DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Relatório Técnico

RT-MAC-9911

A RELIABLE CONNECTIONLESS PROTOCOL FOR MOBILE CLIENTS

Markus Endler and
Dilma M. Silva and
Kunio Okuda

Setembro de 1999

A Reliable Connectionless Protocol for Mobile Clients*

Markus Endler and Dilma M. Silva and Kunio Okuda
Departamento da Ciência da Computação
IME-USP
{endler,dilma,kunio}@ime.usp.br

Abstract

In this paper we present a client-server protocol which implements reliable delivery of messages to mobile hosts. Reliable here means that for every request from a mobile client to a network service, eventually it will receive the result, despite its periods of inactivity and any number of migrations. Such a protocol is suitable for network services based on request-reply style of communication (e.g. RPC) and with long request processing times. For these services, there is a high probability that a client at a mobile host migrates to another cell while waiting for the reply.

The main advantage of the protocol is that although at each moment only a single *Message Service Station* is responsible for retransmitting request results to a given mobile host, the protocol achieves dynamic global load balancing within the set of Message Service Stations.

1 Introduction

With the ongoing improvement and spread of cellular telephony technology, more and more computer network services will be supporting wireless communication. In near future, it will be possible to access a huge variety of information bases from lightweight, inexpensive and hand-held terminals or PDAs, regardless of their current location. Also, many of these information bases will be fed with data sent by these mobile devices, which may range from frequent updates of simple data, such as the current location, to occasional uploads of huge volume of data. In addition, such information bases may be maintained by collections of distributed servers so that whenever data is to be retrieved or updated, first its storage location must be determined.

Currently we are working on a project called SIDAM (Distributed Information Systems for Mobile Agents) that aims at studying the main architectural

*Supported by FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) - Proc. No. 98/06138-2.

and algorithmical issues involved in the implementation of distributed information services for mobile agents, investigating existing methods and technologies and designing efficient protocols and services for data storage and access.

For that, a general model has been proposed as an integrator of the several research areas involved, namely: distributed systems, performance analysis, fault tolerance, software architecture, databases and knowledge representation. The construction of this model was motivated by a real-life application: an online service providing traffic information in a big city such as São Paulo. Project SIDAM aims at developing a distributed information service that can be queried and updated by mobile users. These users may be either common citizens requesting information about traffic situation in a specific part of the city or may be São Paulo's Traffic Engineering Company staff (in a car or helicopter) feeding the system with data about traffic. This data will be stored (eventually in several forms and degrees of accuracy) in an decentralized information base, consisting of several interconnected Traffic Information Servers (TIS). The decentralized nature of the information base is necessary to handle the high load of concurrent queries and updates, and to take advantage of the locality of updates. In this context, queries and updates to the global information base may involve complex searches, interactions and processing within the TIS network.

In order to implement efficiently the operations offered by the system to mobile users (such as *query*, *update*, *subscribe*[1], and *multicast*[2]) we are designing some protocols[6] that handle different aspects of the operations, such as user mobility, data location, data replication, TIS interaction, etc. We expect the resulting model and design to be generic enough to be easily adaptable to a variety of applications, ranging from support systems for strategical actions to electronic mail systems for portable computers.

In the remainder of this paper we focus our attention on a communication protocol for mobile clients that will be primarily used in the *query* and *subscription* operations.

2 System Model and Assumptions

The model of the system consists of two types of machines, the static hosts and the mobile hosts (Mh). Static hosts are connected to each other through a static and reliable network.

Among the static hosts, some also serve as Message Service Stations (Mss) for the mobile hosts and are assumed not to fail. Each Mss defines a geographic region (*cell*) in which it is able to communicate with the set of mobile hosts currently located in the cell. The information about the Mhs within a cell is maintained at each Mss in a data structure *local_Mhs*.

[1] The user specifies an entity and a threshold on any of its attribute values, and the system will notify the user about related changes in the entity.

[2] The user provides its identification, the identification of a group of users (previously configured) and a message to be sent to the group.

Some of the static hosts may be servers (e.g. a Traffic Information Server) that provide application specific services to both static and mobile hosts. It is assumed that each server maintains a fixed address which can be obtained by querying some directory service.

The mobile hosts are disconnected computers that have a system-wide unique identification and which may be in two possible states, namely *active* and *inactive*. In the inactive state (e.g. power save state, turned off) a Mh is unable to receive or send any message. In order to join the system, a Mh sends a `join` message to the Mss in charge for the cell it is currently in, which then becomes the Mss currently responsible for the Mh (*respMss*). A Mh leaves the system by sending a `leave` message to its *respMss*.

Mobile hosts are able to move from one cell to another. Whenever a Mh enters a new cell it sends a `greet(oldMss)` message to the Mss responsible for the target cell, where `oldMss` is the identity of the Mss responsible for cell which the Mh is leaving. With this information the Mss of the new cell is able to initiate a Hand-off protocol with the previous Mss.

This `greet` message is also sent by the Mh when it becomes active again within the same cell where it has been before become inactive. In this particular case, however, the Mss will not initiate a Hand-off protocol, since the previous Mss mentioned in message `greet` is itself.

Actually, in this model we abstract from the details of how a Mh learns that it is entering or leaving a cell, assuming that this can be achieved in different ways according to the wireless technology in use.

Figure 1 shows a system with three Msss and five Mhs (in the corresponding cells or migrating) and where mobile host Mh_1 is issuing a request to server S from one cell, but may pick up the server's reply while visiting another cell.

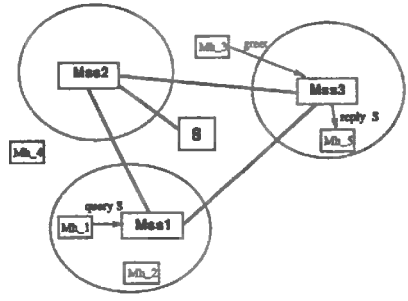

Figure 1: System with three Msss and five Mhs

The main assumptions of the model are the following:

1. Communication among the Msss is reliable and message delivery is in causal order;
2. All Mss are reliable and do not fail;

3. At any time, each active Mh is associated with exactly one Mss (*respMss*).

4. If a Mh is active it must send an `Ack` to all messages received from its *respMss*, and while it is inactive it must not reply to any message. When a Mh migrates between cells it may be considered inactive by both the (Mss_o) or the new Mss (Mss_n) during the period of time of the Hand-off (see Section 3.2).

5. A Mh is able to detect if a message from its *respMss* is a new one or if it is a retransmission.

6. A Mh only leaves the system if it has acknowledged all messages received by its *respMss*.

3 Result Delivery Protocol

In the remainder of this paper we present and analyze a protocol which guarantees reliable delivery of reply messages from a server in the network to a mobile client. This protocol, which we called *Result Delivery Protocol*, is suitable for network services based on a request-reply style of interaction (e.g. RPC) and with long request processing time. For such services, it is likely that the client (on the mobile host) migrates to another cell while waiting for the result of the request. The protocol can be used with a wide range of existing network services since from the perspective of the server, service access is identical to the one by a static client.

Although the presentation of the protocol is focussed on a request-reply style of interaction, like operation *query* of the traffic information service (Section 1), it can be used as well for asynchronous notifications of events to mobile clients. Thus, the RDP may as well be used for implementing the operation *subscribe* (Section 1), by which a mobile client is informed of any major change in the traffic situation.

The RDP protocol is based on the *indirect model* for client-server interactions proposed by Badrinath et al.[2]. In this model the mobile support stations act as representatives for all the mobile hosts within their cell, hold part of the Mh's state, and do the translation between the wired and the wireless communication.

3.1 Outline of the Protocol

The Result Delivery Protocol (RDP) is based on the notion of a *proxy for requests* (or simply *proxy*). A proxy is created on behalf of a Mh that wishes to interact with servers within the wired network. It is created at the Mss responsible for the Mh's current cell (*respMss*) whenever the Mh initiates a new series of service requests. The main purpose of the proxy is to provide a fixed location for the reception of server replies, to keep track of pending requests, store the request's results, and to forward the results to the Mss responsible for the cell in which the Mh is currently located. The proxy is an object which exists only

until all the result forwards are acknowledged by the corresponding Mh, after which it is destroyed. Then, at a later moment, the same Mh may cause the creation of a new proxy at the same or a different Mss, depending on whether it has or not migrated. However, at any time each Mh owns at most one proxy.

In order to decide to which Mss to forward the result of a request, each proxy holds a variable called `currentLoc`, which is updated with the address of the current *respMss* of the corresponding Mh whenever it migrates. A proxy also holds a `requestList` containing identifiers of all pending requests issued by the Mh.

For being able to update `currentLoc` at every migration, each Mh has one *proxy reference* (or simply `pRef`) associated to it, which is hold by its *respMss*. As the name suggests, a `pRef` contains a reference (the i.e. address of the Mss and a proxyID) to the current proxy associated with the Mh. However, when a Mh does not have a proxy (i.e. when it has no pending requests) `pRef` holds a null address. A `pRef` also contains a flag called *Ready to Kill pRef* (RKpR), which signals whether the proxy has forwarded the result of its last pending request (see section 3.3 for more details).

Each time a Mss receives a new request from one of its local Mhs it checks the corresponding `pRef`. If it has an null address, a new proxy is created locally on behalf of Mh and Mss updates `pRef` with its own address and the proxy's objectID. Otherwise, Mss forwards the request to the proxy whose address is mentioned in `pRef`.

When a Mh migrates from Mss_o to Mss_n `pRef` is handed through the *Hand-off* protocol (see Section 3.2) to Mss_n as part of Mh's state. After completion of the *Hand-off* for a Mh, Mh's new location is updated at the proxy. This is done by having Mss_n send a special message (`update currentLoc`) to the proxy. This message is also sent when a *respMss* receives a `greet` message from the Mh announcing its switch from the inactive to the active state. At the proxy, the arrival of the `update currentLoc` message causes the variable `currentLoc` to be updated and any non-acknowledged results from pending requests to be re-sent to the new location.

When the result of any of Mhs pending request (say, request R) arrives at a proxy (located at Mss_p) it is forwarded by Mss_p to the address mentioned in `currentLoc` (i.e. *respMss*) and then delivered to Mh through wireless communication.

In normal conditions, i.e. when a Mh is active and stays in its cell for a sufficient long period of time, the arrival of R's result is acknowledged by the Mh through a $\texttt{Ack}_R$ message, which is forwarded by Mss_n to Mss_p. Once it has arrived, the proxy marks that request R has been completed, removes it from the `requestList`, and possibly sends an acknowledgment to the server, depending on the particular application-level client-server protocol being used.

If *respMss* is unable to reach Mh, because it is migrating or is in inactive state, or simply because wireless communication failed, the *respMss* does not attempt any new forwarding of the result. Instead, it is the proxy that will re-sent the result as soon as it receives the next update of Mh's address.

At each Mss, higher priority is given to forwarding `Ack` messages (from Mhs to Mss_p) than to engaging in any new Hand-off transactions. This avoids that results already acknowledged by a Mh are re-sent to the new cell. However, as soon as *respMss* receives a request (from another Mss) to transfer Mh's state, as part of the Hand-off protocol, it will ignore all future `Ack` messages from this Mh.

From the viewpoint of the proxy, until it does not receive an $\texttt{Ack}_R$ message informing that R's result has been indeed received by the Mh, it keeps re-sending the result to every Mss_n from which its gets a `update currentLoc` message. Notice that this guarantees that every result from a request will eventually reach its destination (the Mh), provided that the Mh does not leave the system with a pending request. The price to pay for this reliability is that eventually a result will be sent more than once to a Mh.

3.2 Hand-off Protocol

The Hand-off protocol is executed between a Mss_o and Mss_n whenever a Mh switches cells, and aims at transferring all of Mh's relevant state to Mss_n. It works as follows:

A Mh entering a new cell registers itself with the corresponding Mobile Service Station (i.e. Mss_n) by sending it a `greet` message containing the identification of its previous *respMss*. After having announced itself to Mss_n, a Mh must not reply to any message from any Mss other than Mss_n.

When receiving the `greet` message, Mss_n sends Mss_o a `dereg` message, asking it to de-register Mh and send back Mh's *proxy reference* (`pRef`). When Mss_o receives `dereg Mh`, it replies to Mss_n with a `deregAck` message containing Mh's `pRef`, and then removes Mh from its *local_Mhs* list. As soon as message `deregAck` from Mss_o arrives, responsibility for Mh is officially transferred to Mss_n, which then includes Mh in its *local_Mhs* list and updates Mh's new location with its proxy, by sending the `update currLoc` message to the proxy, whose address is contained in `pRef`.

3.3 The Proxy Life-cycle

Although every Mh always has at most one proxy that forwards results to it, the location of the proxies in the static network may change over time, by which the protocol achieves dynamic global load balancing within the set of Msss.

As mentioned earlier, a proxy is created whenever a Mh issues a new request and no proxy is yet available, i.e. when address in Mh's `pRef` is empty. After creation, a proxy is capable of handling several concurrent requests from its Mh.

When a proxy (hosted at Mss_p) forwards the result of the last pending request (say, L) to a *respMss*, flag `del-pRef` is set to *true* and is piggy-backed on this result message sent to a Mh's *respMss*. It means that if this message is received by the Mh, then the proxy can be deleted.

When the *respMss* receives a result message with `del-pRef`= *true*, it sets flag `RKpR`=*true* at Mh's `pRef`, meaning that *respMss* will confirm the removal of

the proxy (and erase proxy's address in pRef's) as soon as it receives an $\texttt{Ack}_L$ from Mh that is not preceded by any new request.

If this is the case, *respMss* sends message $\texttt{Ack}_L$ with flag del-proxy= *true* piggy-backed on it to Mss_p, by which this one knows that the proxy can in fact be deleted. However, if in the meantime, a new request from Mh arrives at *respMss*, flag RKpR is re-set to *false* and Ack message is sent with del-proxy= *false*, meaning that the old proxy will also be used for this new request.

Thus, the removal of the proxy is confirmed by *respMss* only if the following condition holds: RKpR= *true* and for all of Mh's requests the corresponding Ack has been received. Notice that since Mh will always use *respMss* as the gateway for new requests, it will never happen that a new request is sent to a Mss which is not hosting a Mh's proxy.

3.4 Examples

In this section we present two examples to illustrate how RDP works.

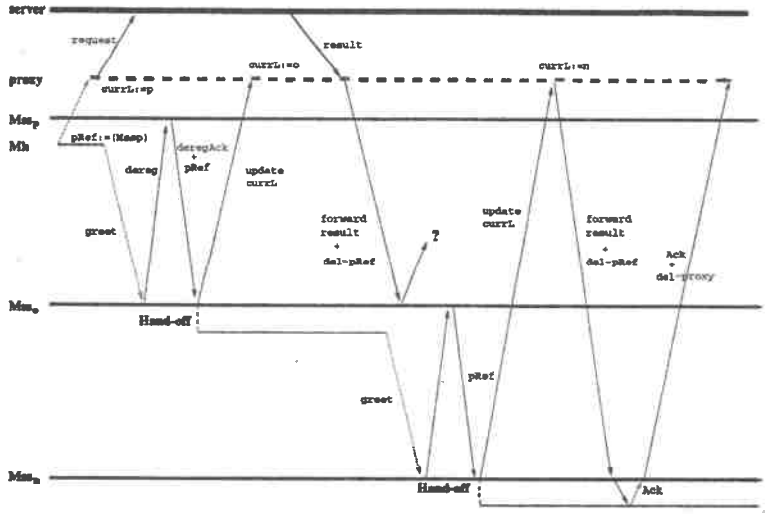

Figure 2: Example of a single request

Figure 2 shows a scenario where a mobile host Mh issues a single request to a server at Mss_p, then migrates to Mss_o and later to Mss_n. Assuming that this is Mh's first request, its pRef will contain no address. Hence a new proxy is created at Mss_p with its variable currentLoc (abbreviated by currL) set to Mss_p and also the address in pRef is set to Mss_p.

Now each time the Mh migrates (and after the Hand-off protocol is completed) the new Mss sends an update currL message to the proxy in order to update its variable currL.

When the response of the request (`result`) arrives at the proxy, Mss_p forwards it to the *Mss* metioned in `currL`, even if in the meantime *Mh* has migrated, as suggested by the qusetion mark in the figure. Piggy-backed on the `result` message goes `del-pRef`= *true*, because proxy's `requestList` has a single entry. Because *Mh* has meanwhile migrated to another cell, the proxy does not receive an `Ack` from Mss_o, hence will repeat the forwarding every time its variable `currL` is updated, until it finally receives an `Ack` from the *Mh*.

When Mss_n receives the `result` with `del-pRef`= *true*, it sets the flag `RKpR`= *true* at *Mh*'s `pRef`, forwards `result` to *Mh* and waits for an `Ack` from *Mh*. As soon as this message arrives, and since still `RKpR`= *true*, Mss_n erases the address at *Mh*'s `pRef` and also sends Mss_p message `Ack` piggy-backed with flag `del-proxy`= *true* to Mss_p. Finally, when this message arrives at Mss_p the proxy is deleted.

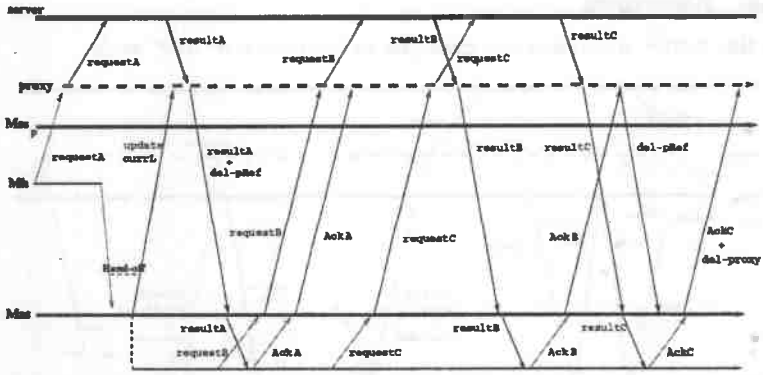

Figure 3: Example of Multiple Requests

Figure 3 depicts a scenario where a *Mh* issues several requests through its proxy, showing when the proxy is actually deleted. In this scenario, the mobile host *Mh* first issues `requestA` at Mss_p (creating a new proxy), and then migrates to *Mss*. Thus, initially the proxy's `requestList` contains only `requestA` and when `resultA` arrives from the server, Mss_p forwards `resultA` with `del-pRef`= *true* to *Mss*. When this message arrives, *Mss* sets flag `RKpR`=true at *Mh*'s `pRef`, and forwards `resultA` to *Mh*. However, since *Mh* issues a new `requestB` before sending an `Ack` for `resultA` (`AckA`) flag `RKpR` is set to false. Since now *Mss* receives `AckA` when `RKpR`= *false*, it leaves `pRef`'s address unchanged.

After receiving `requestB` followed by `AckA`, the proxy's `requestList` ends up holding only `requestB`, and `resultB` is forwarded to *Mss*.

Assume that meanwhile a new `requestC` arrives at the proxy, causing its `requestList` to hold both `requestB` and `requestC`. When `AckB` is received by the proxy, and `resultC` is received and forwarded, `requestB` can be removed from `requestList`, leaving only the single pending `requestC`.

In this particular case, the proxy does not forward again `resultB` (which has already been sent), but sends a special message containing only `del-pRef`= *true* is sent to Mss, where it causes flag `RKpR` at Mh's `pRef` to be set to *true*. Finally, when `AckC` arrives from Mh, `pRef`'s address is set to null, and message `AckC` with `del-proxy`= *true* is sent to proxy, which causes its deletion. This means that at the next time Mh issues a new request, a new proxy will be created at its current *respMss*.

However, suppose that the last `del-pRef` message had arrived at Mss after `AckC`. Since `RKpR`= *false*, `pRef` would be left unchanged and `AckC` would be sent to Mss_p with `del-proxy`= *false*, avoiding the removal of the proxy.

4 Related Work

In our work, we adopt the *indirect model* for client-server interactions originally proposed by Badrinath et al.[2] and now adopted also in many other works[12, 3, 7]. In this model mobility is made explicit to all protocol layers (up to the application layer) and the Mss's serve as static intermediates for any communication between the mobile clients and the servers executing on the network. Compared to other approaches that make mobility explicit only to the protocol layers up to the network layer (e.g. Mobile IP and its optimizations[9]), the indirect model has as its main advantage the possibility to build mobility-aware higher-level protocols which use locality information to dynamically adapt themselves to variant transmission and/or network access conditions, such as communication bandwidth of wireless media, closest server, etc. Moreover, this model has the advantage of allowing the Mss's to perform translations between the wired and wireless medium, and also to store application-specific data related to the state of the 'local' Mhs.

Although RDP is not a generic network protocol (since it is suited only for connectionless request-reply protocols, rather than for connection-oriented transport protocols), it is useful to compare it with Mobile IP due to some common characteristics. In both protocols messages (or datagrams) are delivered to mobile hosts via intermediate elements. In Mobile IP they are called *home* and *foreign agent* and in RDP these are the *proxy* and the *respMss*.

The main differences between the protocols are the following: In Mobile IP the *home agent* is fixed rather than dynamic, making dynamic load balancing impossible.

Moreover, Mobile IP does not guarantee reliable data delivery to the Mhs. For example, IP datagrams may be lost while a new *care-of address* change is on its way to the *home agent*, or during the periods of inactivity of the mobile host. In fact, Mobile IP delegates the duty for detecting and re-transmitting lost datagrams to upper network layers, such as TCP. However, it has been shown that conventional transport-level protocols without mobility-awareness present bad performance when used in a wireless environment[4]. In RDP, on the other hand, Hand-Off is tightly integrated with the message delivery, ensuring that no messages (request results) are lost despite Mh migration or inactivity.

Many other works have investigated the issue of database and network service access by mobile clients: In Project Dataman, Imielinski and Badrinath [8] proposed several query and update strategies for distributed information services that store frequently modified data, such as the geographic location of individual mobile hosts.

Pitoura et al. [10] describe a general architecture of an information system for mobile environment, and in [11] propose support for transactions management for wireless environments. Although recently much attention has been given to the problems of database transaction management[5] with disconnection requirements, this issue is not addressed by our work.

In Project Wireless-View [13] (U.Illinois-Chicago) several data allocation schemes are studied and specific cost models for access to continuously changing databases from clients at mobile hosts were proposed.

5 Protocol Analysis and Conclusion

The protocol presented in this paper guarantees delivery (of the result) with at *least-one semantics*, since the proxy keeps re-transmitting the result until an `Ack` message arrives, signaling that the Mh has indeed received the response. However, a Mh that becomes inactive right after reception of the `result` message (but does not send an `Ack` message) will receive this message again when it becomes active again, either in the previous or in a new cell.

If the Mh already sent an `Ack` to its *respMss* and if wired communication guarantees delivery of messages in causal order, then the protocol ensures delivery of messages with *exactly once semantics*. This is because each Mss is supposed to handle `Ack` forwards with highest priority, and therefore the following sequence of causal dependencies among events (where $send(m)@S$ denotes the sending of message m from server S) always holds:
$send(\texttt{Ack})@Mss_o \rightarrow send(\texttt{Ack+del-proxy})@Mss_o \rightarrow send(\texttt{update currL})@Mss_n$
Hence, the proxy will receive the `Ack` message (forwarded by Mss_o) before the message `update currL`, from Mss_n, and will thus remove the proxy instead of re-transmitting `result`.

In any case, delivery of redundant messages is not a major problem, since it can be assumed that the Mh is able to identify duplicated messages.

If the wireless communication is reliable, re-transmissions of the result with the Result Delivery Protocol (RDP) occur only if the mean time period a Mh spends in a cell is less than: $4*\Delta_{wired}+3*\Delta_{wireless}$, where Δ_{wired} and $\Delta_{wireless}$ are the average transmission times for the wired and wireless communication, respectively. This is unlikely to be the case for current mobile support systems where the diameter of the cells is of reasonable size, e.g. some kilometers.

The major advantage is that except for the proxy reference, neither result forwarding pointers nor other residue (e.g. copies of the `result` message) need to be kept at the Mss, i.e. the Mss can discard the result message after a single attempt to forward it to the local Mh[3].

[3]In fact, if the Mss is able to detect that the target Mh is currently inactive, it may keep

The overhead of this protocol is limited to the following extra messages: (1) one `update currL` whenever the mobile host migrates or becomes active again; and (2) one extra `Ack` message sent from *respMss* to the proxy whenever Mh acknowledges the receipt of `result`. Besides, every request from the mobile host to an application server has to pass through the proxy, so that the proxy can hold it as pending.

Our work extends the indirect model of Badrinath et al.[2] in the sense that the proxy represents yet another communication mediator between the mobile client and the server. The main advantage of using a proxy is that from the server's point of view, the service is being requested from a fixed client (i.e. the proxy), which transparently handles all the mobility of the client it represents. Compared with similar approaches[3, 1] our protocol aims at minimizing the transfer of a Mh's state between the old and new Mss during Hand-off, because most of the data related to the request (e.g. the result) is kept at the proxy.

The Result Delivery Protocol is being implemented as a prototype system in the scope of the SIDAM Project. In this prototype, we are simulating host mobility through random communication between distributed processes (representing Mhs, Mss and *servers*) within a Linux network. Using this prototype, we will test this protocol concerning its efficiency with respect to several patterns of mobility, queries and subscriptions. At a future step, we intend to re-implement it in a network with real wireless communication support.

References

[1] A. Acharya and B.R. Badrinath. Delivering Multicast Messages in Networks with Mobile Hosts. In *Proc. of 13th Intl. Conference on Distributed Computing Systems, Pittsburgh.* IEEE Computer Society, May 1993.

[2] B.R. Badrinath, A. Bakre, T. Imielinski, and R. Marantz. Handling Mobile Clients: A Case for Indirect Interaction. In *Proc. 4th Workstation Operating Systems*, October 1993.

[3] A. Bakre. *Design and Implementation of Indirect Protocols for Mobile Wireless Environments.* PhD thesis, Rutgers University, October 1996.

[4] S. Biaz and N.H. Vaidya. Distinguishing Congestion Losses from Wireless Transmission Losses. In *Proc. of Int. Conference on Computer Communications and Networks*, 1998.

[5] M.H. Dunham and V. Kumar. Impact of Mobility on Transaction Management. In *Proc. of MobiDE/Mobicom99 Workshop, Seattle, WA*, pages 14–21. ACM, August 1999.

the message, and eventually save the re-transmission by the proxy when the Mh becomes active again.

[6] Markus Endler. A protocol for atomic multicast among mobile hosts. In *Proc. of the Workshop DialM/Mobicom'99, Seatle (USA)*, pages 56–71. ACM, August 1999.

[7] M. Haar, R. Cunningham, and V. Cahill. Supporting corba applications in a mobile environment. In *Proc. of MobiCom99, Settle, WA*, pages 36–47. ACM, August 1999.

[8] T. Imielinski and B. R. Badrinath. Querying in highly mobile distributed environments. In *Proc. of the 18th VLDB Conference*, 1992.

[9] D.B. Johnson and D.A. Maltz. Protocols for adaptive wireless and mobile networking. *IEEE Personal Communications*, 3(1), February 1996.

[10] E. Pintoura and B. Bhargava. Building information systems for mobile environments. In *Proc. 3rd International Conference on Information and Knowledge Management*, pages 371–378, 1994.

[11] E. Pintoura and B. Bhargava. Revising transaction concepts for mobile computing. In *Proc. IEEE Workshop on Mobile Computing Systems and Applications*, pages 164–168, 1994.

[12] E. A. Brewer R. H. Katz. The Case for Wireless Overlay Networks. In *Proc. of the SPIE Conference on Multimedia and Networking, (MMCM '96), San Jose, CA*, January 1996.

[13] O. Wolfson Y. Huang. Allocation in Distributed Databases and Mobile Computers. *IEEE Transactions on Parallel and Distributed Systems*, 9(4), April 1998.

RELATÓRIOS TÉCNICOS

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Instituto de Matemática e Estatística da USP

A listagem contendo os relatórios técnicos anteriores a 1996 poderá ser consultada ou solicitada à Secretaria do Departamento, pessoalmente, por carta ou e-mail(mac@ime.usp.br).

Daniela V. Carbogim and Flávio S. Corrêa da Silva
FACTS, ANNOTATIONS, ARGUMENTS AND REASONING
RT-MAC-9601, janeiro de 1996, 22 pp.

Kunio Okuda
REDUÇÃO DE DEPENDÊNCIA PARCIAL E REDUÇÃO DE DEPENDÊNCIA GENERALIZADA
RT-MAC-9602, fevereiro de 1996, 20 pp.

Junior Barrera, Edward R. Dougherty and Nina Sumiko Tomita
AUTOMATIC PROGRAMMING OF BINARY MORPHOLOGICAL MACHINES BY DESIGN OF STATISTICALLY OPTIMAL OPERATORS IN THE CONTEXT OF COMPUTATIONAL LEARNING THEORY.
RT-MAC-9603, abril de 1996, 48 pp.

Junior Barrera e Guillermo Pablo Salas
SET OPERATIONS ON CLOSED INTERVALS AND THEIR APPLICATIONS TO THE AUTOMATIC PROGRAMMINIG OF MMACH'S
RT-MAC-9604, abril de 1995, 66 pp.

Kunio Okuda
CYCLE SHRINKING BY DEPENDENCE REDUCTION
RT-MAC-9605, maio de 1996, 25 pp.

Julio Stern, Fabio Nakano e Marcelo Lauretto
REAL: REAL ATTRIBUTE LEARNING FOR STRATEGIC MARKET OPERATION
RT-MAC-9606, agosto de 1996, 16 pp.

Markus Endler
SISTEMAS OPERACIONAIS DISTRIBUÍDOS: CONCEITOS, EXEMPLOS E TENDÊNCIAS
RT-MAC-9607, agosto de 1996, 120 pp.

Hae Yong Kim
CONSTRUÇÃO RÁPIDA E AUTOMÁTICA DE OPERADORES MORFOLÓGICOS E EFICIENTES PELA APRENDIZAGEM COMPUTACIONAL
RT-MAC-9608, outubro de 1996, 19 pp.

Marcelo Finger
NOTES ON COMPLEX COMBINATORS AND STRUCTURALLY-FREE THEOREM PROVING
RT-MAC-9609, dezembro 1996, 28 pp.

Carlos Eduardo Ferreira, Flávio Keidi Miyazawa e Yoshiko Wakabayashi (eds)
ANAIS DA I OFICINA NACIONAL EM PROBLEMAS DE CORTE E EMPACOTAMENTO
RT-MAC-9610, dezembro de 1996, 65 pp.

Carlos Eduardo Ferreira, C. C. de Souza e Yoshiko Wakabayashi
REARRANGEMENT OF DNA FRAGMENTS: A BRANCH-AND-CUT ALGORITHM
RT-MAC-9701, janeiro de 1997, 24 pp.

Marcelo Finger
NOTES ON THE LOGICAL RECONSTRUCTION OF TEMPORAL DATABASES
RT-MAC-9702, março de 1997, 36 pp.

Flávio S. Corrêa da Silva, Wamberto W. Vasconcelos e David Robertson
COOPERATION BETWEEN KNOWLEDGE BASED SYSTEMS
RT-MAC-9703, abril de 1997, 18 pp.

Junior Barrera, Gerald Jean Francis Banon, Roberto de Alencar Lotufo, Roberto Hirata Junior
MMACH: A MATHEMATICAL MORPHOLOGY TOOLBOX FOR THE KHOROS SYSTEM
RT-MAC-9704, maio de 1997, 67 pp.

Julio Michael Stern e Cibele Dunder
PORTFÓLIOS EFICIENTES INCLUINDO OPÇÕES
RT-MAC-9705, maio de 1997, 29 pp.

Junior Barrera e Ronaldo Fumio Hashimoto
COMPACT REPRESENTATION OF W-OPERATORS
RT-MAC-9706, julho de 1997, 13 pp.

Dilma M. Silva e Markus Endler
CONFIGURAÇÃO DINÂMICA DE SISTEMAS
RT-MAC-9707, agosto de 1997, 35 pp

Kenji Koyama e Routo Terada
AN AUGMENTED FAMILY OF CRYPTOGRAPHIC PARITY CIRCUITS
RT-MAC-9708, setembro de 1997, 15 pp

Routo Terada e Jorge Nakahara Jr.
LINEAR AND DIFFERENTIAL CRYPTANALYSIS OF FEAL-N WITH SWAPPING
RT-MAC-9709, setembro de 1997, 16 pp

Flávio S. Corrêa da Silva e Yara M. Michelacci
MAKING OF AN INTELLIGENT TUTORING SYSTEM (OR METHODOLOGICAL ISSUES OF ARTIFICIAL INTELLIGENCE RESEARCH BY EXAMPLE)
RT-MAC-9710, outubro de 1997, 16 pp.

Marcelo Finger
COMPUTING LIST COMBINATOR SOLUTIONS FOR STRUCTURAL EQUATIONS
RT-MAC-9711, outubro de 1997, 22 pp.

Maria Angela Gurgel and E.M.Rodrigues
THE F-FACTOR PROBLEM
RT-MAC-9712, dezembro de 1997, 22 pp.

Perry R. James, Markus Endler, Marie-Claude Gaudel
DEVELOPMENT OF AN ATOMIC-BROADCAST PROTOCOL USING LOTOS
RT-MAC-9713, dezembro de 1997, 27 pp.

Carlos Eduardo Ferreira and Marko Loparic
A BRANCH-AND-CUT ALGORITHM FOR A VEHICLE ROUTING PROBLEM WITH CAPACITY AND TIME CONSTRAINTS
RT-MAC-9714, dezembro de 1997, 20 pp.

Nami Kobayashi
A HIERARCHY FOR THE RECOGNIZABLE M-SUBSETS
RT-MAC-9715, dezembro de 1997, 47 pp.

Flávio Soares Corrêa da Silva e Daniela Vasconcelos Carbogim
A TWO-SORTED INTERPRETATION FOR ANNOTATED LOGIC
RT-MAC-9801, fevereiro de 1998, 17 pp.

Flávio Soares Corrêa da Silva, Wamberto Weber Vasconcelos, Jaume Agustí, David Robertson e Ana Cristina V. de Melo.
WHY ONTOLOGIES ARE NOT ENOUGH FOR KNOWLEDGE SHARING
RT-MAC-9802, outubro de 1998, 15 pp.

J. C.de Pina e J. Soares
ON THE INTEGER CONE OF THE BASES OF A MATROID
RT-MAC-9803, novembro de 1998, 16 pp.

K. Okuda and S.W.Song
REVISITING HAMILTONIAM DECOMPOSITION OF THE HYPERCUBE
RT-MAC-9804, dezembro 1998, 17pp.

Markus Endler
AGENTES MÓVEIS: UM TUTORIAL
RT-MAC-9805, dezembro 1998, 19pp.

Carlos Alberto de Bragança Pereira, Fabio Nakano e Julio Michael Stern
A DYNAMIC SOFTWARE CERTIFICATION AN VERIFICATION PROCEDURE
RT-MAC-9901, março 1999, 21pp.

Carlos E. Ferreira e Dilma M. Silva
BCC DA USP: UM NOVO CURSO PARA OS DESAFIOS DO NOVO MILÊNIO
RT-MAC-9902, abril 1999, 12pp.

Ronaldo Fumio Hashimoto and Junior Barrera
A SIMPLE ALGORITHM FOR DECOMPOSING CONVEX STRUCTURING ELEMENTS
RT-MAC-9903, abril 1999, 24 pp.

Jorge Euler, Maria do Carmo Noronha e Dilma Menezes da Silva
ESTUDO DE CASO: DESEMPENHO DEFICIENTE DO SISTEMA OPERACIONAL LINUX PARA CARGA MISTA DE APLICAÇÕES.
RT-MAC-9904, maio 1999, 27 pp.

Carlos Humes Junior e Paulo José da Silva e Silva
AN INEXACT CLASSICAL PROXIMAL POINT ALGORITHM VIEWED AS DESCENT METHOD IN THE OPTIMIZATION CASE
RT-MAC-9905, maio 1999, pp.

Carlos Humes Junior and Paulo José da Silva e Silva
STRICT CONVEX REGULARIZATIONS, PROXIMAL POINTS AND AUGMENTED LAGRANGIANS
RT-MAC-9906, maio 1999, 21 pp.

Ronaldo Fumio Hashimoto, Junior Barrera, Carlos Eduardo Ferreira
A COMBINATORIAL OPTIMIZATION TECHNIQUE FOR THE SEQUENTIAL DECOMPOSITION OF EROSIONS AND DILATIONS
RT-MAC-9907, maio 1999, 30 pp.

Carlos Humes Junior and Marcelo Queiroz
ON THE PROJECTED PAIRWISE MULTICOMMODITY FLOW POLYHEDRON
RT-MAC-9908, maio 1999, 18 pp.

Carlos Humes Junior and Marcelo Queiroz
TWO HEURISTICS FOR THE CONTINUOUS CAPACITY AND FLOW ASSIGNMENT GLOBAL OPTIMIZATION
RT-MAC-9909, maio 1999, 32 pp.

Carlos Humes Junior and Paulo José da Silva e Silva
AN INEXACT CLASSICAL PROXIMAL POINT ALGORITHM VIEWED AS A DESCENT METHOD IN THE OPTIMIZATION CASE
RT-MAC-9910, julho 1999, 13 pp.

Markus Endler and Dilma M. Silva and Kunio Okuda
A RELIABLE CONNECTIONLESS PROTOCOL FOR MOBILE CLIENTS
RT-MAC-9911, setembro 1999, 17 pp.