Design of MOBILE MOM: Message oriented middleware service for mobile computing

Conference Paper · February 1999

DOI: 10.1109/ICPPW.1999.800097 · Source: IEEE Xplore

3 authors, including:

Kwang Jin Paek
ROKAF
8 PUBLICATIONS **64** CITATIONS

SEE PROFILE

Design of MOBILE MOM: Message Oriented Middleware Service for Mobile Computing

Do-Guen Jung. Kwang-Jin Paek, and Tai-Yun Kim
Dept. of Computer Science & Engineering, Korea Universit
pkj@netlab.korea.ac.k

Abstract

Message oriented middleware (MOM) is a specific class of middleware that operates on the principles of message passing or message queuing. Existing MOM syste doesn't support the function for mobile computing environment. In the near future, requirements of mobile computing will increase and more dynamic service for mobile computing will be required. And i nteracting with wireless networks is becoming almost commonplace.

In this paper, we propose a MOBILE MOM system supporting mobile computing environment. It bases on existing MOM system. In MOBILE MOM system, MOBILE MOM applications executed on mobile hosts (MH) dynamically and asynchronously are connect with MOBILE MOM me ssage queue managers executed on fixed hosts (FH) through MOBILE MOM message agents executed on base stations (BS). MOBILE MOM can provide a level of fault-tolerance using persistent queues that allow messages to be reco vered when the system fails and very reliable, scalable and performance-oriented distributed application networks in heterogeneous environments.

1. Introduction

MOM model messages as events in an event delivery system and all MOMs share two fundamental characteri stics: they enable message-passing and the message-passing is non-blocking [5]. Existing MOM suppose that it is e xecuted on wired networks. Hence, it doesn't perfor optimizations and functions for wireless network environment. MOM must support functions for mobility so that MO applications may be executed on mobile computing env ironment [2].

There are three points of difference between wired di stributed computing environment and mobile distributed computing environment using wireless communication. 1) Mobility of machines 2) expensive costs and restrictions of uses 3) restrictions of energy uses [7]. Many researches are in progress so as to complement these three points of di fference and make it possible to perform mobile distributed computing. Major research topics are follows:

- Extension of TCP/IP on LAN environment for MH.
- Extension of IP on WAN environment for MH and protocols for mobility management.
- Establishment of directory location for efficient processing location updates according to mobility of MH.
- Design and analysis of reliable multicasting protocol.
- Design and analysis of various algorithms for distributed systems supporting MH. And etc.

The research scope of this paper is to propose MOBILE MOM system based on IP, implement this system, and evaluate its performance. MOBILE MOM is MOM system for mobile computing. MOBILE MOM applications are connected to MOBILE MOM message queue managers supporting asynchronous services through MOBILE MOM message agents executed on BS.

In section 2, we research distributed communication middleware (RPC, MOM, M-RPC), section 3, present M OBILE MOM system, section 4, implement MOBILE MOM system with Java and evaluate its performance, final section 5, reach conclusion and present future work.

2. Distributed Communication & Middleware

The purpose of a distributed communications framework is to provide a good way for the parts of a distributed s stem to communication. Object-oriented frameworks a ccomplish this task by providing distributed objects with a way to message each other. The distributed object-oriented frameworks that get the most attention are those that model messaging as method calls. CORBA and RMI are two excellent examples of this type of framework. These systems are often called remote procedure call systems. The magic of these systems is that they make remote procedure (or method) calls appear to be local procedure calls (LPCs) [5].

The basic requirements of distributed applications are to support transactions and security of data. Many application development systems for perform these operations ar commonly called middleware. Middleware, as popular term, is distributed computing service or application development environment (ADE). As the name implies, middleware products operate between the application logic and the underlying physical networ [4]. Figure 1 illustrates middleware segmentation.

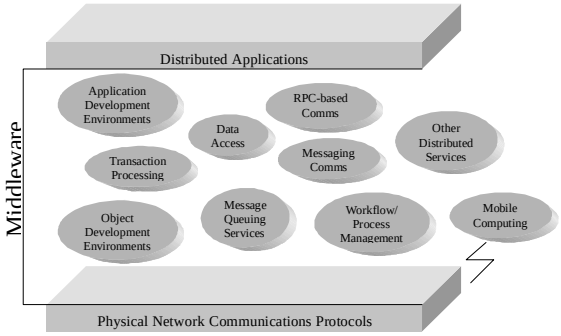

Figure 1. Middleware Segmentation [4]

2.1 Remote Procedure Call (RPC)

An essential problems that RPCs are not procedure calls at all, they are truly process invocations. The invoked p ogram runs across the wire in a different resource domain. RPCs hide the intricacies of the network by using the ordi-

nary procedure call mechanism familiar to every programmer. A client process calls a function on a remote serve and suspends itself until it gets back th results. Parameters are passed like in any ordinary procedure. The RPC, like an ordinary procedure, is synchronous. The process (or thread) that issues the call waits until it gets the results. While RPCs make life easier for the programmer, they pose a challenge for the NOS designers who supply the develo pment tools and run-time environments [6]. Figure 2 illustrates RPC (CORBA) system. In this figure, client objects call methods of server objects.

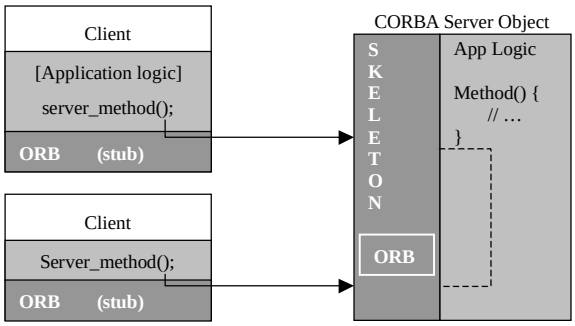

Figure 2. Architecture of RPC system [5]

The better Network Operating Systems (NOSs) provide an Interface Definition Language (IDL) for describing th functions and parameters that a server exports to its clients. An IDL compiler takes these descriptions and produces source code stubs (and header files) for both the client and server. These stubs can then be linked with the client and server.

RPCs are complex to install and use. And c omplicated setup required in RPC. RPC is designed for supporting synchronized communication and servers must first come up before clients can talk to them.

2.2 Message Oriented Middleware (MOM)

"Every DAD needs a MOM " is the unofficial motto of the MOM Consortium. In this context, DAD stands for Distributed Application Development and MOM stands fo Message-Oriented Middleware. MOM is a key piece of middleware that is absolutely essential for a class of cl ient/server products. If the application can tolerate a certain level of time-independent responses, MOM provides the easiest path for creating enterprise and inter-enterprise cli ent/server systems. MOM also helps create nomadic client/server systems that can accumulate outgoing transactions in queues and do a bulk upload when a connection can be established with an office serve [6]. Figure 3 illustrates MOM system. MOM passes messages between applications, which can be peers or client/server.

All MOMs share two fundamental characteristics: the enable message passing and the message-passing is nonblocking [5]. MOM's messaging and queuing allow clients and servers to communicate across a network without being linked by a private, dedicated, logical connection. The cl ients and servers can run at different times. Everybody communicates by putting messages on queues and by taking messages from queues.

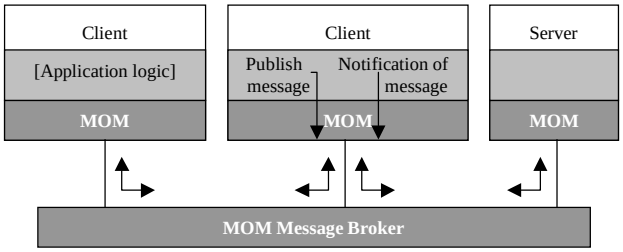

Figure 3. Architecture of MOM system [5]

Messaging queues are very versatile. We can use them to create one-to-many or many-to-one relationships. Many clients are sending requests to one server queue. The messages are picked off the queue by multiple instances of the server program that are concurrently servicing the clients. The server instances can take messages off the queue either on a first-in/first-out basis or according to some priority or load-balancing scheme. In all cases, a message queue can be concurrently accessed. In all cases, a message queue can be concurrently accessed. The servers can also use me ssaging filters to throw away the messages they don't want to process, or they can pass them on to other servers.

Most MOM messaging products make available a simple API set that runs on multiple operating system pla tforms. Most also provide persistent (logged on disk) and non-persistent (in memory) message queues. Persistent messages are slower, but they can be recovered in case of power failures after a system restart. In both cases, messages can be either copied or removed from a queue. A message queue can be local to the machine or remote. System administrators can usually specify the number of messages a queue can hold and the maximum message size [6].

MOM supports messages and therefore is primarily designed to support deferred communication while peer-to-peer and RPC are designed to support synchronous co mmunication. Under RPC, the receiving server must be avai lable to accept messages sent. If the server is down, the message cannot be delivered at that time. MOM, on the other hand, can send messages to servers that are dow without having to resend them. Messages under a MOM system are placed into a queue and retrieved whenever th server requests them. Whether server is available at the time the message is sent is irrelevant. MOM lends itself t event-driven rather than procedural processing [4].

2.3 MOM versus RPC

Comparing RPC and MOM is like doing business via a telephone call versus exchanging letters or faxes. Messa ging is, of course, more flexible, loosely coupled, and time-tolerant than RPC. In the telephone analogy (RPC), we complete the work as it arrives; we don't have to manage stacks of incoming letters (or faxes). Clients are happy to get immediate service. In the mail analogy, letters may start to pile up, and clients may be polling their incoming mailboxes continuously, waiting for a response. We may have made life easier for the server at the expense of the client. On the other hand, messaging does free clients from being synchronized to their servers; this can be very liberating fo

mobile and home users [6].

In contrast to RPCs, MOMs don't model messages as method calls; instead, they model them as events in an event delivery system. Clients send and receive events, or messages, via APIs that the MOM provides. The MO may present directory services that let clients look up an other application, which is acting as a server, or it ma present all-purpose channels that let a group of clients comm unicate as peers, or it may present both options. All applications communicate directly with each other using the MOM. Messages generated by applications are meaningful only to other clients because the MOM itself is only a message router [5]. Table 1 presents Comparing MOM and RPC.

Table 1. Comparing MOM and RPC [6]

Feature	MOM: Messaging and queuing	Remote Procedure Call (RPC)
Metaphor	Post office-like.	Telephone-like.
Client/Server time relationship	Asynchronous.	Synchronous.
Style	Queued.	Call-Return.
Load-balancing	Single queue can be used to implement FIFO or priority-based policy	Requires a separate TP Monitor.
Transactional support	Yes.	No.
Asynchronous processing	Yes. Queues and tr g-gers are required.	Limited. Requires threads and tricky code for managing threads.

2.4 M-RPC [1]

RPC is used as a basis for structuring many client server applications. Conventional RPC implementations however, assume that all the hosts in a network are stationary and are always reachable except in case of failures. The applic ations built on top of conventional RPC also tend to assum that the network is more or less a freely available resource and that network bandwidth is not a constraint. Such applications thus do not attempt to optimize the number of me ssages exchanged between a client and a server. For example, control messages between a client and a server in NFS [12] to refresh file system handle form a significant portion of the overall file system traffic.

The presence of mobile hosts within a network inval idates all the above assumptions. Mobile hosts can move from one cell to another and can remain disconnected from the fixed network for extended periods to conserve scarce battery power. The wireless link connecting a mobile host to the rest of the network usually has lower bandwidth and higher error rate as compared to a wired link. Available bandwidth on wireless is thus a scarce resource and in f uture mobile hosts are expected to incur cost proportional to the connection time on the wireless link. This means that the RPC based applications for mobile clients have to be structured such that the amount of data sent on the wireless is minimized [1].

M-RPC is an RPC service for mobile clients and satisfies the requirements of the mobile applications. An M-RPC client can access connectionless RPC based services on the wired network via an agent located at its current BS which provides the following fun ctionality:

- Support for dynamic server binding
- Reliable transport over the wireless link
- Call retries from the MSR and
- Partial support for disconnected operation.

SunRPC allows the use of both connectionless (UDP) [9] and connection oriented (TCP) [8] transport layers for network communication. Using a connectionless transport protocol affords design of simple stateless servers, which can be replicated for high availability. In practice, NFS implementations using UDP work quite well over a LAN, even though UDP does not provide reliable delivery of datagrams. The reliability needed for NFS is provided b RPC and NFS level retry mechanisms. Using TCP for RPC style request-response kind of communication can be a k-ward because – 1) it is a stream protocol which does not preserve record boundaries and 2) it brings in a lot of add i-tional baggage in terms of congestion and flow control which is quite unnecessary. Connection oriented transport also requires the servers to maintain state for every open connection with a client.

With mobile clients that wish to access services on the wired networks, the choice of a transport protocol for RPC based applications is rather difficult. UDP is clearly inadequate for error-prone wireless links. TCP, although it provides reliable delivery, suffers from the problems m n-tioned earlier. In addition, in a mobile environment TCP performance degrades because of wireless errors and moves. For RPC based applications in mobile environment, we therefore need a transport protocol that provides reliabilit on the wireless link, but at the same time has the simplicit of UDP for the relatively error free wired link [13], [14], [15].

The kind of mobile client applications those which e n-gage in a relatively long lived interaction with the servers making a series of RPC requests in the process lead themselves to M-RPC system. M-RPC system, although useful, does not affect in a significant way those applications, which simply make one RPC request to a server and exit.

Figure 4 shows the overall structure for M-RPC. The M-RPC system has been designed for mobile client which access services from the servers on the wired network.

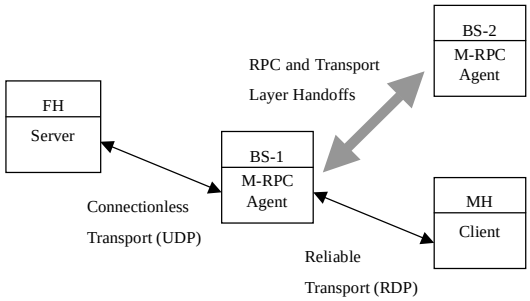

Figure 4. Overview of M-RPC System [1]

In a connectionless client-server application, the retransmissions of failed RPC requests are typically performed b the client side, which uses a retry timeout value for this purpose. With a mobile client using low bandwidth and costly wireless link, call retry from the client is clearl undesirable especiall if the request is correctly delivered to

the wired network but the reply was lost for some reason.

M-RPC is based upon the indirect client-server model [2] for mobile hosts. There are two main reasons for choosing the indirect model for structuring the M-RPC system:

- The mobile hosts and its BS are in the best position to know about the properties of the wireless link and since the mobile has relatively less battery and computing power, the BS is the best place to implement any dynamic scheme of filtering the data on the wireless.
- Since most of the internetwork is unaware of mobility we cannot assume that the servers on the fixed network will adapt to any specialized strategies developed for mobile clients. Thus the BS can provide compatibility with the wired network.

3. MOBILE MOM

In this paper, we present MOBILE MOM system that provides fault-tolerance in the form of persistent queues and can reroute messages to alternate queues in case of a network failure. Entire MOBILE MOM system consists of MOBILE MOM message agents, MOBILE MOM message queue managers and applications. Figure 5 illustrates the overview of MOBILE MOM system.

MOBILE MOM message agent running on BS send messages to proper queue managers and can provide directory service. MOBILE MOM message queue managers run on FH and manage messages received from MOBILE MOM message agents as persistent objects in queues.

A message is an object. To make messages persistent, we give a object unique OID (Object ID) and store objects in persistence supporting media such as file systems and database systems.

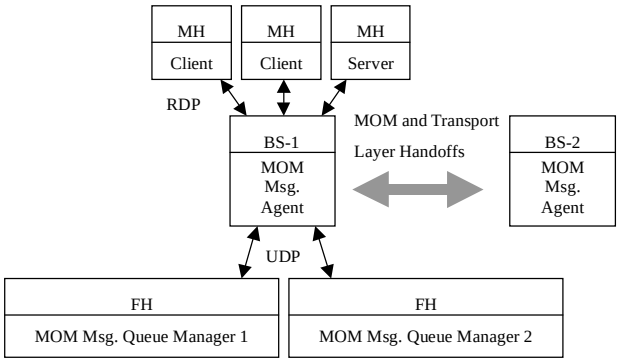

Figure 5. Overview of MOBILE MOM System

Even if MOBILE MOM system must be restarted fo message queue managers fail, MOBILE MOM can recove its integrity and guarantee that a message is delivered to its destination. According to kinds of services, message queue managers handle multi-queue. Each queue provides functions that it can be separately scheduled in accordance with a kind of service.

Figure 6 shows the process of directory-based handoff from message agent 1 to message agent 2. A new service in directory D_2 results in a request sent to main directory$_1$ that indicates the presence of a previous service information in D_1. The main directory$_1$ retransmits the request to directory D_1. This directory is cleaned and returns with its service information to the main directory$_1$ and D_2 [3].

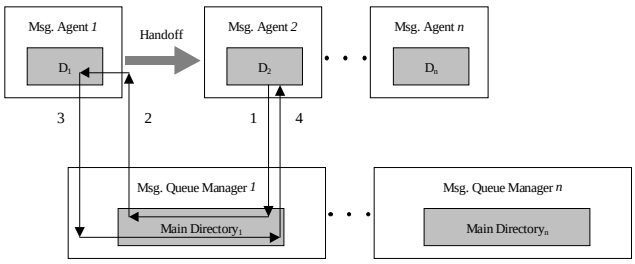

Figure 6. Directory-based handoff scheme of MOBIL MOM

In MH, the application side library for MOBILE MOM is provided as Java package type. An application has a two way message temporary queue, which supports persistence. This queue temporarily holds message objects before eceiving or sending message objects. Applications are indiferent to the management of temporal queues and MOBILE MOM application side library supervises temporal queues. The temporary queue tolerates errors happened in the prcess of message transmission. Applications can avoid th blocking state caused by the fault of wireless networks. Applications convey messages to temporary queues and continuously process next jobs. Temporary queues guarantee the secure transmission of message to MOBILE MOM message queue manager. MOBILE MOM system gives a OID to a message object. OID tolerates faults of applications and wireless networks.

Figure 7 shows the data structure of OID, which is a collection of codes. MOBILE MOM message queue managers use OID in order to discriminate among objects. It includes IP addresses and port numbers of an application and an MOBILE MOM message queue manager. OID is guarnteed uniqueness in the Internet. The collection of MA (MOBILE MOM queue manager IP address) and M (MOBILE MOM Port number) is the service code for clients. MSN (Message Sequence Number is generated b applications. It increase in due sequence.

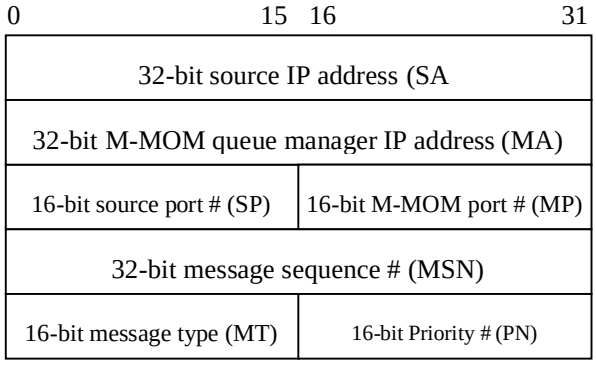

Figure 7. Overview of OID

In MOBILE MOM system, a service has a pair of queue that supports two-way message queuing. Message objects are stored in request queues or result queues according to their types. Figure 8 illustrates multi-queue of MOBILE MOM message queue manager.

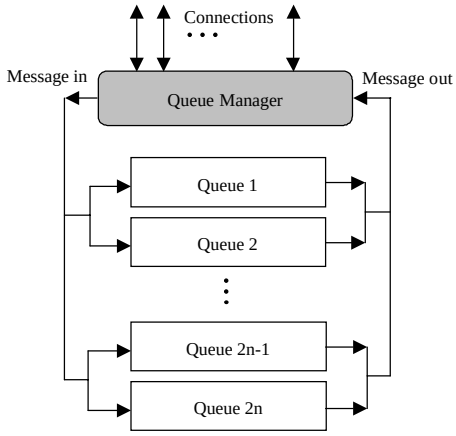

Figure 8. Queues of MOBILE MOM Message Queue Manager

4. Simulation Results

MOBILE MOM system was made of only Java, JDK1.2beta4. Applications made of pure Java can run an platform. Message objects of MOBILE MOM are repr e-sented as Java objects. As to transmit Java objects, we use Java object serialization API offered as core Java package [10]. MOBILE MOM system consists of MOBILE MOM message queue managers as the kernel of MOBILE MOM, MOBILE MOM message agents, and MOBILE MOM API packages for applications.

The packet loss rate of wireless network is higher than that of wired network. We simulated MOBILE MOM a s-suming that mobile cell is the connection that established far across the distance between two hosts: injo and banana. The distance between Pusan and Seoul is 455Km. Table 2 lists the information of hosts, injo and banana. On banana, applications run and on injo, MOBILE MOM queue man-agers and agents run.

Table 2. The specifications of hosts

ITEM	APPLICATION	MOBILE MOM QUEUE MANAGER & AGENT
Machine type	Pentium PC	sun4 sparc
OS	MS-Windows95	Solaris 2.5
Host name	banana	injo
Domain name	korea.ac.kr	pufs.ac.kr
Location	Seoul	Pusan
JDK version	JDK 1.2beta4	JDK 1.1.4

Java Remote method invocation (RMI) is a distributed object model for the Java language that retains the sema n-tics of the Java object model, making distributed objects easy to implement and to use. It is a kind of RPC [11].

In this paper, we simulated the performance of M-RPC and MOBILE MOM by means of evaluating MOM and RMI on error prone and higher packet loss rate WAN envi-ronment. Each experiment had been performed 20 times per day for 12 days. Figure 9 presents the graph of experimen-tal result. Sequentiall packet loss rates are 25%, 22%, 25%, 18%, 11%, 30%, 22%, 21%, 29%, 18%, 25%, and 18%. In the graph, the response of MOM is faster than that of RMI by 6 times on the average. We can induce it from the ex-perimental result that connectless oriented MOM is more suitable to mobile computing, lower bandwidth, and highe packet loss rate environment than connection oriented RPC.

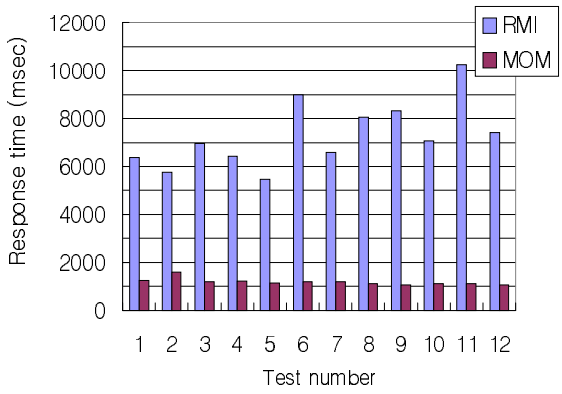

Figure 9. Test Result of Response Time

In above simulations, we excluded the case of packet loss errors in MOBILE MOM. But we can proof that MOBILE MOM response time is practically shorter than M-RPC with mathematical expression. I nequalities (1) show that the practical esponse time of MOBILE MOM.

$$\left.\begin{array}{l} RMI\ \ response\ time = R \\ MOM\ \ response\ time = M \\ Packet\ \ loss\ \ rate = L\ \ (0 < L < 1) \\ MOM\ \ retransmission\ \ time = ML \\ Practical\ \ average\ \ M = M(1+L) \end{array}\right\} (1)$$

$$2M > (1+L)M$$

$$R > 6M > 3 \times Practical\ \ average\ \ M$$

$$\therefore R > 3M(L+1)$$

Figure 10 shows MOBILE MOM system is Java applica-tion version using Java AWT (Abstract Window Toolkit). Due to MOBILE MOM system is written in pure Java, we can make it Java applet version and run it any platform without additional porting jobs.

In consequence of experimentation, we can reason that MOBILE MOM system is more efficient than M-RPC in mobile computing environment.

5. Conclusions

In the near future, mobile computing environment and wireless network environment will be spread and a essential element of computing environment. Therefore not only H/W supporting mobile computing but also ADE based on such H/W needs to be developed. The concept of MOM lends itself to mobile computing environment. This paper has presented the model of MOM that is compatible to mobile computing environment.

We have presented in this paper a Message Oriented Middleware Service fo mobile computing called MOBILE MOM which takes care of some of the unique features of

mobile wireless computing. MOBILE MOM system was made of pure Java. We evaluated its performance. At a viewpoint of mobile computing, MOBILE MOM system provides more convenience and flexibility than M-RPC. Due to physical wireless of mobile computing environment, logical connectless oriented and a synchronous MOBILE MOM mechanism is more suitable to such environment than connection oriented M-RPC. According to results of tests, MOBILE MOM has a level of fault-tolerance su p-porting features of wireless communication.

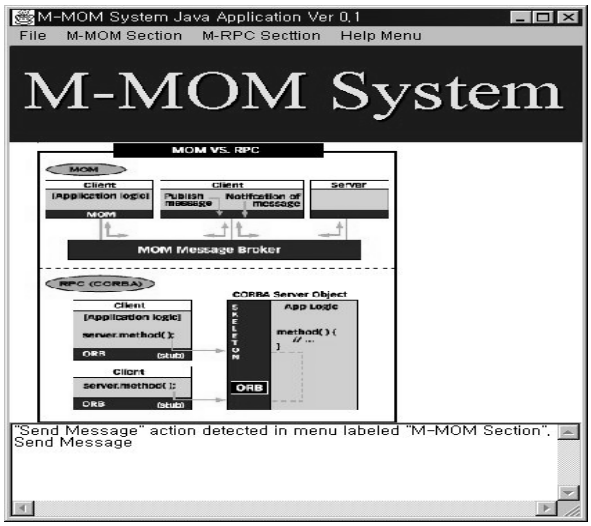

Figure 10. MOBILE MOM System Made of Java

References

[1] Ajay Bakre and B. R. Badrinath, "M-RPC: A Remote Procedure Call Service for Mobile Clients," WINLAB TR-98, Dept. of Computer Science, Rutgers University, 1995.

[2] B. R. Badrinath, A. Bakre, T. Imielinski, and R. Marantz, "Handling Mobile Clients: A Case for Indirect Interaction," Proceedings IEEE Fourth Workshop on Workstation Operating System, Dept. of Computer Science Rutgers University, 1993.

[3] Kai Hwang, Zhiwei Xu, *Scalable Parallel Computing*, The McGraw-Hill Companies, Inc., 1998.

[4] Message Oriented Middleware Association (MOMA), A Middleware Taxonomy, 1998. (URL: http://www.moma-inc.org/)

[5] Michael Shoffner, "Write your own MOM!," JavaWorld, 1998.

[6] Robert Orfali, Dan Harkey, and Jeri Edwards, *The Essential Client/Server Survival Guide: second edition*, John Willey & Sons, Inc., 1996.

[7] T. Imielinski and B. R. Badrinath. "Data Management for Mobile Computing," SIGMOD RECORD, 22(1), 1993.

[8] J. Postel, RFC793: TRANSMISSION CONTROL PROTOCOL, September 1981.

[9] J. Postel, RFC768: User Datagram Protocol, August 1980.

[10] Sun Microsystems, Java™ Object Serialization Specification, 1998.

[11] Sun Microsystems, Java™ Remote Method Invocation Specification, October 1997.

[12] Sun Microsystems, RFC1094: NFS: Network File System Protocol Specification, 1988.

[13] R.Caceres and L. Iftode, "The Effects of Mobility on Reliable Transport Protocols," Proc. of the 14th Intl. Conf. On Distributed Computing Systems, pp. 12-20, June 1994.

[14] A. DeSimone, M.C. Chuah and O.C. Yue, "Throughput performance of transport-layer protocols over wireless LANs," Proc. of Globecom'93, December 1993.

[15] P. Manzoni, D. Ghosal and G. Serazzi, "Impact of mobility on TCP/IP: an integrated performance study," the IEEE Journal on Selected Areas in Communications, 1995.