Synergetic State Evolution under Mobile Computing*

Carlos Baquero†
DI - Universidade do Minho
4700 Braga, Portugal
cbm@di.uminho.pt

May 1995

Abstract

The recent trend towards mobility and ubiquitous computing issued a new perspective over the traditional models of distributed computation. Observation of Human behavior, in particular the study of Human information interchange techniques and protocols presents a simple, yet fruitful, mean of gathering insight on the possible protocols for interaction among mobile hosts.This work will try to go one step further on the study of mobile interactions by leaving the usual semi-centralized approach to mobile computing. Instead of focusing on the reconciliation of mobile hosts with the networked support stations, we will study the possibility of progressive adjustments, both by a mobile host and a support station and between mobile hosts.

1 Introduction

The recent trend towards mobility and ubiquitous computing issued a new perspective over the traditional models of distributed computation. Disconnected operation raises new boundaries in the classical management of network partitions, and calls for enhanced techniques of replica reconciliation and conflict avoidance. Fortunately, and unlike network partitions, disconnected operation is almost always an anticipated action (at lest for nomadic computing).

Observation of Human behavior, in particular the study of Human information interchange techniques and protocols presents a simple, yet fruitful, mean of gathering insight on the possible protocols for interaction among mobile hosts. These protocols will be analyzed from the perspective of persistent data store management, which will be the primary focus of this study.

Humans, as a social species, have developed many forms of interaction for the support of cooperative work. Even, point to point comunication, can include several small differences in the way the same message is transmited. We may transmit an information that we have originated, and by transmiting it to someonoe else, we may be expecting that it keeps it confidential or (on the opposite) we may expect him to disseminate it as broadly as possible. We can be expecting or not a reply to our comunication, and this reply can be needed from all receptors or just from some specific one. In another situation, we can be just forwarding a message, and we may be able, or not, to ensure that is not redudant or up to date. These examples start to show the diversity of direct and indirect comunication that we have at our disponsal, and wich, we hope, will be usefull in mobile comunication.

*Technical Report.

†The autor is supported by JNICT-PRAXIS XXI grant BD / 3123 / 94.

This work will try to go one step further on the study of mobile interactions by leaving the usual semi-centralized approach to mobile computing. Typical frameworks for mobile computation are based on a network of support stations (NSS) that act as connection points for mobile hosts (MH). Instead of focusing on the reconciliation of mobile hosts with the networked support stations, we will study the possibility of progressive adjustments, both by a MH and a NSS and between MHs.

The FICUS distributed file system[6] already pointed in this direction by estabelishing a peer to peer protocol for replica management. The benefits of these kind of protocols are also explored in the Bayou system [3].

Next follows a description of the area of concern of the MATE (Mobile Aplications Transactional Environment) project, under the AMIGOS framework, followed by its presentation through the remaining of this document.

2 Focused connection states

Mobile hosts, when active, can be, at a given time and with respect to the support stations, either (C) connected by a fast reliable link, such as ethernet or serial cable; (PD) partially disconnect[1], when using a slow and possible unreliable link, such as a cellular phone or radio link; (D) or in disconnected operation. Existing frameworks support some or all of the previous possibilities, namely: $\{C, PD, D\}$,$\{C, D\}$,$\{PD, D\}$.

In the remaining of this description we will abstract from issues related to the PD state, concentrating on the duality Connected vs Disconnected. Apart from the low level issues of PD which are subject of ongoing investigation [7], such as the development of communication mechanisms that are aware of and tolerate mobility, we expect that the higher level semantics of PD can be selected from features present for C and D states. As PD varies with the degree of available connection, the selected balance of features will also vary.

[1] Or, alternatively, partially connected.

3 A Global or a Synergetic State

Systems based on MHs and NSSs, have inherently a given structuring hierarchie. Persistent data objects usually have the "official" copies in the fixed network, and MHs act as temporary copy holders that ultimately have to be reconciliated with the copies held on the central hosts. In a more decentralized approach there is no default placeholder for official copies. This leads to the need for an explicit binding of ownership to data objects.

3.1 Data Objects and Replicas

In MATE each persistent data object may have several replicas, beeing one of them held on the *storage manager* (SM) that owns the data object. Replicas will act as *persistent proxys* for the replica held on the owner SM. A SM is associated with an host or a group of tightly coupled hosts, though for simplicity reasons we will assume here a one to one relationship between hosts and SM's. All the replicas must have a field that designates the owner host (in fact the SM). Although some applications, for security reasons, may require that the owner is a fixed (non mobile) host, others may work better or even require the ownership to be held on a mobile host. A tipical example is the use of schedulers, where each user holds its personal schedular data in its private machine. Ownership migration should also be permited, although ensuring that all the replicas point to the same owner. A given owner will know for each owned data object, how many replicas exist and who holds them[2].

[2] The need to know who holds them may be subject of relaxation if a autenticated mechanims of handing-off replicas is devised.

A given SM_i will have a indexed list, $f_{N \hookrightarrow SM}$, of references to other SMs whose owned replicas he holds. Each replica R_i^x in SM_i, will have a index ($\in N$) to the local list of references. With this scheme the change of owner for a data object, is notified by comunicating with the SMs that hold replicas of that object. Naturally, in a mobile setup, not all replica holders can be notificated at the same time. This will lead to a transient phase where the former owner and all the already notified SMs will try to disseminate this change.

3.2 Dissemination Protocol

The propagation of information through the SM comunity follows a simple but very synergetic politic, that mimics human cooperative behavior.

> "If A meets B or A can contact B, and A knows or thinks that knows something that B should know then: A will ask B if he needs it and will eventually hand him that information"

Going back to the example of disseminating the changing of a data object owner, we can devise under this filosophy a suitable protocol.

The former, or the new owner, contacts all reachable SMs that hold replicas, and besides updating them also informs them wich SMs could not be contacted. The updated SMs will try to update the necessary replica holders in the *near* future.

This protocol creates some redundancy as some SMs may receive multiple notifications, which in fact also happens among Human interactions. Possible solutions are asking before trying to update a mate; detecting directly or indirectly that a given information has already been fully spread or is now outdated; giving to only a few SMs the responsability of propagating some specifc information.

3.3 Transactions

It is with the introduction of transactional support, that the notion of owner of persistent data object, shows its relevance in MATE's design. As stated by previous works [8, 10, 4], the transaction models used on non mobile distributed systems, are not suitable for a mobile environment. On a distributed system, transactions can expect to access up to date copies of data objects. When this is ensured and no conflicting actions are detected these transactions can commit and make their changes permanent and visible to other transactions. On a mobile environment, we should be able to make transactions locally visible even before a permanent commit can be estabelished.

MATE supports more than one *commit* level. Although the actual number of commit levels may benefict from not being restricted to 2 commit levels, we will assume for the moment the definition of two commit levels. Using Pitoura and Bhargava terminology [10] we will coin this levels as *loose* and *strict*[3].

A given SM can enforce loose commits on any stored replica and can enforce strict commits on replicas that he owns. From these premises results that loose comits are not durable as they can be aborted if one of the replica owners does not agree to promote a strict commit. The likelywood of abort occurence can be reduced by asking some delegation of responsability from the replica owners, as will be shown latter, on this article.

Consider the following example:

Three users have their personal MHs with one SM per machine (wich yields SM_1, SM_2 and SM_3). Each user runs a appointment scheduler that stores its data in a persistent object c (from callendar). There will be three different calendars and each user will have its own callender and replicas of the other callendars. SM_1 owns its callendar c_{1in1} and keeps two replicas: c_{2in1} that represents the callendar owned by SM_2; c_{3in1} that is owned by SM_3.

[3] Actually, the autors define *strict* and *loose transactions*.

Suppose that user 1 is currently isolated and wants to schedule a meeting with the other users for a given time slot t. A transaction T will start on SM_1 and will examine the availability of time slot t in c_{1in1}, c_{2in1} and c_{3in1}. If t is not available in c_{1in1} then T should abort. Otherwise, if t is not available in c_{2in1} or in c_{3in1} there is still a vague possibility of success (as that time slot may have been set free) and T may choose either to *abort* or to issue a *loose commit*. If t is available in all $c_{?in1}$ replicas stored in SM_1 then a *loose commit* is issued.

Loose commits make the changes visible to subsequent local transactions and keep a log of operations in the accessed replicas for posterior reexecution and conflict detection when trying to estabelish a strict commit. It should be noted that we avoid the use of "global commit" as in MATE the notion of globality is related to the set of replica holders, wich are tipically a subset of all existing SMs. So, a strict commit is, for the intervining participants, a global commit. Actually in real life there is seldom a piece of information that people globaly accept as true and official, while it is much easier to make agreements among circunscrit groups of people.

Latter, if and when SM_1 makes contact with SM_2 (supposing he meets SM_2 before meeting SM_3), the transaction T will try to estabelish a loose commit in SM_2. The official replica C_{2in2} can now be examined and compared with c_{2in1}, if c_{2in2} as not changed since the last c_{2in1} synchronization then we can just update c_{2in2}. This can be done either by copying c_{2in1} into c_{2in2} or by rexecuting the loged operations (held for c_{2in1}) over c_{2in2}.

If c_{2in2} as changed then the log of c_{2in1} must be checked for conflict detection. Conflict detection will be based on compatibility matrixes for operations [2], or the a more expressive specification notation like, invalidate descriptions [1], wich enables the expression of hadoc synchronization mechanisms instead of just read-write and write-write synchronization. Depending on the result, two things may happen: T must be aborted (wich possibly affects other transactions in SM_1) or the log may be allowed to execute over c_{2in2}. In the later case, the settling of a loose commit for T on SM_2 deppends now on the observation of c_{3in2}. The time of synchronization of c_{3in2} and c_{3in1} must be compared and the availability of time slot t checked. If t is available then T can issue a loose commit on SM_2, otherwise and depending on the adopted policy we may still proceed and issue a loose commit or alternatively opt for an abort on transaction T. As a consequence of this contact between SM_1 and SM_2, the six c replicas hold among these SMs become synchronized.

Loose commits act as an unbiased compromise. The SM that issues a loose commit is aware that the transaction may be eventually aborted by another SM, but, despite that, promisses to accept the commited data and agrees to be unable to abort it by himself. Within this compromise we are able to start strict commits when the last replica holder is contacted. We can try to visualize this by seeing loose comits as a wave that starts in a SM and propagates along the other replica holders until reaching the last MH, then the wave refluxes as a strict commit wave until all the intervinient SMs are reached. Unfortunately the loose commit wave may hit some obstacle (outdated replicas or concurrency control conflicts) and refluxe early as a loose comit abort wave. Recall that this interaction follows the dissemination protocol policy prescribed on section 3.2.

In our example, the strict commit phase is reached when SM_2 or SM_1 (wich have issued loose commits for transaction T) contact SM_3. Suppose that SM_2 contacts SM_3. The official replica c_{3in3} can now be tested for reconciliation with c_{3in2} under de described protocol. Deppending on the result, T may proceed into a strict commit on both SMs or originate a loose comit abort on SM_2. In either cases SM_1 should be informed of the resulting action. SM_1 will be notified when he meets SM_2 or SM_3.

4 Locks and Delegation

Lock management is beeing addresses on two AMIGOS subprojects, Vitor Guedes on the management of file replicas for file system support to disconnected operation [5], and Jeppe Damkjaer Nielsen on the study of timed locks for transactional support over the file system [9]. On the MATE subproject, lock management is interpreted as a form of delegation from SMs over their owned replicas.

When a given SM_x contacts another SM_y he can request from the later, locks over its (SM_y) owned replicas after synchronizing them. These granted locks can be assigned an expiration period, after wich the delegated properties cease to be ensured by the owner SM. The delegated properties should be specified acordingly to specific aplication needs, and are not yet fully identified. We can, however, forecast some tipical cases:

Reads The owner SM ensures that there will be no changes to a replica, during some period. This enables isolated transactions to issue strict reads over read-locked replicas. These locks can be granted to more than one replica. In fact, once they are granted, it would be advisable to notify the other replica owners so that they can benefict from the read-lock.

Writes The owner SM ensures that a given SM can have strict write permissions for a given period of time. This allows the borrow of replicas for a given period, without having to initiate a owner migration protocol.

Commits The owner SM ensures that he will not enforce strict commits for some period. These assurance allows postponing the reconciliation of competing loose commits until a given time. With this the aplication may induce some fairness among disconnected updates on MHs that reconnect on predictable periods.

In general, any property which can be assigned to a replica can be subjcet to some delegation from its primary holder. Other good candidate properties can be found on compatibility matrixes and on invalidation specifications.

5 Project Development

The concerns and the design philosophy of the MATE project, here depicted, will be developed on my PhD research, supervised by Francisco Moura. This will encompass the development of a suitable programming framework on C++, for the support of mobility and the expression of transactional properties. The programming framework will then be used to implement and measure these policies.

A possible testbed for the validation of some of these ideas is beeing produced under the GIM project. Working on this project, which will develop a mobile scheduler management aplication, are Orlando, António and Rui Oliveira.

References

[1] P. Anastassopoulos and Jean Dollimore. A unified approach to distributed concurrency control. In *Distributed Computer Systems*, pages 545–571, January 1994.

[2] Naser S. Barghouti and Gail E. Kaiser. Concurrency control in advanced database systems. *ACM Computing Surveys*, 23(3), September 1991.

[3] Alan Demers, Karin Petersen, Mike Spreitzer, Douglas Terry, Marvin Theimer, and Brent Welch. The bayou architecture: Support for data sharing among mobile users. In *IEEE Workshop on Mobile Systems and Aplications*, Computer Science Laboratory, Xerox Palo Alto Research Center, December 1994.

[4] Robert Gruber, Frans Kaashoek, Barbara Liskov, and Liuba Shrira. Disconnected operation in the thor object-oriented database

system. In *IEEE Workshop on Mobile Systems and Aplications*, Laboratory for Computer Science, Massachusetts Institute of Technology, December 1994.

[5] Vitor M. P. Guedes. Mobile computing systems: Research report. See at http://alfa.di.uminho.pt/ mesvpg, September 1994.

[6] Richard G. Guy, John S. Heidemann, Wai Mak, Thomas W. Page, Gerald J. Popek, and Dieter Rothmeier. Implementation of the ficus replicated file system. In *USENIX Conference Proceedings*, pages 63–71. USENIX, June 1990.

[7] Tomasz Imielinski and B. R. Badrinath. Mobile wireless computing: Challenges in data management. *Comunications of the ACM*, 37(10):18–28, October 194.

[8] Vivek R. Narasayya. Distributed transactions in a mobile computing system. Technical report, University of Washington, March 1994.

[9] Jeppe Damkjaer Nielsen. Project on transactions in mobile computing. obtained by email, 1995.

[10] Evaggelia Pitoura and Bharat Bhargava. Revising transaction concepts for mobile computing. In *IEEE Workshop on Mobile Systems and Aplications*, Department of Computer Science, Purdue University, December 1994.