Mobile Devices and Data Synchronization Assisting Medical Diagnosis

ADRIAN SERGIU DARABANT, HOREA TODORAN
Dept. of Computer Science
Babes-Bolyai University
1, Kogalniceanu street, Cluj-Napoca
ROMANIA
dadi@cs.ubbcluj.ro, htodoran@euro.ubbcluj.ro http://www.cs.ubbcluj.ro

Abstract: - In order to be able to establish the most accurate diagnostics as quick as possible, medical doctors need fast access not only to the current patient state and test results but also to its historical medical data. With the diversity of the malady symptoms today a correct diagnostic often requires a valuable time that is not always available due to the patient's condition. Even more, after finishing his/her investigation, the medical doctor should be able to immediately forward the relevant results to other medical personnel (doctors, nurses, hospital administration). In this paper we propose a mobile solution for assisting medical doctors with complete information about their patients even when outside of the medical centers. The software aim is assist medical doctors in two main directions: consult patient data and update the patient's file and propose *early in time* new countermeasures for improving its status. The first direction deals with providing complete information on the patient's history, current status, test results and similar symptomatic cases with their solutions. The second direction deals with patient information updates. All this will be implemented in a mobile solution using handheld devices. The goal of our paper is to present a pilot implementation of a medical database system with dynamic and efficient data synchronization using wireless technologies.

Key-Words: mobile applications, wireless data synchronization, personal digital assistants, medical applications.

1 Introduction

Being able to perform virtually the same tasks as desktop computers or laptops, Personal Digital Assistants (PDAs) are constantly increasing their attractiveness. Although originally created to run simple applications like electronic diaries, address books or planning calendars, the handheld devices eventually evolved into real pocket computers capable of undertaking more complex tasks[11,13], from word processing and spreadsheet editing to multimedia authoring[14]. Current models support data transfer over communication networks via the common wireless protocols (infrared, bluetooth, WiFi, GPRS), therefore providing access to convenient services, including web browsing, messaging, email, and so on.

Besides the *lower power consumption* (the battery life is approximately two times longer than in the case of laptops), the most important advantages of the handheld computers over other mobile devices achieving similar performance consist in their higher *portability* and *mobility*. PDAs are much lighter than laptops or TabletPCs (approximately 100-200g), fit into the jacket's pocket (wearable), can be hold into one hand and operated with the other (handhelds), and can be operated even on the move. Furthermore, they are very easy to use and prove economic viability [1], having much of the computing capability and storage capacity of laptops at a fraction of the cost (some authors even called them "equity computers"- e.g. Andrew Trotter in [2]). As a consequence, PDAs are more and more exploited in various fields, including mobile business ([3]), mobile education, medicine, and leisure.

Nonetheless, there are also inconveniences when using handheld devices. The most significant are related to the *small size of the screen*, which confines the amount of information displayed or requires the intensive use of navigation bars. *Data input* is more difficult than in the case of desktop computers or laptops, since the keyboard and the mouse are very small (if present). Even the stylus pens are rather narrow, therefore requiring accurate operation on the screen pad. Handhelds have relatively limited storage capabilities, are difficult to upgrade and much less robust than TabletPCs, laptops, and desktop computers. Taking all these restrictions into account, software producers must design applications running on PDAs more carefully than those for the other types of PCs. As a good practice example, the Windows Mobile for PocketPC family includes scale down

versions of the Microsoft Windows operating systems, very similar to the desktop versions, though adapted in terms of minimal requirements (memory, processor speed) and visual elements (windows, menus, lists, buttons) to the characteristics of PocketPCs. An exhaustive list of general design requirements for Windows Mobile-based PocketPC applications is given in [4].

2 Problem Formulation

Medical doctors need to be efficient when consulting their patients in the daily routine, in terms of both saving time and taking the appropriate professional decisions. A crucial prerequisite for achieving a high level of efficiency is the quick access to a whole range of information about the current patient: medical history, results of previous medical investigations, opinions of other specialists on the case, and so on. Moreover, the medical doctor should be able to easily disseminate the results of his/her own investigations on the patient.

The traditional solution is to use paper-based patient files that have to be carried by the medical doctor or by the accompanying staff to the patient's bed. The more information required for a certain investigation, the bulkier the dossier and the more difficult the search for relevant data.

A modern alternative is to use a computer-based solution, with a piece of equipment that is small enough to fit easily into the jacket's pocket, and with sufficient computing and communication power to rapidly bring patient's data on demand from a central database onto the device.

Although there are still situations where it is feasible for medical doctors to carry the paper-based patient files with them, these do not always provide the most up-to-date information, which is essential when taking the decisions. For instance, the most recent results of laboratory investigations might have not yet been recorded in the dossier. Furthermore, even if the raw data has already been recorded, it has not yet been interpreted according to medical procedures.

In this paper, we propose an innovative software architecture on mobile devices with communication capacities, solution that can be easily used with any exiting medical data management software. The application not only facilitates the management of various sets of information (doctors, patients, investigations, and so on), using the PDA, but also synchronizes the data between the mobile device and the hospital's database server.

We implemented the proposed framework on mobile assistants (PDAs) with cellular phone and/or wireless 802.11 capabilities [6, 7] in a pilot project that has been experimentally linked with the management system of a hospital [8]. With the aid of the new framework a hospital that has many premises is able to manage, at any moment, updated information about its patients, even coming from far locations[15]. It is also able to keep in contact with general practitioners who, after visiting their patients at home, can immediately send the results of their investigations to the specialists.

The novelty of our system consists in using an incremental data synchronization mechanism [5] based on timestamps, as described in section 4.1 below. The system architecture (see section 3 for details) also ensures a high degree of independence between the mobile system and the hospital's data management system, which is crucial in case of temporary failure of one of the components. Optimized network traffic is achieved by means of a data compression solution, illustrated in section 4.3 of the present paper

2.1 Data flow

The system relies on the existence of the hospital's central database server to which medical staff (doctors, nurses, laboratory assistants), as well as administrative personnel shall have access anytime and from any location within hospital's premises. They not only frequently ask for updated information concerning the patients, but also send new information to the database server (e.g. results of their investigations).

Medical doctors and their assisting staff query the hospital's database to get useful information that could help them conducting their investigations on the patient. As a result of these investigations, the medical staff attains new information that should be inserted into the database for future use. Various pieces of the recorded information could also be used by the administrative staff, for instance to estimate the cost of the treatment or to plan the use of equipment.

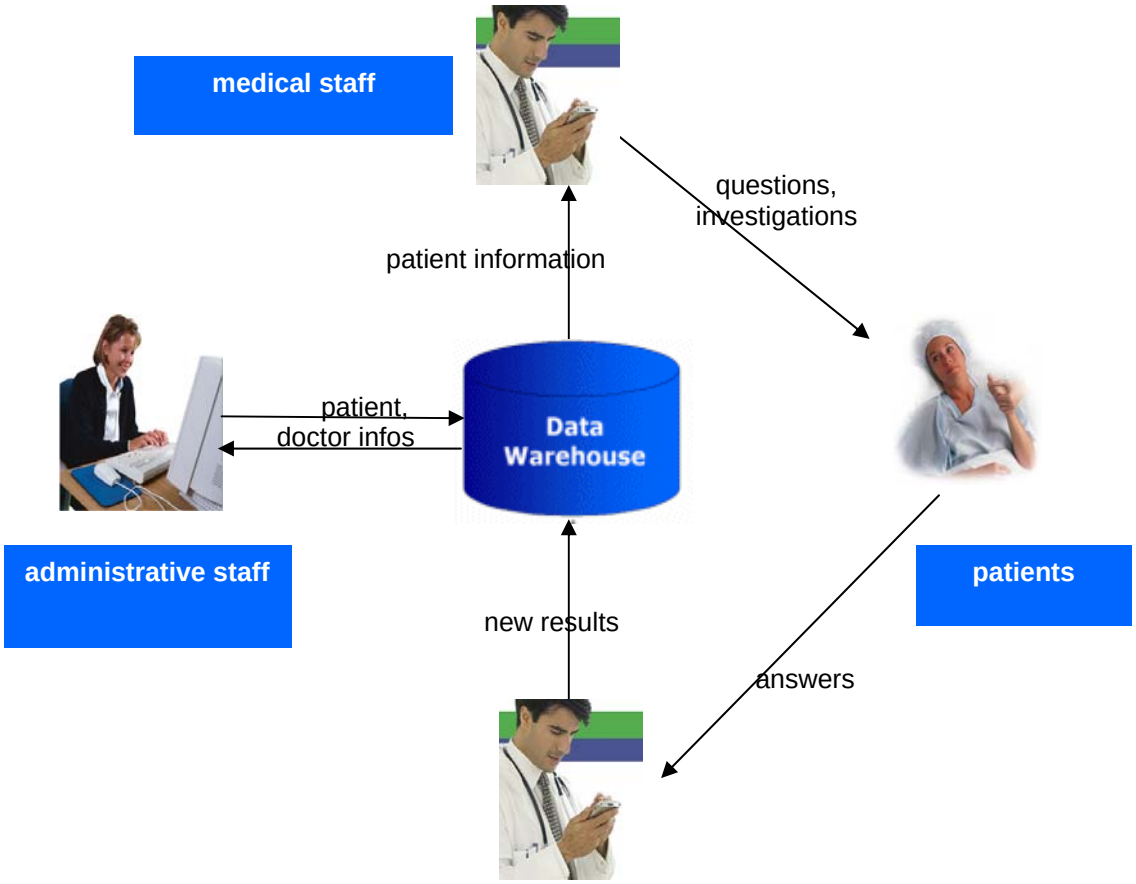

Fig.1 Medical information flow

Fig. 1 depicts the typical information flow within the medical architecture we are trying to model.

The general practitioner is in our scenario a mobile user that travels with his/her PDA. Usually, traveling means either visiting the patients in the hospital, or receiving them in his/her office, but general practitioners might as well visit the patients at their residence in certain situations. Given this *volatile* environment, we would like the medical doctor to have as much information as possible about the patients he/she is consulting at a given time. Furthermore, a medical doctor could get assignments for visiting patients directly. The patient's consultation request at the clinic is either directly pushed by the software system on the general practitioner's mobile device or collected in the course of the next data synchronization between the central system and the mobile device.

During a consultation the general practitioner usually evaluates the patient's state using various medical analysis methods: blood sampling, temperature, blood pressure, etc. The results of all these measurements consist in various data sets that the general practitioner would normally write in the patient's record. Using our mobile application – *MobMed* – the general practitioner no longer needs to fill in paper reports and records about the patient. All he has to do is to input the various data sets he obtained into the database, using the friendly interface we designed. Data is temporarily stored in its own portable device (PDA) and then synchronized with the central warehouse of the hospital. Virtually, the general practitioner has a small sized tool capable of storing patient's records, recalling history data about the patients and therefore helping the medical doctor establish an early potential diagnosis.

But our solution deals not only with generalists but also for special intervention medical teams. In the case of a severe accident the often unconscious patients need to be quickly transported to a specialized hospital. In many cases preparing the patient arrival at the hospital involves preparing the equipments and scheduling out of order clinical tests. All these procedures need to be prepared in accordance with the patient history. Instant consultation of patient history might reveal to the intervention team certain severe

incompatibilities with the proposed resuscitation techniques or proposed treatments. In many cases these incompatibilities have lead to tragic deaths or patient permanent injuries that could have been otherwise avoided. It appears thus vital for a medical practitioner or intervention team to instantly consult a patient or victim medical history. We offer this possibility by implementing a software solution on PDAs or mobile devices. The patient medical records are retrieved to the medical staff PDA using any available communication link and stored temporarily. Upon history checks the medical personnel can inquire for similar symptomatic cases in order to be assisted in their diagnosis. A list of similar cases in then retrieved together with their solution and success indicators. By inspecting all this information the medical specialist can verify the chances of applying an inadequate treatment given a patient current status and history. The *backoffice system* in fig 1 is the central data storage facility. For the purpose of our application this could be any database server storing patient historical records, treatment and schedules, as hospital administration data, schedules, operation planning, etc. The data warehouse could be implemented by the same database that handles the hospital back office data or in a different database that will be synchronized/partially replicated [12] with the hospital backoffice.

3 System Architecture and Services

In the following paragraphs we present the proposed system architecture that has been already implemented in a pilot project called *MobMed*. One of our major requirements was to build a system that is as least intrusive as possible in the existing hospital or clinic management software (HMS). With this goal in mind, we needed to link the mobile system facilities to an existing software solution in such a way that no major adjustments are required for the already implemented solution.

We used the following model to implement the mobile architecture on the top of existing hospital management software, which is running on an MS Windows-like operating system and storing data in an SQL Database Server [10]. We present this particular architectural model here in order to show the degree of independence between the mobile system and the existing management solution.

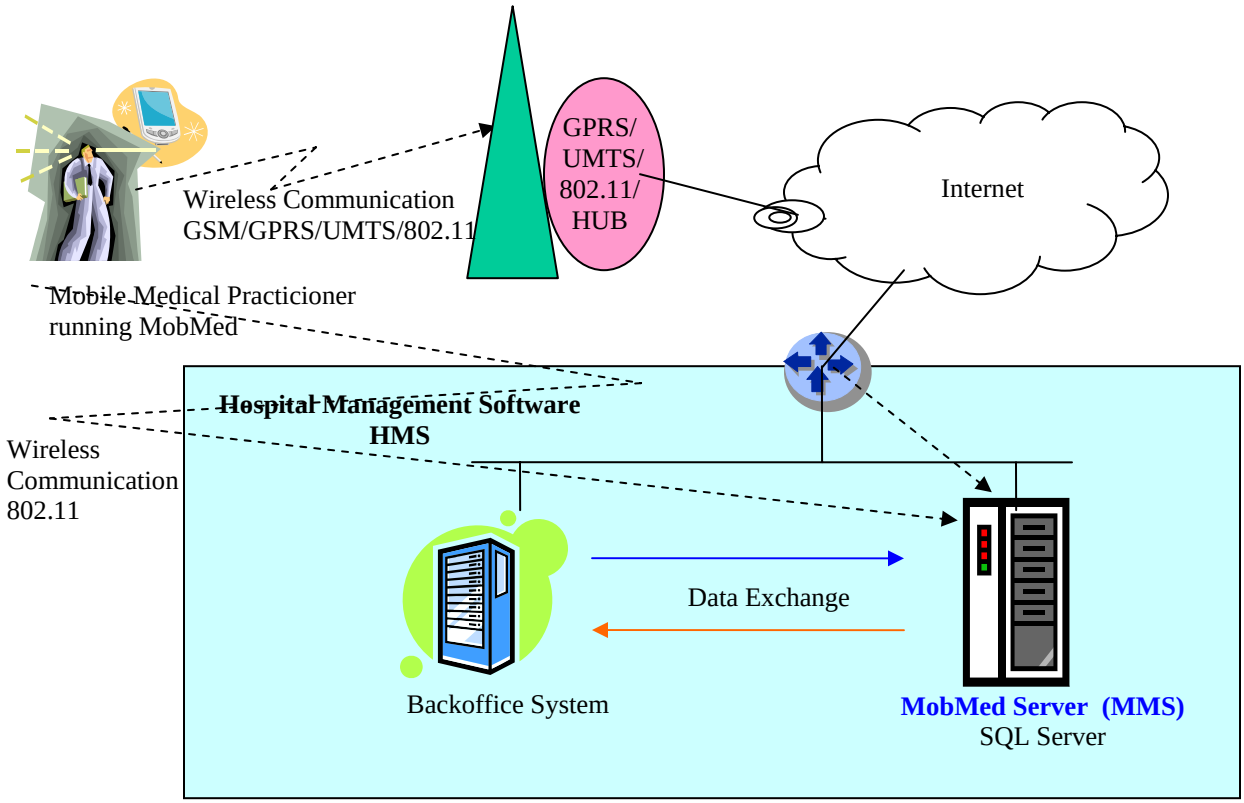

Fig. 2 – The MobMed System Architecture and Integration with the Hospital Management Software

Even if it is a challenging task, the integration between any existing management software (HMS) and *MobMed* is always possible with little or no intrusion into the HMS at all. In order to implement the *MobMed* solution we suppose that the clinic (hospital) already has at least an Intranet system, possibly connected to the Internet by some highly secure connection. We further suppose that, in most of the cases, the internal network of the hospital is protected by a firewall that separates the Intranet from the Internet.

In Fig. 2 the general practitioners are running MobMed on Pocket PC devices with incorporated GSM phones or wireless radio cards (WiFi). The choice of GPRS/UMTS or wireless is conditioned by the necessary connection persistence. If the connection is desired at any particular moment, the best choice would be GPRS/UMTS, both requiring a GSM line. For connections only in hospital's perimeter the use of the 802.11 standard and WiFi hotspots might be a convenient alternative. For mixed uses the applications will use the communication method with the smallest cost. When the user of the MobMed application is within a Wifi hotspot area that allows him to reach the MMS then the Wifi connection can be used. If the Wifi connection is not possible then the application will establish first (automatically or with the help of the operator) a 3G/GRPS data connection by using the GSM line. When the connection is available the application is free to exchange information with the MMS server. We will describe in the following sections the techniques used to exchange data between MobMed and MMS. The most used communication scenario involves data downloading on the device, handling and then, when needed, synchronization back to the MMS. A medical user will query data for a patient. The query for data regarding a patient will trigger a download operation if data not already present on the device. For that, either the entire set of patients cured by the same medical doctor or just data for the specific patient is downloaded from the MMS server. Once on the device the data is stored in local mini Mobile SQL Server 2005 in a relational format that is a miniaturized filtered copy of the MMS dataset. Upon the download operation the entire information concerning the patient is available to the medical doctor and presented by the application. The medical status of the patient can be assessed by looking at the pathological tests conducted on the patient and by consulting its recent medical history. Every recent treatment can be inspected and all available test results will help the medical doctor take the right diagnosis decisions. If there is not enough data available for formulating a conclusion and issuing a diagnosis the medical doctor can update the patient's record and request and schedule in the same time new medical tests on its local device. Once all the tests are scheduled the entire information is *synchronized* again with the MMS server. From the MMS server the data is pushed back into the hospital backoffice application and will trigger the appropriate actions. The backoffice and the MMS server are tightly coupled and synchronized at every data modification. Being generally on a local high bandwidth network, keeping them in sync will not penalize the entire system.

4 Data Exchange and Synchronization

Since we want to give access to the mobile general practitioners to the database server where patient, diagnosis and consultation schedules data are stored, some features of the Microsoft based platforms can be used. First of all, the SQL Database Server, in its 2000 and 2005 variants, offers support for mobile subscribers to data publications.

The MobMed server runs SQL Server and provides for data synchronization with the mobile devices. The synchronization procedure uses the HTTP protocol for data transport in order to be easily accessible on highly secured platforms. HTTP is used because of its high implementation availability on all platforms and because it allows easier through firewall communication.

The mobile devices are running a .NET application platform with Mobile SQL Server 2005 as data storage and synchronization engine. Microsoft SQL Server 2005 Mobile Edition (SQL Server Mobile), the "descendant" of Microsoft SQL Server 2000 CE Edition 2.0 (SQL Server 2000), extends the Microsoft enterprising solutions for *line-of-business* and for the management of personal information applied on a device. SQL Server Mobile delivers the functionalities necessary to a relational database, transposed to a lower scale: robust information storage, query preparation, connectivity capacities. Microsoft SQL Server 2005 was projected to support an extended list of mobile devices and Table PCs. The mobile devices include any device that runs Microsoft Windows CE 5.0, Microsoft Mobile Pocket PC 2003, Microsoft

Mobile Version 5.0 Pocket PC, or Microsoft Mobile Version 5.0 Smart Phone

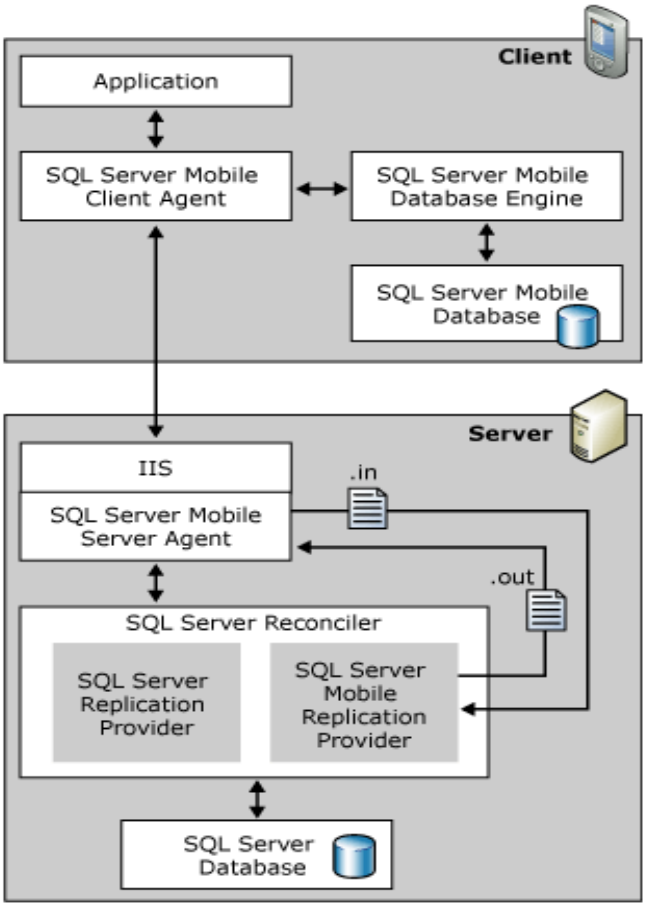

Fig 3 – Merge Replication Architecture[1].

.

The internal architecture of the data replication between the mobile devices and the HMS server is depicted in fig.3. The *application* runs on the mobile devices together with an instance of the SQL Server Mobile database engine for storing data. An additional component: the *SQL Server Mobile Client Agent* helps synchronizing the two databases: the local mobile database with the HMS SQL Server database. On the server side the data flow passes trough an Internet Information Server (IIS) (the Microsoft's implementation service of a Web server). This is needed for handling the communication trough the HTTP protocol. When installed on a local area network the IIS needs a helper component: *the SQL Server Mobile Server Agent.* This component will identify all requests for data synchronization arrived at the IIS server entry point and will differentiate them from normal HTTP sessions. The IIS server and component are also required in order to implement authentication. All requests passing trough the IIS server and destined for data synchronization carry an authentication token that is validated or invalidated by the *SQL Server Reconciler* (against the SQL Server engine). All authenticated requests are then translated in SQL requests for the SQL Server by the *SQL Mobile Replication Provider.* The fact that the IIS takes part into the replication process leads to the idea that data synchronization works on *pull* model. Indeed the MMS server cannot trigger a data synchronization operation on a mobile device that doesn't ask for it. The model is thus a client-server one where the client (the MobMed application) always initiates data exchanges. The *SQL Server Replication Provider* in fig. 3 deals with data replication between the MMS system and the hospital backoffice (HMS) data management system. The internal data replication architecture we presented here uses the *Merge replication* protocol in order to synchronize the data.

Merge replication is an ideal synchronization mechanism when using mobile devices, as it allows the automatic and independent data update on both the mobile device and the server. As soon as the device is connected, the data is synchronized, changes being sent from the client to the server, along with the new information being pushed from the server to the mobile device. Merge Replication needs more server configurations and maintenance than its alternative method (Remote Data Access), but it is more advantageous for applications that involve several mobile devices. Merge Replication also has the capacity of detecting and solving shown up conflicts and of rejoining data from several tables at the same time. It allows instrument survey by using SQL Server and provides more data rejoining options such as the article types and filtering.

Merge Replication is a proper solution when conflict solving is required, when information has to be propelled to and from the desktop or laptop computers and when working with larger databases.

Merge Replication uses some components of the Microsoft SQL Server 2005 Mobile Edition (SQL Server Mobile): SQL Server Mobile Database Engine, SQL Server Mobile Client Agent, SQL Server Mobile Server Agent, SQL Server Mobile Replication Provider as described above.

[1] Microsoft TechNet [http://technet.microsoft.com /en-us/library/ms171927.aspx]

4.1 Replication Implementation

SQL Server and the SQL Server Mobile versions generally allow for two types of interactions when it comes to data synchronization: *Remote data access (RDA)* and *Merge replication.* The two methods are distinct from the point of view of managing data conflicts. All data replication mechanisms are prone to data conflicts i.e. the same data updated in multiple mobile locations and then applied to the server. RDA offers a simple manual conflict interception technique as opposed to the merge replication method that handles data conflicts automatically when possible.

RDA provides a simple way for a Windows CE-based application to access (pull) data from a remote SQL Server database table and store that data in a local SQL Server Mobile database table. The application can then read and update the local SQL Server Mobile database table. SQL Server Mobile can optionally track all changes that are made to the local table. The application can later update (push) the changed records from the local table back to the SQL Server table. Windows Mobile-based applications can also use RDA to submit SQL statements to be executed on a remote SQL Server database. For example, an application could submit SQL statements that insert, update, or delete records to a remote SQL Server table. RDA is appropriate when conflict resolution is not required.

Merge replication is ideally suited to portable devices because it lets data be updated autonomously and independently on the portable device and the server. The data on the device and the server can be synchronized when the device is connected, to send changes from the client to the server and to receive new changes from the server. Merge replication requires more configuration and maintenance at the server than Remote Data Access (RDA), but it is appropriate for applications that have many devices. Merge replication also provides built-in and custom conflict resolution capabilities, allows for replication of data from multiple tables at the same time, lets you use monitoring tools by using SQL Server, and provides rich data replication options such as article types and filtering to improve performance depending on the data needs and identity range management.

Merge replication is also a good solution when conflict resolution is required, when data must be propagated to and from desktop or portable computers and devices and when rich data distribution capabilities are required.

Common scenarios for merge replication include the following:

- Downloading read-only data
- Entering and uploading new data
- Updating and synchronizing data

The functioning of *merge replication* from Microsoft SQL Server 2005 Mobile Edition implies the following processes:

1. The data is published on the SQL Server
2. One or multiple *subscribers* is/are created for the publication (mobile devices)
3. The data in the subscriber is updated
4. The data is synchronized.

Details on data publication and subscriber registration setup are presented in [16].

When a SQL Server Mobile *subscriber* is synchronized with the SQL Server, all the changes on data are recovered by the database publication. However, when a SQL Server Mobile subscriber is synchronized for the first time, it can recover the data either directly from the database publication or from the snapshot file. The sealed class *SqlCeReplication* is part of the space names *System.Data.SqlServerCe* and allows the synchronization of SQL Server Mobile databases with the SQL Server databases. Its interface is described in [9].

The typical steps to perform data synchronization when everything is setup are depicted bellow:

```
SqlCeReplication repl = null;
Try{
  repl = new SqlCeReplication();
  repl.InternetUrl = "http://www.myHMSserver.ro
    /sqlmobile/sqlcesa30.dll";
  repl.InternetLogin = "MyInternetLogin";
  repl.InternetPassword = "<password>";
  repl.Publisher = "MyPublisher";
  repl.PublisherDatabase =
    "MyPublisherDatabase";
  repl.PublisherLogin = "MyPublisherLogin";
  repl.PublisherPassword = "<password>";
  repl.Publication = "MyPublication";
  repl.Subscriber = "MySubscriber";
```

```
    repl.SubscriberConnectionString = "Data Source=
      MyDatabase.sdf";
    repl.AddSubscription(AddOption.CreateDatabase);
    repl.Synchronize();
} catch (SqlCeException) {     // Error Handling/ }
finally   {    repl.Dispose();  }
```

For all synchronization units we have an incremental synchronization mechanism based on timestamps. Each mobile device keeps the date and time of the last successful synchronization for each entity type (column, row, record) and brings in only newly modified data from the gateway server. The timestamps or trackers are added automatically to both MMS SQL Server and SQL Server Mobile by the publication setup.

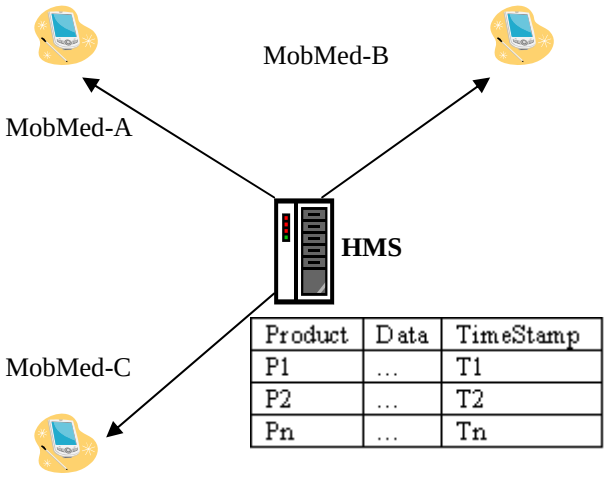

Fig 4 – Merge Replication using timestamps.

The SQL Server trackers or timestamps are never removed after the publication is created. They stay on for the entire life of the publication. On the client side (the MobMed mobile device) the SQL Server Mobile has two types of synchronization: *recovery* and *incremental.* The recovery synchronization recovers the entire publication from the publisher. Generally this type of synchronization is requested at the initialization of a new mobile device with the MobMed application. There is no initial database on the subscriber (MobMed). The first and initial copy is created by the first recovery synchronization. The incremental synchronization on the other hand is used for subsequent data replications when entities have been changed either on the subscriber or on the publisher. The former synchronization type only downloads data from the server – is a one phase step. The later method is composed of two steps: a download process that brings in modified data from the publisher and an upload process for integrating local changes into the publication database.

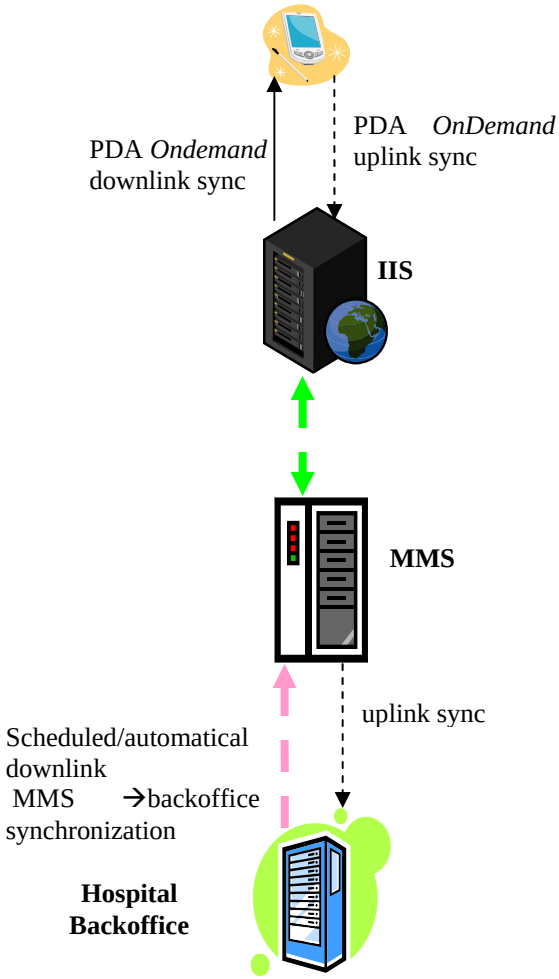

Fig 5 – Logical/Physical Data Path during Synchronization

Fig 5 shows the data path during synchronizations between the MobMed application and the hospital backoffice. In order to be as least intrusive as possible we deal in our architecture with two distinct synchronization operations. The first one occurs between the MobMed mobile device and the MMS Server running SQL Server, were the data is published. The second synchronization occurs

internally in the hospital's intranet and synchronizes data between the MMS system and hospital management solution. As the hospital management system is unknown/black box from our point of view we do not deal with data integration between the two systems. The only important thing from case to case is to ensure that there is a synchronization possibility between the hospital management database system and the MMS SQL Server. As already SQL Server provides synchronization methods for most available database servers this point should not be a blocking one. Even in the situation where the already implemented data replication with the hospital backoffice is not suitable for our needs there is always the solution of directly implementing this layer. As a final note – the two synchronization procedures MobMed-IIS-MMS and MMS-Hospital backoffice can be disconnected or tightly coupled together.

4.2 Conflict Detection and Resolution

An important aspect in data synchronization with multiple data sources is conflict resolution. Devices might update the same entity locally with different data and then upload changes to the MobMed server in a random order. The problem is choosing the correct final value/state of the entity. Automated conflict resolution is usually tightly dependent on the business rules that govern the conflicting entity's type ([6]). SQL Server Mobile detects *client-side* conflicts but does not manipulate their settlement. The conflict information is sent to the Publisher in view of settlement during the following synchronization. Most of the conflicts are settled by the Publisher following the synchronization.

Changes made to the local database during synchronization could cause a local conflict. If a row from the publisher cannot be applied at the subscriber, it is considered a subscriber conflict, and the SubscriberConflicts property is set. With SQL Server subscribers, conflicts are resolved at the Subscriber. However, SQL Server Mobile does not have a reconciler. All conflicts must be resolved at the Publisher. When developing an application, one can design the application to examine the Subscriber Conflicts property after each synchronization. If it is set to a nonzero value, the data must be resynchronized so that the publisher can resolve the conflicts.

There are thus two major types of conflicts: one at the subscriber when downloading data and one at the publisher due to multiple subscribers updating the same entities.

SQL Server Mobile 2005 allows for both row and column tracking when synchronized with SQL Server 2005. With row-level tracking the SQL Server Mobile Database Engine invokes tracking when any row is inserted, updated, or deleted. When conflicts are detected at the row level, changes made to corresponding rows are considered a conflict, regardless of whether the changes are made to the same column. For example, a change is made at the Publisher to the address column of a row, while another change is made at the Subscriber to the phone number column of the corresponding row. With row-level tracking, a conflict is detected because changes were made to the same row.

After a conflict is detected, the system launches the conflict resolver that is selected for the article. This might be the default resolver, one of the other supplied resolvers, or a custom resolver. The accepted changes are chosen according to the rules of the conflict resolver. SQL Server Mobile Subscriber conflicts are always detected, resolved, and logged at the Publisher. Resolvers can use the source of the data change, or the priority value of the Subscriber, to resolve conflicts. For example, the default resolver follows the rule that the changes on the Publisher always override changes on the Subscriber. One can opt to use a different resolver that always favors the changes on the Subscriber over those on the Publisher.

When using the default priority-based conflict resolver, one does not have to set the resolver property of an article. For using an article resolver instead of the default resolver, one must set the resolver property for the article that will use it by selecting an available resolver on the Publisher. Any specific information that needs to be passed to the resolver can also be specified in the resolver information property.

Merge replication offers four types of conflict resolvers:

- The default priority-based conflict resolver. The default resolution mechanism behaves differently, depending on whether a subscription is a client subscription or a server subscription. If different priority values are assigned to individual Subscribers that use server subscriptions; changes made at the node with the highest priority win any conflicts. For client subscriptions, the first change written to the Publisher wins the conflict. After a

subscription is created, it cannot be changed from one type to another.
- A business logic handler. The business logic handler framework allows writing a managed code assembly that is called during the merge synchronization process. The assembly includes business logic that can respond to conflicts and a number of other conditions during synchronization.
- A COM-based custom resolver. Merge replication provides an API for writing resolvers as COM objects in languages such as Microsoft Visual C++ or Microsoft Visual Basic. For more information, see COM-Based Custom Resolvers.
- A COM-based resolver supplied by a third party. SQL Server 2005 includes a number of COM-based resolvers.

4.3 Data Compression

An important aspect when dealing with GSM transports is the cost of the data transfer. As the GPRS/UMTS solutions are still costly, the employment of a data compression mechanism is beneficial. The application has native compression support based on Merge replication compression. Another feature that reduces traffic is column and cell based synchronization. When just a single column of database row has been updated, only that value is sent to the server avoiding thus an entire row transmission.

5 User Friendly Interface

As already mentioned in the introductory section of this paper, software applications for PDAs have to be carefully designed in order to overcome the limitations of the handheld devices, especially those related to the small size of screen and the difficult use of data input accessories (keyboard, mouse, stylus pen). We certainly took these constraints into account when designing the user interface of the MobMed solution.

Following are the main characteristics of the MobMed user-friendly interface (some of them are noticeable in Fig 4 below):

- the navigation bar always displays the name of the topmost window, thus avoiding confusion;
- common menus appear on the leftmost position of the MenuBar in a known order;
- whenever text fields are present on the screen, they are accessible with the Soft Input Panel (SIP) up, thus facilitating data input from mobile devices without keyboard;
- the application maintains the regional and language settings specified by the client.
- in order to ease user's navigation through the successive application's windows, we employed suggestive graphical elements wherever appropriate. It is also the case of the main menu, made of attractive graphical buttons.

Fig. 6 MobMed's login window

Fig 6 shows the login form of the MobMed application. Each medical doctor will use a single ID to identify against the system. Upon authentication the system knows the list of active patients for the given doctor and allows him to deal only with data regarding its patients. All the synchronization procedures are filtered against the doctor's authentication information so not-needed data will not be downloaded to the mobile device. Additional information can always be made available upon special request from the user.

In the fig 7 the main and a patient form are presented. The main form directs the user to the main functions of the application. The patient form allows for new examinations to be requested and scheduled together with handling the patient details.

Fig. 7 MobMed's main and patient form

All data input methods are targeted for a mobile device. This means that long user inputs by using the keyboard or keying in strings should be avoided. Assisted search forms will guide the user when patient or malady identification and descriptions should be searched for and displayed.

6 Conclusions and Future Work

The current paper proposes a novel multi-tier system to enhance the mobility of medical staff, thus bringing an important efficiency advantage to the hospital/clinic implementing it.

The main functional characteristics of our proposal are related to medical staff having access to up-to-date, complex information on their Pocket PCs, virtually wherever and whenever they need it. This is provided by on demand secure and fast synchronization of the local database from the mobile device with the central data warehouse of the hospital.

The application running on the mobile device (MobMed) has an user-friendly interface, which is designed according to the general requirements described in [4].

As far as the system architecture is concerned, our model ensures application isolation and independence in the case of temporary failure between the three tiers: the mobile device, the data management server, and the client's existing management software. Moreover, MobMed successfully deals with important aspects related to data synchronization, like conflict resolution and data compression.

Incremental data synchronization, optimized network traffic, and cvasi-permanent availability are features that make our system valuable and different from other similar implementations.

Future work is intended to building a general framework allowing the smooth integration with any HMS. It should also improve the security of data exchange and the control of mobile agents' navigation from the handheld devices.

7 References

[1] E. Dieterle, *Handheld devices for ubiquitous learning and analyzing*, 2005 National Educational Computing Conference, Philadelphia, PA, available at [http://center.uoregon.edu/ISTE/uploads/NECC2005/KEY_7287575/Dieterle_NECC2005Dieterle_RP.pdf], visited June 2007.

[2] A. Trotter, "Palm computing moving from the workplace to the classroom", in *Education Week*, October 1999.

[3] A.S. Darabant, H. Todoran, "Building an Efficient Architecture for Data Synchronization on Mobile Wireless Agents", in *WSEAS Transactions on Communications*, Issue 8, Volume 5, August 2006, ISSN 1109-2742, pp. 1384-1391.

[4] ***, *Designed for Windows Mobile™ Software Application Handbook for Pocket PCs*, Microsoft Corporation, May 2004.

[5] M.D. Sutton, *Data Synchronization: Which Technology?*, Intel Software Network, [http://www.intel.com/cd/ids/developer/asmo-na/eng/52893.htm], visited June 2007.

[6] W. Wheeler, *Integrating Wireless Technology in the Enterprise, First Edition: PDAs, Blackberries, and Mobile Devices*, Elsevier Digital Press, 2003.

[7] M. Mallick, *Mobile and Wireless Design Essentials*, Wiley Publishers, 2003.

[8] S. Fischer et al., *Handheld Computing in Medicine*, Journal of the American Medical Informatics Association, 2003

[9] MSDN, *SqlCeReplication Class*, Microsoft Corporation, [http://msdn2.microsoft.com/en-us/library/system.data.sqlserverce.sqlcereplication.aspx], visited June 2007.

[10] R. Laberge, S. Vujosevic, *Building PDA Databases for Wireless and Mobile Development*, Wiley Publishers, 2002.

[11] O. Cret, Z. Mathe, C. Grama, L. Vacariu, F. Roman, A. Darabant, "Solving the Maximum Subsequence Problem with a Hardware Agents-based System", WSEAS Transactions on Circuits and Systems, vol. 5, no. 9,September 2006, pp. 1470-1478.

[12] A. Darabant, H. Todoran, O. Cret, G. Chis, "The Similarity Measures and their Impact on OODB Fragmentation Using Hierarchical Clustering Algorithms", WSEAS Transactions on Computers, vol. 5, no. 9, September 2006, pp. 1803-1811.

[13] O. Cret, Z. Mathe, C. Grama, L. Vacariu, F. Roman, A. Darabant, "Solving the Maximum Subsequence Problem with a Hardware Agents-based System", Proceedings of the 10th WSEAS International Conference on CIRCUITS, Vouliagmeni, Athens, Greece, July 10-12, 2006, pp.75-80.

[14] M. Wagner, *Handhelds nudge PCs*, InternetWeek, May, 2001.

[15] P. McDougall, *Full-Fledged Business Apps Make PDAs Indispensible*, InformationWeek, October, 2001.

[16] Microsoft TechNet, *How to: Create a Publication and Define Articles (SQL Server Management Studio)*, Microsoft Corporation, [http://technet.microsoft.com/en-us/library/ms151160.aspx], visited June 2007.