Surrogate Object Model: A New Paradigm for Distributed Mobile Systems

M.A. Maluk Mohamed, D. Janakiram and Mohit Chakraborty
Distributed & Object Systems Lab,
Dept. of Computer Science & Engg.,
Indian Institute of Technology, Madras, India.
http://lotus.iitm.ac.in
{maluk, djram}@cs.iitm.ernet.in, mohit@peacock.iitm.ernet.in

Abstract:
Distributed mobile systems are characterized by asymmetry in both network connectivity and computing power, mobility of hosts and their constraints. To address these issues, existing approaches attempt to completely redesign distributed applications or algorithms to make them work in a distributed mobile environment. This paper proposes an alternate mechanism namely, the *surrogate object model* to handle the asymmetry problem in distributed mobile systems. The surrogate object is a representative for a particular Mobile Host (MH) in the wired network that maintains application specific data structures and methods. The surrogate object model maintains a cache of data stored in the mobile host and reduces wireless data transfers. The proposed novel model also provides an ideal placeholder for MH location information, thus solving the location management problem to handle mobility of hosts. A comprehensive simulation study of the surrogate object model has been done. Performance studies of a sample application with and without the surrogate object model illustrates the efficiency of developing applications over the surrogate object model.

Keywords Distributed Mobile Systems, Asymmetry Problem, Surrogate Object, Mobile Environments, Location Management.

1 Introduction

The proliferation of mobile devices coupled with the increasing technological growth of devices and wireless communication, has made distributed mobile systems a potential computing paradigm. Different forms of devices exist, including portable computers (laptops) with wireless interfaces, Personal Digital Assistants (PDAs), cellular phones etc. Most of these devices have constraints: finite energy source; unreliable and bandwidth varying wireless connectivity [Sat96]. These constraints along with the mobility of the devices pose serious challenges for mobile computing [FZ94].

Mobile devices connect to the static network through Mobile Support Stations (MSS) which have the capability to communicate through wireless interfaces. The MSSs are also connected to each other through wired connections. This leads to a basic asymmetry in network connectivity and computing power. The MSSs have comparatively greater computing power and the wired network is more reliable and has a higher unvarying bandwidth. This asymmetry has been a fundamental problem for distributed mobile computing researchers. However, such a conceptualization of the problem does not seem to exist in the literature.

A common problem addressed by several efforts is mobility itself. One approach to solve this problem has been to completely redesign distributed algorithms to work in a mobile environment [BAI94]. Similar approach for the development of a reliable multicast protocol has been proposed in [BS98, ABS01, ARV95]. A formal approach to model distributed mobile system using the notions of time, place and action has been proposed in [WR96]. Although applications can be transparent to these notions, the system software (including the communication protocol) would need to take into account all the intricacies of distributed mobile systems. However, if asymmetry is addressed as the key problem, even the system software can be simplified to a large extent. Mobile agent based approaches for mobile grid computing follow a similar approach, but handle mainly the computing power asymmetry [BSZP03]. Further, the mobile agent resides on the MSS to which the mobile host is attached, implying that for every handoff, the agent has to be migrated. This approach does not address location management of mobile hosts.

A client-mediator-server model is proposed in [ZD95] in order to handle network connectivity problems of MHs in Mobile IP based networks. The proxy server used as the intermediary is located at a fixed point in the network, irrespective of mobility of MH. This may complicate the communication between the source and destination, leading to the dogleg routing problem. Mobility Extender has been proposed as a mechanism for seamlessly integrating mobility into distributed systems [PR97]. It is an application level design pattern, where the various objects such as constraint meta-object, proxy etc. are non-transparent to the application developer. Further, the proxy object is migrated with every handoff, leading to increased network traffic. However, both the above efforts do not address asymmetry as the key problem.

Difficulty in locating mobile hosts is one immediate consequence of the asymmetry problem. Existing approaches for handling location management of mobile hosts such as [MRV02] increase the granularity of tracking a mobile node to within a subnet. However, in this as well as in other efforts the most important issue for location management is where to place the location information, whether to keep it in all network locations or in one central node or at selected locations [PS01]. The trade-off between reducing the number of updations and querying time is not easy to address. This paper presents the surrogate object model, that provides an elegant solution by having a place holder for storing location information: the surrogate object. Further, many surrogate objects can be distributed throughout the wired network, each maintaining location information about a particular host.

The key features of the surrogate object model are as follows:

- It provides an elegant solution for handling the asymmetry in distributed mobile systems. The surrogate 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 surrogate object model.

- It also provides an ideal placeholder for 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 [JHE99].
- The surrogate 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 surrogate object knows the current network connectivity and other constraints of its corresponding host.

The rest of the paper is organized as follows: Section 2 explains the basic distributed mobile system model over which the surrogate object model has been built. Section 3 describes the surrogate object model in detail, including surrogate object migration and how it handles constraints of the devices. Section 4 gives a sample contour mapping application realized with and without the surrogate object. Section 5 compares the performance of the contour application with and without the surrogate object. It also illustrates other performance considerations in the surrogate object model, including surrogate object migration and caching. Section 6 concludes the paper and provides directions for future research.

2 System Model

A distributed mobile system is viewed as a two-tier architecture consisting of a set of mobile hosts (MHs) and stationary hosts. A MH can move (can change its location with time) while retaining its network connections [JDM91]. 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 Stations (MSS).

A MSS communicates directly with the MHs which are within its cell[1] through a wireless channel. A MH may belong to only one cell at any given time. A MH can communicate with other MHs and MSSs only through the MSS to which it belongs. The model also supports incomplete coverage of wireless cells. When an MH roams and moves out of the cell and enters a new cell, a handoff procedure is executed between the two MSSs associated with the cells. The communication between hosts in the network is through message passing.

The MSS maintains separate data structures to identify the list of MHs which are within its cell. MSS regularly communicates with the MHs which are within its cell using a beacon message to keep track of the MHs within its cell.

[1]The geographical coverage area under an MSS.

3 The Surrogate Object Model

3.1 An Overview

The surrogate object model for distributed mobile systems defines an architecture that allows mobile devices to participate seamlessly in computing and communication. The model involves bridging the mobile device and its support environment, using a place-holder namely the surrogate object. This is done by creating the surrogate in the static network to act on behalf of each mobile device. One consequence of using the surrogate object model is that mobile devices would be transparent to the instability of wireless communication. The surrogate object can remain active, maintaining information regarding the current state and plays an active role on behalf of the device.

The major advantages of using a surrogate architecture are:

- Maintains the location information about the mobile device.
- Acts as a place-holder that can realize local caching for faster information access.
- Handles message delivery for the MH, when it is out of reach from the MSS.
- Acts as data sink that can collect data from diverse sources and delivers appropriate data to the MH depending upon the current location of the MH and its connectivity constraints.

3.2 The Complete Picture

Surrogate object is a software 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. This model helps in handling mobility of the mobile devices and helps in effective handling of the available bandwidth. In addition it 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 surrogate object is as shown in figure 1.

3.2.1 Surrogate Object Identifier (SOID) Management

In the proposed model, 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 Surrogate Object Identifier (SOID). The middleware adds the following entry in the naming service: the object name (the MHID is the object name) and the corresponding object reference.

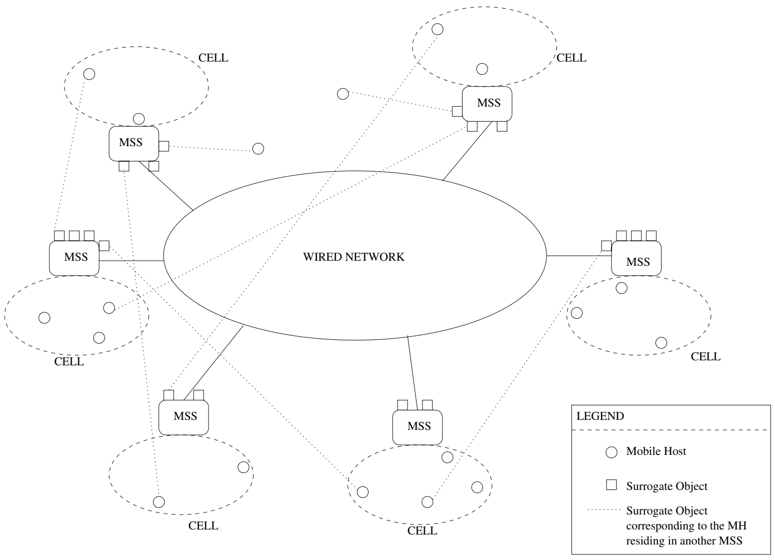

Figure 1: Structure of Distributed Mobile Systems with Surrogate Objects

When MH_s wants to communicate with MH_d, the MSS to which MH_s belongs, uses the naming services of the middleware and gets the object reference of the surrogate corresponding to MH_d. Using the object reference, the surrogate object associated with MH_d is located and MH_s starts communicating with the surrogate object of MH_d, instead of actually communicating with MH_d. This is the case if there is a cache hit at the surrogate object. If there is a cache miss, then MH_d need to be contacted.

3.2.2 Asymmetry Bridging, Mobility and Disconnected Operations

Mobile connectivity is highly variable in performance and reliability. The wireless communication channels used by the MHs also have a lower bandwidth than the wired channels. This asymmetry between the wired and wireless network is bridged using the proposed surrogate object model, as, the case is now similar to delivering the message to a static node.

Mobility implies that a MH changes its location. Thus, the location management for the targeted MH becomes an indispensable task of any application that runs on the distributed mobile system. This complicated issue could be handled effectively without affecting the delivery using our proposed model. The sender of a message targeted to an 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 surrogate object which is residing in the static portion of the network. The surrogate object takes opportune time to deliver the message to its MH considering the availability of the wireless bandwidth and the traffic on the network.

Mobile hosts often get disconnected from the rest of the system due to mobility or by switching into doze mode to save power. Disconnected operations is a regular feature in mobile computing and is distinct from failure [KS92]. Disconnected operation due to doze mode is voluntary in nature and a mobile host can be required to execute a disconnection protocol before its detachment. Thus, the algorithms have to accommodate such voluntary disconnections and make progress during the disconnection of mobile hosts. This is done by caching the current state of the MH in its place holder namely the surrogate object. However in case of disconnection due to mobility the proposed surrogate object model could help in continuing with the execution of the application using the available information cached in the surrogate object.

3.2.3 Surrogate Object Migration

The surrogate object is free to move independently of the MH anytime. Though the surrogate object resides on the wired network, the MH to which the surrogate object corresponds, is free to move. 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. 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 surrogate object as near as possible. However, migrating the surrogate 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 surrogate 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 criteria for migrating the surrogate objects are:

- Move from 'oldMSS' to 'newMSS' after the MH has made 'n' [2] 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.
- Move to the cell which is the source of maximum queries.
- Move to a less loaded MSS.

Surrogate object migration needs to be handled at the middleware level, as the surrogate object resides in the wired portion of the network. Existing middlewares such as CORBA do not handle object migration efficiently. The granularity of migration cannot be at the level of individual objects, as this affects the scalability of the implementation repository [Hen98]. We have developed an object migration mechanism based on the concept of message filters to allow individual objects to migrate freely in a distributed system [JS02].

[2]Appropriate values of 'n' are discussed in the performance studies.

A message filter is attached to each surrogate object. The filter has the same interface as the surrogate object in addition to certain special methods for plugging/unplugging the filter and to update the location of the object. The filter maintains the current location of the object and is responsible for redirecting client requests to the current location. The filter also handles the migration window, i.e., the period during which the surrogate object is migrating. Full details of the object migration mechanism is beyond the scope of this paper and is available in [JS02].

4 Sample Application: Contour Mapping

To show the usefulness of the Surrogate Object model, the following application has been developed.
Consider the following application scenario: Contour maps of a large geographical area need to be drawn. Normally, field workers use compasses and collect data. All workers later put together the data and verify it. If there are data inconsistencies, they may be required to recollect the data and repeat the whole process. Instead, the workers can be provided with mobile devices having GPS capability. The devices now become sources of data which could be collected and analyzed. We shall consider only a single MH acting as a data source and several clients need access to this data. This application is representative of a class of applications such as earth science application [VCR+02].

The application can be designed using the surrogate object model and without it. In the latter case, clients end up querying the device directly, but in the former case, clients need to query only the surrogate object.

The following algorithm describes how this application would be developed without the surrogate object model.

4.1 Algorithm without Surrogate Object Model

1. (a) The client host (say MH_s) sends a query message to its local MSS (say MSS_s) stating the destination host (say MH_d) and the query index.
   (b) MH_s enters the details of the query in its 'Sent Query Buffer' including the send time of the message and its status (NOT_ACKD).
2. When MSS_s receives the query message:
   (a) it sends a message to the location server requesting it the current location of MH_d.
   (b) enters the details of the message in the local 'Query Detail Buffer' with status as 'NOT_SENT'
3. Location server looks up the entry for MH_d and returns the current known location information to MSS_s in the form of a message. (Assume, MH_d currently resides

in MSS_d). An error in lookup results in an error message being returned.

4. Upon receipt of the location reply message from the location server, MSS_s removes all query details from the 'Query Detail Buffer' which are destined to MH_d and whose status is 'NOT_SENT'. For each such query, either of the following 2 is done:

   Case 1 The reply is an error message. The error message is forwarded to MH_s and the query is deleted.

   Case 2 The reply contains the location information. A query message is constructed and sent to MSS_d and the status of the query is marked as 'SENT'.

5. MSS_d, upon receiving the query message checks if MH_d is a registered mobile host.

   - If yes,
     (a) The query details are entered in the 'Received Query Detail Buffer'.
     (b) The query is forwarded to MH_d
     (c) A copy of the sent message is stored in the 'Sent Message Buffer'.
   - Else,
     An error message is returned to MSS_s

6. When MH_d receives a query message from MSS_d, it looks up its file and returns the data corresponding to the query index in the form of a reply message to MSS_d. (If the query index is invalid, the reply value is a special one to denote this).

7. MSS_d, upon receiving the reply message:

   (a) Retrieves (and removes) the details of the corresponding query from the 'Received Query Detail Buffer'.

   (b) Sends a reply message to MSS_s.

   (c) Removes the copy of the corresponding sent message stored in the 'Sent Message Buffer'.

8. MSS_s, upon receiving the reply message:

   (a) Retrieves (and removes) the details of the corresponding query from the 'Query Detail Buffer'.

   (b) Forwards the reply message to MH_s.

   (c) A copy of the sent message is stored in the 'Sent Message Buffer'.

9. When MH_s receives the reply message, it checks its 'Query Sent Buffer' for a corresponding entry.

   - If such an entry is found,
     it sends an acknowledgment of the receipt to MSS_s.

- Else,
  it discards the reply message. (It is a duplicate reply received due to host movement).

10. When MSS_s receives the acknowledgment from MH_s, it removes the copy of the corresponding sent message stored in the 'Sent Message Buffer'.

If a query arrives for MSS_s when a location query for MH_d is outstanding, steps 2 and 3 are avoided. Also, if the destination MSS (MSS_d) happens to be the same as the source MSS (MSS_s), steps 2, 3, 4 and 7 are not required. This represents the best case over this model requiring just 4 packet transmissions over the wireless portion of the network.

As described in the algorithm above, 8 packet transmissions over the network are required for the retrieval of a query result, 4 of those transmissions over the wired portion of the network and another 4 are over the wireless portion. This represents the average case for the number of packet transfers for this application.

The worst case has theoretically no limits over the number of packet transfers. Even in the case of no packet loss, the movement of the mobile hosts (either MH_s or MH_d) results in the packets delivered to the wrong MSSs. Sometimes, the packets can be forwarded to the MSS of the new cell while in other cases there is no option but to drop the packets. For this reason, a copy of the sent packets needs to be kept until being acknowledged. This is especially true for the wireless medium which is highly error prone as compared to the wired network. Thus, a large amount of overhead is involved for any application developed over the cellular network. 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 response time higher in addition to clogging the network bandwidth with copies of lost packets.

4.2 Algorithm with Surrogate Object Model

1. This is same as step 1 of algorithm described in section 4.1.
2. When MSS_s receives the query message, it
   (a) checks if the SO for MH_d is currently hosted.
   If YES,
   - The query details are entered in the 'Received Query Detail Buffer'.
   - The query is forwarded to the SO
   - enters the details of the message in the local 'Query Detail Buffer' with status as 'SENT'

   Else,
   - checks the 'Query Detail Buffer' for the existence of a query which has MH_d as its destination and status as 'NOT_SENT'

    - If not found, it sends a message to the location server requesting it the object reference of the Surrogate Object of MH_d.
    - enters the details of the message in the local 'Query Detail Buffer' with status as 'NOT_SENT'

3. The Location Server looks up the entry for the Surrogate Object of MH_d and returns the object reference of the surrogate object to MSS_s in the form of a message. (Assume, the SO currently resides in MSS_o). An error in lookup results in an error message being returned.

4. This is same as step 4 of algorithm described in section 4.1.

5. MSS_o, upon receiving the query message checks if the SO for MH_d is currently hosted.

   If YES,

   - The query details are entered in the 'Received Query Detail Buffer'.
   - The query is forwarded to the SO.

   Else,

   - Pushes the message in a buffer for later delivery to the SO (as part of the deregistration process).

6. When the SO for MH_d receives the query message, it checks its file cache for the data.
   If FOUND,

   - It returns the result in the form of a message.

   Else,

   - It sends a query message to the MSS where MH_d currently resides (say MSS_d).

7. MSS_d upon receiving the query message from the SO of MH_d

   (a) The query details are entered in the 'Received Query Detail Buffer'.
   (b) The query is forwarded to MH_d
   (c) A copy of the sent message is stored in the 'Sent Message Buffer'.

8. When MH_d receives a query message from MSS_d, it looks up its file and returns the data corresponding to the query index in the form of a reply message to MSS_d.

9. MSS_s, upon receiving the reply message from MH_d, checks if the SO for MH_d is currently hosted.
   If YES,

   - the reply message is passed on to the SO.

Else,

- The message packet is ignored.

10. The SO updates its cache using the reply message contents and sends a reply to MSS_s

11. Steps 11, 12 and 13 are same as steps 8, 9 and 10 respectively of algorithm described in section 4.1.

The different possible scenarios that could occur are:

Case 1 MH_s is located in the cell of MSS_s and the SO of MH_d is hosted on the same MSS. Also, the SO finds the requested data in its cache. It require just 2 wireless packet transmissions. This represents the best case for the new model.

Case 2 The SO of MH_d is hosted on MSS_o (o != s) and the SO finds the requested data in its cache. Here it require 2 wireless packet transmissions and 4 wired packet transmissions. This represents the average case for cache hit.

Case 3 The SO of MH_d is hosted on MSS_o (o != s) and the SO does not find the requested data in its cache. MH_d is located in the cell of MSS_d. Here we require 4 wireless packet transmissionsand 6 wired packet transmissions. This represents the average case for cache miss.

5 Performance Studies

To study the performance of the model, the query execution scenario explained in previous section was considered, and was implemented over a simulated model of a cellular network. The scenario was studied over both the surrogate and without surrogate object models. Throughout this section, the algorithm without the surrogate object model is referred to as the old model and the one with the surrogate object model is named as the new model. Typical performance studies include actual number of packets lost for different packet loss probabilities in both old and new models; proportionate increase in the query time for different packet loss probabilities in both the models; impact of movement of the surrogate object in terms of packet loss and message traffic.

5.1 Comparison of Old and New Models

Figure 2 shows the comparison of the query latencies in the caching application over both the old and new models. In each graph, the query time variance is plotted against simulation time for different packet loss probabilities. The study shows that as the packet loss probability increases, the query time increases. However, in the new model, this increase is not significant compared to the same in the old model. Further, it can be observed that the maximum query time with zero packet loss probability is much higher in the old model, compared to the maximum query time in the new model, even with 5% packet loss probability. Figure 3 shows the percentage increase in query time as packet loss probability in the wireless network increases for both the old and the new models.

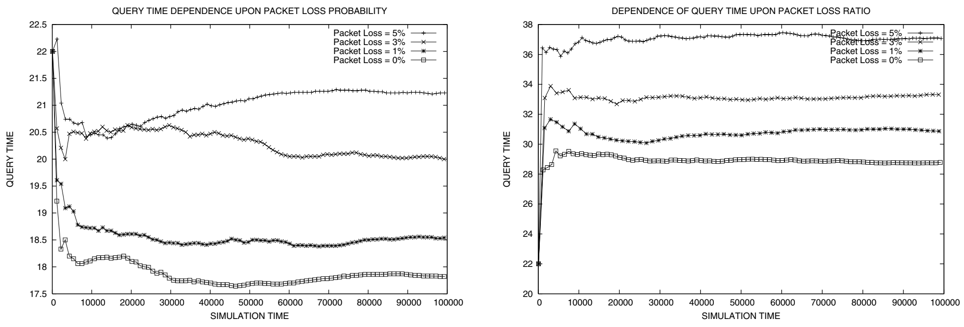

Figure 2: Comparison of Query Latencies in the New and Old Models

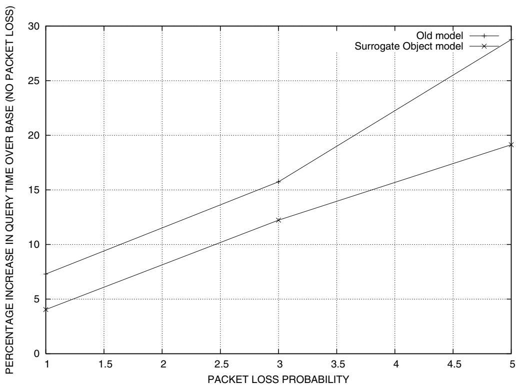

Figure 3: Percentage Increase in Query Time with Increasing Packet Loss Probability

5.2 Effect of Surrogate Object Migration

One of the key advantages of the surrogate object model is that it is free to be migrated to any node in the wired network. This flexibility is desirable for various reasons, including failure recovery, load balancing and network latency reduction. Figure 4 shows the effect of surrogate object migration on the query latency for various migration frequencies. It can be seen that if the surrogate object is migrated every time the MH moves, the query latencies are considerably high. This is due to the increase in the percentage of time spent in migration itself. But as the move frequency is reduced (the surrogate object moves only once for every 'n' moves made by the MH to various cells), the query time improves. Surprisingly, further changes in the move frequency seems to have almost no impact on the query time. This means that for any 'n' other than 1, the query time is the same as if the surrogate object is static, never migrated. The reason for this behaviour is that query response time is more dependent on the nearness of the surrogate object to the originating MSS and the queries are generated randomly from different parts of the network. A closer study of the network traffic generated as a result of surrogate object migration reveals the reason for providing migration freedom for the surrogate object.

Figure 5 shows the network traffic in terms of number of messages exchanged versus simulation time for various move frequencies. It can be seen that if the surrogate moves with every movement of the MH, the traffic is an order of magnitude higher. As the move frequency reduces, the traffic generated is much lesser. But the frequency value (other than 1) does not affect the network traffic. This implies that it is not a very good strategy to move the surrogate object every time the MH moves. This would be the case if, instead of using the surrogate object model, a similar effect is to be achieved using data structures at the MSS. These data structures would have to be moved with every handoff.

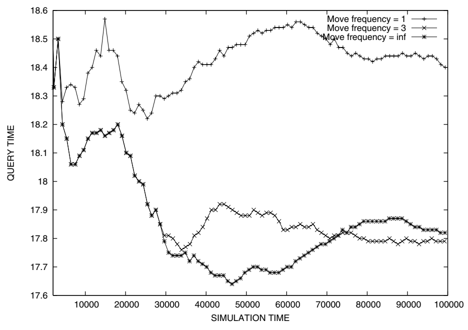

Figure 4: SO migration: Query Time

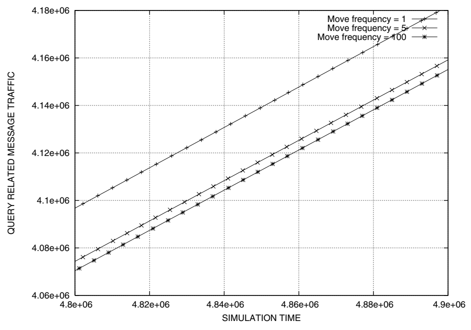

Figure 5: SO migration: Packet Loss

6 Conclusions

The surrogate object model presented in this paper provides a new perspective for designing distributed mobile systems. It focuses on the fundamental issue of asymmetry. Addressing asymmetry automatically takes care of other issues such as device constraints or mobility. The surrogate object 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.

Detailed performance studies of a sample application with and without using the surrogate object model reveals the real benefit of using the surrogate object model, both in terms of response time and network traffic. Surprisingly, it was found that the frequency of surrogate object migration had little effect on response time. Any frequency other than 1 (migrate for every handoff) seems to be good enough.

An interesting direction of future research is to explore the theoretical foundations of the surrogate 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 surrogate object model.

References

[ABS01] Giuseppe Anastasi, Alberto Bartoli, and Francesco Spadoni. A Reliable Multicast Protocol for Distributed Mobile Systems: Design and Evaluation. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1009–1022, October 2001.

[ARV95] 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.

[BAI94] B.R. Badrinath, Arup Acharya, and Tomaiz Imieslinski. Structuring Distributed Algorithms for Mobile Hosts. In *Proceedings of the 14th International Confernce on Distributed Computing Systems*, june 1994.

[BS98] K. Brown and S. Singh. RelM: Reliable Multicast in Mobile Networks. *Journal of Computer Communications*, 21(16):1379–1400, April 1998.

[BSZP03] 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*. Tokyo, Japan, june 2003.

[FZ94] G. H. Forman and J. Zahorjan. The Challenges of Mobile Computing. *IEEE Computer*, 27(4), April 1994.

[Hen98] Michi Henning. Binding, Migration and Scalability in CORBA. *Communications of the ACM*, 41(10):62–71, October 1998.

[JDM91] 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.

[JHE99] Jin Jing, Abdelsalam (Sumi) Helal, and Ahmed Elmagarmid. Client-Server Computing in Mobile Environments. *ACM Computing Surveys*, 31(2):117–157, June 1999.

[JS02] D Janakiram and A Vijay Srinivas. Object Migration in CORBA. *Journal of the CSI*, 32(1):18–27, March 2002.

[KS92] J J Kistler and M Satyanarayanan. Disconnected Operation in the Coda File System . *ACM Transactions on Computer Systems*, 10(1):3–25, February 1992.

[MRV02] 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.

[PR97] Anjeneyuly Pasala and D. Janaki Ram. Mobility Extender: A Design Pattern for Seamless Integration of Mobility into Distributed Systems. In *Proceedings of National Conference on Object Oriented Technology (NCOOT)*, pages 105–114. Warangal, India, August 1997.

[PS01] Evaggelia Pitoura and George Samaras. Locating Objects in Mobile Computing. *IEEE Transactions on Knowledge and Data Engineering*, 13(4):571–592, July/August 2001.

[Sat96] M. Satyanarayanan. Fundamental Challenges in Mobile Computing. In *Proceedings of the 15th ACM Symposium on Principles of Distributed Computing*, pages 1–7, May 1996.

[VCR$^+$02] E R Vivoni, R Camilli, M A Rodriguez, D D Sheehan, and D Entekhabi. Development of Mobile Computing Applications for Hydraulics and Water Quality Field Measurement. *Hydraulics Information Management*, 10, 2002.

[WR96] C. Donald Wilcox and Gruia-Catalin Roman. Reasoning About Places, Times, and Actions in the Presence of Mobility. *IEEE Transactions on Software Engineering*, 22(4):225–247, April 1996.

[ZD95] B. Zenel and D. Duchamp. Intelligent communication filtering for limited bandwidth environments. In *Proceedings of IEEE Fifth Workshop on Hot Topics in Operating Systems*, pages 28–34, May 1995.