

An Agent Module for a System on Mobile Devices

Praveen Madiraju, Sushil K. Prasad,
Rajshekhhar Sunderraman, and Erdogan Dogdu

Department of Computer Science,
Georgia State University,
Atlanta, GA 30302
{cscpmx, sprasad, raj, edogdu}@cs.gsu.edu

Abstract. A Middleware is the software that assists an application to interact or communicate with other applications, networks, hardware, and/or operating systems. We have earlier proposed an RMI-based middleware for mobile devices called System on Mobile Devices (SyD). A middleware on mobile devices is a challenging issue, as it has to deal with problems such as limited memory, frequent disconnections, low bandwidth connection, and limited battery life. The mobile agent module fits in the context of the middleware for mobile devices as it quite naturally alleviates the above mentioned problems. Communication between devices and method invocation capabilities, among other things are carried out by employing agents. In this paper, we provide the design and implementation of an agent module for SyD. We also present practical experiences gathered from carrying out experiments on the agent module.

Keywords: Agent Middleware for Mobile Devices, Agent Based Execution Engine, Mobile Agents, System on Mobile Devices Middleware.

1 Introduction

It has been widely acknowledged that a middleware is essential for application development on mobile devices. However, application on mobile devices introduces multiple challenges in a mobile setting. Mobile devices suffer from: frequent disconnection, low bandwidth connection, limited battery life, and limited memory. A middleware for mobile devices is a software that assists an application to interact or communicate with other applications on other mobile devices. A middleware should support the following basic set of services:

- *Communication Services:* enables communication between different mobile devices.
- *Execution and Listening Services:* provides capability to execute method calls on remote devices and also be able to listen to incoming method calls.
- *Data Access and Connectivity Services:* makes data on one mobile device be accessible to authorized groups of devices and also provide ability for mobile devices to connect to other devices.

We have earlier proposed the System on Mobile Devices middleware (SyD) [11], [12], [13], [14]. SyD provides all the above mentioned services based on Remote Method Invocation (RMI). An emerging middleware approach is the agent oriented middleware approach. The services that a middleware provides, can be realized by employing mobile agents. Mobile agent approach inherently has advantages when compared to RMI. Once the agent is transported to a destination host, the agent can go ahead and execute even in case of a disconnection and when the connection is alive, the agent returns the result to the source host. This is the basic motivation for implementing an agent module for SyD.

In this paper we show the design and implementation of an agent module for SyD. One of the challenges of designing the agent module for mobile devices is the memory size overhead of the mobile agent framework. We have used μ Code agent framework [10], as it is lightweight and also provides the basic required services of an agent framework. We also describe the experiments conducted on the agent module for SyD.

The rest of the paper is organized as follows: In Section 2, we give a brief overview of SyD middleware. We present background on agents and design of the mobile agent module for SyD in Section 3. The details of the experiments conducted on the agent module are described in Section 4. In Section 5, we compare our work with other peer's work and finally we offer our conclusions in Section 6.

2 SyD Middleware

We give a very brief overview of System on Mobile Devices (SyD) middleware [11], [12], [13], [14]. SyD is a new middleware technology that addresses the key problems of heterogeneity of device, data format and network, and that of mobility. SyD allows rapid development of a range of portable and reliable applications. SyD combines ease of application development, mobility of code, application, data and users, independence from network and geographical location, and the scalability required of large enterprise applications concurrently with the small footprint required by handheld devices. Each device is managed by a SyD deviceware that encapsulates it to present a uniform and persistent object view of the device data and methods. Groups of SyD devices are managed by the SyD groupware that brokers all inter device activities, and presents a uniform world view of the SyD application to be developed and executed on. All objects hosted by each device are published with the SyD groupware directory service that enables SyD applications to dynamically form groups of objects hosted by devices, and operate on them in a manner independent of devices, data, and underlying networks. The SyD groupware hosts the application and other middleware objects, and provides a powerful set of services for directory and group management, and for performing group communication and other functionalities across multiple devices. SyD middleware can function with or without a backbone network infrastructure on weakly connected networks as well as on ad-hoc networks providing varying levels of QoS guarantees.

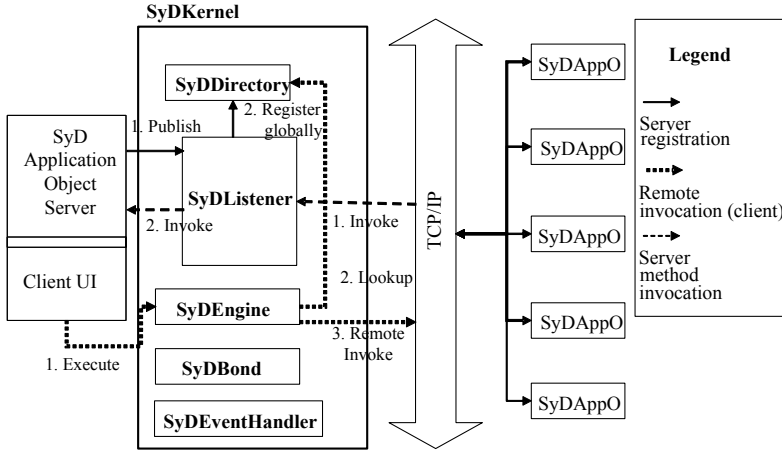


Fig. 1. SyD kernel architecture

We have earlier designed and implemented a modular SyD Kernel utility in Java. Fig. 1 describes SyD Kernel with the following five modules :

- **SyDDirectory:** Provides user/group/service publishing, management, and lookup services to SyD users and device objects and also supports intelligent proxy maintenance for users/devices.
- **SyDEngine:** Allows users to execute single or group services remotely and aggregate results.
- **SyDListener:** Enables SyD device objects to publish their services (server functionalities) as “listeners” locally on the device and globally via the directory services. It then allows users on SyD network to invoke single or group services via remote invocations seamlessly (location transparency).
- **SyDEventHandler:** Handles local and global event registration, monitoring, and triggering.
- **SyDBond:** Enables an application to create and enforce interdependencies, constraints and automatic updates among groups of SyD entities.

3 Agent Module for SyD

Here, we first give introduction to mobile agents and then present the design of agent module in the context of SyD.

3.1 Mobile Agents

Mobile agents can be considered as an incremental evolution of the earlier idea of process migration. A mobile agent is an autonomous, active program that can move both data and functionality (code) to multiple places within a distributed system. The state of the running program is saved and transported to the new

host, allowing the program to continue execution from where it left off before migration [5],[8].

Mobile agents require two components for their successful execution. The first component is the agent itself. The second component being the place where in an agent can execute. This is often referred to as the software agent framework. It provides services and primitives that help in the use, implementation and execution of systems deploying mobile agents. This generic framework allows the developers to focus on the logic of the application being implemented, instead of focusing on the implementation details of the mobile agent system. Specifically, It should support the creation, activation, deactivation and management of agents, which include mechanisms to help in the migration, communication, persistence, failure recovery, management, creation and finalization of agents. Additional services as naming and object persistence can also be provided. This environment must also be safe, in order to protect the resources of the machine from malicious attacks and possible bugs in the implementation of the agent code. Some of the popular examples are: IBMs Aglets[6], Mitsubishi Electric ITAs Concordia [7] and Object Spaces Voyager [4].

3.2 Design of Agent Module

In Fig. 2, we describe the agent module in the context of SyD. A mobile device serving as a server (SyD Application Object Server in Fig. 2), registers it's services and then provides services to clients. We illustrate this using the following steps (arrow labels with legend "Server registration"):

1. In the event of a new publish, the SyD Application Object Server sends a publish request to the Agent Module.
2. The Agent Module publishes and registers the services offered by the SyD Application Object Server to the SyDDirectory.

A mobile device serving as a client (Client UI in Fig. 2) can execute object services located on remote devices using the Agent Module. We illustrate the Client UI process in the following three steps (arrow labels with legend "Remote invocation(client)"):

1. The Client sends an execute of a remote service as a local call to the Agent Module.
2. The Agent Module dispatches an agent on to the SyDDirectory to get the remote user/service information of the remote server
3. With the user/service information (typically the URL of the remote server), the agent module dispatches another agent to complete the remote invocation.

We have used μ Code[10] as our agent framework for mobile devices. Fig. 3 gives the internal details of the Agent Module. μ CodeServer is running on each device listening for incoming mobile agents and is also capable of executing mobile agents on remote devices. Fig. 3 shows a sample method being executed by mobile device 1 on device n .

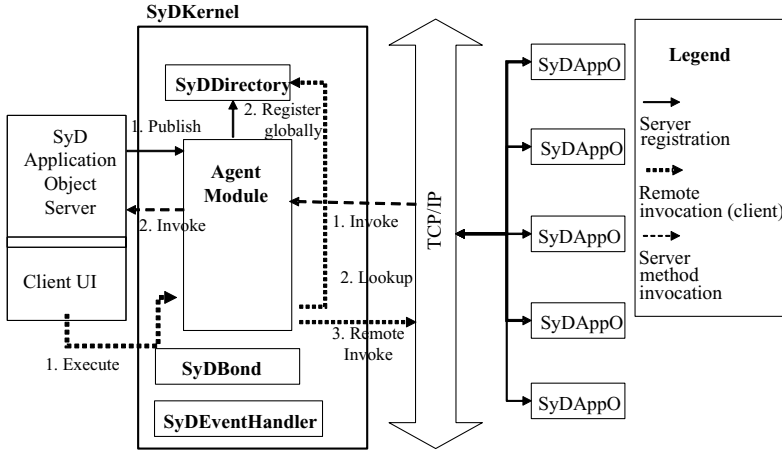


Fig. 2. Design of agent module for SyD

1. Mobile Device 1 sends an agent to the directory service to get the physical location information(URL) of device n.
2. The μ Code agent framework has a listener, which listens for incoming mobile agents. Mobile Device 1 now has the physical location(URL) of the Mobile Device n.
3. Mobile Device 1 dispatches mobile agent to Mobile Device n to execute a method call.
4. Mobile Device n returns the result of executing the method call through an agent on to mobile device 1.

However, it should be noted that, steps 1 and 2 from above could also be realized by simply exchanging messages between the agents rather than sending the agent by itself.

μ CodeServer

μ Code[10] is a lightweight software agent framework for mobile devices (small footprint - the core package is less than 18 Kbytes of jar file). It is a small Java API that aims at providing a minimal set of primitives to support mobility of code and state (i.e., Java classes and objects). It provides good abstractions for doing only a single thing, that is, moving code and state around. It also constitutes the kernel, providing small and efficient mechanisms for code mobility.

4 Experiments on the Agent Module

Experimental Test Bed: We ran our experiments on a high performance/low power SA-1110 (206 MHz) Compaq iPAQs H 3600, with 32 MB of SD RAM and 32MB of flash ROM. The handheld devices are connected through a mobile wireless network using a 2.4GHz wireless router. The operating system is

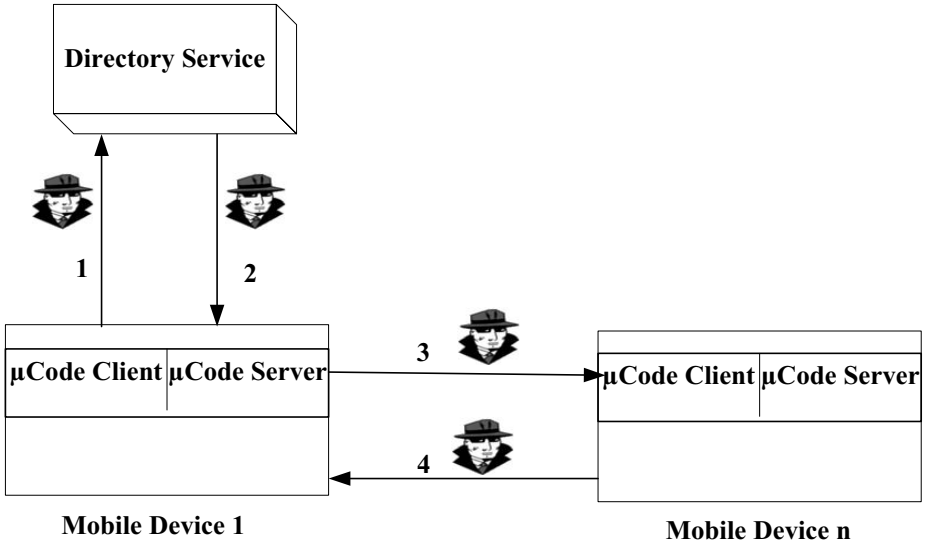


Fig. 3. Internal architecture of the agent module

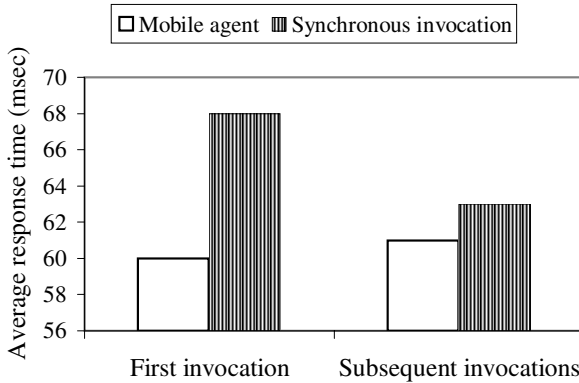


Fig. 4. Mobile agent vs Synchronous invocation - number of invocations

Windows CE. The mobile agent frame work on the iPAQ is μ Code version 1.03. We use jdk version 1.3 to code our programs and JVM for iPAQ is Jeode VM Version 1.9. The DBMS of the directory server is Oracle 8i.

The proposed agent module for SyD replaces the SyDEngine-SyDListener pair. Here we compare synchronous invocation (SI) via SyDEngine-SyDListener pair with mobile agent (MA) using: response time based on multiple method invocations and response time based on size of data processed.

Response time is the total time required to execute a method call on a remote host. Number of method invocations is the number of times a particular method is called. In order to be consistent, we transferred a message of size 16 kilo bytes and the results of it are shown in Fig. 4. For a single method invocation, the

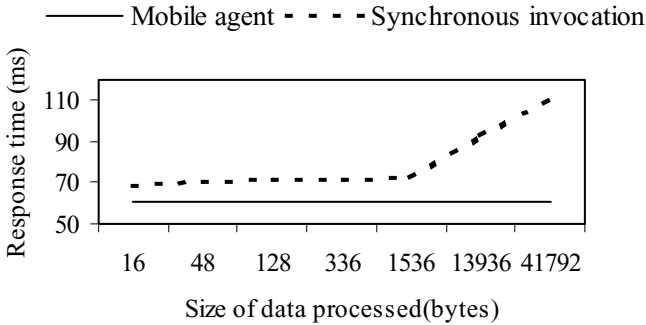


Fig. 5. Mobile agent vs Synchronous invocation - size of data processed

average response time of synchronous invocation (SI) of SyDEngine-SyDListner pair is much higher than mobile agent (MA) approach. However for subsequent method invocations, SI response time gets better. SI starts out with a higher response time and this could be attributed to the fact that, it needs to load stubs and skeletons and also factors such as marshalling/un marshalling and the implicit object serialization that is involved with any RMI-based approach. However for subsequent method calls, the client side stubs and skeletons are already bound to the server and RMI registry look up is faster. In the MA approach, the only initialization time is the time required for the μ Code to start up. For all other subsequent method invocations, it's the same.

Fig. 5 gives the comparison based on the size of the data processed. The response time for MA is much lesser than SI. In the MA approach, data is processed at individual sites and processed data is sent across the network. In the SI approach, data is collected from multiple sites and then processing takes place on the gathered data and therefore results in higher response time.

5 Related Work

Our design of the mobile agent module is chiefly inspired from Limone[1]. Limone provides rapid application development over ad hoc network's consisting of logically mobile agents and physically mobile hosts. Lists of agents that satisfy the policy of host agent are stored into its acquaintance list. The host agent retains full control of the local tuple space since all remote operations are simply requests to perform a particular operation for a remote agent and are subject to policies specified by the operation manager. This high degree of security encourages a collaborative type of interaction among agents. This coordination model and middleware promises to reduce development time for mobile applications. We don't have the concept of a tuple space in our model description. However limone uses tuple space, as it's underlying model. Our directory service serves the purpose of the acquaintance list.

A programmable event based middleware[3] is developed for pervasive mobile agent organizations. A concept of organization oriented framework for the design

of mobile agent application in pervasive computing scenarios is discussed. This middleware is an event-based approach based on the definition of a minimal event-kernel, which is suitable for deployment in resource-constrained devices.

A mobile agent based PC Grid[2] is a mobile agent based middleware where remote computers users wish to mutually offer their desktop computing resource to other internet group members. Each agent represents a client user, which carries out their requests, searches for the available resources, executes the job at suitable computers, and migrates it to others when the current ones are not available.

Data Lockers[15] is another research activity under mobile agent middleware. Data lockers allow users of mobile devices to rent space at the fixed network. This helps mobile users to perform computations remotely with out bothering about the memory space and computation capacity of mobile devices.

A plenitude of research is available on mobile agent based approach for middleware as discussed in the programmable event based middleware[3], PC Grid[2] and Data Lockers[15]. However, we have not seen much research comparing the different middleware approaches. A close line of study to ours can be found in [9]. They discuss performance evaluations of different java based approaches to web database access. We compare middleware approaches and [9] compares java based approaches.

6 Conclusions

We already have a full scale design and implementation of a RMI-based middleware (SyD). We proposed the design and implementation of an agent module for SyD. We have implemented, evaluated and compared the agent module versus the synchronous invocation of SyDEngine-SyDListener Pair. We have also presented performance comparisons of average response time based on varying number of method invocations. We have not taken in to account of the security drawbacks that mobile agents imposes on the system. The security aspect is ignored in this paper as it is out of the scope for the performance evaluations.

As part of the future work, we plan to carry out experiments and do performance evaluations based on: response time(n), where n is the no. of disconnections and the agent framework overhead vs the SyDListener overhead. We aim to design and implement a hybrid engine that extracts the best of the features of agent and RMI approaches by automatically switching between them depending on a decision algorithm.

References

1. C.-L. Fok, G.-C. Roman, and G. Hackmann. A lightweight coordination middleware for mobile computing. Technical report, Technical Report WUCS-03-67, Washington University, Department of Computer Science and Engineering, 2003.
2. M. Fukuda, and Suzuki N. Tanaka, Y., L.F. Bic, and S. Kobayashi. A mobile-agent-based pc grid autonomic computing. In *Fifth Annual International Workshop on Active Middleware Services (AMS'03)*, pages 696 – 703, Seattle, Washington, June 25 - 25, 2003.

3. M. Gazzotti, M. Mamei, and F. Zambonelli. A programmable event-based middleware for pervasive mobile agent organizations. In *11th IEEE EUROMICRO Conference on Parallel, Distributed, and Network Processing*, pages 517–525, Genova, Feb. 2003.
4. G. Glass. Overview of voyager: Objectspace’s product family for state-of-the-art distributed computing. Technical report, ObjectSpace, 1999.
5. C. G. Harrison, D.M. Chessm, and A. kershenbaum. Mobile agents: Are they a good idea? Technical report, Research Report, IBM Research Division, 1994.
6. G. Karjoth, D. Lange, and M. Oshima. A security model for aglets. *IEEE Internet Computing*, 1(4), 1997.
7. R. Koblick. Concordia. In *Communications of the ACM*, march, 1999.
8. P. Madiraju and R. Sunderraman. A mobile agent approach for global database constraint checking. In *ACM Symposium on Applied Computing (SAC’04)*, pages 679–683, Nicosia, Cyprus, 2004.
9. S. Papastavrou, P.K. Chrysanthis, G. Samaras, and E. Pitoura. An evaluation of the java-based approaches to web database access. *International Journal of Cooperative Information Systems*, 10(4), 2001.
10. Gian Pietro Picco. μ code: A lightweight and flexible mobile code toolkit. In *Mobile Agents, Procs. of the 2nd Intl. Workshop on Mobile Agents (MA)*, volume 1477, pages 160–171. Springer, LNCS, Stuggart, 1998.
11. Sushil K. Prasad, V. Madiseti, et al. System on mobile devices (SyD): Kernel design and implementation. In *First Intl. Conf. on Mobile Systems, Applications, and Services (MobiSys), Poster and Demo Presentation*, San Francisco, May 5-8, 2003.
12. Sushil K. Prasad, V. Madiseti, et al. Syd: A middleware testbed for collaborative applications over small heterogeneous devices and data stores. In *5th ACM/IFIP/USENIX International Middleware Conference*, Toronto, Ontario, Canada, October 18th - 22nd, 2004.
13. Sushil K. Prasad, Vijay Madiseti, et al. A middleware for collaborative applications over a system of mobile devices (SyD): An implementation case study. Technical report, Technical Report CS-TR-03-01, Department of Computer Science, Georgia State University, July 16, 2003. <http://www.cs.gsu.edu/~cscskp/PAPERS/CONF/TechRep/SyDTechReport.doc>.
14. Sushil K. Prasad, M. Weeks, et al. Toward an easy programming environment for implementing mobile applications: A fleet application case study using SyD middleware. In *IEEE Intl Workshop on Web Based Systems and Applications, at 27th Annual Intl. Computational Software and Applications Conf. (COMPSAC)*, pages 696 – 703, Dallas, Nov 3-6, 2003.
15. Y. Villate, A. Illarramendi, and E. Pitoura. Data lockers: Mobile-agent based middleware for the security and availability of roaming users data. In *IFCIS International Conference on Cooperative Information Systems (CoopIS’2000)*, September, 2000.