Protocols for maintaining inventory databases and user service profiles in mobile sales applications*
(Extended abstract)

Narayanan Krishnakumar and Ravi Jain
Bell Communications Research
445 South Street
Morristown, NJ 07960
email: {nkk@marble,rjain@thumper}.bellcore.com

Abstract

An important and challenging set of issues in mobile computing is the design of architectures and protocols for providing mobile users with integrated Personal Information Services and Applications (PISA), such as personalized news and financial information, and mobile sales and banking. We present a system architecture for delivery of PISA based on replicated distributed servers connected to users via a personal communications services (PCS) network. The PISA architecture takes advantage of many of the basic facilities incorporated in proposed PCS network designs, and is analogous to it.

We focus on the mobile sales and inventory application as an example of a PISA with a well-defined market segment. We describe a database design which supports both mobile and stationary salespersons. A key principle behind our design choices is to minimize those aspects which are needed solely to support user mobility. We propose using a site escrow method and redistribution protocol for supporting sales transactions, and discuss how mobile salespersons can be accommodated in this scheme. Finally we discuss how the service profiles of the mobile salespersons can be maintained using a two-level hierarchy of profile databases, show that a protocol more robust than that used for maintaining their location is needed, and present such a protocol.

1 Introduction

An important and challenging area of mobile information systems is the design of architectures and protocols for providing mobile users with Personal Information Services and Applications (PISA). Examples of PISA include personalized financial and stock market information, electronic magazines, news clipping services, traveler information, as well as mobile shopping, banking, sales, inventory, and file access. As a concrete running example in this paper, we use the mobile sales and inventory application. In sec. 2 we describe this application in more detail.

We consider the situation in which PISA are primarily provided by a commercial entity called the Information Service and Applications Provider (ISAP). The ISAP maintains a set of servers which contain the appropriate information and run applications, and which are connected to the mobile user via a personal communications services (PCS) network. The ISAP need not be the same commercial and administrative entity as the PCS network provider.

The mobile user's terminal runs application software to interact with the ISAP. These interactions are divided into logical, application-dependent segments called *sessions*. For example, in the case of

*A portion of this research has been submitted for publication in a journal, and has appeared in very preliminary form, in the *IEEE Conference on Networks for Personal Communications* (NPC '94), Long Branch, NJ, Mar. 1994.

the mobile sales and inventory application, a session may consist of the salesperson connecting to a remote server, downloading images and text describing a product, and then logging out. Sessions may be initiated by the user or by the ISAP. The salesperson may be an employee of the ISAP, or the ISAP may be a third party providing integrated sales and inventory information (e.g. a real estate multiple listing service). It is desirable that when a session is in progress, the user is not aware of any disruption in service as the user moves.

In order to meet reliability, performance and cost objectives when the number of users is large and geographically dispersed, a distributed server architecture will be necessary. In sec. 3 we describe the system architecture for mobile sales and inventory applications where the ISAP has a distributed replicated database architecture and uses the underlying PCS network for communicating with the user. As the user moves or network load and availability changes, the server interacting with the user may need to change. Thus, real mobility on the part of the user may result in the *virtual mobility* of the server. This is accomplished by means of a *service handoff*, which is broadly analogous to a PCS call handoff or user location update (i.e., registration/deregistration) procedure [1, 16], but relatively less frequent. We have previously designed a service handoff protocol [10, 11] and described the context information which must be transferred from the old to the new server for various classes of applications, including mobile transactions.

In sec. 4 we describe how the semantics of the sales application can be exploited to provide an appropriate database design, and provide a framework for running mobile transactions in sec. 5. Just as for the underlying PCS network, the ISAP will need to maintain user service profiles to determine what services the user needs and is authorized to obtain. A PCS network typically uses a two-level hierarchy of databases (Home Location Register and Visitor Location Register - HLR and VLR) to maintain this information, and a user location update protocol (e.g. IS-41 [1]). For the same reason as for the PCS network, we propose in sec. 6 that the ISAP also use a two-level database hierarchy for service profiles, and discuss scenarios in which the need arises for an access protocol more robust than those (e.g. IS-41 [1]) used for managing user location information in the PCS network. We present a protocol for managing the service profile databases. We end with some concluding remarks in sec. 7.

2 Application scenario: Mobile sales and inventory

We consider a scenario in which the user is a mobile salesperson selling financial products (like insurance, bonds, etc.), or consumable products (e.g. doctor's office supplies like gloves, syringes, etc.). The ISAP is either the company the salesperson works for, whose products are being sold, or a third-party supplier of information. The salesperson uses a personal digital assistant (PDA) as a mobile database when discussing and completing sales. To avoid confusion, we will adopt the following terminology. The salesperson is also referred to as the user of the ISAP's services. The PDA or other end equipment which the salesperson uses is called the mobile or the client of the ISAP's servers. The person to whom the sale is being made is referred to as the customer.

The user visits numerous customer offices during the course of a day and discusses their requirements, shows images of products, initiates new orders or queries about the status of previous orders, etc., using the PDA. The PDA is capable of doing these functions using either a wireless or a wireline link to the ISAP's servers. The mobile database contains customer records as well as information regarding policies, prices and availability of the product being sold. The time available to the salesperson for meeting with the customer may be extremely limited [19]. In order to ensure that this time is utilized most effectively, the mobile database must have current information available about dynamically varying quantities such as inventory levels, delivery dates, etc. The mobile database is periodically

updated from the ISAP's database. The user has a service profile stored with the ISAP which ensures that the mobile database is updated at appropriate times with the appropriate information. Orders or queries placed by the salesperson are transmitted to the ISAP database.

Observe that this is not a far-fetched scenario. Mobile sales applications are already being tested and marketed [20, 19]. Moreover, if current market projections are realized, this scenario will not be rare in the future. PDAs are forecasted to become a commonplace business accessory, with sales of PDAs exceeding 3.6 million units by 1997 [12].

3 System architecture

In general, mobile users will desire access to private and corporate databases which, for reliability, performance and cost reasons will necessitate a *distributed server* architecture when the number of users and their geographic dispersion becomes large. In a distributed server architecture the information is (partially) replicated across multiple interconnected servers but the system functions as a single logical information base.

There are several possible ways of interconnecting the servers, e.g. using a private ISAP network attached to the PCS network via a gateway, or using the PCS network itself as the inter-server communication backbone. The geographical coverage area for the information service is partitioned into *service areas*, analogous to PCS registration areas. It is likely that a service area will cover several PCS registration areas. Each service area is served by a single information server, called the *local server*, analogous to the PCS network's Mobile Switching Center (MSC) or VLR database. The connection between the ISAP and the mobile user can be set up by either side dialing the other's non-geographic telephone number.

The most basic support[1] required by the ISAP from the PCS network is that the physical connection between the user and the ISAP be maintained without interruption during a session as the user moves. Two key functions needed for this support are to *locate the mobile user* and to *perform a physical connection transfer as the user moves*. Protocols for performing the physical connection transfer function in a store-and-forward-packet-switched network have been proposed by Keeton et al. [13]. However, we note that both functions above are already provided by the user location facilities and call handoff mechanisms specified in PCS standards. Thus we will assume that the following levels of protocols are already provided by the PCS network:

1. A call handoff protocol, similar to that specified in Bellcore's WACS [3] or GSM standards [16] for physical connection transfer of the wireless link when a mobile client moves from one cell to another.

2. A user location protocol, similar to that specified in the IS-41 [1] or GSM [16] standard, for registering a mobile client in a registration area and for locating and delivering calls to the client when it moves between registration areas.

We also assume that the application is running a link-level protocol which recovers from bit errors, as well as packet losses, duplication and reordering, for both wireless and wired links. (Examples of such protocols include LAPR for wireless links and LAPM for wireline voiceband modems; see [18]).

As a mobile user moves from cell to cell but within the same service area (so that the user is in contact with the same server during the move), the PCS network can perform a physical connection transfer, i.e., keep the connection continuous with the same server, using the usual PCS call handoff procedure. The call handoff may result in errors at the physical layer of the connection, e.g. bit errors or packets being dropped, which can be recovered from by using the link level protocol.

[1] The PCS network can also provide additional services such as billing etc, which are outside the scope of this paper.

As the user moves out of one service area into another, it is desirable that the local server at the new service area take over providing the service. This *service handoff* for the *virtual mobility* of the server is broadly analogous to the PCS call handoff procedure (except that it occurs between ISAP servers rather than PCS base stations), and also has the requirement that service appear to continue transparently without interruption.[2] In [10, 11], we have described protocols and capabilities required in both the ISAP and the PCS network to implement service handoffs. Briefly, a service handoff consists of a transfer of context information from the old server to the new server, followed by a physical connection transfer between the old and new servers. The context information depends upon the application in progress, but essentially informs the new server of where to pick up the session after the old server left off. For example, if the mobile user was simply reading through a file, the context information would be the name of the file and a pointer (e.g. line number or byte position) from where the new server should start sending information to the mobile.

The service handoff is initiated by an ISAP process called the *matchmaker*, which is responsible for mapping users to appropriate servers, and for setting up initially and managing the connection between the user and servers of the ISAP. (The term "matchmaker", and some of its functionality, has been borrowed from [22]. However our notion of service handoffs, the protocols we have developed and the applications we consider are quite different [10, 11].) The matchmaker can be implemented in a centralized or a distributed manner across several ISAP servers.

4 Database system design

In this section we describe the design of the ISAP's database. The design choices are motivated by the need to accommodate both stationary and mobile users. A principal aim of the design is to minimize the aspects of the database which are specific to mobile users. In this section we describe the design, and in the following section describe the few special operations which need to take place to handle user mobility.

The use of a replicated database entails a concurrency and replica control protocol for ensuring correctness. The correctness criterion commonly used by concurrency control techniques in replicated databases is the notion of *one-copy serializability* [5] of transactions. Since this criterion can be quite restrictive, recent approaches in the literature have taken into account the semantics of the application to relax this criterion and improve transaction throughput. In this paper, we take a similar approach not only because it improves throughput but also because, as it turns out, it accommodates mobile users easily.

Sales and inventory applications, in particular, are amenable to our approach. Consider the situation where the salesperson is selling items from inventory, where each instance of the item is indistinguishable from the others (e.g. the item is a medical supply item, and each instance is, say, one box of the item). For ease of exposition, consider a single item, i, and let $Total_i$ be the total number of instances of that item in stock. As salespersons make sales of that item, the problem is to ensure that the total number of items they have sold for immediate delivery, $Sales_i$, satisfies the constraint $Sales_i \leq Total_i$ at all times. As sales orders are taken or canceled, salespersons launch transactions which update the number of instances sold, $Sales_i$. These updates will typically be made as operational updates instead of value updates, i.e., instead of reading and writing the actual value of the variable $Sales_i$, transactions will issue operations to increment or decrement it. (Operational updates can be preferred over value updates for a number of reasons relating to concurrency control, and rollback and recovery of aborted transactions [5]).

If two salespersons launch long-running transac-

[2] Note that virtual mobility differs from *service mobility* [2], which is the ability of a user to have a consistent set of services even though the user may move.

tions which contain update operations, a traditional concurrency control algorithm would require that the variable $Sales_i$ be locked by each transaction, so that one transaction cannot begin until the other commits and releases the lock. In a replicated system, this further entails using a distributed protocol such as quorum locking [8] and then a two-phase commit to ensure consistency. However, it is possible to exploit the semantics of the sales application in order to improve the system throughput, in particular by the idea of placing instances of the item being sold in *escrow*.

Several escrow schemes have been proposed in the literature including transaction escrow [17], site escrow [15, 21, 4], and generalized site escrow [14]. The transaction escrow scheme was initially proposed for access to hot-spots in a single copy database. In this algorithm, a transaction executes an *escrow* operation to try to place in reserve the resources that it will (potentially) use. All successful escrow operations are logged in an *escrow log*. Before executing an escrow operation, each transaction accesses the log and sees the total escrow quantities of all uncommitted transactions. The transaction then makes a worst-case decision to determine whether it can proceed. For instance, suppose the total number of items in stock is 100, and the number of items sold due to committed transactions is 20. Suppose there are currently two uncommitted transactions each requesting one item. Let transaction T wishing to reserve ten items now be initiated. Since the log indicates that $Sales_i$. is at most 22, T can proceed irrespective of whether the other two transactions commit or abort: the constraint is maintained in any case. Note that when transaction T executes an escrow operation, T (short-term) locks the data item to access and update the log and releases the lock after the log has been updated (the lock release need not wait for the commit or abort of T as in a traditional transaction execution). This feature allows long-running transactions that contain escrow operations to run concurrently so that throughput is increased.

We describe the site escrow method briefly, that involves a distributed environment with each site executing a transaction escrow algorithm. In site escrow algorithms, the total number of instances of a given item available, *Total*, is partitioned across the number of sites (servers) in the system. This can be thought of as each site (as against a transaction) holding a number of instances in escrow. A transaction launched by a user runs at only one site (typically, the one closest to the user). Each transaction at a site uses a transaction escrow scheme to allocate and deallocate resources that are (site-)escrowed at that site. Thus, a transaction can successfully complete only if the number of instances it requires does not exceed the number of instances available in escrow at that site. This scheme results in sites having considerably more autonomy than in the traditional locking scheme, since each site can deplete its own instances without consulting any other site and there is no requirement for a two-phase commit at the end of the transaction. When one site requires more instances, a *redistribution protocol* such as the point-to-point demarcation protocol [4] or a dynamic quorum-based protocol [14] is executed, so that the site can get a portion of some other sites' unassigned instances.

We will assume that the ISAP database uses a site escrow scheme for maintaining inventory information, using suitable algorithms (as in [14, 4]) for maintaining the escrow information.

5 Mobile sales transactions

We now consider how mobile sales transactions can be handled given our system architecture and database design outlined above. It turns out that site escrow methods, such as used in our design, are particularly appropriate for mobility.

Firstly, consider the situation sometimes referred to as *discrete mobility*, where (in our example) a user makes some sales at one service area, completes all transactions, closes the session, disconnects from the PCS network and moves to another service area. The user can now be connected to the local server of the

new service area and make further sales. The sales transactions at the new service area can simply operate on the escrow values stored at the new server. The decision as to whether there is sufficient inventory to make the sales, for instance, can be resolved locally by the new server, or via the existing redistribution protocol.

Now consider the situation in which the user is running a long transaction involving the allocation of multiple inventory items, and the user moves between service areas while the transaction is running. (Henceforth, unless otherwise stated, we deal only with transactions which allocate resources - the typical sales scenario. The discussion below is easily generalized to cases where transactions can have both allocation and deallocation of resources.) In our model, we assume that a service handoff occurs, so that the user starts communicating with the local server of the new service area. The service handoff protocol [10, 11] maintains continuity of the physical communication link between the user and the ISAP while the handoff occurs. However, before the physical connection transfer can actually be carried out, the context of the interactions of the user with the ISAP needs to be transferred from the old to the new server.

Let us study the context information transfer of the handoff in greater detail. Essentially, a transaction involves numerous operational updates. When the user moves to the new service area, the current operation in progress at the old server is completed, and then the context of the transaction is transferred to the new server. (Observe here that the mobile can continue interacting with the old server while the context is being transferred. Only after the context is transferred does the mobile start interacting with the new server.) If a traditional locking scheme had been used for concurrency control, the context information would include the set of locks held by the transaction, along with the transaction id. Additional information relating to replica control would also need to be transferred. For instance, suppose a pessimistic quorum consensus protocol [8] with operational updates is used. In this protocol, it is not necessary that at any point of time, a server i know of the updates done at all the other servers. It is necessary though that when a transaction wishes to execute at server i, server i is brought up to date of all the updates done at the other servers. Thus server, i, before initiating an operation o of a transaction T, locks the data items accessed by o at a set of *quorum* servers. The quorum servers send the committed (timestamped) updates that they know of along with the lock grant response to i. On receiving the lock grant responses, server i merges in timestamp order the committed updates it has obtained from the quorum servers with the set of committed updates it knows about. If the quorum is a majority of servers in the system, one can ensure that at this stage [8], server i is up to date of all the committed updates in the system and also that since the lock for o was acquired, there is no other conflicting operation executing in the system. Based on the updated state, i determines the update, u_T. u_T is then appended to an *intentions list* of uncommitted updates for the transaction T. When T is ready to commit, a two-phase commit is executed across all the quorum servers of its operations, and if the decision is to commit, the intentions list is added to the list of committed updates at i and the quorum servers, otherwise the list is discarded. Thus, if this protocol is used, the context transferred from the old server to the new server would also include (a) the list of updates seen at the old server before the most recent operation in T was executed, (b) the intentions list for T, and (c) the list of quorum servers for each operation in T executed so far. The new server will have to first incorporate the list (a) into its state, set up (b) as an intentions list for T, update the lock table to remember the locks that T has already acquired, and store the list of quorum servers (so that they can participate in a two-phase commit when T is ready to commit). Using site escrow methods makes this context transfer and set-up much easier.

In our design using the site escrow method, decisions in the sales transaction to allocate resources are made locally (as far as possible). So quorum in-

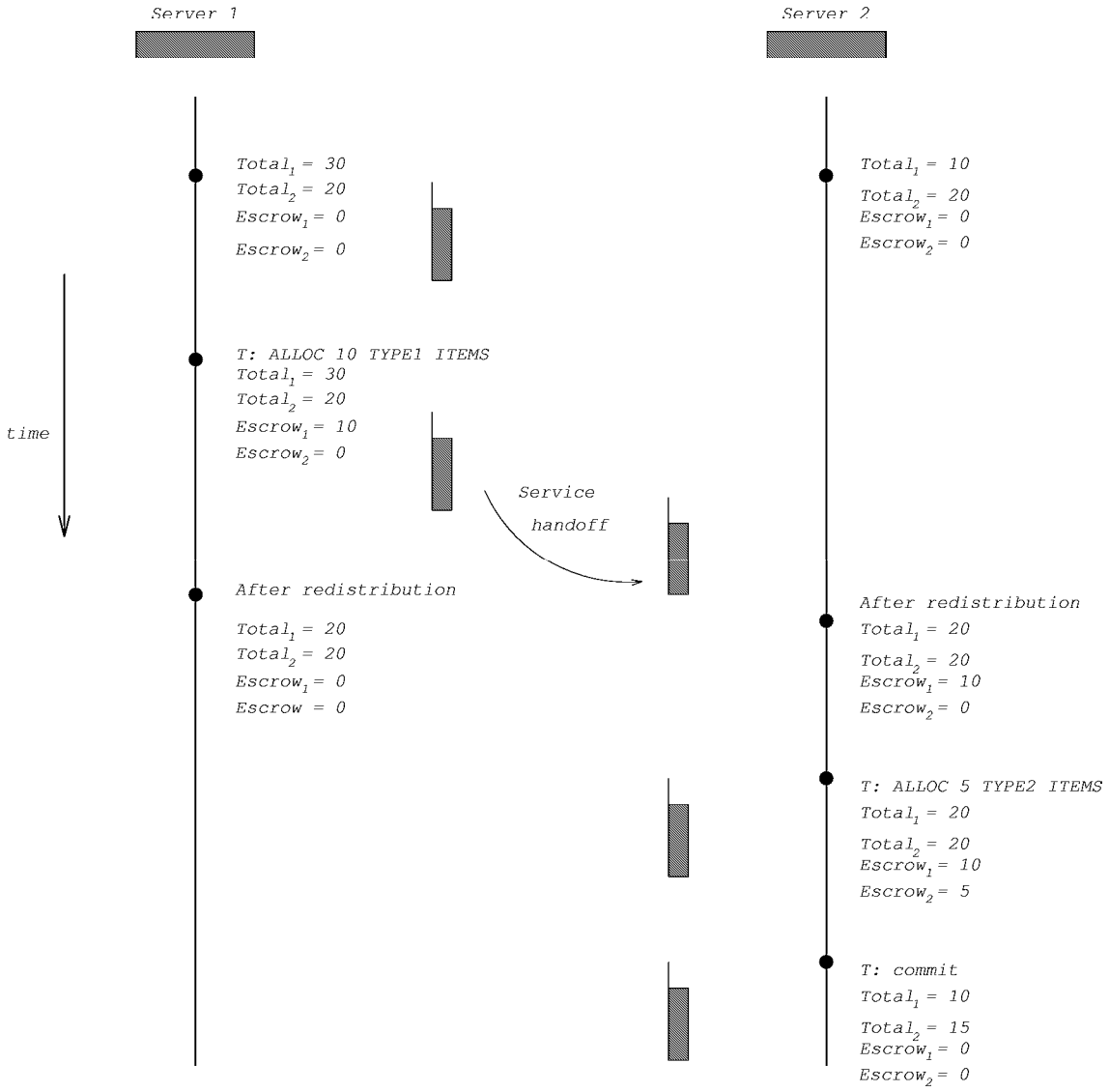

Figure 1: A mobile sales transaction example

formation as described earlier does not have to be maintained or tranferred as context. Furthermore, since sites use transaction escrow locally, locks are released after the completion of each operation. So no lock information needs to be transferred to the new server. In addition, suppose the operations in the sales transaction deal only with escrowable resource types. The new server does not need information about the operations already performed at the old server, since the old server made its decisions based on its locally escrowed resources. Thus it is possible that the context transferred is only the information about which operation in the transaction is to be run next at the new server. However, when the transaction commits, a two-phase commit would also be required between the old server and the new server since part of the transaction has been run at the old server and the rest at the new server. (Note that the two-phase commit might be between several servers if the user moves between several service areas during the duration of the transaction, but for simplicity, we are considering only two servers here.) The benefit of the site escrow idea in the case of a salesperson who does not move is that most of the time, a two-phase commit would not be required *at the end of the transaction* since requests for resources might be satisfied locally. If the salesperson moves around a lot between service areas, then a two-phase commit is almost always required at the end of the transaction, and this is undesirable.

To account for this problem, we associate a redistribution of item instances with a service handoff. Recall that a redistribution protocol is provided as part of the site escrow framework. On a service handoff, a redistribution of instances is performed as follows. The total number of instances at the old server is decremented by the number of instances allocated by the transaction, and the total number of instances at the new server is incremented by the same amount. Furthermore, these newly acquired instances are entered as having been escrowed by the transaction at the new server. Thus, the redistribution protocol is invoked to transfer the required number of instances from the old to the new server. If the transaction commits at the new server, the new server's total is decremented accordingly; if the transaction aborts, the total is not modified. This scheme thereby eliminates the need for the two-phase commit protocol at the end of the transaction, allowing the transaction to perform a simple one-phase commit at its current server. The context information which needs to be transferred during service handoff, with the site escrow method, thus includes only the transaction id and the intentions list already done at the old server. The redistribution protocol can be performed independently of the context information transfer - the only requirement is that the redistribution(s) complete before the transaction commits. The redistribution itself involves a two-phase commit between the old and new servers, but the two-phase commit now is not necessarily at the end of the transaction: we overlap it with the rest of the processing in the transaction.

An example of a mobile sales transaction is shown in Figure 1. There are two types of resource items, and $Total_1$ and $Total_2$ at each server indicates the total number of instances available there. $Escrow_1$ and $Escrow_2$ indicate the number of instances escrowed by uncommitted transactions at each site. The mobile allocates 10 items of type 1 at server 1 and then moves from server 1 to server 2. The redistribution protocol updates $Total_1$ and $Escrow_1$ at both the sites. The mobile submits another operation to allocate 5 items of type 2 as part of the same transaction (this could potentially have been run in parallel with the redistribution), and then commits.

We now discuss disconnections in more detail. Either the mobile could voluntarily disconnect from the ISAP (as in discrete mobility), or the mobile could be disconnected due to a failure in communications or of a server. In the former case, the salesperson could desire to continue selling items while being disconnected. It is then necessary that the salesperson have some idea of how many resource instances of each kind he/she expects to sell. The salesperson can allocate that many resources from the server and "cache" those resources at the mobile before disconnecting. The mobile could thereby continue to sell

items by using transaction escrow on the resources that it had "cached" from the server. At some point, when the mobile reconnects, the resources that are left over could be deallocated. (Business policies would have to determine a maximum on the number of resources that can be cached, since the ISAP could quickly run out of resources even if they were not being sold: all salespersons might cache resources and not be able to sell them.) The second case of disconnection is when the server fails or there is a communications failure. It is then possible that a lock is held by a transaction running on a mobile, or some items have been escrowed at a server on behalf of an uncommitted transaction. Some timeout mechanism would have to be used in either case to abort the transaction and free up the resources. If the redistribution protocol fails during a service handoff, it is retried until the commit point of the transaction or until the timeout on escrowed resources expires at the other server, at which point the transaction is aborted.

Therefore, by using the ideas of escrow, service handoffs and redistribution of resources, we have provided a clean transaction framework for performing mobile sales transactions.

6 Maintaining service profiles

We now discuss the need for user service profiles and how they can be maintained.

Just as the PCS network maintains user profiles for PCS users, to determine what communication services the user needs and is authorized to obtain, it is likely that the ISAP will also need to maintain service profiles i.e., what are the information service requirements and access rights of the user. For instance, in our application, the mobile database contains client records as well as information regarding policies, prices and availability of the product being sold. It is desirable that the mobile database have reasonably current information available about quantities such as inventory levels, etc. The service profile would therefore indicate that the server should initiate a transaction which is to be run on the mobile database at appropriate instances of time, (say hourly or in response to specific market changes), to update the mobile inventory. Such a transaction could be an operational summary of the items consumed at all the sites since the last such update (such as, x instances of item 1 and y of item 2 were consumed, etc.).

In a PCS network, a two-level hierarchy of databases is used to store profiles, with the HLR at one level connected to several VLRs at the lower level. Each VLR serves one or more PCS registration areas (and MSCs). When a user moves between registration areas served by the same VLR, only the VLR needs to be notified that the user has moved. Thus the two-level database scheme reduces the signalling network traffic and database load at the HLR, so helping to prevent the HLR from becoming a performance bottleneck. For the same reasons as for the PCS network, we propose that the ISAP store the service profiles in a similar logical two-level hierarchy, using a Home Service Database (HSD) and Visitor Service Databases (VSD). With each ISAP server is associated a VSD. When the ISAP detects that the user has moved into a service area, the associated VSD is updated with information from the HSD about the user's service profile. The server in the user's current service area can use this profile to determine what interactions are required with the user, and when.

The use of a two-level hierarchy of profile databases entails a protocol for managing them. For instance, when a user moves into a new service area, the new VSD needs to obtain the user's service profile, and it may be necessary to delete the profile stored in the user's old VSD. This is analogous to a PCS location update (registration/deregistration) procedure. Therefore, it would seem that one could use a PCS user location strategy, such as that specified in the IS-41 or GSM standards, for service area registration and deregistration. However, the semantics of most PCS user location protocols (see [9] for a survey) do not ensure that a user is registered in the visited database of *exactly one* registration area

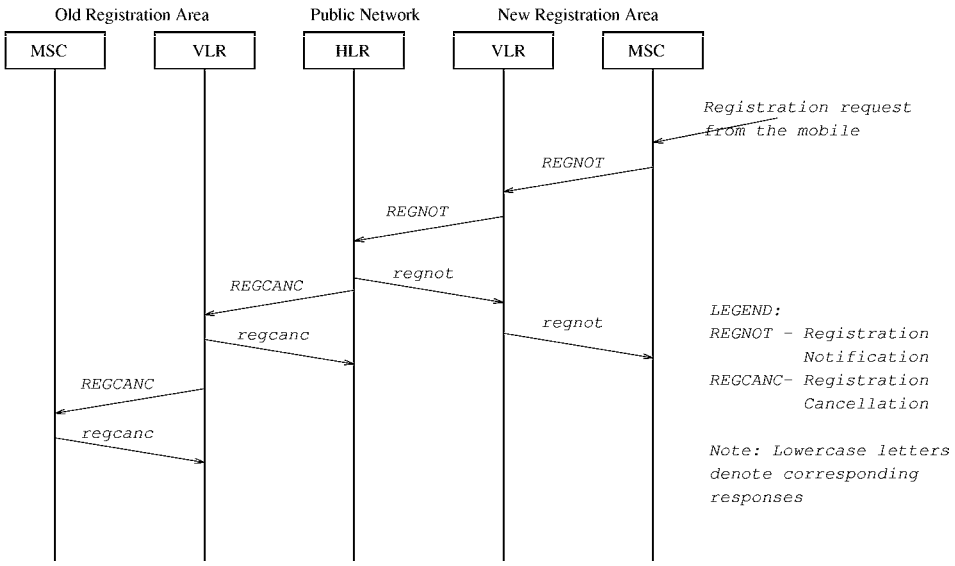

Figure 2: Signalling message flow in the IS-41 protocol

at any given time, which can lead to race conditions.

As an example, consider the location update procedure in the IS-41 protocol [1], restricting attention to the databases involved. Suppose the user moves between regions served by different VLRs (see Fig. 2). The new VLR is notified (by the new MSC) that the user has moved into a registration area served by that VLR. The new VLR sends a registration notification message (abbreviated "REGNOT") to the user's HLR. The HLR is updated to reflect the new VLR as the user's serving VLR. The HLR sends a confirmation message (called "regnot") to the new VLR, and then sends a registration cancellation message ("REGCANC") to the old VLR. The new VLR completes the registration of the user after it receives the "regnot" message. The old VLR deletes the user's registration after it receives the "REGCANC" message, and sends a confirmation message ("regcanc") to the HLR. With this protocol, there exists a race condition between the arrival of the "regnot" message at the new VLR and the arrival of the "REGCANC" message at the old VLR, so it is possible that for some finite time the user is registered in both the VLRs.

The negative effects of race conditions during user location updates are not likely to be serious; for instance, a call to the user might not be completed during the race interval, and the caller might get a busy signal. In contrast, depending upon the application in question, race conditions during service handoffs could be serious. For instance, if a protocol similar to IS-41 is used to update VSDs, it is possible that when a user moves two VSDs contain the user's profile. In our mobile sales application example, the VSD alerts the server, at the time specified in the user's profile, to initiate the appropriate transaction on the client database. If a protocol similar to IS-41 is used to manage the VSDs when a user moves from one service area to another (whether a call is in progress or not), the race condition described above can occur, resulting in both servers running the same update transactions twice and the mobile database having inconsistent information.

We propose one possible solution for preventing race conditions. First, note that the race condition we have described in IS-41 can result in the user being registered in two VLRs, but it can also result in the user being "active" in neither VLR. The latter situation arises because although the new VLR records the user's location before it sends the "REGNOT" registration message to the HLR, it does not consider the user "active" until it has received the

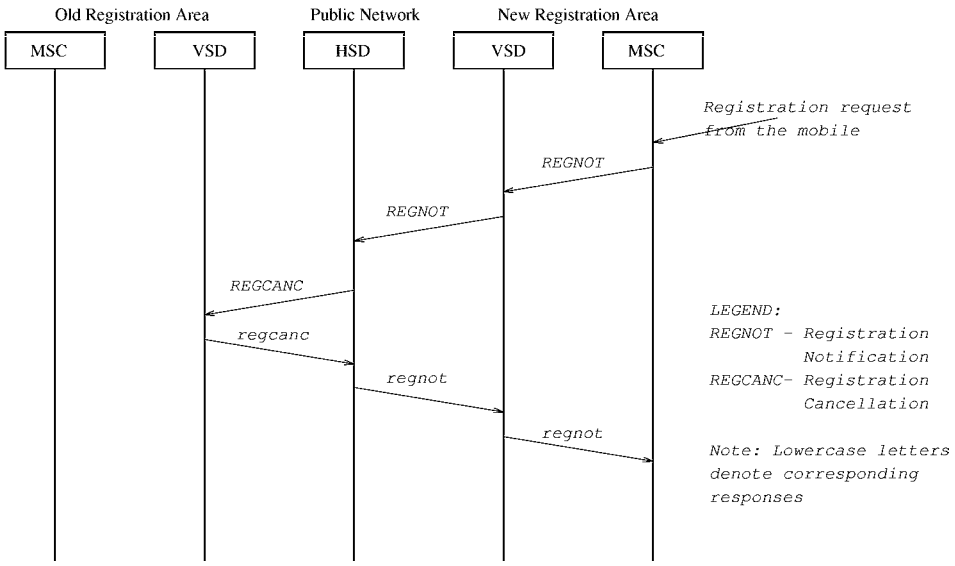

Figure 3: Message flow for a service profile management protocol, based upon a modified version of the IS-41 protocol

"regnot" confirmation message from the HLR. Thus if the "REGCANC" cancellation message arrives at the old VLR before the "regnot" message arrives at the new VLR, the user will be active in neither. In other words, the protocol as it stands allows the user, at different times, to be active in zero, one or two databases.

We propose using a simple modification of the IS-41 protocol for managing service profile databases, as follows (see Fig. 3). Upon receiving a "REGNOT" message from a the new VSD, the HSD first sends a "REGCANC" cancellation message to the old VSD to deactivate the user's profile. Upon receipt of the "REGCANC" message the old VSD stops all further operations related to the user's profile. The HSD then waits until it receives a "regcanc" confirmation message from the old VSD before sending a "regnot" registration confirmation message to the new VSD. This sequence of messages removes one ambiguity in the protocol, by ensuring that the user can never be active in two VSDs. Note that in principle the user's service profile is transferred to the new VSD (either from the old VSD or the HSD, as appropriate) after this activation procedure is complete; in practice, the service profile may arrive at the new VSD along with the "regnot" message.

The modified protocol also has the effect of ensuring that there is a small interval during every service handoff when the user is active in neither VSD. Consider again our example scenario, where some transaction T is initiated based upon the user's profile at a fixed time, say t. Suppose the old VSD receives the "REGCANC" message and deactivates the user at some time $t_o < t$, where t_o is the time according to the local clock at the old VSD. Suppose the user becomes active at the new VSD at time t_n, where t_n is the local time at the new VSD. As it stands, if $t_n > t$, the new VSD will not alert the new server to perform transaction T. To prevent this, the old VSD timestamps the user's profile with time t_o as its last operation upon the profile. (Alternatively, the old VSD can send this information to the new VSD via a separate signalling message.) When the new VSD receives the profile, if $t_n \geq t_o$ the new VSD initiates all transactions which fall in the interval $[t_o, t_n]$, including transaction T. Otherwise $t_n < t_o$ and the new VSD does not initiate any transactions based upon the user's profile until the new VSD's local clock exceeds t_o. This convention ensures that the user being active in neither VSD does not cause some

transactions to be dropped, and also does not require that the clocks of the two VSDs be synchronized.

7 Conclusions

We have presented a distributed replicated server architecture for delivery of PISA, and identified the notion of service handoffs in this architecture. We have presented a database system design, based on a site escrow method, suitable for sales and inventory applications supporting both stationary and mobile users. We have discussed how the site escrow idea provides a simple method for performing service handoffs when a mobile user moves from one service area to another. Finally, we discussed a two-level hierarchy of databases for maintaining service profiles for salespersons, and designed an appropriate update protocol for it. We are currently investigating some of the issues discussed in this paper further. We have previously discussed [10, 11] how mobile transactions which access distinguishable instances, for which site escrow methods are typically not appropriate, can be supported. We are considering the feasibility of combining such transactions with those accessing indistinguishable items.

Acknowledgements

We thank A. Grinberg, D. Hakim, M. Kramer, R. Wolff, and T. Whitaker for their helpful comments on an earlier version of parts of this paper.

References

[1] "Cellular radiotelecommunications intersystem operations, Rev. B", EIA/TIA, July, 1991.

[2] "Feature description and functional analysis of Personal Communications Services (PCS) Capabilities", Bellcore Special Report, SR-TSV-00230, Apr. 1992.

[3] "Generic criteria for Version 0.1 Wireless Access Communications Systems (WACS)", Bellcore Technical Advisory, TA-NWT-001313, Issue 1, July 1992.

[4] D. Barbara and H. Garcia-Molina, "The Demarcation Protocol : A technique for maintaining arithmetic constraints in distributed database systems", Proceedings of International Conference on Extending Data Base Technology, 1992.

[5] P.A. Bernstein, V. Hadzilacos and N. Goodman, "Concurrency Control and Recovery in Database Systems", Addison Wesley Publishing Company, 1987.

[6] D. Ferrari, A. Banerjea and H. Zhang, "Network support for multimedia - a discussion of the Tenet approach", Technical Report TR-92-072, Intl. Comp. Sci. Inst., Berkeley, CA, Nov. 92.

[7] J. Gray and A. Reuter, "Transaction Processing: Concepts and Techniques", Morgan Kaufmann, 1993.

[8] M. P. Herlihy, "Concurrency vs. availability: Atomicity mechanisms for replicated data", ACM TOCS, 5 (3), 249-274, Aug. 1987

[9] R. Jain, "A survey of user location strategies in personal communications services systems", Submitted for publication, 1993.

[10] R. Jain and N. Krishnakumar, "Network Support for Personal Information Services to PCS Users", *IEEE Conf. Networks for Pers. Comm. (NPC)*, Long Branch, NJ, Mar. 1994.

[11] R. Jain and N. Krishnakumar, "Service handoffs and virtual mobility for delivery of personal information services to mobile users", Submitted for publication, 1994.

[12] J. Jerney, "A conversation with Dataquest's Jerry Purdy", Pen-based computing, pp. 7-8, Aug./Sep., 1993.

[13] K. Keeton, B. A. Mah, S. Seshan, R. H. Katz, D. Ferrari, "Providing connection-oriented network services to mobile hosts", Proc. USENIX Symp. Mobile and Location-Independent Computing, pp. 83-102, Aug. 93.

[14] N. Krishnakumar and A. Bernstein, "High throughput escrow algorithms for replicated databases", Proceedings of the 18th Intl. Conf. on Very Large Data Bases, 175-186, Aug. 1992

[15] A. Kumar and M. Stonebraker, Semantics based transaction management techniques for replicated

data, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 379-388, 1988.

[16] M. Mouly and M. - B. Pautet, "The GSM System for Mobile Communications", 49 rue Louise Bruneau, Palaiseau, France, 701 pp., 1992.

[17] P. E. O'Neil, "The escrow transactional model", ACM TODS, 11 (4), 405-430, Dec. 1986.

[18] A. R. Noerpel, L. F. Chang and D. J. Harasty, "Radio link access procedure for a wireless access communications system", *Proc. Intl. Conf. Comm.*, May, 1994.

[19] V. Schnee, "An excellent adventure", *Wireless*, pp. 40-43, Mar./Apr., 1994.

[20] J. Schwartz, "Upgrade lets salespeople share data", *Comm. Week*, pp. 47-48, May 24, 1994.

[21] N. Soparkar and A. Silberschatz, "Data-value partitioning and virtual messages", *Proceedings of the 9th ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pp. 357-367, 1990.

[22] C. Tait and D. Duchamp, "An efficient variable-consistency replicated file service", *Proc. USENIX File System Workshop*, May 92.