Customized Delegate Object Migration Model for Distributed Mobile Environment

N.Shenbagavadivu[1] , Dr E.Kirubakaran[2] ,R.S.Ponmagal[3]
[1] Lecturer, Department of Computer Science and Engineering , Anna University, Tiruchirappalli
[2] Chairman, Computer Society of India, BDU Technology Park, Tiruchirappalli
[3] Lecturer, Department of Computer Science and Engineering, Anna University, Tiruchirappalli

Summary
Mobile computing environment provides the information access to mobile users irrespective of their locations. The main objective of this paper is to develop a customized delegate object migration model for distributed mobile systems. The delegate object is an ambassador for a particular Mobile Host (MH) that maintains application specific data structures and methods which are required on mobility of the mobile. A mobile host should be able to access its application specific details anytime and anywhere. There is a need to migrate and distribute delegate object to the current location of the mobile host. The proposed model is a customizable approach, meaning that host specific and application specific constraints can be enforced and also improves the performance of delegate object migration.

Key words:
Delegate Object, Mobile Host, Distributed Systems

1. Introduction

Mobile communications has experienced explosive growth in the past two decades. Today millions of people around the world use cellular phones. Cellular phones allow a person to make or receive a call from almost anywhere. Likewise, a person is allowed to continue the phone conversation while on the move. Cellular communications is supported by an infrastructure called a cellular network, which integrates cellular phones into the public switched telephone network [2].
Mobile Computing represents a new paradigm that aims to provide continuous network connectivity to users regardless of their location. A wide spectrum of portable, personal computing devices has recently been introduced in the market such as portable computers (laptops) with wireless interfaces, Personal Digital Assistants (PDAs),cell phones etc. 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 [1,2].
Delegate object is a run time entity that acts on behalf of a Mobile Host (MH), hosted on Mobile Support Station. This paper describes the new model to create and migrate a smaller version of the delegate object instead of passing the complete surrogate object.

2. Distributed Mobile Systems

A distributed mobile environment consists of a set of static hosts and a set of mobile hosts. A MH is a host that can move (can change its location with time) while retaining its network connection. A stationary host in contrast, as its name implies, does not change its location and communicates with other stationary hosts via a wired fixed network. Some stationary hosts also serve as an infrastructure computer to support communication between mobile hosts and are called mobile support systems (MSS) [3,4].

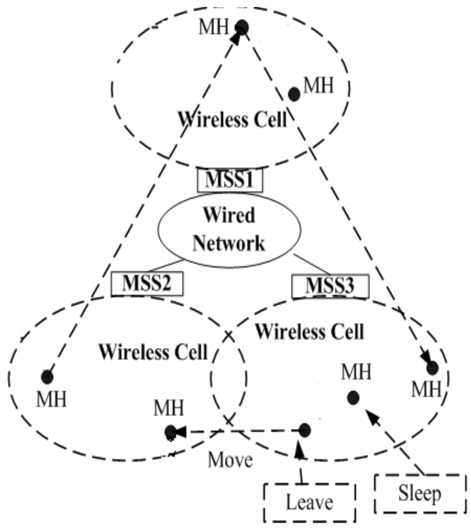


Fig. 1: System model for Distributed Mobile Systems

The geographical area within which an MSS supports mobile hosts is called a cell. An MSS communicates directly with all the mobile hosts in its cell through a wireless channel. A MH can communicate directly with an MSS only if the MH is located within the MSS's cell.

Manuscript received November 5, 2009
Manuscript revised November 20, 2009

An MH communicates with other hosts through its MSS. The static hosts and the communication channels among them constitute the static part of the mobile computing environment. The set of all mobile hosts constitute the mobile part of the environment. An example of a mobile computing environment is shown in Figure1.
A mobile host can migrate from one cell to another cell at any time. After moving to the cell of another MSS, a mobile host establishes contact with the new MSS using the beacon protocol [5], and then informs the new MSS the id of its old MSS. A handoff procedure is then executed between the new MSS and the old MSS. The new MSS informs the old MSS about the mobile host's migration, and gets the data structures and other relevant information associated with the mobile host from the old MSS.
The communication between hosts in the hosts in the network is through message passing [6].The MSS maintains separate data structures to identify the list of MHs which are within its cell's regularly broadcasts and talk with the MHs which are within its cell using a beacon message to keep track of the MHs presence within its cell.

3. Delegate Object Model

The model involves bridging the device and its support environment, using a place-holder namely the delegate object. This is done by creating the delegate in the static network to act on behalf of each mobile device. With wireless communication being unstable, the availability of mobile devices will not be much felt with the delegate object architecture. The delegate object can remain active, maintaining information regarding the current state and plays an active role on behalf of the devices.
The major advantages of using the surrogate architecture are:

- It provides an elegant solution for handling the asymmetry in distributed mobile system. The delegate object model is a customizable approach, meaning that host specific and Application specific constraints can be enforced. This would be difficult to achieve without the delegate object model [4,5].
- It also provides an ideal for MH location MH location information, thus solving the Location management problem in distributed mobile systems.
- 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 MHs 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.

3.1 The Structure of Delegate object Model

Delegate object is a run time entity that is hosted on some mobile support station and acts on behalf of a mobile device. It contains data structures relevant to the MH and methods to act upon them .Some of the methods are the interfaces being called upon by the MSS. The model helps in handling mobility of the mobile devices and helps in effective handling of the available bandwidth. In addition its helps in hiding the asymmetry between the wired and the wireless network in the distributed mobile systems. The structure of a distributed mobile system with delegate object is as shown in figure 2.

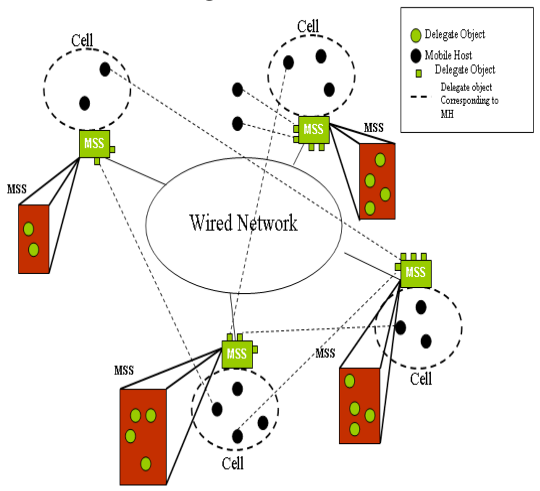
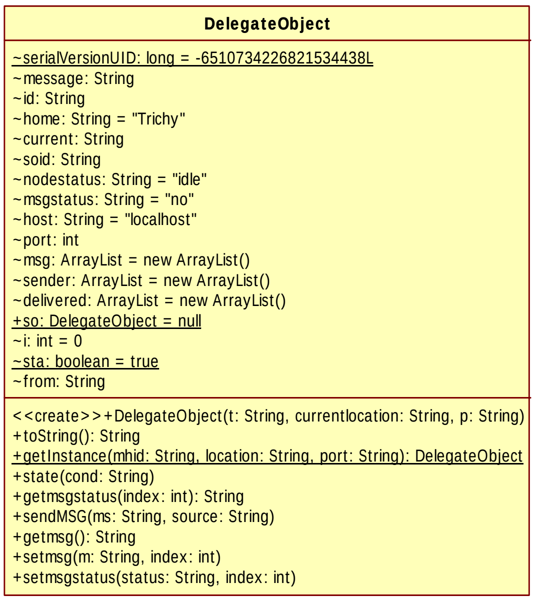


Fig. 2: Structure of a Distributed Mobile Systems with Delegate Objects

3.2 The Structure Of Delegate Object

The above class diagram represents the structure of a Delegate Object. Data Members message and msg are used to hold current message and list of messages respectively. "soid " 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 getInstance method. It contains methods to set and get the status of message.



```
DelegateObject
~serialVersionUID: long = -6510734226821534438L
~message: String
~id: String
~home: String = "Trichy"
~current: String
~soid: String
~nodestatus: String = "idle"
~msgstatus: String = "no"
~host: String = "localhost"
~port: int
~msg: ArrayList = new ArrayList()
~sender: ArrayList = new ArrayList()
~delivered: ArrayList = new ArrayList()
+so: 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)
```

3.3 Delegate object Identifier (DOID)

In the proposed modal, whenever a MH newly joins the distributed mobile system, it registers itself with an MSS. The MSS assigns a unique identification for the MH namely, the mobile Host Identifier (MHID) and passes the information to the underlying middleware about the entry of a new MH to the system. The middleware creates an object corresponding to the MH and assigns a unique Delegate object Identifier (DOID). The middleware adds the following entry in the naming service: the object name (the MHID is the object name) and the corresponding object reference.

When MH_1 wants to communicate with MH_2, the MSS to which MH_1 belongs uses the naming services of the middleware and gets the object reference of the surrogate corresponding MH_2. Using the object reference, the delegate object associated with MH_2 is located and MH_1 starts communicating with the delegate object of MH_2, instead of actually communicating with MH_2. The delegate object identifier and the mobile host identifier could be some random numbers generated using random number engines. The object corresponds to SOID and MHID can be obtained by means of remote invocations.

4. Proposed Customized Delegate Object

Migration Model

The new model proposes to create 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. Instead of passing the complete delegate object a new object (customized object) can be created based the services which were registered by the MH for roaming. This improves the performance of data transfer and also improves the accesses to services since only limited number of services is available.

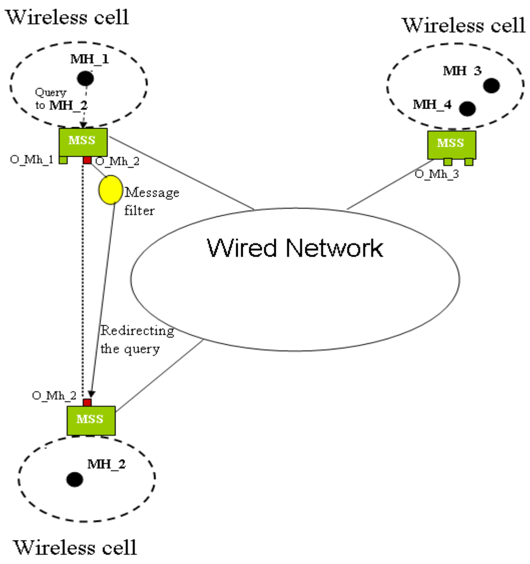


Fig. 3: Delegate object Migration

4.1 Need for Migration

Mobility implies that a MH changes its location. The location management for the targeted MH becomes an indispensable task of any application that runs on the distributed mobile system [13]. This complicated issue could be handled effectively without affecting the delivery using our proposed model. The sender of a message targeted at a MH need not bother about whether the MH is in motion or out of coverage region. All that needs to be done is to deliver the message to the delegate object which is residing in the static portion of the network [14]. The delegate object takes opportune time to deliver the message to its MH considering the availability of the wireless bandwidth and the traffic on the network. Thus, after some point of time the MH might have moved to a far off place with respect to the surrogate object. 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 surrogate object [15]. This is because the messages need to travel more hops as they are physically separated out by a long distance. Hence, it will be advisable to keep the MH and its associated delegate object as near as possible.

4.2 Condition for Migration

Migrating the delegate object from one MSS to another with the movement of the MH from one cell to the other will also be inefficient. There will also be cases where the MH may keep moving back and forth between cells. In such cases swapping the delegate object between the MSSs will be highly inefficient. Thus, both the cases of never moving the surrogate objects or always moving the surrogate objects are ineffective. This requires identifying the criterion for migrating the surrogate objects. Some of the criterion for migrating the surrogate objects can be:

- Move to the cell which is the source of maximum queries.
- Move to a less loaded MSS.
- Move from "oldMSS" to "newMSS" after the MH has made 'n' cell changes away from the "oldMSS" and is currently in region of the "newMSS".
- Move to a cell based on the earlier movement pattern which is recorded as histogram in each MSS, about the movement of each MH.

4.3 Migration algorithm

1. Registration of Mobile host (new MH) in a MSS-provides Roaming Service (RS) details

2. 1. (a) The client host (say MH_S) enters the query including the send time of the message and its status (NOT_ACKD).
(b) MH_S sends the query message to its local MSS (say MSS_S)

Case 1: MH_S is located in the cell of MSS_S and the DO of MH_d is hosted on the same MSS. Also, the DO finds the requested data in its cache.
We require just 2 wireless packet transmissions:
1. Step 1 => MHs —> MSS_S
2. Step 2 => MSS_S —> MH_S

This represents the best case for the new model.

Case 2: The DO of MH_d is hosted on MSS_0 (o! = s) and the DO finds the requested data in its cache.
Here we require 2 wireless packet transmissions:
1. Step 1 => MH_S —> MSS_S
2. Step 2 => MSS_S —> MH_S

and 4 wired packet transmissions:
1. Step 1 => MSS_S —> Location server
2. Step 2 => Location server —> MSS_S
3. Step 3 => MSS_S —> MSS_0
4. Step 4 => MSS_0 —> MSS_S

This represents the average case for cache hit.

Case 3: The DO of MH_d is hosted on MSS_0 (o! = s) and the DO finds the requested data in its cache. If it needs object migration then
Here we require 2 wireless packet transmissions:
1. Step 1 => MH_S —> MSS_S
2. Step 2 => MSS_S —> MH_S

and 4 wired packet transmissions:
1 Step 1 => MSS_S —> Location server
2 Step 2 => Location server -> MSS_S
MSS_S generates a customer specific customized delegate object (CDO) based on RS where size of CDO < DO
3. Step 3 => MSS_S —> MSS_0
CDO is migrated from MSS_S *to* MSS0 (All requests are handled by (CDO)
4. Step 4 => MSS_0 —> MSS_S
(Synchronization processes is handled)

This represents the average case for cache hit... In case replies to queries do not arrive even after an expected delay, MSS_S and MH_S can independently take action by resending the query. In the face of packet losses, such cases become more and more common, pushing the average query time higher in addition to hogging the network bandwidth with copies of lost packets.

5. Simulation Test bed

The application was implemented over a simulated model of a Distributed Mobile Systems with the following characteristics:

* The simulation is discrete event in nature. Some of the events may generate more events which are inserted in their proper position in the event queue. Thus, the process is a generic discrete event simulation system with the front-end having no knowledge about the details involved.

* A similar thinking has gone into the design for the message transmissions in the network. All entities being transferred over the network are inherited from a packet class. So, addition of new types of messages does not affect other portions of the program. The messages have a method named 'execute' which encapsulates the behavior of the message. So, a receiver simply calls this method for any/ every message it receives.

* The cellular area is specified through a file. It gives the number of MSSs and the region they cover. One interesting feature is that the cells are allowed to be of arbitrary shapes and sizes.

5.1 Implementation of Prototype

The system was implemented based on the above design. we implemented the prototype system entirely in Java. The Sun Java J2ME Wireless Toolkit2.5 simulator is used to simulate hand held terminal device. Java RMI has been used for simulating distributed environment. The Tomcat5.0 was used to maintain the location based applications.

5.1.1 Communication Between The MH's Within The Same Cell

When MH_1 wants to communicate with MH_2, the MSS to which MH_1 belongs uses the naming services of the middleware and gets the object reference of the delegate corresponding MH_2. Using the object reference, the delegate object associated with MH_2 is located and MH_1 starts communicating with the delegate object of MH_2, instead of actually communicating with MH_2.The Fig. 4 shows the communication between the MHs within in the same cell.

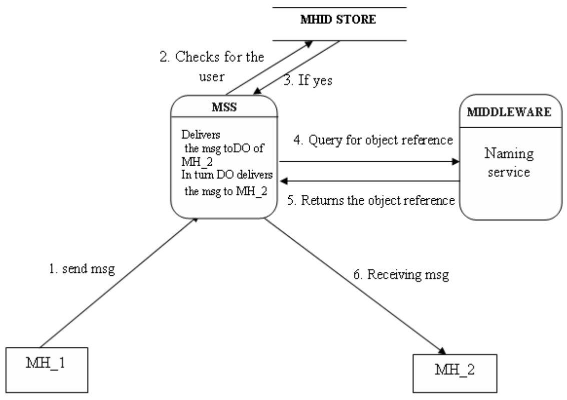

MHID STORE
2. Checks for the user
3. If yes
MSS
Delivers the msg toDO of MH_2 In turn DO delivers the msg to MH_2
MIDDLEWARE
Naming service
4. Query for object reference
5. Returns the object reference
1. send msg
6. Receiving msg
MH_1
MH_2

Fig 4 Communication between the MHs with in the same cell

5.1.2 Implementation of Location Server and Communication Between MH's in Various Cell

When the MH_1 sends a message destined to MH_2, it will be forwarded to local MSS (say MSS_1). When MSS _1 receives the message it checks if the DO for MH_2 is currently hosted. If not found, it sends a message to the location server requesting to the MSS reference where the MH_2 is resides. The location server looks up the entry for the reference table $_{and}$ returns the reference of the corresponding MSS_2 in the form of a message.(Assume, the DO currently resides in MSS_2). An error in lookup results in an error message being returned. if the reply contains the location information. The message is sent to MSS_2. MSS_2, upon receiving the query message checks if the DO for MH_2 is currently hosted. If YES, the message is forwarded to the DO. When the DO for MH_2 receives the message it sends a message to the MH_2. Fig. 5 shows the communication between the MHs with different cell.

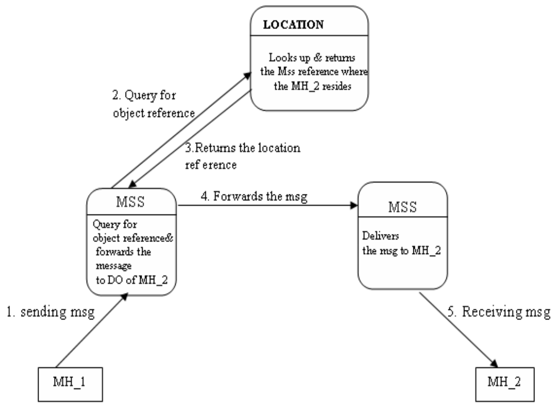
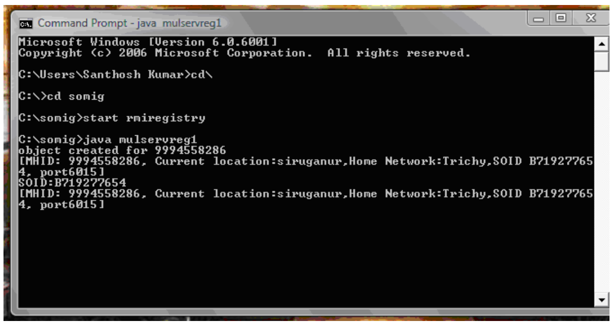

LOCATION
Looks up & returns the Mss reference where the MH_2 resides
2. Query for object reference
3.Returns the location ref erence
MSS
Query for object reference& forwards the message to DO of MH_2
4. Forwards the msg
MSS
Delivers the msg to MH_2
1. sending msg
5. Receiving msg
MH_1
MH_2

Fig 5 implementation of Location server and Communication between the MHs with different cell

5.1.3 Snapshot of prototype system

The following figure 6 shows snapshot for the creation of delegate object, which shows the mobile id, current location and its home network from where the mobile has moved. Fig 7 shows the transmission of messages between mobile hosts



```
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation.  All rights reserved.

C:\Users\Santhosh Kumar>cd\

C:\>cd somig

C:\somig>start rmiregistry

C:\somig>java mulservreg1
object created for 9994558286
[MHID: 9994558286, Current location:siruganur,Home Network:Trichy,SOID B71927765
4, port6015]
SOID:B719277654
[MHID: 9994558286, Current location:siruganur,Home Network:Trichy,SOID B71927765
4, port6015]
```

Fig 6.Creation of Delegate Object

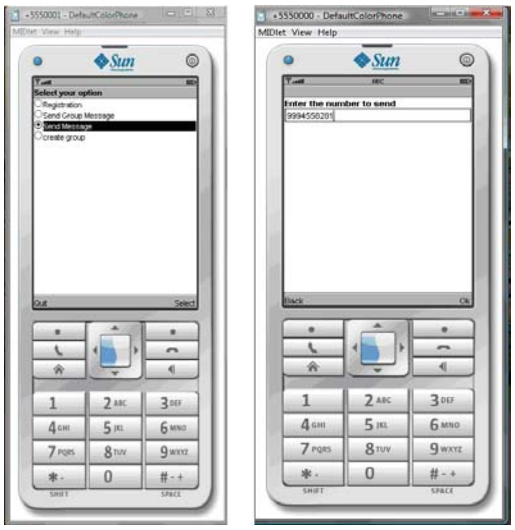


Fig 7. Transmission of Messages between MHs

6. System Evaluation

Fig 8 illustrates the comparison of response times for the delegate object model and customized delegate object model. The migration starting time is plotted against the migrated time of the object. The study shows the response time is quicker for the customized object model than the delegate object model. Hence, the mobile customers are able to get the required information in a shorter time

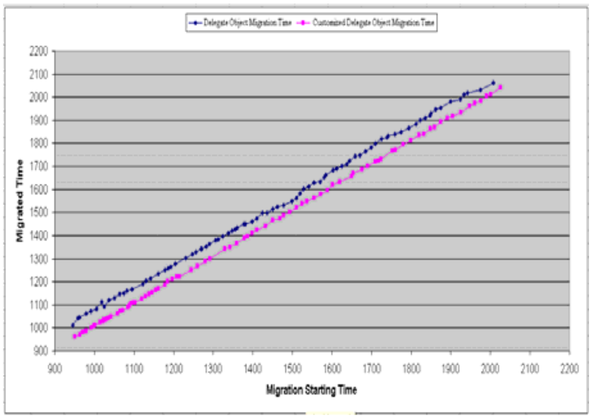


Fig 8. Response Time Comparison For Delegate Object Model And Customized Delegate Object Model

7. Conclusions

The customer specific delegate object migration model presented in this paper provides a new perspective for designing distributed mobile systems. It reduces the data transferring time and increase the performance of handling the client requests. It focuses on the asymmetry being the fundamental issue to be addressed; whish automatically takes care of other issues such as device constraints or mobility. The customer specific 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. It also provides an ideal placeholder for host specific information, thus enabling application or host specific constraints to be enforced. This facilitates building customizable applications over distributed mobile systems.

An interesting direction of future research is to explore the theoretical foundations of the delegate object model. Whether the notions of time and space traditionally used to model distributed systems are sufficient or not to model distributed mobile systems is an important question. We are also currently developing various applications as well as protocols on top of the delegate object model.

References

[1] M. Satyanarayanan. Fundamental challenges in mobile computing. In Proceedings of the 15th ACM Symposium on Principles of Distributed Computing, pages 1-7, May 1996.
[2] G. H. Forman and J.Zahorjan. The challenges of mobile computing. IEEE computer, 27( ), April 1994.
[3] B.R. Badrinath, Arup Acharya, and Tomaiz Imieslinski. Structuring distributed algorithms for mobile hosts. In proceedings of the 14th International Conference on Distributed Computing Systems, June 1994.
[4] K.Brown and S. Singh. RelM: Reliable multicast in mobile networks. Journal of Computer Communications, 21( ):1379-1400, April 1998.
[5] Giuseppe Anastasi, Alberto Bartoli, and Francesco Spadoni. A reliable multicast protocol for distributed mobile systems: Design and evaluations. IEEE Transaction on Parallel and Distributed Systems, 12( ):1009-1022, October 2001.
[6] S. Alagar, R.Rajagopalan, and S. venkatesan. Tolerating mobile support station failures. In Proceedings of the First Conference on Fault Tolerant Systems, pages 225-231, Dec. 1995.
[7] C.Donald Wileox and Gruia –Catalin Roman .Reasoning about places, times, and actions in the presence of mobility.IEEE Transaction on Software Engineering, 22(4):225-231, April 1996.
[8] Dario Bruneo , Marco Scarpa , Angelo Zaia, and Antonio Puliafito.Communication paradigms for mobile grid users. In Proceedings of the 3rd IEEE/ACM International Symposium on Cluster computing and Grid .Tokoyo, Japan, june2003.
[9] Amy L. Murphy, Gruia-Catalin Roman, and George Varghese. Tracking mobile units for dependable message delivery. IEEE Transactions on Software Engineering, 28(5):433-448, May 2002.
[10] Evaggelia Pitoura and George Samaras. Locating objects in mobile computing. IEEE Transactions on Knowledge and Data Engineering, 13(4):571-592, July/August 2001.
[11] Jin Jing, Abdelsalam (Sumi) Helal, and Ahmed Elmagarmid. Client-server computing in mobile

environments. ACM Computing Surveys, 31(2):117 157, June 1999.

[12] J.Ioannidid, D. Duchamp, and G. Q. Maguire. IP based protocols for mobile internetworking. In Proceedings of the ACM SIGCOMM Symposium on Communication, Architectures and Protocols, pages 235-245, Sep. 1991.

[13] J J Kistler and M Satyanarayanan. Disconnected Operation in the Coda File System . ACM Transactions on Computer Systems, 10(l):3 25, February 1992.

[14] Michi Henning. Binding, Migration and Scalability in CORBA. Communications of the ACM, 41(10):62-71, October 1998.

[15] D Janakiram and A Vijay Srinivas. Object migration in corba. Journal of the CSI, 32(1):18-27, March 2002.