A Case Study of Dynamic Application Partitioning in Mobile Computing — An E-mail Browser*

Grace Hai Yan Lo and Thomas Kunz; {hylo, tkunz}@uwaterloo.ca
University of Waterloo, ON, Canada

October, 1996

Abstract

Mobile computing needs new designs in applications that can adapt to network performance. Dynamic application partitioning is a potential useful technique in developing mobile applications. This paper describes the case study of the design of an email application that aims to partition dynamically according to the wireless network environment. It also discusses the mobile computing research work at the University of Waterloo and the future development of a load-balancing scheme for partitioning applications in the mobile environments.

1 Introduction

The wireless networking technology has engendered a new era of computing, called mobile computing. Portable devices like laptop and palm top computers make it possible for mobile users to access diverse sources of global information anywhere and at any time.

However, today's wireless technologies face major problems of limited network bandwidth, high and variable latency, and high error rates etc., which do not exist in the traditional distributed system environment. New designs in mobile applications are needed to minimize dependence upon continuous connectivity, better utilize the network bandwidth and allow for dynamic allocation of work between client and server. In particular, application partitioning ([WatA] and [WatE]) is a way to achieve the above objectives.

This paper describes the concept of application partitioning for the mobile environment and presents a case study of an e-mail application. It also discusses our research goals and approach at the Mobile Computing Group of the Shoshin Group at the University of Waterloo.

2 Application Partitioning

In the past, traditional sequential applications just run on one machine. As distributed applications evolve, the client/server architecture is the most popular design of applications where the client application is split statically between the client and the server side.

According to [WatA], [WatE] and [HokA], application partitioning is a concept where the application functionality is divided or replicated between the mobile device and the wired network. The interface

*This paper is presented at the Workshop on Object Replication and Mobile Computing (ORMC'96) of OOPSLA'96 at San Jose, Califonia in Oct 1996

boundary can be chosen based on efficiency and the network conditions. Most of the current mobile applications are partitioned statically between the mobile client and a proxy server. In other words, the way the application is split is determined at compile time. The proxy server, or *service proxy* in [HokA], is on a stationary machine connected to the wired network and it acts just like a server for the mobile client.

Dynamic partitioning of an application allows the application to dynamically split between the client and the proxy server at run-time. The Rover toolkit [JosR] and the Wit World Wide Web (WWW) browser application, W* ([WatE], [Wathtml] and [WatU]) both incorporate the concept of application partitioning in their designs. Rover uses the concept of relocatable dynamic object (RDO), which is an object with a well-defined interface that can be dynamically loaded into a client from a server or vice versa. Wit uses annotated object graphs of the Wit objects for cooperative resource management design. The graphs are used for anticipating data accesses and resource management.

3 Our View of Dynamic Application Partitioning

We think that mobile applications should be dynamic enough that the same application could be run under different bandwidth conditions, so as to provide location transparency to the users. We also consider the possibilities of migrating the mobile objects in dynamic partitioned applications. If the objects are not allowed to be migrated, their placement decisions could still be postponed until creation time. The application is split differently in low bandwidth wireless environment and high bandwidth wired environment. If object migration is allowed, then the movement of the objects are decided at run-time and may react to the changes in the wireless environment.

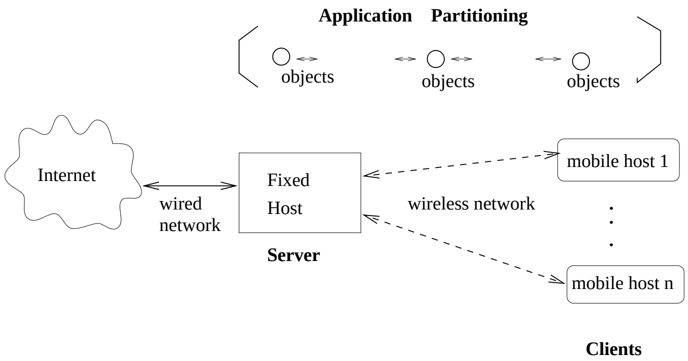


Figure 1 : Dynamic Application Partitioning Between Fixed Server and Mobile Clients

Figure 1 illustrates the design of dynamic partitioned applications running between a proxy server and several clients which are mobile. The fixed proxy server is connected to a fixed network, for instance, the Internet, and to the mobile hosts through a wireless network. An application can be split to run on both the client and proxy server with distributed objects migrating in the wireless network.

4 Case Study : An E-mail Application

To employ the above Dynamic Application Partitioning model in the wireless environment, we have designed an email browser that allow objects to migrate in the wireless network. The goal is to derive a load-balancing algorithm for migrating objects in the wireless network to efficiently partition applications dynamically and provide access and data transparencies to the application developers.

4.1 Mentat

Before discussing the design of the e-mail application, we first describe the software system we used for building the object-oriented distributed mobile applications — Mentat. The Mentat system is an object-oriented parallel-processing system built by the Mentat Research Group at the University of Virginia ([Mentat2.9] & [GriPort]). There are two primary components of Mentat : the Mentat Programming Language (MPL) and the Mentat Run-Time System (RTS). MPL is an extended C++ language designed to simplify the task of writing parallel applications by providing parallelism encapsulation. It contains two types of objects : independent objects and contained objects. Mentat objects are independent objects which possess a distinct address space, a system-wide unique name, and a thread of control. They are analogous to UNIX processes. Contained objects are like C++ objects. In our design, we use ovals to symbolize mentat objects and rectangles to symbolize contained objects.

The Mentat Run-Time System provides a set of services needed by the application programs including instantiation and scheduling of Mentat objects, program graph construction and management. In particular, the instantiation managers (IMs) and the unbound token matching units (TMUs) are daemons required on each host of an active Mentat network. The IMs are used in making scheduling decisions, thus are useful in the object load-balancing mechanism to be derived in the future.

The reasons we choose Mentat as the software system for writing applications include its completeness and availability, its ability to provide a distributed programming environment, and its support of distributed independent objects. With the above features, different dynamically partitioned mobile applications can be developed.

4.2 Message Handling (MH) System

There are different mail user agents available that utilize the SMTP [RFC821] or POP3 [RFC1081] standards. The MH Message Handling System ([MHS] & [MHfaq]) is one of them, with the advantage of allowing users to execute commands from a Unix shell prompt. In this way, the e-mail application can be easily built by invoking the MH routines, rather than calling the different mail protocols directly.

The MH Message Handling System provides facilities that help users to deal with messages in various ways. MH consists of the message database to store messages, the user's profile to specify certain actions of the message handler and the commands for dealing with messages.

Under MH, each message is stored as a separate UNIX file. A UNIX directory in which messages are stored is called a folder. The messages in a folder have numerical names. The user profile is specified in a file called *.mh_profile* in the home directory. It specifies the path name to the directory containing the folders. Also, another file, called *context*, stores the "current" folder the user last referenced.

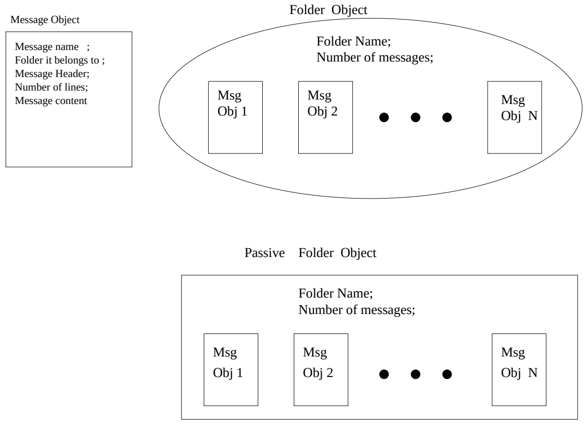
MH executable commands are available in the Base Operating System/ Mail Handler on AIX 3.2, 4.1 and 4.2. Other interfaces, like Exmh, using Tcl/Tk, are developed for the MH routines. In this paper, we will illustrate the mobile versions of the MH commands "inc" and "comp".

4.3 Design

In our design, we model folders as mentat objects. Their methods can be invoked remotely or locally, and transparent to the users and the programmers. At one time, there should be only one instance of the folder object except at the point of object migration. All methods support remote invocation and local execution, but the decision of using which is only known at run-time. The goal is to make the different method invocations and the migration of the objects dependent on the wireless network condition only. The network parameters are returned by calling the Application Programming Interface (API) written in our mobile computing research group [NiddW].

Figure 2 shows the basic data structure of the objects used in our design. The rectangle ones are the data objects (contained objects) while the oval one is the mentat object. A folder is chosen as mentat object because most operations/methods will be applied on the folders. The messages are implemented as data contained in the mentat object rather than mentat objects to minimize the number of processes created. The passive folder object contains the same information as the mentat folder object except that it is only used in object migration and could be returned by a method. Currently, Mentat does not support object / process migration.

Figure 2 : The data structure of the objects used in the mobile e-mail browser application

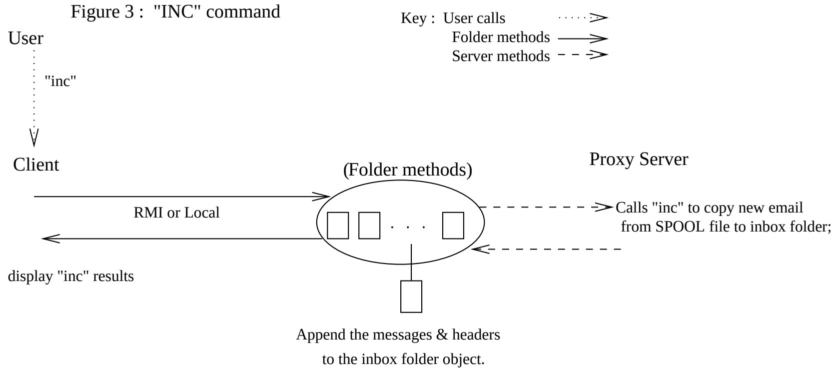


4.4 Two scenarios of Mobile MH commands

The folder objects could be instantiated at the mobile client or the proxy server. The following two scenarios show how the method invocations work.

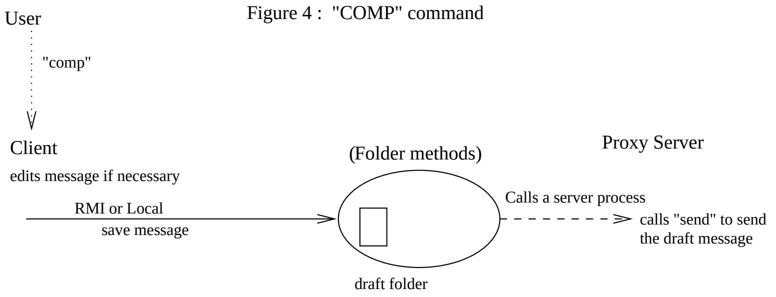
1. inc — gets a scan listing of the newly arrived messages and adds them to the user's "inbox" folder.

Figure 3 : "INC" command



2. comp — allows users to edit a message and send off the message

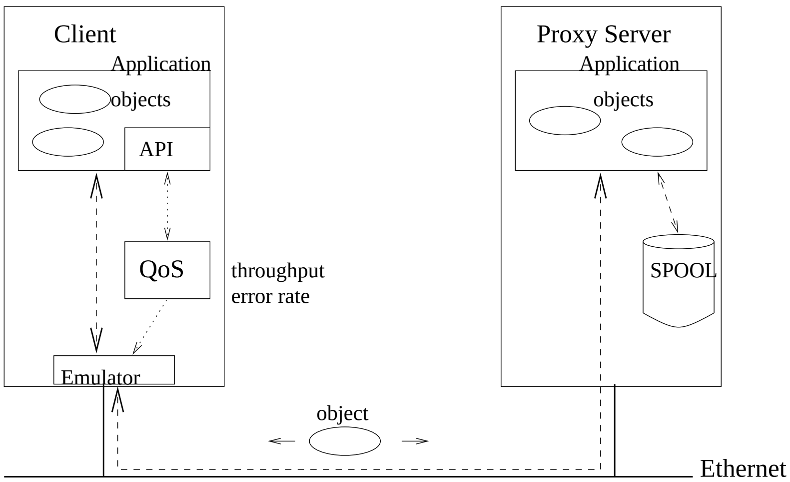
Figure 4 : "COMP" command



5 Architecture

Figure 5 shows the architecture of the wireless testbed in our research group at the University of Waterloo. Our research at the University of Waterloo focuses on "Client-Server Computing in Wireless Networks" [NiddW]. It extends existing QoS research into design techniques that minimize user frustration on a wireless platform. This plan includes the development of reactive applications that can adapt to existing network performance, and an API that can support them. A generic application-layer package is developed to allow network parameter tolerances, and the response to violations, to be adjusted at run-time. We also use an emulator to emulate the wireless environment to execute mobile applications. A simulator is developed to allow quality-of-service parameters to be estimated and used to drive the emulator in delaying or dropping packets.

Figure 5 : Architecture of the Mobile Client-Server Applications



Our object-oriented partitioned application will call the API to access network statistics, like throughput and error rate. The goal is to incorporate a load-balancing algorithm for migrating objects and control the load on the client and proxy server, in order to reduce the latency.

The efficiency of the load-balancing mechanism will be measured. In the case of our e-mail application, suppose there are two types of folders : one of size 30KB and one of size 1KB. The time spent on creating the folders, fetching messages from the folders, etc., will be taken. A comparison study of the performance of each scenerio under remote method invocation and local invocation will be done. Rough estimates of the throughput will be calculated to show the plausibility of the balancing algorithm.

Other mobile applications like news reader and web browser can be developed for testing and evaluating the feasibility and efficiency of this balancing scheme. Currently, our wireless testbed is also used by other members to extend audio and groupware applications with mobility awareness.

6 Conclusions

In this paper, we have described the dynamic application partitioning technique in developing mobile applications. A case study of designing an e-mail application is presented and being implemented in our research group. The goal of the research is to derive a load-balancing scheme to migrate the objects according to the network condition. Due to the huge demand of new mobile applications, we believe that the issue of designing dynamically partitioned applications for the mobile environment is a large potential research area.

Acknowledgement
Thanks to the Mentat Research Group for the availability of the Mentat system.

References :

[GriPort] Grimshaw, Andrew S., Weissman, Jon B., Strayer, W. Timothy. Portable Run-Time Support for Dynamic Object-Oriented Parallel Processing. ACM Transactions on Computer Systems, Vol. 14, No. 2, May 1996, Pages 139-170

[HokA] Hokimoto, A., Kurihara, K., Nakajima, T. An Approach for Constructing Mobile Applications Using Service Proxies. Proceedings of the 16th ICDCS, p.726 - 733. 1996

[JosR] Joseph, A.D., deLespinasse, A.F., Tauber, J.A., Gifford, D.K., Kaashoek, M.F. Rover : A Toolkit for Mobile Information Access. SIGOPS'95 12/95.

[Mentat2.9] Mentat 2.9 Programming Language Reference Manual.

[MHS] The Rand MH Message Handling System: User's Manual.

[NiddW] Nidd, Michael; Kunz, Thomas; Black, James P. Wireless Application and API Design. Fourth International IFIP Workshop on Quality of Service - IWQoS'96. Februrry 1996.
Homepage : http://ccnga.uwaterloo.ca/mobile/index.html

[RFC821] Postel, Jonathan B. Simple Mail Transfer Protocol, August 1982

[RFC1081] Rose, Marshall. Post Office Protocol - Version 3, November 1988

[WatA] Watson, Terri. Application Design for Wireless Computing. In Proceedings of the workshop on Mobile Computing Systems and Applications, p.91-94, Santa Cruz, CA, U.S. December, 1994
Homepage : http://snapple.cs.washington.edu:600/mobile/mcsa94.html

[WatE] Watson, Terri. Effective Wireless Communication Through Application Partitioning.
http://snapple.cs.washington.edu:600/wit/presentations.html

[WatU] Watson, Terri. Using application data semantics to guide system network policies.

[Wathtml] Watson, Terri. Homepage of Wit II : http://snapple.cs.washington.edu:600/wit/witII/