Asian Journal of Information Technology 15 (1): 46-53, 2016
ISSN: 1682-3915
© Medwell Journals, 2016

Enrich Mobile Data Availability Using Customized Delegate Object Migration Model in Distributed Mobile Environment

[1]N. Shenbagavadivu, [2]M. Bhuvaneswari and [3]I. Bremnavas
[1]Department of Computer Applications,
Anna University-BIT Campus, Trichy, Tamil Nadu, India
[2]Department of ECE,
[3]Department of Computer Engineering and Networks,
College of CS and IS, Jazan University, Saudi Arabia

Abstract: In present days, mobile computing is a most popular and growing trend as it provides the information access to mobile users irrespective of their locations. The main objective of this study is to develop a customer specific delegate object migration model for distributed mobile systems to enhance the availability and manipulation of mobile information. The delegate object is a run time entity that is acts on behalf of a Mobile Host (MH) hosted on Mobile Support Station (MSS) which holds application specific data structures and methods. A mobile host should be able to access its application specific details in anytime and anywhere. There is a need to migrate and customize delegate object to the current location of the mobile host. The proposed model is a customizable approach, meaning that host and application specific constraints can be enforced and improves the performance of updated mobile information retrieval when the mobile host travels intensively to different locations using delegate object model.

Key words: Delegate object, mobile host, distributed mobile systems, migration, Customize

INTRODUCTION

In past two decades, mobile communications has experienced in a tremendous explosive growth. Today millions of people around the world use mobile phones. Mobile phones allow a person to make or receive a call from almost anywhere. Mobile computing represents a new paradigm that aims to provide continuous network connectivity to users regardless of their location. Coupled with the advent of wireless networking this has given rise to a new style of computing wherein the computer can move with the user and yet maintain its network connections, isolated computers to share resources giving rise to distributed computing (Satyanarayanan, 1996; Forman and Zahorjan, 1994). Today, mobile environment is wide spread and the mobile device resource capabilities are high. This makes them not only to be used as a communication medium but also as a data carrier. This implies that mobile phones will be a popular client system in the network (Khedo and Subramanian, 2009).

Object oriented technology is widely accepted as a suitable methodology for the construction of distributed applications. However, this approach can lead to higher resource consumption than other methodologies. The proposed model is implemented using a delegate object in a distributed mobile environment. The delegate object is a run time entity that acts on behalf of a Mobile Host (MH) hosted on Mobile Support Station (MSS) (Janakiram *et al*., 2005).

This research describes the new model to create and migrate a smaller version of the delegate object instead of passing the complete delegate object. This customization of the delegate object improves the performance of mobile applications.

MATERIALS AND METHODS

Distributed mobile systems: A wireless/mobile computing system is a distributed system which consists of both MH and static MSS nodes. A set of dynamic and wireless communication links can be established between a MH and a MSS and a set of high speed communication link is assumed between the MSS. The MSS may communicate with a number of MH but a MH at a time communicates with only one MSS (Gupta *et al*., 2008). The MH communicates with the rest of the system via its MSS. Distributed computation in mobile computing environment is performed by a set of processes concurrently on MH and MSS in the network (Shenbagavadivu and Savithri, 2012). Figure 1 shows the system model for distributed mobile systems.

When an MH roams and moves out of the cell and enters a new cell, a handoff procedure is executed

Corresponding Author: N. Shenbagavadivu, Department of Computer Applications, Anna University-BIT Campus, Trichy, Tamil Nadu, India

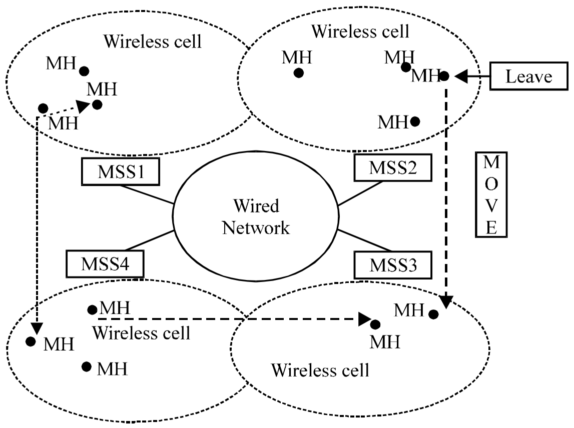


Fig. 1: System model for distributed mobile systems

between the two MSS associated with the cells. The communication between MH in the network is through message passing (Shenbagavadivu and Savithri, 2012). The MSS maintain separate data structures to identify the list of MH which are within its cell's regularly broadcast and talk with the MH which are within its cell using a beacon message to keep track of the MH presence within its cell.

Conventional, communication model: In the distributed mobile system a mobile device acts as client device as well as a data source for providing relevant information needed for business process. When one Mobile Host (say MH_Source) needs to communicate with other Mobile Host (say MH_Dest) for access the business information conventionally there would be direct communication established between the two MH through MSS. The following procedure (Shenbagavadivu and Savithri, 2012) describes the flow of data between the destination device and source device:

- The MH_Source sends a query message to its local Mobile Support Stations (say MSS1)
- MSS1 receives the query message and sends a request to the location server for the current location of MH_Dest
- Location server looks up the reference table for the entry of MH_Dest and returns the current location information (Assume, MH_Dest currently resides in MSS2) to MSS1 in the form of a message
- The query message is transferred from MSS1 to MSS2 if the location information is valid otherwise the error message is forwarded to MH_Source
- MSS2 accept the query message and check if MH_Dest is a registered mobile host. If it is a registered mobile host then the message is forwarded to MH_Dest otherwise an error message is returned to MSS1
- MH_Dest receives a query message from MSS2 and returns the requested business data in the form of reply message to MSS2
- MSS2, receives the reply message and forward it to MSS1
- MSS1, receives the message from MSS2 and forward it to MH_source

In the above discussion, the destination MSS (MSS2) happens to be the same as the source MSS (MSS1) therefore, steps 2-4 and 7 are not required. As described in the algorithm above, eight packet transmissions over the network are required for the retrieval of a query result, four of those transmissions over the wired portion of the network and another four are over the wireless portion. This represents the average case for the number of packet transfers for this application. Normally, the wireless communication is being unstable; the availability of mobile devices will be much felt during the business transaction. To increase the response time and content availability the delegate object is introduced as a middleware component.

Delegate object model: The delegate object acts as a representative of the MH. This bridges the MH and its support environment. The wireless network is highly unstable. This causes the mobile host to frequently connect and disconnect from the mobile network. Irrespective of the availability of the mobile host, the delegate object remains active and maintains information about the mobile host. The major advantages of using the delegate object architecture are:

- It provides solution to the instability of mobile network in a distributed mobile system
- It acts as a data source for handling data dissemination to provide mobile data access, in both server-push and client-pull models
- The delegate object can also cache mobile host specific data and reduce the response times for many client queries. It also supports disconnected operations of the MH by buffering client requests or using the cached data to handle them
- It provides optimal utilization of wireless bandwidth as the delegate object knows the current network connectivity and other constraints of its corresponding host

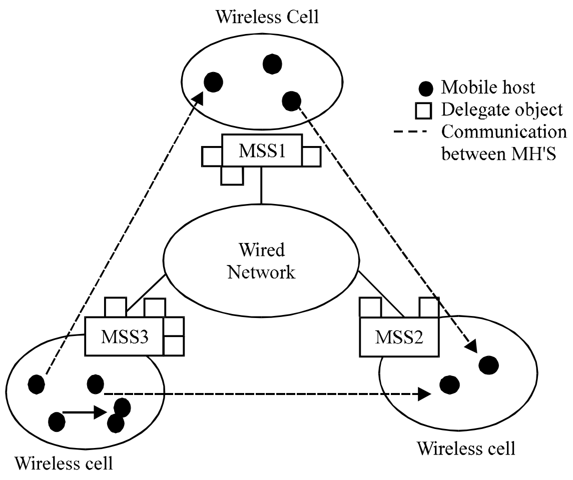
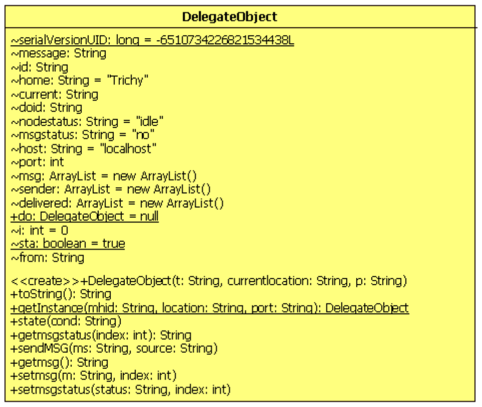


Fig. 2: Distributed mobile system with delegate object model



```
DelegateObject
~serialVersionUID: long = -6510734226821534438L
~message: String
~id: String
~home: String = "Trichy"
~current: String
~doid: String
~nodestatus: String = "idle"
~msgstatus: String = "no"
~host: String = "localhost"
~port: int
~msg: ArrayList = new ArrayList()
~sender: ArrayList = new ArrayList()
~delivered: ArrayList = new ArrayList()
+do: DelegateObject = null
~i: int = 0
~sta: boolean = true
~from: String

<<create>>+DelegateObject(t: String, currentlocation: String, p: String)
+toString(): String
+getInstance(mhid: String, location: String, port: String): DelegateObject
+state(cond: String)
+getmsgstatus(index: int): String
+sendMSG(ms: String, source: String)
+getmsg(): String
+setmsg(m: String, index: int)
+setmsgstatus(status: String, index: int)
```

Fig. 3: Class diagram of delegate object

The structure of delegate object model: The structure of a distributed mobile system with delegate object is as shown in Fig. 2.

The delegate object structure also contains methods to process the MH and m-business data. In the distributed environment each mobile host is has a delegate object.

Structure of delegate object: The class diagram shown in Fig. 3. It represents the structure of a delegate object. Data member's message and msg are used to hold current message and list of messages, respectively.

The data members home and current are used to store home MSS and current location details.The mode status helps to identify whether the MH is in active mode or not. Similarly, the message status helps to check the message delivery. Delegate object identifier is unique identifier assigned for each object. Methods are available to send and receive messages. A new delegate object for a particular MH is created using get Instance method. It contains methods to set and get the status of message.

Customized delegate object migration model: This model focuses on creating a smaller version of the delegate object. This can be achieved by making a new MH to register for the application services needed while the MH is roaming. This information can be stored in the data structure and can be used when the MH leaves its home cell to another MSS. There can be multiple customized data structures created according to the business type. These data structures reside in all MSS in a distributed mobile system. Instead of passing the complete delegate object a new object (customized object) can be created based on the services which are registered by the MH and a corresponding data structure. The customized object which is serialized and sent can be handled by the Foreign MSS as the same data structures are available on all the MSS which allows deserialization and reconstruction of the object on the Foreign MSS.

Need for migration: The MH is in Foreign MSS the communication between the mobile host and its delegate object takes long time. This leads to increased latency and heavy traffic in the wired network, when there is a need to maintain consistency between current state of the MH and its associated object (Henning, 1998). This is because the messages need to travel more hops as they are physically separated out by a long distance. Therefore, there is a need to hold the MH and its corresponding delegate object in the same location. This can be achieved by migrating the delegate object from home MSS to Foreign MSS. But, the challenge is involved in identifying the real destination MSS as and when a MH moves from home MSS to destination MSS. It might pass through a lot of intermediate MSS. Migration in this scenario would cause delegate object to be moved too and fro on reaching each MSS. So, the optimized solution is to move the object to a MSS from which maximum queries are received or managing a cache which holds movement pattern of a MH.

Need for customization: In the above scenario, the need for object migration is analysed. The model results in migration of entire object to the Foreign MSS. The entire delegate object is serialized and sent to the other MSS. This causes a heavy network load and has an adverse effect on performance of the MSS. Moreover, all business services and data are not required at all locations. When a filtration is applied it reduces the amount of data to be serialized and transferred. This would improve the performance of MSS and reduces the network traffic. This is achieved by the customized delegate object model which is discussed as follows.

Migration algorithm for customized delegate object model:

- Registration of Mobile Host (new MH) in a Mobile Support Stations provides object access privileges and Roaming Service (RS) details
- The MH_Source wants communicate to MH_Dest then MH_Source sends a query message to its local Mobile Support Stations (say MSS1)

Case 1: The MH_Source and MH_Dest is located in the same MSS:

- The MSS1 receives query message and check the delegate object of MH_Dest (DOd) is currently hosted within the MSS1
- DOd is in MSS1 then the query message is forwarded from MSS1 to DOd

Case 2: The DOd finds the requested data in its cache then it transferred the information to MH_Source. It requires just two wireless packet transmissions:

- MH_Source->MSS1
- MSS1->MH_Source. This represents the best case for the new model

Case 3: The DOd is empty (DOd does not contains requested information) DOd communicates to MH_Dest then forward the message to MH_Source. It requires four wireless packet transmissions:

- MH_Source->MSS1
- MSS1->MH_Dest
- MH_Dest->MSS1
- MSS1->MH_Source

Case 4: The MH_Source and MH_Destis located in different MSS:

- The MSS1 receives query message and check the delegate object of MH_ Dest (DOd) is currently hosted within the MSS1
- DOd is not in MSS1 then it communicates to the location server looks up the reference table for the entry of DOd and returns the object reference of DOd to MSS1 (Assume that DOd currently resides in MSS2)
- The response from the location server is accepted by the MSS1 then send a message to MSS2
- The MSS2 receives the query message and check the DOd is currently hosted within the MSS1. The DOd is in MSS2 then forward the query message to Dod

Case 5: The DOd finds the requested data in its cache then it transferred the information to MH_Source. It requires just two wireless packet transmissions and four wired packet transmission:

- MH_Source->MSS1
- MSS1->Location Server
- Location Server->MSS1
- MSS1->MSS2
- MSS2->MSS1
- MSS1->MH_Source

Case 6: The DOd is empty (DOd does not contains requested information). DOdcommunicates to MH_Dest then forward the message to MH_Source. It requires four wireless packet transmissions and four wired packet transmissions:

- MH_Source->MSS1
- MSS1->Location Server
- Location Server->MSS1
- MSS1->MSS2
- MSS2->MH_Dest
- MH_Dest->MSS2
- MSS2->MSS1
- MSS1->MH_Source

Case 7: MH_Dest is in roaming. MH_Source and MH_Dest are belong to same MSS(MSS1) but MH_Dest is located in Foreign MSS(MSS2) (DOd is in home MSS and MH_Dest located in different MSS)

- The MSS1 receives query message and check the delegate object of MH_Dest (DOd) is currently hosted within the MSS1
- DOdis in MSS1 then the query message is forwarded from MSS1 to DOd

Case 8: The DOd finds the requested data in its cache then it transferred the information to MH_Source. It requires just 2 wireless packet transmissions:

- MH_Source->MSS1
- MSS1->MH_Source

This represents the best case for the new model.

Case 9: The DOd is empty (DOd does not contains requested information). DO1 communicates to MH_Dest then forward the message to MH_Source. It requires four wireless packet transmissions and four wired packet transmission:

- MH_Source->MSS1
- MSS1->Location Server
- Location Server->MSS1
- MSS1->MSS2
- MSS2->MH_Dest
- MH_Dest->MSS2
- MSS2->MSS1
- MSS1->MH_Source

Case 10: MH_Dest is in roaming. MH_Source and MH_Dest are belong to different MSS (MSS1, MSS2). Assume that MH_Dest is located in foreign MSS (MSS3) (DOd is in home MSS (MSS2).

- The MSS1 receives query message and check the delegate object of MH_Dest (DOd) is currently hosted within the MSS1
- DOd is not in MSS1 then it communicates to the location server looks up the reference table for the entry of DOd and returns the object reference of DOd to MSS1 (Assume that DOd currently resides in MSS2)
- The response from the location server is accepted by the MSS1 then sends a message to MSS2
- The MSS2 receives the query message and check the DOd is currently hosted within the MSS2. The DOd is in MSS2 then forward the query message to DOd

Case 11: The DOd finds the requested data in its cache then, it transferred the information to MH_Source. It requires just two wireless packet transmissions and four wired packet transmissions:

- MH_Source MSS1
- MSS1Location Server
- Location ServerMSS1
- MSS1MSS2
- MSS2MSS1
- MSS1 MH_Source

Case 12: The DOd is empty (DOd does not contains requested information). DOd communicates to MH_Dest then forward the message to MH_Source. It requires four wireless packet transmissions and six wired packet transmissions:

- MH_Source->MSS1
- MSS1->Location Server
- Location Server->MSS1
- MSS1->MSS2
- MSS2->MSS3
- MSS3->MH_Dest
- MH_Dest->MSS3
- MSS3->MSS2
- MSS2->MSS1
- MSS1->MH_Source

In the above procedure Case 9 and 10 deal with roaming services of a MH that is the delegate object of MH is in home MSS and the MH is located in foreign MSS.

Implementation of prototype: The prototype system is implemented entirely in Java environment. The Sun Java J2ME Wireless Toolkit 2.5 simulator is used to simulate hand-held terminal device. Java RMI has been used for simulating distributed environment. The Tomcat 5.0 is used to maintain the location based web applications and data are maintained in oracle database.

Registration of new mobile host and creation of delegate object: The new MH enters in the distributed mobile system, it registers itself in the MSS. In the registration process is shown in Fig. 4.

The user enters his/her personal information and mobile number in the J2ME midlet applications. This information is sent to the registration server. The registration server is a server side t program which stores the information in the database then it invokes a remote method createMHID for generating MH Identifier and instantiate a delegate object using getInstance method and assign a unique delegate object identifier.

Implementation of location server and communication between mobile hosts's in different mobile support stations: The location server activities are an implemented using RMI service which is represented in Fig. 5. When the MH_Source sends a message destined to MH_Dest, it will be forwarded to local MSS (say MSS1). When MSS1 receives the message and check if the delegate object for MH_Dest is currently hosted. If not found, it sends a message to the location server for requesting the

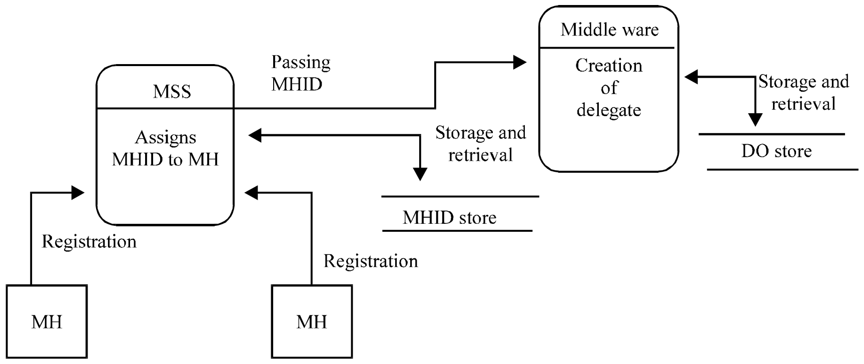


Fig. 4: Registration of new MH with MSS

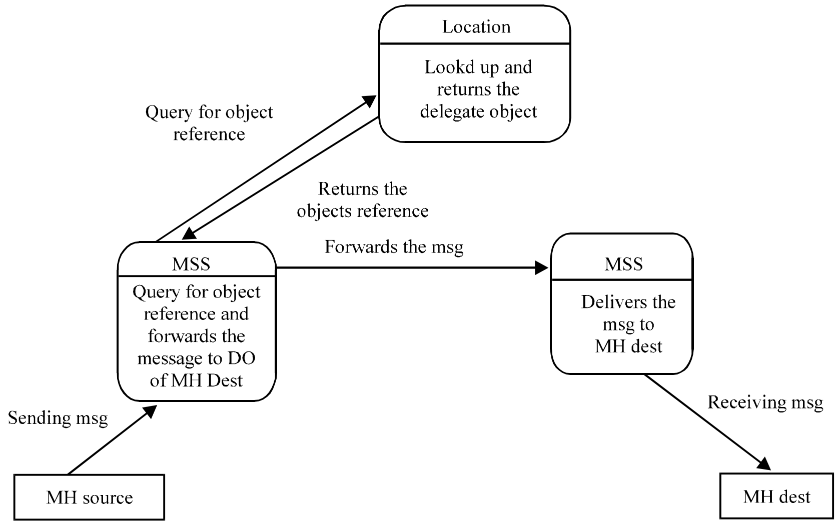

Location
Lookd up and returns the delegate object
Query for object reference
Returns the objects reference
Forwards the msg
MSS
Query for object reference and forwards the message to DO of MH Dest
MSS
Delivers the msg to MH dest
Sending msg
Receiving msg
MH source
MH dest

Fig. 5: Implementation of location server

object reference of the delegate object of MH_Dest. The location server looks up the reference table entry for the location of delegate object of MH_Destand returns the location information in the form of object reference to MSS1.

RESULTS AND DISCUSSION

Performance analysis: Two testing applications are developed to analyze the performance of delegate object in m-business application system. These two tests are:

- Test A: Performance analysis of thin client (mobile client) model without delegate object
- Test B: Performance analysis of thin client model with delegate object These test are used to measure Round Trip Time (RTT)

In test A, client request is processed by data retrieval web component. The data retrieval web component invokes a related web services. The web service communicates with the database and retrieves the updated value and returns to the client. Each time, a new data or updated data is to be retrieved, the web component has to send a new request to the database. In this case, the web component is unaware of the changes made in the database. Therefore, this model uses a polling pattern to place a request to the database at frequent time intervals. The round trip time is measured for all the client requests.

In Test B, the performance is evaluated for thin client architecture using a delegate object. The web component forwards the request to middle tier. The middle tier or the application layer contains delegate object and web service to manage the delegate object. A delegate object is created for each business need and a separate web

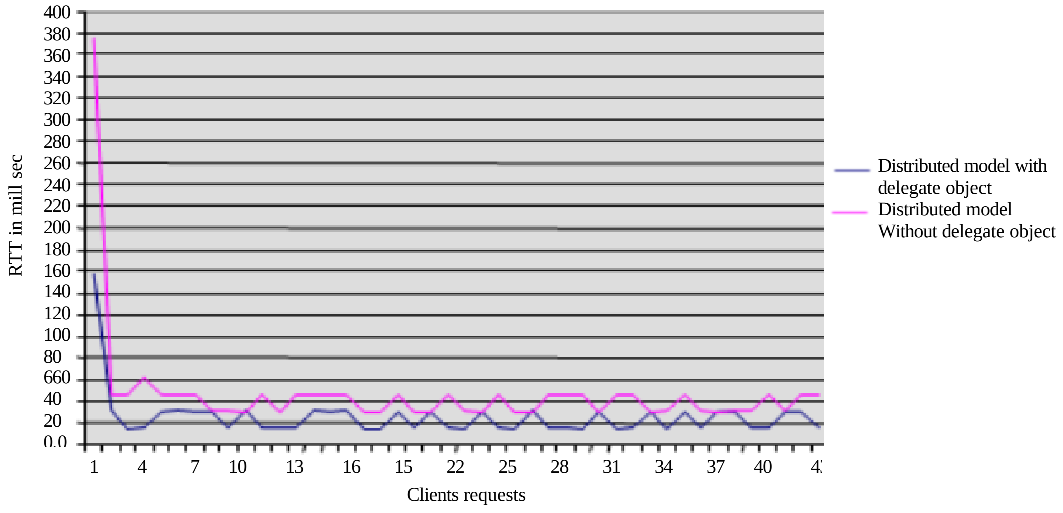


Fig. 6: Performance analysis using RTT; thin client model

Table 1: Average round trip time in milli seconds for thin client model

Distributed model	Average RTT in milli sec
Distributed model with delegate object	25.97727
Distributed model without delegate object	47.65909

service implements the logic for the business function. To process the request, the data retrieval component invokes the corresponding web service . The web service gets the instance of the appropriate delegate object that is associated with the requested service. The updated information from the delegate object is retrieved by the web service. The web service also manages the data held by the delegate object. The web service responds to the presentation layer.

The RTT is measured for all the client requests. In both the models the initial RTT time is higher as the web component establishes a connection with the database. The further queries are implemented asynchronously thereby reducing the RTT value. In the above two cases, the RTT measures in terms of milli seconds are calculated and plotted as a graph which is shown in Fig. 6. The average RTT is evaluated in terms of milli seconds which is shown in Table 1.

By comparing the RTT computed for both the models it is determined that using delegate object model's average RTT time is less than the model without using delegate object.

The major factor that analyses the performance of the delegate object model is RTT. The average RTT of thin k client model is shown in Table 1. It is determined that using delegate object model's average RTT time is less compared to the model without using delegate object. The proposed delegate object model is considered as one of the best possible alternatives for m-business environment since, it follows asynchronous communication patterns.

CONCLUSION

The delegate object model is developed for distributed mobile system to enhance the content availability in m-business applications. In order to eliminate the object migration time and to enhance the reliability of the distributed mobile system, it is proposed to customize the delegate object at the time of migration.

The customized delegate object migration model elegantly solves a whole set of problems in distributed mobile systems: location management, mobile data access in client-server systems, disconnected operations, etc. The delegate object model has an effect on performance of m-business transaction by reducing the network overhead of huge number of transaction requests. This is tested in multiple simulation environments and their performance evaluation is carried out with respect to RTT between the request and response. The developed distributed models can be tested with large scale e-business and m-business environment in real time for portability across different service providers and heterogeneous network scenario.

REFERENCES

Forman, G.H. and J. Zahorjan, 1994. The challenges of mobile computing. Computer, 27: 38-47.

Gupta, S.K., R.K. Chauhan and P. Kumar, 2008. Backward error recovery protocols in distributed mobile systems: A survey. J. Theor. Applied Inform. Technol., 4: 337-347.

Henning, M., 1998. Binding, migration and scalability in CORBA. Commun. ACM., 41: 62-71.

Khedo, K.K. and R.K. Subramanian, 2009. A service-oriented component-based middleware architecture for wireless sensor networks. Int. J. Comput. Sci. Network Secur., 9: 174-182.

Satyanarayanan, M., 1996. Fundamental challenges in mobile computing. Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing, May 23-26, 1996, ACM, Philadelphia, PA, USA, pp: 1-7.

Shenbagavadivu, N. and S.U. Savithri, 2012. Enhanced information security in distributed mobile system based on delegate object model. Procedia Eng., 30: 774-781.