A Human Centered Perspective for Mobile Information Sharing and Delivery

António Sousa Carlos Baquero José Orlando Pereira Rui Oliveira*
Francisco Moura

Universidade do Minho, Braga, Portugal
{als, cbm, jop, rco, fsm}@di.uminho.pt

May 1996

1 Introduction

Recent developments in portable computing made possible widespread use of small hand held computers [7], in particular, the so called *personal digital assistants* (PDAs). Unsurprisingly, user expectations on these portable devices are quite different from those usual on their static counterparts. In addition to host mobility, the availability of portable communication devices leads to qualitative changes in the problems faced when building distributed systems in this environment.

Developers targeting mobile systems must be aware of these changes in order not to frustrate user expectations, namely, concerning which applications people wish to carry in their pockets. The most obvious of these are personal information management (PIM) and personal communication applications.

This paper focus on the design and implementation of a suitable communication layer to be used with groups of mobile systems, targeted to support personal information management and exchange applications. A PIM was used as case study, focusing on the problems raised by the process of scheduling meetings.

As a first step, the environment is evaluated in order to establish some facts and assumptions that should shape the system. We also think that the 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 possible protocols for interaction among mobile hosts. This is specially true when mobile computing is used to support human information exchange. Finally, the object replication system developed is introduced and a prototype implementation are briefly described.

This work was developed within the AMIGOS project. The main goal of this project is to provide transparent support for semi-connected operations on mobile computers running standard operating systems.

2 Environment

The first step to clearly understand and reason about a distributed mobile system is to understand the differences from a traditional distributed system without support for mobile hosts.

In this kind of system, portable machines can spend some of their time connected to a global internetwork, connected to another mobile machine through a peer-to-peer link, or, most of their time, disconnected. In addition, the connection to the internetwork may be established through an high bandwidth traditional local area network or through an wireless network like GSM. Of course, these different forms of connection should be taken into consideration as they have very different cost/bandwidth ratios.

Due to their hierarchical fixed structure, traditional distributed systems subliminally induce users to communication patterns that closely follow this

*Current address: Laboratoire de Systemes d'Exploitation, EPFL, Switzerland

structure [5, 6]. Examples can be found on the structured location to location channels implicit in e-mail, FTP and HTTP communications. Such mechanisms force the users to know the locations of their communication peers, instead of just having to know with whom they intend to communicate.

Additionally, traditional routing protocols used to deliver data rely on the assumption that machines are not physically moving, even if they consider the fact that more than one path can be used to reach them. In other words, in a network of fixed hosts, each one of them has probability 1 of being in a well known place and 0 probability of being anywhere else. With mobile hosts, there are a lot of places where the probability of the host being there is greater than 0 even if it never reaches 1. Anyway, data addressed to a specific host should be spread to all locations where the probability of the host being there exists.

While in traditional distributed system the network is expected to be available most of the time, a mobile host is only useful as long as it is able to operate disconnected for long periods. In short, while the first might have to wait for the fault to be corrected to restore normal behavior, the second is expected to perform well even if it will not be able to reach the network for a long time.

The fact that mobile hosts spend most of their time disconnected leads to network partitions that are much more frequent and not transient. This means that the probability of being able to find two hosts that need to exchange information in the same partition, so as to establish an end-to-end transfer, is scarce. This calls for the use of third parties to mediate transfers [1, 4, 2].

One way to increase the probability of two machines being in the same partition is to consider peer to peer interaction between mobile hosts. This allows direct exchange of data among them as well as the use of each other as a mediator in multi-step process as described above. This should increase the possibility of data reaching its destination as soon as possible.

Even if mobile communication devices such as cellular phones and wireless networks allow portable machines to be permanently connected to internetworks, the fact that bandwidth in these communication channels is often limited and expensive, is for some applications as useless as no connection at all. This is usually called semi-connected operation: the mobile hosts is considered both connected and not connected depending on the application.

An interesting is that much of the above reasoning is valid for very large scale networks too. The opposite is also valid, if we think of networks of mobile computers as wide area networks where distances grow to infinity, and therefore, some machines become unreachable. This kind of observation is interesting because solutions to problems in wide area networking can be reused in a mobile environment and vice-versa.

3 Human interaction patterns

When dealing with problems involving human communication, like scheduling meetings, people usually are able solve them even if it is not possible to establish some sort of real-time conferencing between all interested parties. This leads us into thinking that it should be possible for computers to deal with similar problems even if they are not simultaneously connected to a network. Therefore, it is reasonable to assume that the patterns of human communication that are responsible for human success in this areas could be adopted as computer communication patterns.

For instance, let us review some methods used by people to schedule some appointment involving a group. When proposing the meeting, the initiator will usually try to contact as many of the participants as possible. For that, he will possibly talk to them, directly to as many as it is possible to group or to each of them by phone. Then, he will try to notify them using e-mail, fax, mail or even by scattering notes in their desks or bulletin boards in places where they usually go. In order to improve the chances of success, a smart person will use some or all of this procedures, probably in the order they were enumerated.

Some useful lessons can be drawn from the *modus operandi* described. First, when possible, direct real-time communication to as many of the participants is desirable. This can be compared to group communication protocols. If not possible, deferred media are used. These can be further classified as many-to-one and one-to-many. The first is useful when the location of the target is uncertain but predictable and consists in spreading the information in places where it will probably be. The second is

useful when the target itself is unknown, making the information available where targets can pick it up. The initiator, might also encourage everyone he talks to spread the word around and try to achieve agreement with whoever he meets. This process can be repeated recursively. When two people from separate groups, who have already agreed in separate with the proposed schedule meet, they can merge their knowledge and use it in future contacts.

This means that indirect multi-step communication might be used to convey information when an end-to-end channel cannot be established. Again, the usefulness of spreading information around, is confirmed, even when considering third parties involved. Another very important lesson to learn is that the possibility of individuals acting on their local incomplete views of the system is very important. However, this is only useful as long as different partial views can be merged whenever two information carriers meet, which sets aside the possibility of restricting the order in which information exchanges take place.

From the point of view of each of the people contacted, it would be nice to have an assistant to collect information from different sources in an incoming folder. Yet, it would be nice if this assistant tries very hard to keep the information in that folder updated and removes duplicate and out-of-date messages. On the other hand, answering to each of these messages should be as simple as placing them in an outgoing folder. The assistant will then figure out how to send them to all interested parties, whoever and wherever they are. From this last observations, we can conclude that people are not interested in dealing with the aspects of communication, specially when there is complexity involved in exchanging messages. What is desirable is to have an updated copy of the status of the shared information available, where it is possible to make changes that will be known by others. In other words, replication of mutable objects is a good attempt to hide the complexity involved in exchanging information in heterogeneous, partitioned networks.

An existing mechanism that resembles this model, is the distribution of Usenet News. News groups are replicated across news servers, that communicate among them by a variety of protocols. Each group can be edited in any of the replicas, by adding news. Eventually, all news will be present in all servers. This validates the idea that problems faced in wide area networks are somewhat similar to those found in mobile environments.

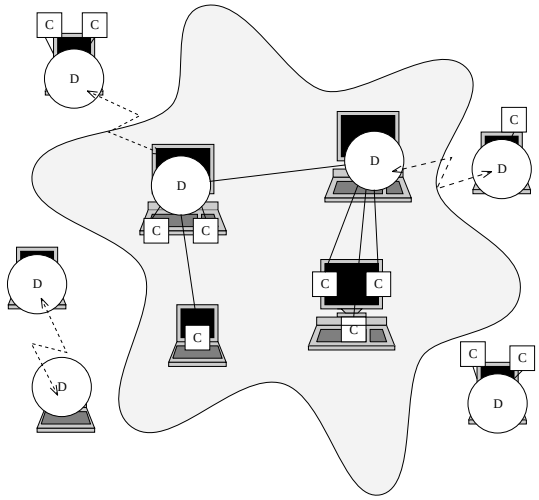

Figure 1: Mobile replication model. (D) Deposit; (C) client.

4 Sharing objects

our model focus on replicating data objects. Therefore, no explicit messages are exchanged. Communication between elements of the group is accomplished by reading and writing their locally available copies.

Information to be shared is partitioned into *objects*. These objects reside in *deposits*. *Clients* connect to deposits in order to inspect and modify information contained in individual objects. This means that one deposit must exist in each possible partition of the network (figure 1).

When a client creates a new object in a deposit it will eventually be copied to other deposits, in order to make it available to whoever it may concern.

Everytime a client wishes to read an object, it retrieves a copy of the local representative of the object from a deposit. It can also write on this copy. When it desires to make changes permanent, it sends its copy of the object back to the deposit, thus issuing a new *revision*. The most recent revision of each object in each deposit is considered

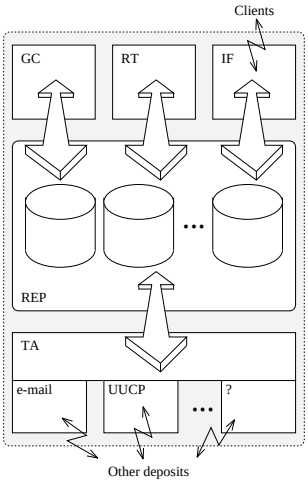

Figure 2: Implementation architecture. (GC) Garbage collector; (RT) routing subsystem; (IF) client interface; (REP) repositories; (TA) transport agents.

the local representative of this object. Deposits are free to discard revisions which are superseded by newer revisions.

When two deposits manage to exchange information, they attempt to merge their contents. This means that revisions that are not yet shared, are copied. After that, more than one revision of an object may be found in any of the deposits. If neither revision derives from the other they are said to be *concurrent*, and thus, both carry unique information. In order to find a local representative of the object, the deposit will ask the first client available to issue a suitable merge which will supersede both concurrent revision and become the local representative.

5 Implementation

This section focus on the implementation of the replication layer proposed. The reference architecture for the deposit is shown in figure 2. Its main feature is a set of passive *object repositories*. Each repository is just raw storage for object revisions. The existence of multiple repositories per deposit is necessary to distinguish incoming, outgoing and locally available revisions.

The *interface module* (IF) is in charge of handling clients requests. It provides revisions as needed and notifies clients of the arrival of new revisions to the deposit. This can be initial revisions (i.e. new objects) targeted to the client or new revisions of objects already in use. It also handles new revisions issued locally, which it stores in the repositories. An important function of this module is detecting the need to merge concurrent revisions. This must be done in order to provide a single local representative of each object. The merging itself is done by the client.

The *routing subsystem* just moves objects around the repositories according a set of rules. It thus is used to select objects from any of the incoming repositories and make them locally available or ready to leave, by copying them to one of the outgoing repositories.

The *garbage collector* is in charge of deleting revisions which are obsolete, either because they were superseded by newer ones or due to real time constraints. The garbage collector is also used to perform smart loop avoidance.

Transport agents are responsible for delivering objects stored in outgoing repositories to one or more foreign deposits, where the corresponding agent places them in an incoming repository. This architecture provides optimum flexibility to integrate different protocols, suited to the characteristics of the underlying communication infrastructure.

A prototype implementation closely following the architecture outlined above, was developed on the Unix operating system. Repositories are represented as file system directories, with each revision being stored in a separate file. An API is available to C++ programmers, consisting of a proxy for the deposit and a base class for replicated objects. This is an abstract class, as methods for handling notifications from the deposit must be overridden.

The routing daemon relies on external tables and revision headers, in order to establish where each file should be sent to. Although currently these tables are configured by the system administrator, they can be in the future generated and kept up to date by a specialized protocol.

In this implementation, transport agents are anything able to take files from one directory corresponding to an outgoing repository to another directory, probably in another machine, belonging to a second deposit. In Unix, there is a plethora of mech-

anisms satisfying this requirement such as NFS, FTP, UUCP, mail, Usenet News and even floppy disks and manual intervention.

6 Conclusions

Although a centralized server or a reliable group communication protocol would allow us to maintain a consistent view of the system, they would be prohibitive in a mobile computing environment, as frequent network partitions would render them unavailable most of the time. Similar conclusions were drawn by [3], although our work is less concerned with consistency and more with the difusion of data.

To circumvent the problem, weak consistency protocols provide high availability, allowing concurrent reads and writes in each of the replicas. On the other hand, only the careful study of each application guarantees a correct behavior in the presence of inconsistency. The replication platform described is thus an valuable tool to evaluate real life problems, collect data, draw conclusions and plan future work.

The flexible architecture of the implementation is also very important, in order to test different communication protocols suited to the heterogeneous and rapidly changing nature of communication networks.

References

[1] A. Acharya and B. Badrinath. Delivering multicast messages in networks with mobile hosts. In *13th Conference on Distributed Computing Systems*. May 93.

[2] Arup Acharya. *Structuring Distributed Algorithms and Services for Networks with Mobile Hosts*. PhD thesis, Graduate School - New Brunswick Rutgers, New Jersey, USA, May 1995. Available as GIT-CC-95-12.

[3] I.Greenberg A.Downing and J.Peha. OSCAR: A system for weak-consistency replication. In *Proceedings of IEEE Workshop on the Management of Replicated Data*, 1990.

[4] S. Alagar and S. Venkatesan. Causally ordered message delivery in mobile systems. Technical report, Santa Cruz, USA, dec 1994.

[5] B. Badrinath, A. Acharya, and T. Imielinski. Handling mobile clients: A case for indirect interaction. Technical report, Rutgers University, New Brunswick, 1994.

[6] B. Badrinath, T. Imielinski, and A. Virmani. Locating strategies for personal communication network. In *IEEE GLOBECOM 92 Workshop on networking of personal communications applications*. December 1992.

[7] Mark Weiser. Ubiquitous computing. *IEEE Computer Hot Topics*, October 1993.