Master's Thesis

A secure XML based synchronization scheme for mobile devices

Panayiotis Andreou

UNIVERSITY OF CYPRUS

COMPUTER SCIENCE DEPARTMENT

June 2006

UNIVERSITY OF CYPRUS
COMPUTER SCIENCE DEPARTMENT

A secure XML based synchronization scheme for mobile devices

Panayiotis Andreou

Supervisor

Prof. Andreas Pitsillides

The current MSc thesis has been submitted for the partial fulfilment of the MSc degree of the Computers Science Department of the University of Cyprus

Acknowledgments

First of all, I would like to express my sincere gratitude to my supervisor, Professor Andreas Pitsillides, for his constant support all these years and for giving me the opportunity to participate in several interesting research projects and attend various international conferences. His help was valuable to finish this thesis work.

I would also like to thank my fiancée and my family for their continuous support and understanding during these past two years.

Executive Summary

This thesis proposes a new approach to mobile database synchronization using latest standards like XML. This new approach is more appealing than the ones offered by modern mobile synchronization software, in the fact that it can be employed to organizations that involve virtual team management. Virtual team management adds more complexity and constraints to the task and therefore existing synchronization algorithms do not apply and they have to be adjusted.

Furthermore, the underlying network and its state affect the performance of the algorithm. This is crucial, and must be taken under great consideration when developing any mobile application.

Finally, in addition to a new algorithm of synchronizing mobile data, this thesis presents a complete platform architecture that enhances performance and security of the synchronization process by employing compression and encryption mechanisms.

CONTENTS

A new synchronization scheme

Enhancing the synchronization platform

Case Study – DITISv2 Mobile System

INTRODUCTION

1.1 General

1.2 Research motivation

Commercial mobile database systems differ a lot from desktop database systems. Mobile transactional coordination has to deal with weak connectivity and frequent disconnections, asynchronous, dynamic replication with profiling, publish & subscribe for data recharging & propagation, large scale replication, user interaction & feedback, long running tasks and decentralized commitments, real-time constraints and much more.

Modern commercial mobile database systems like SQL Server CE, SyncML, HotSync, IBM DB2 Everyplace, Oracle Lite, Sybase Anywhere strive to deal with the problems mentioned above, but they employ single user synchronization to a single database. By that, we mean that a single record is updated based only on timestamps rather than a policy ran by the organization. Virtual team management at a database level adds more complexity to the synchronization task. Virtual team tasks and management are governed by the rules and policies of the organization it resides. As a result a simple merge replication algorithm based on timestamps is not preferred or does not apply, because it does not take into account any specific characteristics of the virtual team. These characteristics may include predefined rules or policies that may apply in specific situations.

Furthermore, employing a good synchronization algorithm is not enough in the case of mobile devices, as weak connectivity and different disconnection may render the exchange of mobile data useless. The network and its state affect the performance of the synchronisation algorithm. As a result a stable and efficient infrastructure must exist for the synchronization algorithm to perform efficiently. This infrastructure must include mechanisms that will ensure both network and application efficiency and robustness.

Also, we cannot ignore security when building wireless applications. Data transmitted over a wireless medium are easily recovered if no security mechanism exists. Of course, security adds a negative performance effect to any application and it must not be handled lightly when dealing with mobile devices.

Having in mind the Greek saying “Παν μέτρον άριστον”, which means do not overdo something even if you know is good, in this thesis we aim to investigate above mentioned problems and propose a complete solution for synchronizing data with mobile devices.

1.3 Research outline

This thesis is organized as follows. In chapter 2 we describe the problems concerning synchronization of data with mobile devices in terms of the mobile device and network constraints. In addition, our first assumptions on what are the requirements for a new solution are presented. In chapter 3 we describe the technologies that are proposed for our solution. These technologies are evaluated based on the standards that are going to be used in the final solution, like XML, and a selection is made between them. Chapter 4 presents the proposed synchronization algorithm along with its advantages and disadvantages. In chapter 5 we integrate the synchronization algorithm to our architecture and employ compression and encryption techniques to increase the performance and security of our system. Our case study is presented in chapter 6, where we employ our solution to an already existing system with high demands on mobile data. Finally in chapter 6 and 7 we present our conclusions along with open issues for future work.

PROBLEM DEFINITION

1.4 Introduction

In computing, synchronization is the process of ensuring that two or more computing hosts hold the same up-to-data information. If any computing host creates, updates, or deletes a record or file at one location, the synchronization process should create, update, or delete the same record or file at the other location.

Synchronization can be divided into two simple subsets, one-way or two-way. One-way synchronization is more trivial than two-way, because records or files are transmitted from the primary computing host (source/publisher) to the secondary computing host (target/subscriber) only in that direction. The subscriber host will never transmit any records or file to the publisher. On the other hand, in two-way synchronization, both hosts transmit records or files to the other establishing that both locations are always synchronized.

Two-way synchronization is not a trivial task. One must consider a number of factors when synchronizing. These factors include network connectivity, clock synchronization, crashing frequencies, CPU performance, memory, security, and many others.

Mobile device synchronization adds more stiffness to the task. Network disconnections, network cost, mobile device CPU performance, and mobile device power increase the synchronization infrastructure and algorithms' complexity to a very high degree.

In the following sections we discuss the constraints involved when building a mobile application. Specifically, we analyse how the CPU performance, memory, network connection and data load, affect the solution and discuss possible solutions.

1.5 Synchronizing data with modern mobile devices

As discussed in the previous section, mobile devices add new constraints to the synchronization algorithm and infrastructure. These constraints include the CPU, memory and network capabilities of the mobile device and will be thoroughly described in the section below as well as their effect in mobile applications.

1.5.1 Mobile device constraints

Mobile devices, such as Palms, Pocket PCs, SmartPhones, and Mobile Phones, nowadays are primarily designed as miniature replicas of desktop computers: that is they have the ability to edit documents, play audio/video files, run programs in addition to executing a vast variety of other functions; and yet still fit into the palm of our hand or in our pocket.

Even though modern mobile devices have a lot more processing power than their predecessors, they still pose many problems. Some of the basic problems are the small screen size and resolution, low processing power, low memory and low battery life. In this section we will only elaborate on the constraints that affect the data transfer and processing.

1.5.1.1 Processing power

Nowadays modern desktop computers have a very high processing power (1.6GHz-3.6GHz) and thus any desktop application besides video rendering or image filtering runs acceptably fast. Even laptops which are optimized for the mobile user run with a processing power which is considerably fast (1.6GHz – 2.1GHz).

This is not the case for mobile devices though. Most modern mobile devices run with a processor of 400MHz[1,2,3]. This fact leads us to the conclusion that there is a huge gap in the performance of desktop applications vs. mobile applications. As a result, new methods or algorithms must be designed with these environments in mind and should be tested for their efficiency. By efficiency we mean both performance and power or even better performance/power.

The performance of today’s mobile device processors is shown in the benchmarks[4] below.

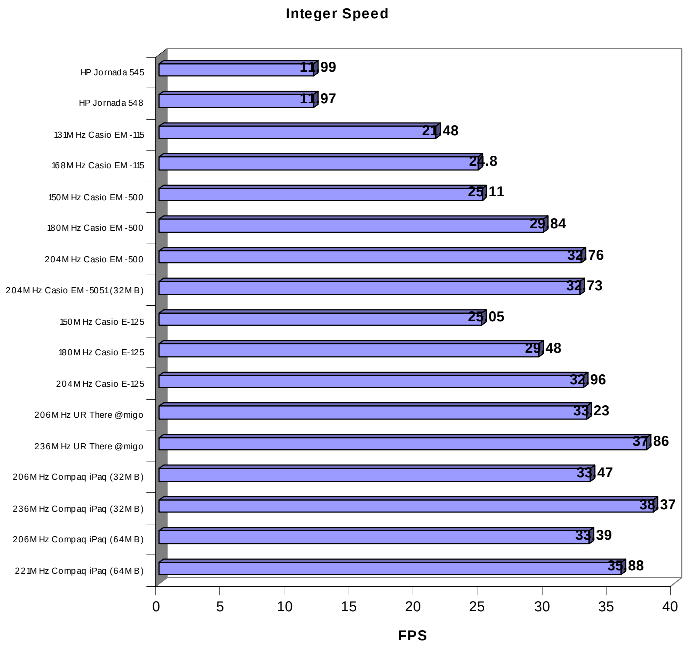


Figure 1: Pocket PC CPU comparison (integer benchmark) (Reproduced from [])

CANNOT READ Y AXIS???

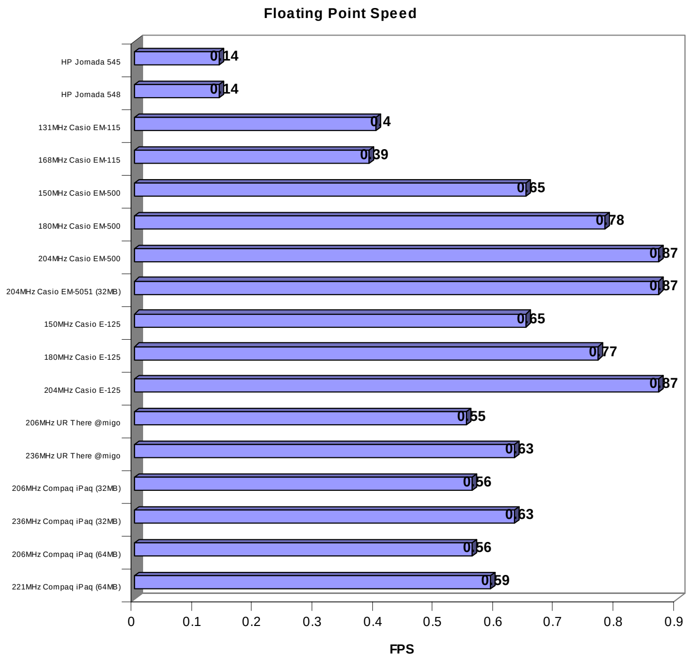


Figure 2: Pocket PC CPU comparison (floating point benchmark) (Reproduced from [])

1.5.1.2 Memory

The same observations that apply to the CPU performance apply to the memory aspect of mobile devices as well.

Memory is one of the most important components of today’s mobile devices. Memory performance is a crucial aspect (as is the processing performance, the user acceptance of the mobile application interface, and the wireless communication modules) and cannot be ignored.

The memory component of each mobile device is where all the running applications and data are loaded. As in desktop computers, the memory module has to perform three main tasks, write data into memory, preserve and read them. All the information is of course stored as binary data. Nowadays there are a lot of different kinds of memory that can be used in computers but we are going to focus only on the type of memory that is used in mobile devices.

Two basic types of memory are included in almost all desktop and mobile device architectures, that is ROM (Read-Only Access Memory) and RAM (Random Access Memory). ROM memory is used for storing data that the user wants to preserve on the mobile device, whilst RAM can be used for loading an application into memory as well as the data that this application requires. As a result, RAM should be faster since it directly affects the performance of the computing host.

In addition to RAM and ROM, the vast majority of mobile devices support the use of external memory modules like memory card-readers, USB interfaces or other. These external memory modules can be used for storing permanent data. In the following sections, when we refer to memory we mean the RAM module of the mobile device.

When dealing with large amounts of data, the memory component becomes crucial. Desktop computers with vast amounts of memory can load much more data into their memory than a mobile device. That is not the case with mobile devices. The out of memory exceptions are more than frequent in mobile application. When dealing with multi column tables and over 100000 rows then these exceptions become more than frequent. An idea of how great these delays can be, will be studied in section 1.8.4.

As a result one must be extra careful of the memory limitation of the mobile device or even better employ smart memory management in the application, according to the device capabilities.

1.5.2 Network constraints

Even if performance was not a problem with mobile devices, the fact that they are being used on the move over a mobile/wireless network can be a performance bottleneck. Mobile/wireless networks differ from fixed networks in many respects. One of the major problems is the low transmission rate. Depending on the connection being used this can vary from 54Mbits (Wi-Fi) to 9,6Kbits (GSM). In many cases this fact can limit the mobile application’s efficiency a great deal, especially when we are dealing with large amounts of data transfer and real time applications, such as video conferencing. Higher loss rates appear due to interference from any power emitting device like engines, lightning, etc. In addition there are higher delays and jitter.

As if these we are not enough, the security aspect becomes a nightmare when transmitting through a wireless medium. Anyone can sniff the medium and retrieve the packages, and if they are not protected somehow (e.g. encryption) then they original files can be easily retrieved.

On top of all that comes the fact that the connection to the wireless medium is not cost free. Even if the previously mentioned problems did not exist, the cost factor always affects the final decisions for the proposed solution. A mobile application that synchronizes perfectly with a remote database through the internet would not be appealing if the cost was, say a €1000 per month. So, one must be extra careful when building mobile services by trying to optimize the network’s use to a minimum.

1.5.3 Data Management and Logging

As discussed in the previous sections, processing power, memory and the network affect traffic flow. As a result, when designing a synchronization algorithm, the data load must be considered as a crucial factor, affecting to the performance of the mobile application. If, for example, we have two applications that synchronize with a remote server and both have the same performance but the first one minimizes network connections and download size but has an acceptable negative effect on performance, then the former is considered better than the latter because it also minimizes cost. This is a crucial point when building cost effective mobile applications.

Optimization of the transaction log is also a crucial aspect of the synchronization performance. One way of establishing synchronization is by re-executing the log at the mobile device. This, of course, may not be the best case scenario for mobile applications. A developer must try to keep the log as small as possible because of two reasons:

- Shorter period of transmission of the transaction log
- Low memory requirement for loading the log into main memory.

This can be done by propagating only the latest update on a single data item. For a better understanding let’s look at the example below:

Assume that the mobile application holds a record for a product whose ID is 15. This record is shown below:

Product ID	Description	Price	VAT
15	Chocolate	£15.00	15%

This record was synchronized at time T0 at the server. At some later time points the server performed the following updates.

T1: SET Description = ‘praline chocolate’
T2: SET Price = ‘£14.50’
T3: SET VAT = ‘18%’

This results in the following record:

Product ID	Description	Price	VAT
15	praline chocolate	£14.50	18%

If we filter the transaction log according to this record we would receive these 3 updates. Let’s assume that the record has not been updated at the mobile host and now a synchronization process is triggered with the server. The resulting transaction log would of course include these three changes but only one is needed to obtain the final record.

Translated to SQL, instead of

```
UPDATE      [PRODUCTS]
SET         [Description]=’praline chocolate’
WHERE       [Product ID]=15

UPDATE      [PRODUCTS]
SET         [Price]=£14.50
WHERE       [Product ID]=15

UPDATE      [PRODUCTS]
SET         [VAT]=18%
WHERE       [Product ID]=15
```

We would only need

```
UPDATE      [PRODUCTS]
SET         [Description]=’praline chocolate’,
            [Price]=£14.50,
                [VAT]=18%
WHERE           [Product ID]=15
```

This is a relatively simple example, but it reveals a rather major hazard. Imagine if the Product ID=15 was updated 1000 times and finally deleted from the server! We would only need the final delete statement to execute at the mobile host and preserve a consistent state.

Conclusively, the data management and logging must be considered when synchronizing data with a mobile device.

1.6 Preliminary Conclusions

Having listed all the problems and considerations we can now make some preliminary assumptions on how we can achieve our goals. First of all we need a clear definition of how the synchronization algorithm will work and what are the advantages and disadvantages in relation with existing commercial synchronization algorithms.

In the next step, we are going to study the download time of compressed and decompressed data and we are going to correlate them with the processing power needed for compressing and decompressing the data.

To extend our architecture and make it more secure we aim to employ an encryption mechanism that can be efficient in terms of processing power.

Finally, the essential question that we need to clarify is when should a mobile application use this platform architecture and when adjustments have to be made.

To achieve these goals we are going to study different technologies and mechanisms that can be employed in our solution. Afterwards we are going to propose a new solution and employ it to an already existing and running system, DITIS, and present the benchmark results.

TECHNOLOGICAL INFRASTRUCTURE

3 Technological Infrastructure

1.7 Introduction

In the following sections we are going to analyze the technologies that were considered and hence develop and implement our solution.

1.8 XML data

In the following sections we briefly describe the eXtensible Markup Language (XML) and the reasons why it is becoming more and more popular in our days. In addition, we list the advantages and disadvantages of XML which apply in our case. Finally, we examine some real scenarios of transferring XML data onto mobile devices and elaborate on the results.

1.8.1 eXtensible Markable Language

According to the World Wide Web Consortium, the eXtensible Markup Language is a general purpose markup language with the capability of developing markup languages that have the ability of describing different kinds of data.

In other words, XML can be used for describing the data (XSD files) and contain data as well, similar to database files with meta-data and records. XML is a simplified subset of Standard Generalized Markup Language (SGML). Its major goal is to assist the sharing of data across different computing hosts; computing hosts connected through the internet in particular. Several languages are based on XML, specifically GML, MathML, PML, RSS, RDF/XML, SyncML and many others. Because of XML's formal definition, applications can validate documents transmitted in XML without prior knowledge of their structure.

1.8.2 Features of XML

XML is text based, that is all information is stored as text, divided by special markers that dictate the hierarchy of the document. The major components that are used in an XML file containing data are the element which can be thought of as container like structures that include attributes about them. For example, imagine a table in a database containing the demographics information for the personnel of an organization. Below are the first five records in this table and are presented in a table-like structure.

PERSON_ID	FIRSTNAME	MIDNAME	SURNAME	SSN	DATE OF BIRTH	WORK_PHONE
0	Panayiotis	G	Andreou	123456	01/01/1978	+357 (22) 123456
1	Andreas	K	Georgiou	123789	01/01/1979	+357 (22) 123789
2	Dimitris	G	Sotiriou	456789	01/01/1980	+357 (22) 456789
3	Kostas	H	Michael	789456	01/01/1981	+357 (22) 789456
4	Giorgos	A	Prodromou	789123	01/01/1982	+357 (22) 789123

In our example, an element node may be considered each row of the above result set. Each column value is considered an attribute of the element. Translated to XML we may produce one the following XML file.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<PERSON>
 <row  PERSON_ID="0"
             FIRSTNAME="Panayiotis"
             MIDDLENAME="G"
             SURNAME="Andreou"
             SSN="123456"
             DATE_OF_BIRTH="01/01/1978"
             WORK_PHONE="+357 (22) 123456" />

 <row  PERSON_ID="1"
             FIRSTNAME="Andreas"
             MIDDLENAME="K"
             SURNAME="Georgiou"
             SSN="123789"
             DATE_OF_BIRTH="01/01/1979"
             WORK_PHONE="+357 (22) 123789" />

 <row  PERSON_ID="2"
             FIRSTNAME="Dimitris"
             MIDDLENAME="G"
             SURNAME="Sotiriou"
             SSN="456789"
             DATE_OF_BIRTH="01/01/1980"
             WORK_PHONE="+357 (22) 456789" />

 <row  PERSON_ID="3"
             FIRSTNAME="Kostas"
             MIDDLENAME="H"
             SURNAME="Michael"
             SSN="789456"
             DATE_OF_BIRTH="01/01/1981"
             WORK_PHONE="+357 (22) 789456" />

 <row  PERSON_ID="4"
             FIRSTNAME="Giorgos"
             MIDDLENAME="A"
             SURNAME="Prodromou"
             SSN="789123"
             DATE_OF_BIRTH="01/01/1982"
             WORK_PHONE="+357 (22) 789123" />

</PERSON>
```

In another case, all the attributes can be represented as inner elements of the row element. This particular example is shown below:

```
<?xml version="1.0" encoding="utf-8" ?>
<PERSON>
 <row>
       <PERSON_ID>0</PERSON_ID>
       <FIRSTNAME>Panayiotis</FIRSTNAME>
       <MIDDLENAME>G</MIDDLENAME>
       <SURNAME>Andreou</SURNAME>
       <SSN>123456</SSN>
       <DATE_OF_BIRTH>01/01/1978</DATE_OF_BIRTH>
       <WORK_PHONE>+357 (22) 123456</WORK_PHONE>
 </row>
 <row>
       <PERSON_ID>1</PERSON_ID>
       <FIRSTNAME>Andreas</FIRSTNAME>
       <MIDDLENAME>K</MIDDLENAME>
       <SURNAME>Georgiou</SURNAME>
       <SSN>123789</SSN>
       <DATE_OF_BIRTH>01/01/1979</DATE_OF_BIRTH>
       <WORK_PHONE>+357 (22) 123789</WORK_PHONE>
 </row>
 <row>
       <PERSON_ID>2</PERSON_ID>
       <FIRSTNAME>Dimitris</FIRSTNAME>
       <MIDDLENAME>G</MIDDLENAME>
       <SURNAME>Sotiriou</SURNAME>
       <SSN>456789</SSN>
       <DATE_OF_BIRTH>01/01/1980</DATE_OF_BIRTH>
       <WORK_PHONE>+357 (22) 456789</WORK_PHONE>
 </row>
 <row>
       <PERSON_ID>3</PERSON_ID>
       <FIRSTNAME>Kostas</FIRSTNAME>
       <MIDDLENAME>H</MIDDLENAME>
       <SURNAME>Michael</SURNAME>
       <SSN>789456</SSN>
       <DATE_OF_BIRTH>01/01/1981</DATE_OF_BIRTH>
       <WORK_PHONE>+357 (22) 789456</WORK_PHONE>
 </row>
 <row>
       <PERSON_ID>4</PERSON_ID>
       <FIRSTNAME>Giorgos</FIRSTNAME>
       <MIDDLENAME>A</MIDDLENAME>
       <SURNAME>Prodromou</SURNAME>
       <SSN>789123</SSN>
       <DATE_OF_BIRTH>01/01/1982</DATE_OF_BIRTH>
       <WORK_PHONE>+357 (22) 789123</WORK_PHONE>
 </row>
</PERSON>
```

Both representations assume that the target computing host has the “knowledge” of what is the data type for each attribute. If that is not the case then an XML Schema definition must be sent to the target computing host or else the data will be useless. The XML Schema Definition (XSD) contains the description of the data (meta-data in DBMS terms) that are stored in the XML file.

A possible XSD file for our XML file above is the following:

```
<?xml version="1.0" ?>
<xs:schema
      id="PERSON"
      targetNamespace="http://localhost/Top_5_PERSON_elements.xsd"
      xmlns:mstns="http://localhost/Top_5_PERSON_elements.xsd"
      xmlns="ttp://localhost/Top_5_PERSON_elements.xsd"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
      attributeFormDefault="qualified"
      elementFormDefault="qualified">

      <xs:element name="PERSON"
                  msdata:IsDataSet="true"
                  msdata:Locale="en-GB"
                  msdata:EnforceConstraints="False">
       <xs:complexType>
             <xs:choice maxOccurs="unbounded">
                   <xs:element name="row">
                         <xs:complexType>
                               <xs:sequence>
                                          <xs:element
                                                name="PERSON_ID"
                                                type="xs:int"
                                                minOccurs="0" />
                                          <xs:element
```

```
                                                name="FIRSTNAME"
                                                type="xs:string"
                                                minOccurs="0" />
                                          <xs:element
                                                name="MIDDLENAME"
                                                type="xs:string"
                                                minOccurs="0" />
                                          <xs:element
                                                name="SURNAME"
                                                type="xs:string"
                                                minOccurs="0" />
                                          <xs:element
                                                name="SSN"
                                                type="xs:string"
                                                minOccurs="0" />
                                          <xs:element
                                                name="DATE_OF_BIRTH"
                                                type="xs:date"
                                                minOccurs="0" />
                                          <xs:element
                                                name="WORK_PHONE"
                                                type="xs:string"
                                                minOccurs="0" />
                               </xs:sequence>
                         </xs:complexType>
                   </xs:element>
             </xs:choice>
       </xs:complexType>
 </xs:element>
</xs:schema>
```

Translated in English, this XSD file illustrates that the XML file contains data that consist of seven columns named

- PERSON_ID
- FIRSTNAME
- MIDNAME
- SURNAME
- SSN
- DATE_OF_BIRTH
- WORK_PHONE,

and their data-types are

- Integer
- String
- String
- String
- String
- Date
- String

Having said all that, we can compare the XML definition to the LISP programming language and conclude that they are very similar to LISP's statements where in tree structures each node contains its own property list.

The following conclusion can be easily derived from the above example. Everything in XML is strings. An XML document is a combination of strings and special characters that define the structure of the XML file as well as its data. Each document consists of one or more entities, each of which consists of characters or series of characters and is stored in a simple text file.

Few languages, prior to XML, were general purpose, very easy to learn as well as internet protocol friendly. In fact, the vast majority of them were binary in format, based on bit sequences. In addition, the target should have prior knowledge of the format that is going to be transmitted.

In contrast, by allowing hierarchy, data meta-type description, elements, attributes and naming conventions, XML provides the means of creating custom markup languages based on XML. Each document must "obey" the syntactic rules of XML and thus assuring that all XML-aware applications can parse and comprehend the arrangement of data in an XML file. The XSD file supplements the syntactic representation of an XML file by enforcing constraints to the syntax rules and thus assuring data consistency between computing parties. In our example, the XSD file adds data-type constraints for each column in the XML document. In this way, it assures that the computing applications will for example provide an integer for the PERSON_ID attribute/column, a date field for the DATE_OF_BIRTH attribute, and strings for all other columns. Of course, more complex constraints exist.

A common misconception is that XML and HTML are similar. On the contrary, HTML and XML are very little alike. HTML is a general purpose language that dictates, how the document is rendered, that is displayed, whereas XML is the actual data and their description.

Another very important difference is the strictness embodied in the syntax of XML. Because of this formal syntax, it is much easier to develop software applications that are able to access an XML document's information, since the data structures are expressed in a formal relatively simple way.

XML makes no prohibitions on how it is used. Although XML is fundamentally text-based, software quickly emerged to abstract it into other, richer formats, largely through the use of datatype-oriented schemas and object-oriented programming paradigms (in which the document is manipulated as an object). Such software might only treat XML as serialized text when it needs to transmit data over a network, and some software doesn't even do that much. Such uses have led to "binary XML", the relaxed restrictions of XML 1.1, and other proposals that run counter to XML's original spirit and thus garner an amount of criticism.

1.8.3 Advantages/Disadvantages of XML

The advantages of XML which we consider relevant for our application are summarized below.

- XML is both human and machine readable
- XML support Unicode, therefore it support communication of information in any language
- XML has the ability to represent the most generic data structures, that is records, list and trees
- XML is self-documented; it describes the field names, structures, data types as well as specific constraints and values
- XML has strict syntax and parsing requirements; as a result, all parsing applications remain simple, efficient and consistent
- XML is widely used for storing and processing documents, both online and offline
- XML is based on international standards
- XML can be verified through its format and therefore is robust
- XML supports a hierarchical representation of data
- XML is license-free since it manifests as plain text files
- XML is platform independent
- XML is being used since 1986, so a lot of experience has been gained from already existing applications and there is a variety of XML supporting software application available

The disadvantages of XML are summarized in the points below.

- The XML syntax is highly redundant. Elements constantly enclose the attribute names. This fact causes a drop in application efficiency. Sometimes it possible to affect the human readability factor as well. In cases where bandwidth efficiency is important, XML is probably not a good choice. Of course, compression can solve this problem in some cases.
- Designing parsers for XML is not a trivial task if the XML document's format is not straightforward or too complex. Parsers must detect improperly formatted syntax and recursively traverse nested data structures.
- Security is essential when communicating with XML since it is text-based.
- XML by it self can not indicate what are the data types for each field. For example, the decimal number "0.5" could be interpreted as string. Of course, this can be solved with the attachment of an XSD file.
- The relational and object oriented paradigms can not be easily mapped to XML if no assumptions are made.
- XML can be used for data storage only if the files are of low volume.

Given the above advantages and disadvantages, we consider XML as a good candidate for transmitting data over a wireless medium. Next we will investigate its data load requirements.

1.8.4 Data load

To demonstrate the data load of transferring XML data from a database we have performed several readings. These readings aim to demonstrate the high volume storage weakness of XML. The following tests were performed using 3 different types of wireless network connections that are supported by various mobile devices, namely GSM, GPRS and WiFi.

The following figures demonstrate a simple scenario; the mobile application requests a number of records from the database server in XML format and the following measurements take place:

- o Download time (in seconds)
- o Write time of the XML data into file for later use (in seconds)
- o Total time (in seconds)
- o File size needed for the XML file (in bytes)

Two different tables were used for the measurements in order to give a more comprehensive view. Table 1 contains demographic information for some persons in a test database. Table 2 contains the calendar (appointments) record for those persons in the same database.

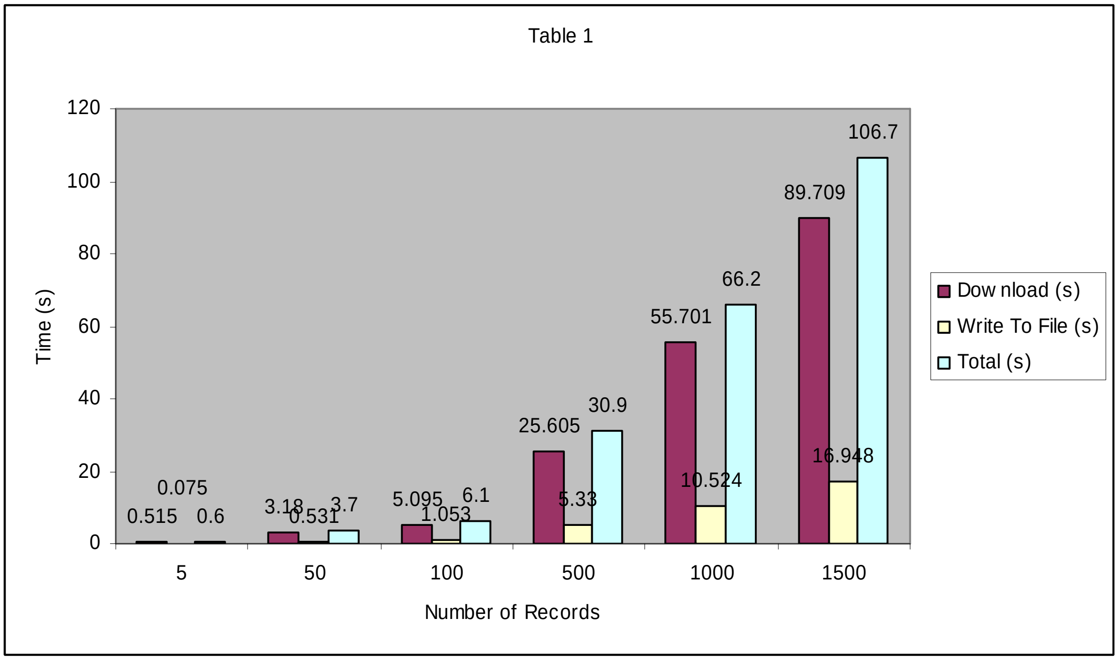


Figure 3: XML data loading times with different number of records

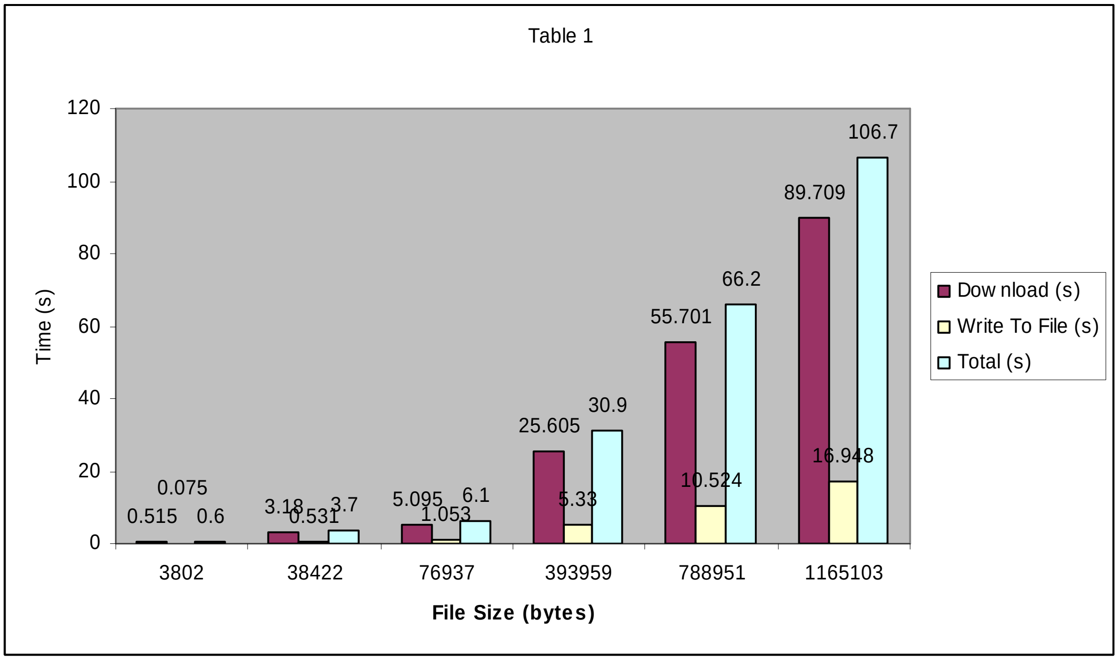


Figure 4: XML data loading times/ File size requirements

An obvious conclusion that can be made from Figure 3: XML data loading times with different number of records and Figure 4: XML data loading times/ File size requirements, is that as the application demands increase (number of records increases) the response time (download time) increases in a rate that is not analogous to the records transferred. This fact could render an application useless.

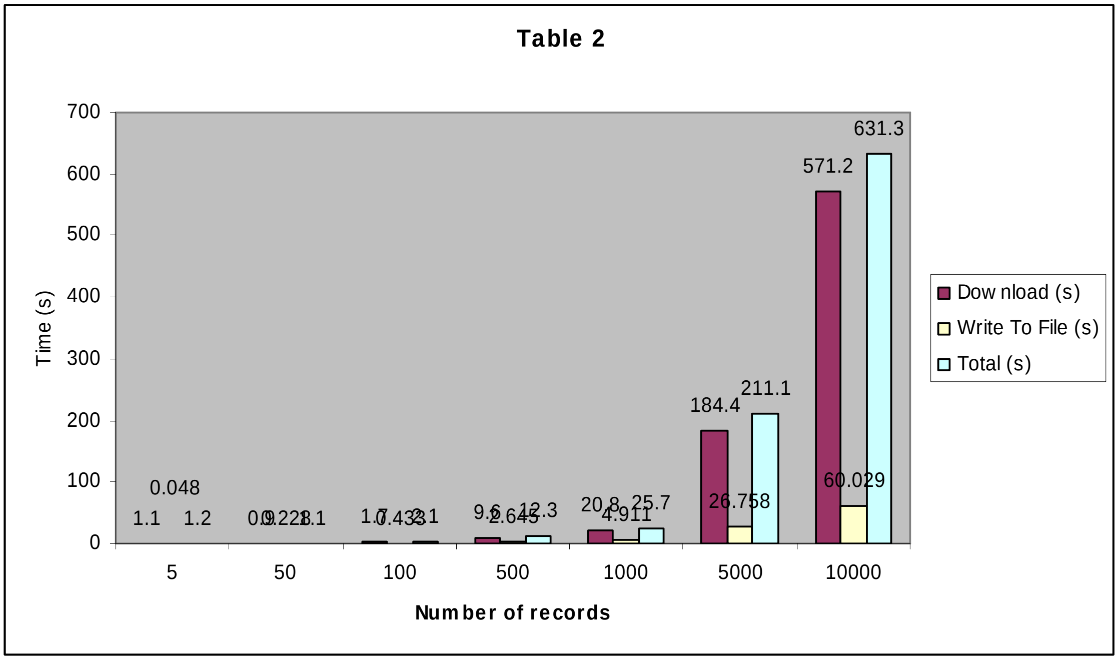


Figure 5: XML data loading times with different number of records

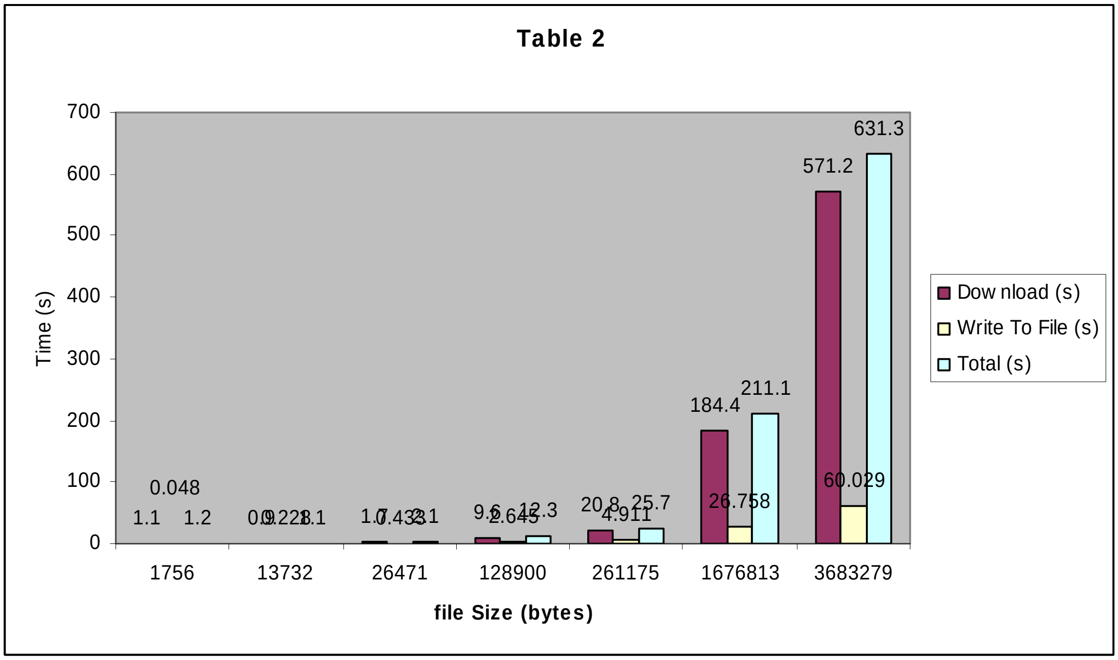


Figure 6: XML data loading times/ File size requirements

The same results apply to Figure 5: XML data loading times with different number of records and Figure 6: XML data loading times/ File size requirements. In this example we can see that the response time for 10000 records is nearly 10.5 minutes! Imagine the user having to wait 10.5 minutes for a result, and note that we are using WiFi!

According to Figure 3-Figure 6, the loading time exhibits exponential increase tendencies as the number of records increases. In the case of 10000 records the loading time is as high as 631 seconds. This means that the user will wait 10.5 minutes until the download is complete. The 3.5MG that are downloaded will cost an average of 6 - 8 CYP[*] in Cyprus. Imagine if this process takes place 5 or more times a day. The application will cost 30 - 40CYP per day and user!

1.9 Web services

One of today's trends is the transformation of desktop applications to platform independent web applications. This process involves the dismantling of desktop applications into procedures and methods and integrating them into a web environment so as to be accessible from any browser enabled host.

Similarly to components, web services provide an interface for methods or services to developers that can be used without worrying how they are implemented. These interfaces are also called contracts and describe the services that they provide as well as a list of the parameters and data types that they require both for input and output.

In this way, following the component based software cycle paradigm[5,6], developers can create their own applications by simply gluing the different pieces (methods/service) together, just like a kid playing with LEGO. Imagine that you want to build a web application for handling the online orders for an organization. A developer could use the following remote services for this particular application:

- For authentication, a passport service (e.g. Microsoft Passport) can be used for authenticating the users based on email and password.
- For the online transactions, let's assume that a credit card service will be used
- An inventory/stock management service could be used for handling the organization's DVD database

Unlike current component technologies, however, Web services do not use protocols that are specific to certain object models, such as Distributed Component Object Model (DCOM), which requires specific, homogeneous infrastructures on the computers that run the client and the server. Web services communicate by using standard Web protocols and data formats such as HTTP, XML and SOAP. Any system that supports these Web standards can support web services.

[*] CYP: Cyprus Pounds

1.9.1 What is a web service?

A Web service, according to the W3C, is remote software system that is designed to support requests from any HTTP enabled host in a network. Web Services have interfaces that are written or automatically generated from an IDE[†] and defines the protocol that a method of the web service is used. This interface is described by WSDL files. If any hosts wants to invoke a web service, it prepares a message according to the interface and encloses it in a SOAP envelope.

SOAP is a protocol for the exchange of XML formatted messages over a network using HTTP. SOAP forms the foundation layer of the Web services stack, providing a basic messaging framework which more abstract layers can build on. SOAP can be used to facilitate a Service-Oriented architectural pattern.

There are several different types of messaging patterns in SOAP, but by far the most common is the Remote Procedure Call (RPC) pattern, where one network node (the client) sends a request message to another node (the server), and the server immediately sends a response message to the client. Indeed, SOAP is the successor of XML RPC.

These messages are typically conveyed using HTTP, and normally comprise XML in conjunction with other Web-related standards. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (for example, between Java and Python, or Microsoft Windows and Linux applications) is due to the use of open standards. OASIS and the W3C are the primary committees responsible for the architecture and standardization of web services. To improve interoperability between web service implementations, the WS-I organization has been developing a series of profiles to further define the standards involved.

1.9.2 Web service invocation

Figuratively speaking, invoking a web service is similar to going to a restaurant for dinner and ordering something from the menu. In this case, requesting the menu from a waiter is similar to the access of the WSDL file on the web server. In the first case, the result will be the menu with different selections of food, and in the latter, the web service description which includes the list of methods provided by the web service.

[†] Integrated Development Environment (e.g. Visual Studio .NET, IBM VisualAge)

Finally, the order of a specific item on the menu is similar to executing a specific method form the web service. The process of finding the restaurant through the yellow pages is similar to looking up a web service in the UDDI[7] directory.

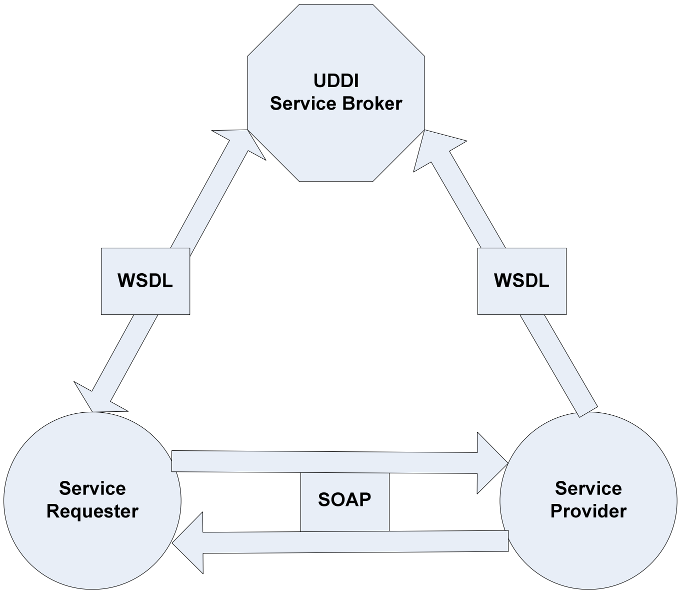


Figure 7: Invoking a web service

The Web Services Description Language (WSDL) is an XML format published for describing Web services. Version v1.1 has not been endorsed by the World Wide Web Consortium (W3C), however it has released several drafts for version 2.0, that will be a W3C recommendation, and thus endorsed by the W3C.

It is commonly abbreviated as WSDL in technical literature and is usually pronounced wiz-dell.

WSDL describes the public interface to the web service. This is an XML-based service description on how to communicate using the web service; namely, the protocol bindings and message formats required to interact with the web services listed in its directory. The supported operations and messages are described abstractly, and then bound to a concrete network protocol and message format.

WSDL is often used in combination with SOAP and XML Schema to provide web services over the internet. A client program connecting to a web service can read the WSDL to determine what functions are available on the server. Any special datatypes used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL.

1.9.3 Advantages/Disadvantages of a web service?

The advantages of a web service are summarized in the points below.

- Web services can be invoked from various software applications running on different platforms and operating systems. As a result, web services provide high interoperability
- Web services are based on XML, SOAP and other open standards. Since these standards are text-based and therefore human readable, developers can easily comprehend them
- Web services are invoked using HTTP. This does not require any change in current firewall or intrusion detection software settings. Note that most forms of RPC are blocked as a default in the majority of security systems
- Web services from different software companies and network hosts can be easily integrated in an application
- Web services can be built from already existing methods or components that are in use in an organization. This is a great help for developers in the fact that they do not have to rewrite code

The disadvantages of a web service are summarized in the points below.

- Web services are a relatively new technology and it can not be compared with already existing and well established middle tier platforms like CORBA. Of course, since the vast majority of vendors have committed to the OASIS standards to implement QoS, this fact is likely to diminish
- Since Web services rely in XML they also inherit its disadvantages. XML is text based and therefore very slow in most cases.

Given above advantages and disadvantages, we selected web services for the interoperability that they provide and the fact that every information exchanged between web services is in XML format.

1.10 Mobile devices

1.10.1Introduction

Mobile devices have evolved greatly in the last 10 years. When Palm Computing launched the first Palm Pilot 1000 in 1996 no one had foreseen the evolution that would follow. Weighing in at only 5.7 ounces, the Pilot wasn't the first handheld device to hit the market. But it was the first one to catch fire, becoming not only a market leader but a cultural phenomenon. The pocket-sized devices have organized information and lives in the boardroom, the emergency room and countless hotel rooms. It's even made it to the top of Mt. Everest and flown in the Space Shuttle!

Figure 8: Modern SmartPhones

In our study we consider a mobile device that we assume has an XML browser. A generic XML reader is relatively simple software that can be build using even the most old programming languages. This module will be responsible for transmitting XML requests and receiving XML responses with the remote server. Figure 9: XML receiver/Sender module displays the generic architecture for this model.

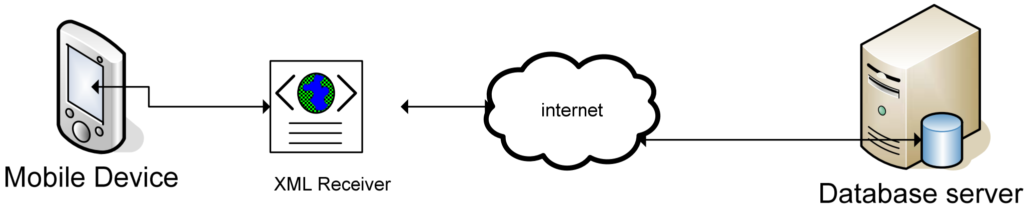


Figure 9: XML receiver/Sender module

1.10.2Device Selection

Since any device that can support a generic XML module can be used, we have selected a Pocket PC 2002 device.

Figure 10: Qtek 9090

1.11 Networks

1.11.1Mobile/wireless networks

Mobile/wireless networks play a vital role when it comes to mobile computing. Network connectivity can sometimes render a mobile application useless due to the disconnection or weak signal strength that may occur. In the following sections we will examine the wireless networks supported in most countries and organizations so as to give an idea of what are the differences between them.

1.11.1.1 GSM

The Global System for Mobile Communications (GSM) is the most popular standard for mobile phones in the world. GSM service is used by over 1.5 billion people across more than 210 countries and territories. The ubiquity of the GSM standard makes international roaming very common between mobile phone operators, enabling subscribers to use their phones in many parts of the world. GSM differs significantly from its predecessors in that both signaling and speech channels are digital, which means that it is considered a second generation (2G) mobile phone system. This fact has also meant that data communication was built into the system from very early on. GSM is an open standard which is currently developed by the 3GPP.

1.11.1.2 GPRS

General Packet Radio Service (GPRS) is a mobile data service available to users of GSM mobile phones. It is often described as "2.5G", that is, a technology between the second (2G) and third (3G) generations of mobile telephony. It provides moderate speed data transfer, by using unused TDMA channels in the GSM network. Originally there was some thought to extend GPRS to cover other standards, but instead those networks are being converted to use the GSM standard, so that is the only kind of network where GPRS is in use. GPRS is integrated into GSM standards releases starting with Release 97 and onwards.

1.11.1.3 UMTS

UMTS stands for "Universal Mobile Telecommunications System", and represents an evolution in terms of capacity, data speeds and new service capabilities from second generation mobile networks.

Today, more than 60 3G/UMTS networks using WCDMA technology are operating commercially in 25 countries, supported by a choice of over 100 terminal designs from Asian, European and US manufacturers. Japanese operator NTT DoCoMo launched the world's first commercial WCDMA network in 2001.

3G/UMTS in its initial phase offers theoretical bit rates of up to 384 kbps in high mobility situations, rising as high as 2 Mbps in stationary/nomadic user environments.

1.11.1.4 Wi-Fi

Wi-Fi is a brand licensed by the Wi-Fi Alliance to products which pass testing demonstrating that they implement a set of product compatibility standards for wireless local area networks (WLAN) based on the IEEE 802.11 specifications. New standards beyond the 802.11 specifications, such as 802.16 (WiMAX), are currently in the works and offer many enhancements, anywhere from longer range to greater transfer speeds.

Wi-Fi was intended to be used for mobile devices and LANs, but is now often used for Internet access. It enables a person with a wireless-enabled computer or personal digital assistant (PDA) to connect to the Internet when in proximity of an access point. The geographical region covered by one or several access points is called a hotspot.

1.11.1.5 Wireless networks speed comparison

The following table demonstrates the speed disparities between the different network types

Network	**Data Rate (KB)**
GSM	9.6
GPRS	172 (theoretical limit)
UMTS	2000 (theoretical limit)
WiFi	11000 or 54000

1.12 Compression

1.12.1 Widely accepted compression algorithms

The process of encoding data, using fewer bits (or other information-bearing units) than any other non-encoded representation would use is called data compression or source coding, in computer science and information theory. For instance this thesis could be encoded using fewer bits by just accepting the convention that the word "compression" would be substituted by the shortcut "comp". The ZIP file format is one of the most popular instances of compression, which not only provides compression but also acts as an archiver, by storing a series of files in just a single output file.

It is though necessary, in order to establish a compressed data communication, that both ends participating in this communication, are familiar of the encoding scheme used. An example for illustrating exactly this is that this text makes sense only if the receiver knows that it is intended to be interpreted as characters representing the English language. Hence, in order to be able to understand compressed data, the receiver must know the decoding method.

What makes compression possible in real world data scenarios is the fact that such data are characterized by statistical redundancy. What is meant by this is the fact that, for example the letter 'e' is much more common than the letter 'z' in English text whereas the probability that the letter 'q' will be followed by the letter 'z' is rather small. This statistical redundancy is usually exploited by lossless compression algorithms, so that the data sent are represented in a more efficient but nevertheless perfect way.

In the following sections, we will mainly focus on popular lossless compression algorithms that can be employed to XML data.

1.12.1.1 Huffman Coding

In computer science, Huffman coding is an entropy encoding algorithm used for lossless data compression. The term refers to the use of a variable-length code table for encoding a source symbol (such as a character in a file) where the variable-length code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol. It was developed by David A. Huffman as a Ph.D. student at MIT in 1952, and published in A Method for the Construction of Minimum-Redundancy Codes[8].

Huffman coding uses a specific method for choosing the representation for each symbol, resulting in a prefix-free code (that is, the bit string representing some particular symbol is never a prefix of the bit string representing any other symbol) that expresses the most common characters using shorter strings of bits than are used for less common source symbols. Huffman was able to design the most efficient compression method of this type: no other mapping of individual source symbols to unique strings of bits will produce a smaller average output size when the actual symbol frequencies agree with those used to create the code. (Huffman coding is such a widespread method for creating prefix-free codes that the term "Huffman code" is widely used as a synonym for "prefix-free code" even when such a code was not produced by Huffman's algorithm.)

For a set of symbols with a uniform probability distribution and a number of members which is a power of two, Huffman coding is equivalent to simple binary block encoding.

Assertions of the optimality of Huffman coding should be phrased carefully, because its optimality can sometimes accidentally be over-stated. For example, arithmetic coding ordinarily has better compression capability, because it does not require the use of an integer number of bits for encoding each source symbol. LZW coding can also often be more efficient, particularly when the input symbols are not independently-distributed, because it does not depend on encoding each input symbol one at a time (instead, it batches up a variable number of input symbols into each encoded syntax element). The efficiency of Huffman coding also depends heavily on having a good estimate of the true probability of the value of each input symbol.

1.12.1.2 RLE – Run Length Encoding

Run-length encoding (RLE) is a very simple form of data compression in which runs of data (that is, sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run. This is most useful on data that contains many such runs; for example, simple graphic images such as icons and line drawings.

For example, consider a screen containing plain black text on a solid white background. There will be many long runs of white pixels in the blank space, and many short runs of black pixels within the text. Let us take a hypothetical single scan line, with B representing a black pixel and W representing white:

WWWWWWWWWWWWBWWWWWWWWWWWWBBBWWWWWWWWWW
WWWWWWWWWWWWWWBWWWWWWWWWWWWWW

If we apply a simple run-length code to the above hypothetical scan line, we get the following:
12WB12W3B24WB14W

Interpret this as twelve W's, one B, twelve W's, three B's, etc. The run-length code represents the original 67 characters in only 16. Of course, the actual format used for the storage of images is generally binary rather than ASCII characters like this, but the principle remains the same. Even binary data files can be compressed with this method; file format specifications often dictate repeated bytes in files as padding space. However, newer compression methods such as deflation often use LZ77[9]-based algorithms, a generalization of run-length encoding that can take advantage of runs of strings of characters (such as BWWBWWBWWBWW).

Common formats for run-length encoded data include PackBits, PCX and ILBM.

Run-length encoding performs lossless data compression and is well suited to palette-based iconic images. It does not work well at all on continuous-tone images such as photographs, although JPEG uses it quite effectively on the coefficients that remain after transforming and quantizing image blocks.

1.12.1.3 RLBE – Run Length Binary Encoding

Run-length binary encoding (RLBE) is very similar to RLE but deals with binary data.

1.12.1.4 LZW

LZW[10] (Lempel-Ziv-Welch) is a universal lossless data compression algorithm created by Abraham Lempel, Jacob Ziv, and Terry Welch. It was published by Welch in 1984 as an improved implementation of the LZ78[11] algorithm published by Lempel and Ziv in 1978. The algorithm is designed to be fast to implement, but not necessarily optimal since it does not perform any analysis on the data.

The compressor algorithm builds a string translation table from the text being compressed. The string translation table maps fixed-length codes (usually 12-bit) to strings. The string table is initialized with all single-character strings (256 entries in the case of 8-bit characters). As the compressor character-serially examines the text, it stores every unique two-character string into the table as a code/character concatenation, with the code mapping to the corresponding first character. As each two-character string is stored, the first character is outputted. Whenever a previously-encountered string is read from the input, the longest such previously-encountered string is determined, and then the code for this string concatenated with the extension character (the next character in the input) is stored in the table. The code for this longest previously-encountered string is outputted and the extension character is used as the beginning of the next string.

The decompressor algorithm only requires the compressed text as an input, since it can build an identical string table from the compressed text as it is recreating the original text. However, an abnormal case shows up whenever the sequence character/string/character/string/character (with the same character for each character and string for each string) is encountered in the input and character/string is already stored in the string table. When the decompressor reads the code for character/string/character in the input, it cannot resolve it because it has not yet stored this code in its table. This special case can be dealt with because the decompressor knows that the extension character is the previously-encountered character.

Since we are dealing with XML data we will need a compression algorithm that works best with text based data. As a result Huffman coding is considered a good candidate as it produces high compression ratios for text-based data. Extensive Huffman will also be studied for its ability to code strings instead of characters. This particularly useful in XML as a lot of XML tags are constantly repeated throughout XML files.

1.13 Security

SEE COMMENTS LATERS AS TO WHETHER YOU NEED PARTS OF THIS SECTION.

1.13.1Introduction

“A security system is as strong as its weakest link”. Indeed, this is a defacto for security mechanisms. The objectives of security are to establish procedures and mechanisms to protect sensitive data and systems. For example it includes such procedures for preventing:

- Unauthorized access
- DoS (Denial of Service) attacks
- Interception of data

1.13.2Security mechanisms

In the following sections we will briefly describe the security mechanisms that appear in commercial software systems as well as some considerations with regards to mobile devices.

1.13.2.1 Authentication

Authentication (Greek: αυθεντικός, from 'authentes'='author') is the act of establishing or confirming something (or someone) as authentic, that is, that claims made by or about the thing are true. Authentication of an object may mean confirming its provenance. Authentication of a person often consists of verifying their identity.

Below is a brief description of most authentication methods:

- Username/Password

  This is the simplest form of authentication. A password is a form of secret authentication data that is used to control access to a resource. The password is kept secret from those not allowed access, and those wishing to gain access are tested on whether or not they know the password and are granted or denied access accordingly.

  Username and password can be of course used together with other security mechanisms like encryption.

- Public Key Infrastructure (PKI)

  PKI ensures a secure method for exchanging sensitive information over unsecured networks. In addition, PKI provides authenticated, private and non-reputable communications.

  PKI makes use of the technology known as public key cryptography. Public key cryptography uses a pair of keys to scramble and decipher messages, a public key and a private key. The public key is widely distributed whereas the private key is held secretly by an individual. Messages are protected from malicious people by scrambling them with the public key of the recipient. Only the recipient can decrypt the message by using his / her private key, thus retaining the privacy of the message. The public key is distributed with a digital certificate that contains information that uniquely identifies an individual (for example name, email address, the date that the certificate was issued, and the name of the certificate authority which issued it). Also, by using digital certificates we can digitally sign messages to protect the integrity of the information itself and achieve non-repudiation (digitally signing a transaction is legally binding and no party can deny his /her participation).

1.13.2.2 Authorization

Authorization is a part of the operating system that protects computer resources by only allowing those resources to be used by resource consumers that have been granted authority to use them. Resources include individual files or items data, computer programs, computer devices and functionality provided by computer applications. Examples of consumers are computer users, computer programs and other devices on the computer.

The authorization process is used to decide if person, program or device X is allowed to have access to data, functionality or service. Various authorization methods exist and are summarized below:

- Role based access

  A system administrator can specify what resources or services a user may or may not use. In database systems this is translated to permission to SELECT/INSERT/DELETE/UPDATE on a table/view or EXECUTE procedures

- Authorization tokens

  Authorization tokens are like authentication tokens but specify the level of access on data.

1.13.3Conclusions

Security mechanisms are a vital part of any application that transmits sensitive data over a network. As a result, a blending of security mechanisms must be implemented in order to avoid any type of attacks to the system.

Security mechanisms though do not come at small cost. Implementing even a simple encryption algorithm may need a lot processing power to run; mobile devices do not have desktop computer capabilities. As a result, we need to investigate the trade-off in performance versus the gain in providing adequate security. In this thesis we will only investigate encryption mechanisms to provide the secure transmission of data

SYNCHRONIZATION

1.14 Introduction

In the following sections we review existing commercial synchronization software that is applicable to mobile device synchronization and indicate their straightness and weaknesses. From this review we extract some basic knowledge of the synchronization schemes and the intervals in which the synchronization process handles. This knowledge motivates our proposed solution.

In the next chapter, we propose a new synchronization methodology, generated from our conclusions and point out its benefits as well as its limitations.

1.15 Commercial Mobile Database Systems

In the following section we analyze existing commercial mobile database systems. For each system, we present an overview of its synchronization architecture and analyze its synchronization algorithm.

This study presents strengthens and weakness of each system when used in a mobile device, and provide us with a clear path of what our synchronization platform should include or not.

1.15.1SQL Server CE

SQL Server CE extends one of Microsoft’s most established Database Management Systems (DBMS), which is Microsoft SQL Server, to Microsoft Windows CE-based mobile devices.

SQL Server CE provides developers the means to rapidly develop database applications for mobile devices running the windows platform. For succeeding this, it includes a footprint relational database which provides substantial database functionality as well as a robust data store, an optimizing query processor; and reliable, scalable connectivity capabilities.

This compact database is installed on the Windows CE mobile device and synchronizes with a master database running SQL Server using the ActiveSync.

The following illustration shows the relationship of three typical environments in which SQL Server CE can be used.

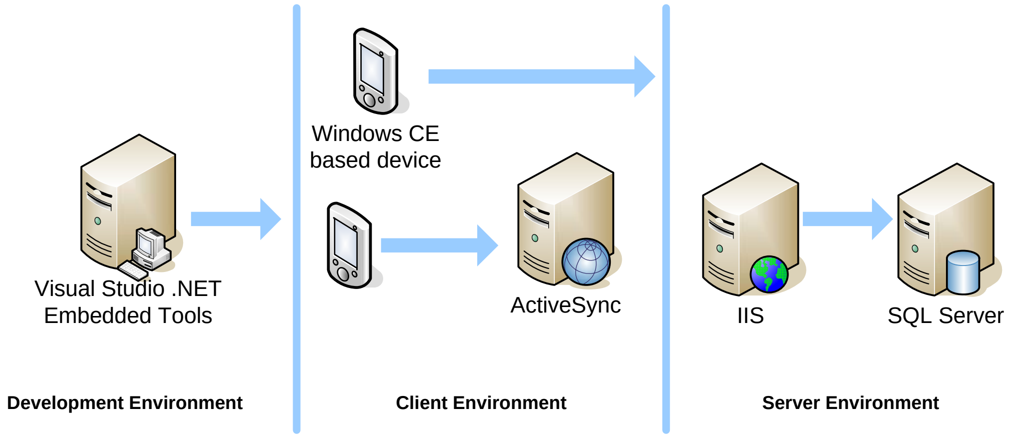


Figure 11: Microsoft SQL Server CE Edition Architecture

The main architecture components are described below:

- Development Environment

  The development environment includes the computer that the Windows CE application is developed. It includes a version of the Visual Studio tools. Specifically, Windows CE applications can be developed using one of the tools below:

  - Microsoft Visual Studio .NET
  - Microsoft eMbedded Visual Tools version 3.0 installed with at least one of the following software development kits (SDKs): Handheld PC 2000 SDK, Pocket PC SDK, or the Windows Powered Pocket PC 2002 SDK.

- Client Environment

  The Client environment includes the Windows CE powered devices that the mobile applications will be deployed. Windows CE can use ActiveSync to connect to the fixed host via cradle. If a cradle is not available, RemoteActiveSync can be used, using any network connection available.

- Server Environment

  The Server environment includes a web server running Microsoft Internet Information Services (IIS) and a database server running Microsoft SQL Server. Of course, only one computer can be used that is both configured as a web server and a database server. IIS is use for remote data access (RDA) with SQL Server.

SQL Server CE provides three types of publication/replication[12]:

1. Snapshot Publication
2. Transactional Publication

3. Merge Replication

1.15.1.1 Snapshot Publication

Snapshot replication is also known as static replication. Snapshot replications copies and distributes data and database objects exactly as they appear at current moment in time.

- The changes to data at subscriber are not updated to the subscriber continuously
- Subscribers are updated with complete modified data and not by individual transactions
- Propagating the changes to the subscribers takes more time as it is one time process or scheduled process.

1.15.1.2 Transactional Publication

Transactional replication is also known as dynamic replication. In transactional replication, modifications to the publication at the publisher are propagated to the subscriber incrementally.

- Publisher and the subscriber are always in synchronization
- Transaction boundaries are preserved, i.e. if there is modification to 5 rows of data, either all the 5 modified rows are propagated to the subscriber or none
- The publisher and the subscriber should be connected always.

1.15.1.3 Merge Replication

SQL Server CE merge replication uses this process:

1. Data is published.

   You create a publication containing SQL Server data that will be published to other sites, including Microsoft Windows CE-based devices running SQL Server CE. (A publication is a collection of articles. An article is a table that is enabled for replication.)
2. Subscriptions to the publications are created.

   A Windows CE-based application subscribes to the publication by using the SQL Server CE Replication object with Microsoft eMbedded Visual Tools or the SqlCeReplication class with the .NET Compact Framework Data Provider for SQL Server CE. When the subscription is created, the initial snapshot is downloaded from the Distributor to create the subscription database on the Windows CE-based device.
3. Data at the Subscriber is updated.

The subscription database on the Windows CE-based device is updated by applications running on that device.

4. Data is synchronized.

   Periodically, updates made at the Subscriber are sent to the Publisher and merged with updates made at the Publisher and updates propagated to the Publisher from other Subscribers. Similarly, changes made at the Publisher and other Subscribers since the initial download or the most recent merge are sent to the Windows CE-based device, where they are merged into the subscription database.

1.15.1.4 Publishing Data

Publications specify the data that is published. Publications are tailored to different users or groups of users. In some cases, all users need exactly the same data. For example, every employee might need a copy of the company employee directory. In other cases, different groups of users need different partitions of data. For example, sales representatives might need one set of data, and customer support technicians need a different set of data. Individuals might also need data specifically filtered for them. For example, a sales representative might need the data for his or her own customer accounts.

You can create a publication and specify which articles it contains. Although SQL Server publications may contain other database objects, such as stored procedures, views, and user-defined functions, SQL Server CE replication ignores these objects and only includes tables in the SQL Server CE subscription. You specify which of the table rows and columns are included in the article.

1.15.1.5 Subscribing to Publications

After you define the publication, a Windows CE-based application can subscribe to it by calling the methods exposed by the SQL Server CE Replication object or the SqlCeReplication class on the Windows CE-based device. When the subscription is created, the initial snapshot from the Distributor is applied at the subscription database on the Subscriber.

1.15.1.6 Updating Data at the Subscriber

Windows CE-based applications can update the subscription database. Merge replication allows each subscription database to be updated autonomously. Updates can occur whether or not the Windows CE-based device is connected to the network and to the Publisher. Each

SQL Server CE database uses change tracking to keep track of INSERT, UPDATE, and DELETE statements made at the SQL Server CE Subscriber.

1.15.1.7 Synchronizing data

Typically, users periodically connect the Windows CE-based device to the network. This allows the Windows CE-based application to synchronize changes made at the Subscriber with changes made at the Publisher. The Windows CE-based application initiates synchronization by calling the synchronization methods exposed by the Replication object or the SqlCeReplication class. Synchronization is a four-step process:

1. Extract changes and create the input message file.

   The SQL Server CE Client Agent extracts all inserted, updated, and deleted records from the subscription database at the SQL Server CE Subscriber and propagates them to the SQL Server CE Server Agent through HTTP. The SQL Server CE Server Agent creates a new input message file on the computer running Microsoft Internet Information Services (IIS) and writes into that file the insert, update, and delete changes sent by the SQL Server CE Client Agent.

2. Run the SQL Server Reconciler process and apply changes to the publication database.

   When all requests have been written to the input message file, the SQL Server CE Server Agent initiates the SQL Server Reconciler. The SQL Server Reconciler loads the SQL Server CE Replication Provider, which reads the input message file and informs the SQL Server Reconciler of changes made to the SQL Server CE subscription database that must be applied to the publication database at the Publisher. During processing, the SQL Server Reconciler detects and resolves conflicts; a conflict occurs when more than one Subscriber or Publisher updates the same record.

   The SQL Server Reconciler resolves conflicts with conflict resolvers. Use the conflict resolvers provided with SQL Server to implement simple forms of conflict resolution for your SQL Server CE replication applications or write conflict resolvers to implement more sophisticated solutions.

3. Create the output message file.

   The SQL Server Reconciler informs the SQL Server CE Replication Provider of changes made at the Publisher that must be applied to the subscription database on the Windows CE-based device. The SQL Server CE Replication Provider writes these changes to an output message file it creates on the computer running IIS.

The SQL Server CE Client Agent processes both the input and output message files in logical blocks as it reads from or applies changes to the subscription database on the Windows CE-based device. By processing each message file in this way, the SQL Server CE Client Agent avoids writing the entire message file on the Windows CE-based device, conserving storage space.

4. Read the output message file and apply changes to the SQL Server CE subscription database.

   When the SQL Server Reconciler process is complete, the SQL Server CE Server Agent locates the output message file created by the SQL Server CE Replication Provider. This file contains the changes that have occurred at the Publisher and that must be applied to the subscription database on the Windows CE-based device. The SQL Server CE Server Agent reads the output message file and transmits it to the SQL Server CE Client Agent on the Windows CE-based device. The SQL Server CE Client Agent applies the changes from the output message file to the SQL Server CE subscription database.

After the SQL Server CE Client Agent has incorporated all changes into the subscription database on the Windows CE-based device and conflicts (if any) have been resolved, the publication and subscription databases are synchronized and data is converged. However, before data values would be identical at the Publisher and at the Subscribers (because updates can occur continuously), you would need to stop all updates and then run several merges.

1.15.1.8 Limitations

Although SQL Server provides substantial database management solutions through a small, footprint database it has some limitations.

- Interoperability with all mobile devices is not supported. That is, SQL Server CE specifically requires that the device is running Windows CE.
- Column-level synchronization. When synchronizing two records, both whole records are transmitted between mobile host and fixed host and not only the columns that were updated.
- Customizability of the synchronization engine is not provided

1.15.2Microsoft SQL Server 2005 Mobile Edition

1.15.2.1 Overview

Microsoft SQL Server 2005 Mobile Edition is the new Microsoft SQL Server 2005 Mobile Edition is the new version of Microsoft SQL Server CE provided by Microsoft. SQL Server

2005 Mobile Edition[13] is a mobile database solution designed for developers who target Microsoft Windows mobile-based devices. New features and enhancements in SQL Server Mobile focus on the following key areas:

- Integration with SQL Server 2005 and Microsoft Visual Studio 2005
- Increased reliability and performance
- Faster development of mobile applications

The fact that SQL Server 2005 Mobile Edition targets only Microsoft Windows mobile-based devices is a major disadvantage though since it obligates the organization to buy new hardware which may not be the case in case of an already existing infrastructure.

Microsoft SQL Server 2005 Mobile Edition (SQL Server Mobile) architecture includes both a development environment and a client and server environment. This section describes the components that make up each environment.

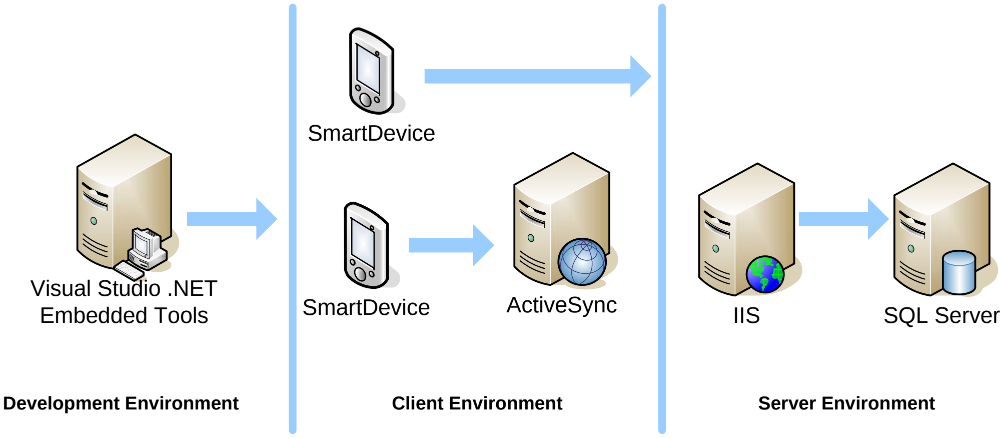


Figure 12: Microsoft SQL Server 2005 Mobile Edition Architecture

We provide an overview of the main architecture components below:

- Development Environment

  The development environment includes the computer on which applications are developed. This computer must have Microsoft Visual Studio 2005, including the .NET Compact Framework, to create applications for SQL Server Mobile. You can create managed applications by using either Microsoft Visual Basic or C#, or you can use Microsoft Visual C++ for Devices, previously referred to as Microsoft eMbedded Visual C++ 4.0, to create native applications.
- Client Environment

  In the SQL Server Mobile architecture, the client environment is made up of one or more supported devices on which the application and SQL Server Mobile are deployed. When the devices do not contain network connectivity, you can use Microsoft ActiveSync to connect SQL Server Mobile to the server environment.
- Server Environment

The server environment is made up of one or more computers that run Microsoft Internet Information Services (IIS) and an instance of Microsoft SQL Server or data propagated for a heterogeneous data source. You can run IIS and SQL Server on the same computer or configure them over multiple computers. IIS is required to connect to and exchange data between servers and clients.

1.15.2.2 Replication Publishing Model Overview

Replication uses a publishing industry metaphor to represent the components in a replication topology, which include Publisher, Distributor, Subscribers, publications, articles, and subscriptions. It is helpful to think of Microsoft SQL Server replication in terms of a magazine:

- A magazine publisher produces one or more publications
- A publication contains articles
- The publisher either distributes the magazine directly or uses a distributor
- Subscribers receive publications to which they have subscribed

Although the magazine metaphor is useful for understanding replication, it is important to note that SQL Server replication includes functionality that is not represented in this metaphor, particularly the ability for a Subscriber to make updates and for a Publisher to send out incremental changes to the articles in a publication.

A replication topology defines the relationship between servers and copies of data and clarifies the logic that determines how data flows between servers. There are several replication processes (referred to as agents) that are responsible for copying and moving data between the Publisher and Subscribers. The following illustration is an overview of the components and processes involved in replication.

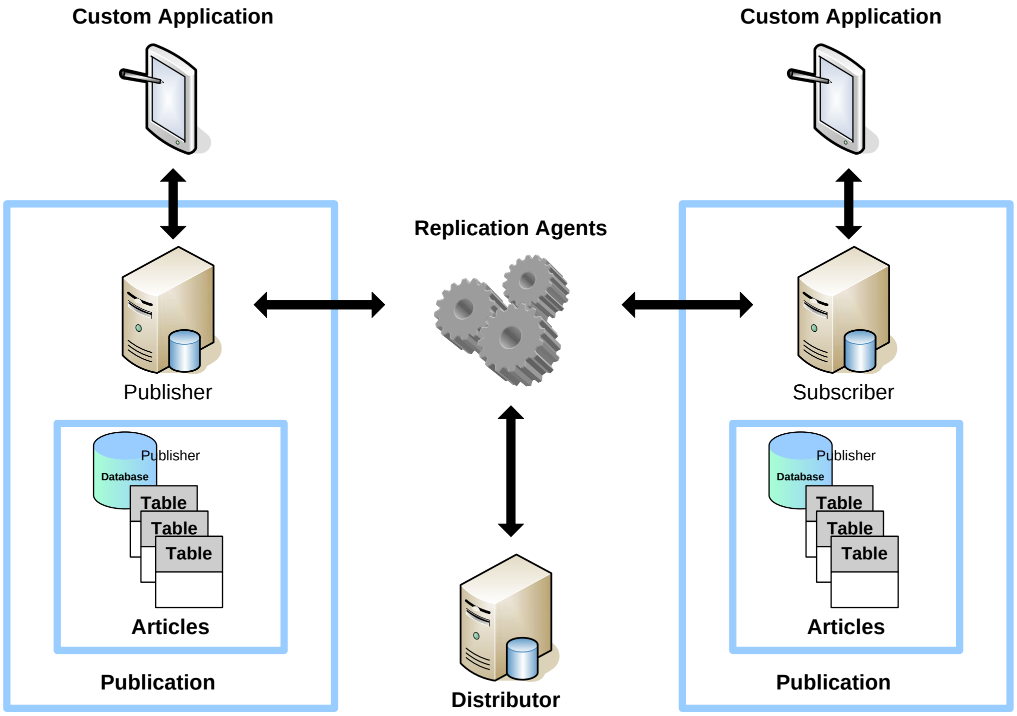


Figure 13: Microsoft SQL Server 2005 Mobile Edition Replication Publishing model overview

- Publisher

  The Publisher is a database instance that makes data available to other locations through replication. The Publisher can have one or more publications, each defining a logically related set of objects and data to replicate.

- Distributor

  The Distributor is a database instance that acts as a store for replication specific data associated with one or more Publishers. Each Publisher is associated with a single database (known as a distribution database) at the Distributor. The distribution database stores replication status data, metadata about the publication, and, in some cases, acts as a queue for data moving from the Publisher to the Subscribers. In many cases, a single database server instance acts as both the Publisher and the Distributor. This is known as a local Distributor. When the Publisher and the Distributor are configured on separate database server instances, the Distributor is known as a remote Distributor.

- Subscriber

  A Subscriber is a database instance that receives replicated data. A Subscriber can receive data from multiple Publishers and publications. Depending on the type of replication chosen, the Subscriber can also pass data changes back to the Publisher or republish the data to other Subscribers.

- Article

An article identifies a database object that is included in a publication. A publication can contain different types of articles, including tables, views, stored procedures, and other objects. When tables are published as articles, filters can be used to restrict the columns and rows of the data sent to Subscribers.

- Publication

  A publication is a collection of one or more articles from one database. The grouping of multiple articles into a publication makes it easier to specify a logically related set of database objects and data that are replicated as a unit.

- Subscription

  A subscription is a request for a copy of a publication to be delivered to a Subscriber. The subscription defines what publication will be received, where, and when. There are two types of subscriptions: push and pull.

1.15.2.3 Synchronizing Data

Synchronizing data refers to the process of data and schema changes being propagated between the Publisher and Subscribers after the initial snapshot has been applied at the Subscriber. Synchronization can occur:

- Continuously, which is typical for transactional replication
- On demand, which is typical for merge replication
- On a schedule, which is typical for snapshot replication

When a subscription is synchronized, different processes occur based on the type of replication you are using:

- Snapshot replication. Synchronization means that the Distribution Agent reapplies the snapshot at the Subscriber so that schema and data at the subscription database is consistent with the publication database. If modifications to data or schema have been made at the Publisher, a new snapshot must be generated in order to propagate modifications to the Subscriber.
- Transactional replication. Synchronization means that the Distribution Agent transfers updates, inserts, deletes, and any other changes from the distribution database to the Subscriber.
- Merge replication. Synchronization means that the Merge Agent uploads changes from the Subscriber to the Publisher and then downloads changes from the Publisher to the Subscriber. Conflicts, if any, are detected and resolved. Data is converged, and the Publisher and all Subscribers eventually end up with the same data values. If

conflicts were detected and resolved, work that was committed by some of the users is changed to resolve the conflict according to policies you define.

Snapshot publications completely refresh the schema at the Subscriber every time synchronization occurs, so all schema changes are applied to the Subscriber. Transactional replication and merge replication also support the most common schema changes.

1.15.2.4 Limitations

Microsoft SQL Server 2005 Mobile edition is a significant update to SQL Server CE Although SQL Server provides substantial database management solutions through a small, footprint database it has some limitations.

- Interoperability with all mobile devices is not supported. That is, SQL Server 2005 Mobile Edition requires that the device is running Windows CE.
- Customizability of the synchronization engine is not provided

1.15.3SyncML

SyncML[14,15] (Synchronization Markup Language) is the former name (currently referred to as: Open Mobile Alliance Data Synchronization and Device Management) for a platform-independent information synchronization standard. Existing synchronization solutions have mostly been somewhat vendor-, application- or operating system specific. The purpose of SyncML is to change this by offering an open standard as a replacement. Several major companies such as Motorola, Nokia, Sony Ericsson and Siemens AG already support SyncML in their products. Philippe Kahn was instrumental in the founding vision for synchronization with Starfish Software, later acquired by Motorola. The founding vision as expressed by Kahn was: "Global synchronization and integration of wireless and wireline devices".

SyncML is most commonly thought of as a method to synchronize contact and calendar information between some type of handheld device and a computer (personal, or network-based service), such as between a mobile phone and a personal computer. The new version of the specification includes support for push email, providing a standard protocol alternative to proprietary solution like BlackBerry.

Some products are now using SyncML for more general information synchronization purposes, such as to synchronize project task information across a distributed group of team members. SyncML can also be used as a base for backup solutions.

The current version of the protocol is 1.1.2. Version 1.2 is a candidate enabler and is available on OMA Release Program and Specifications page.

The SyncML specification defines seven synchronization types:

- Two-way sync

  This is the normal type of synchronization, in which client and server exchange information about any new changes to the data in their databases. This sync type is always initiated by the client, which sends a synchronization request to the server. The server then unifies the client's data and the data in the server's database and sends the unified data to the client. The client then updates its own database and sends back to the server any necessary mapping information.

- Slow sync

  In this form of two-way sync, all items in the client's database are compared with all items in the server's database, field by field. Either the client or the server may request such a sync if it loses its change log.

- One-way sync from client

  The server gets all changes from the client but doesn't send any changes back.

- Refresh sync from client

  The client sends all its data to the server, which is expected to replace all its own data with the client's data.

- One-way sync from server

  The client gets all changes from the server but doesn't send any changes back.

- Refresh sync from server

  The server sends all its data to the client, which is expected to replace all its own data with the server's data.

- Server-alerted sync

  The server notifies the client that there is a need to initiate a specific type of synchronization.

1.15.3.1 Limitations

SyncML is totally based on XML and thus any mobile device that has a generic XML parser can parse a SyncML packet. On the other hand, SyncML is not optimized for database record consistency that is it was built for synchronizing all the data between mobile device and computer including files, calendar, contacts and other. In addition the synchronization types are not user customizable and do not provide table level support. Finally, column level support is not provided.

1.15.4HotSync

HotSync [16] is the Palm's answer to synchronizing data between mobile devices and fixed hosts. Is supports synchronization via the cradle or remote synchronization via wireless network connection.

The HotSync specification[17,18] defines two synchronization types:

Non-schema databases on the handheld include flags that HotSync Manager uses to improve the efficiency of synchronization operations:

- a database "dirty" flag, which indicates whether the non-schema database has changed since the most recent synchronization
- record "dirty" flags, which indicate whether each record has been changed since the most recent synchronization

If HotSync Manager determines that it can perform a fast sync, conduits can rely on the validity of the "dirty" flags and therefore work only with database records that have changed.

Note that there are some situations in which a conduit must synchronize all of the records in an non-schema database, even if they have not been changed since the last HotSync operation:

- If a database has been deleted on either the handheld or desktop computer, all of the records in the surviving database must be transferred to the other during synchronization.
- When a new application is installed on a handheld or desktop computer, all of the corresponding data records must be transferred to the other.

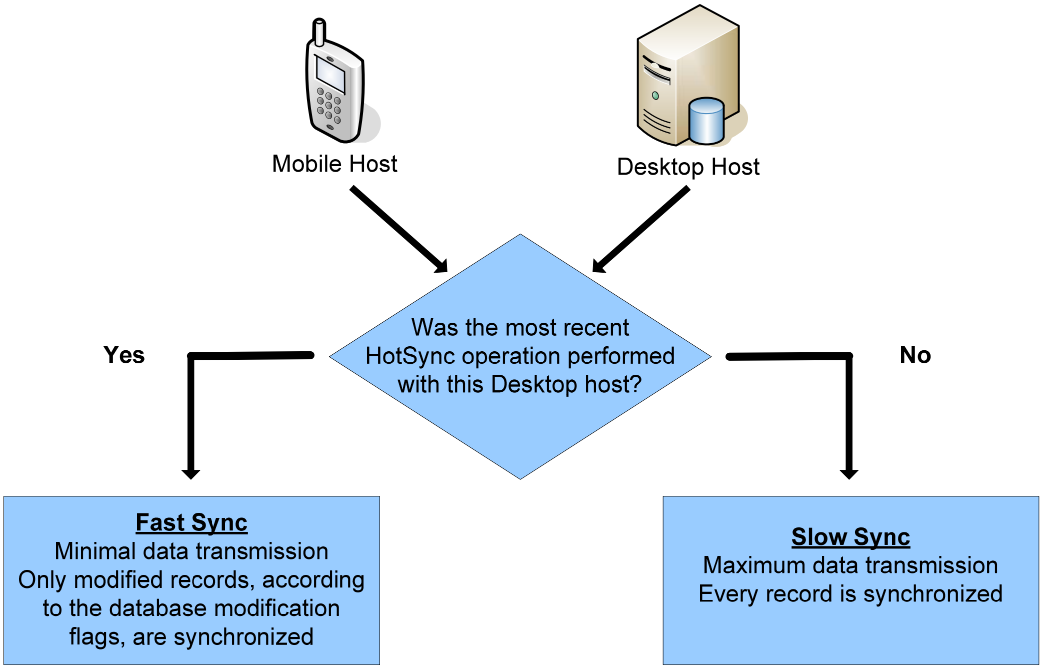

Mobile Host
Desktop Host
Was the most recent HotSync operation performed with this Desktop host?
Yes
No
Fast Sync
Minimal data transmission
Only modified records, according to the database modification flags, are synchronized
Slow Sync
Maximum data transmission
Every record is synchronized

Figure 14: HotSync's fast and slow synchronization

1.15.4.1 Limitations

HotSync provides a good synchronization scheme for mobile devices that supports two types of synchronization, Fast and Slow sync. If a user tries to sync with a different desktop computer then all the data are overwritten at the mobile device. This is a major security hazard especially in the case where the mobile device is stolen.

1.15.5IBM DB2 Everyplace

IBM DB2 Everyplace[19,20] features a small footprint relational database and high performance data synchronization solution that enables enterprise applications and data to be securely extended to mobile devices such as personal digital assistants (PDAs), smart phones and other embedded mobile devices. With DB2 Everyplace, the mobile work force in industries such as health care, telecommunications, retail, distribution, transportation and hospitality can now easily access the information they need to perform their work -- from any location, at any time, right from the palm of their hand. DB2 Everyplace can also be embedded into devices and appliances to increase their functionality and market appeal. DB2 Everyplace can be used as a local independent database of the mobile device or query information on remote servers when a connection is available.

1.15.5.1 The synchronization process

Mobile workers require a reliable software solution that allows them to access their organization's data locally on a mobile device, modify this data, and synchronize the changes with a database on a remote server in a timely fashion. IBM's DB2 Everyplace solution provides this capability, allowing two-way synchronization of data between the enterprise data source and mobile and embedded devices. DB2 Everyplace is also capable of one-way synchronization of files. DB2 Everyplace Sync Server can also manage one-way subscriptions where DB2 Everyplace only inserts data into the data source. The synchronization process has two major steps: v Mobile users submit changes that they made to local copies of source data. Users receive any changes that were made to source data residing on the enterprise server since the last time they synchronized. This two-step process is known as a synchronization session.

The synchronization process from Client to Server can be summarized in the figure below:

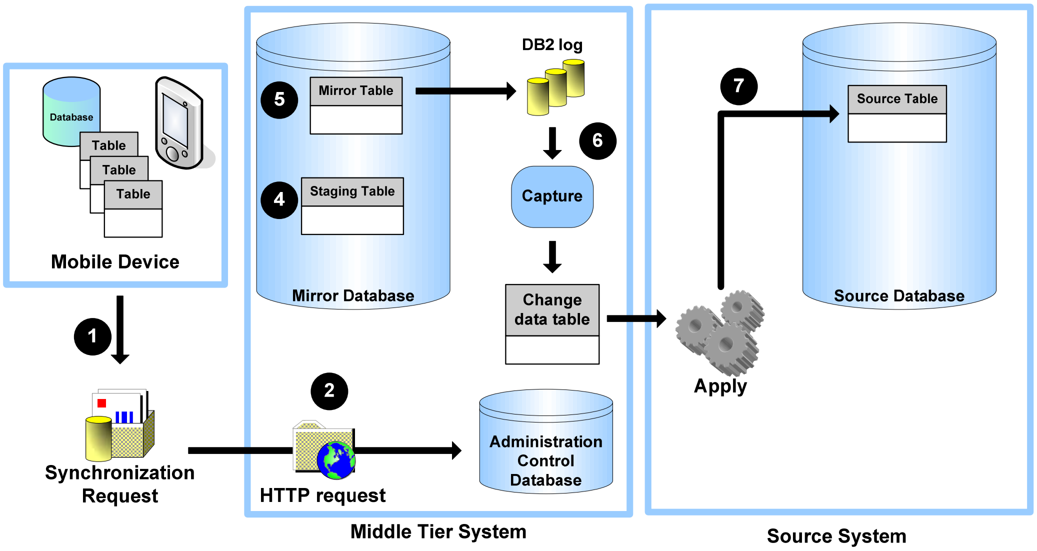


Figure 15: IBM DB2 synchronization (Client to Server)

1. The user initiates synchronization by tapping the IBM Sync icon. alternatively, synchronization can be requested automatically by an application using the IBM Sync Engine API. IBM Sync sends a synchronization request to the Sync Server.
2. The synchronization request is authenticated and then placed on the input queue of the Sync Server. The synchronization client software on the device waits for a synchronization reply from the source server.
3. After the synchronization request is received by the Sync Server on the mid-tier system, the username and password that the client provides are authenticated against the administration control database to verify that the client is a valid Sync Server user.

4. The data is placed into a staging table. Staging tables help improve throughput capacity of synchronization requests because changes can be staged while other updates are taking place.
5. The data is copied from the staging table to the mirror table (M_ECCATALOG in this example), and potential update conflicts are resolved. Changes to the mirror table are recorded in the DB2 log.
6. The DB2 Data-Propagator Capture program captures changes to the mirror table from the DB2 log and writes them to a change data table.
7. The DB2 Data-Propagator Apply program applies changes from the change data table to the source table, ECCATALOG.

The synchronization process from Server to Client can be summarized in the figure below:

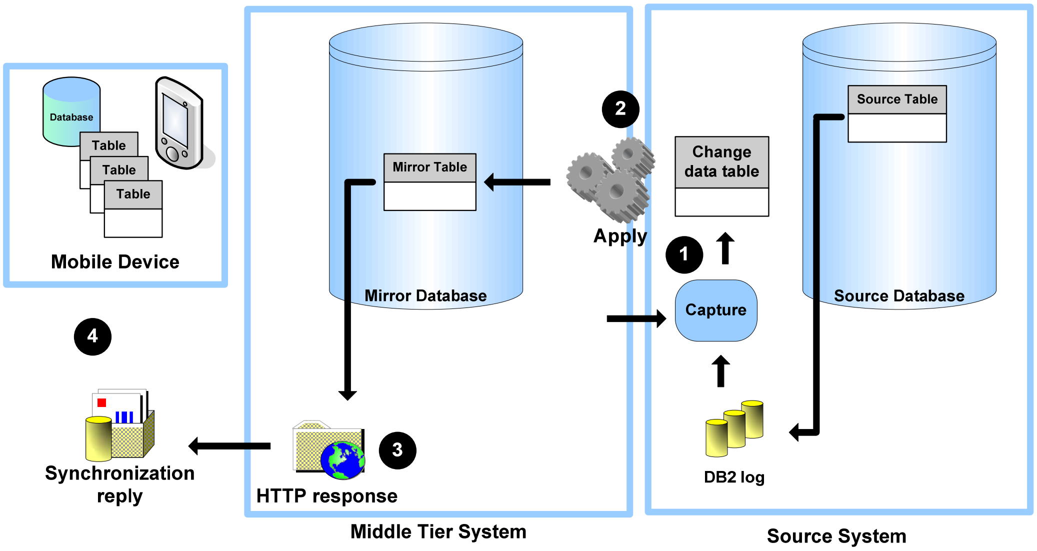


Figure 16: IBM DB2 synchronization (Client to Server)

1. The DB2 Data-Propagator Capture program, running continuously on the source system, captures changes to the ECCATALOG source table from the DB2 log and writes them to the change data tables.
2. The DB2 Data-Propagator Apply program applies changes from the change data table to the mirror table, M_ECCATALOG. Update conflicts might be identified and resolved during this step of the process.
3. The changes to data are sent to an output queue on the mid-tier system as a synchronization reply message.
4. The synchronization client software retrieves the synchronization reply message from the output queue.

5. The changes to data are applied to the client's local copy of the table. The synchronization session ends.

1.15.5.2 Limitations

IBM DB2 EveryPlace provides an excellent synchronization scheme for mobile devices using a small footprint database. The chief disadvantage is that all of the function and data reside at the server; the PA would have to establish a wireless or wired connection to the server to perform any function or see any data. Stopping to connect, interact, and disconnect takes time, and maintaining a wireless connection throughout the day is usually too costly.

1.15.6Oracle Lite

Oracle Database Lite is the first database to extend the power of grid computing to the mobile workforce. Oracle Database Lite is a complete and integrated solution for developing and deploying vital database applications for mobile and embedded environments and delivers features common in mission-critical systems to mobile and embedded devices. Enabling persistent access to critical information and applications without requiring continuous connectivity to back-end enterprise systems, Oracle Database Lite enables users to increase efficiency, productivity and responsiveness of mobile workforces, reduces costs, and improves customer satisfaction.

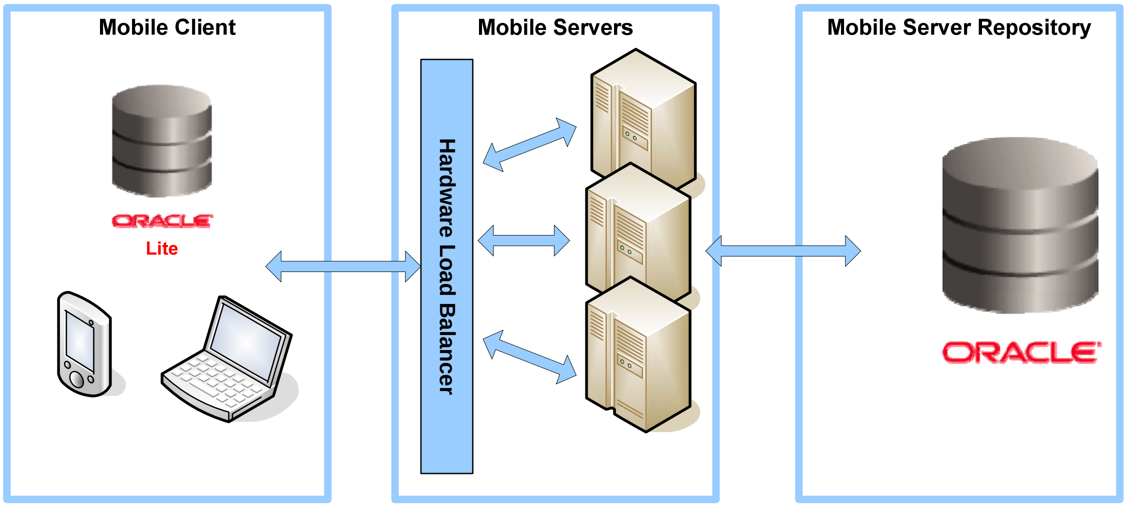


Figure 17: IBM DB2 synchronization (Client to Server)

Synchronization is accomplished by invoking the Mobile sync client, mSync, which interacts with the Mobile Server. The Mobile Server uses synchronization objects such as users, publications, publication items, and subscriptions to process client and server data changes. This technique is referred to as a publish/subscribe model.

Synchronization Process

Oracle Database Lite contains a subset of data stored in the Oracle database. This subset is stored in snapshots in the Oracle Lite database. Unlike a base table, a snapshot keeps track of changes made to it in a change log. Users can make changes in the Oracle Lite database while the device is disconnected, and can synchronize with the Oracle database server.

There are basically three types of publication items that can be used to define synchronization; fast refresh, complete refresh, and queue based. The most common method of synchronization is a fast refresh publication item where changes are uploaded by the client, and changes for the client are downloaded.

Meanwhile, a background process called the Message Generator and Processor (MGP) periodically collects the changes uploaded by all clients and applies them to database tables. It then composes new data, ready to be downloaded to each client during the next synchronization, based on predefined subscriptions. Oracle Lite Sync is a two step asynchronous process:

1. Sync moves updated rows from the client database to the In Queue and from the Out Queue to the client database
2. MGP (Message Generator Processor) applies In Queue changes to base tables and composes the changes to the base tables to client Out Queues

The synchronization process is shown in the figure below

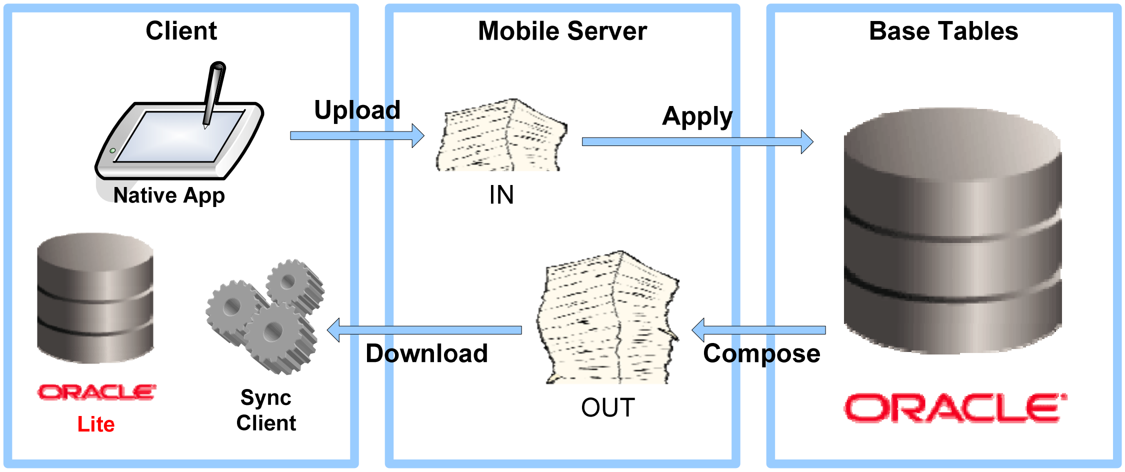


Figure 18: IBM DB2 synchronization (Client to Server)

1.15.6.1 Limitations

Oracle Lite provides an excellent synchronization scheme for mobile devices using a small footprint database. The chief disadvantage is that it not XML based and therefore can only be used on specific mobile devices it supports.

1.16 Synchronization types

Having analyzed the various synchronization types that exist in commercial systems we can now evaluate these types and select which ones fit into our solution.

1.16.1Synchronization schemes

Below is a list of the synchronization schemes that can be used:

- Time ordered synchronization

  Time ordered synchronization is performed by comparing date/time fields for each record. Consequently each record must have a special column/field that can reveal the last time it was edited. This field must contain both the date and time values. The time precision must be set according to the application. In some applications, synchronization may be performed by comparing the hour and minute values while in others the precision must be to the millisecond.

  This method assumes that the clocks between Mobile Device (MD) and Server (SR) are perfectly synchronized or else many problems may arise.

- Time offset synchronization

  Time offset synchronization is more accurate than time ordered synchronization because of the fact that it does not require the MD and SR clocks to be synchronized.

  The offset is measured using the local time of each host and therefore it is more accurate. Offset values are compared and accordingly the necessary actions are performed.

  The offset's precision can again be set by the application and it can vary from hours to milliseconds.

- Clock synchronization

  Clock synchronization is necessary to ensure that the MD and SR compare correct date-time values when using time ordered synchronization.

Because of the fact that these synchronization schemes do not take into account the user's access levels we propose a new scheme here:

- Role based synchronization

  In this scheme, the access rights of the users that updated the records are compared and whoever has the higher level of access wins. This scheme must be used in conjunction with either time ordered or time offset synchronization.

1.16.2Synchronization intervals

Synchronization can occur any time, either synchronously (periodic time intervals) or asynchronously (on demand). We will now present a brief description of these types as well as their pros and cons.

- Synchronous (at specified time intervals)

  The MD or SR triggers the synchronization process at specified time intervals. This type requires the user to be able to connect to the SR periodically and perform the synchronization process. This, of course, is not trivial. It is not guaranteed that the MD will be able to connect to the SR each time. If for example the MD tries to connect remotely to the SR (no cradle is present) and the wireless medium is insufficient (frequent disconnections, weak signal strength) then the process will not complete and fail safe mechanisms must take place.

- Asynchronous (per user request)

  Most mobile devices use this type of synchronization. It is far simpler than the synchronous type and it does not require any special hardware for connectivity. The only requirement is that when the MD requests synchronization, it connects to the SR by some means (cradle, wireless connection) and performs the synchronization process.

Since both methods have their advantages we will select a combination of both synchronization types. The user will of course perform synchronization whenever he demands it (as long as a connection to the SR is available). To ensure that the records are up-to-date most of the times we will also recommend a frequent periodic synchronization.

A NEW SYNCHRONIZATION SCHEME

1.16.3Introduction

Having analyzed most of the existing synchronization platforms already in the market, the various synchronization schemes that they use as well as the latest standards and technologies in the previous chapter, we now in the position to decide what the main criteria are, as well as the properties of the synchronization algorithm.

1.16.4Synchronization Methodology

In this section we will provide a thorough description of our methodology in the form of criteria. We will use the following notations in our description:

MH: Mobile Host
FH: Fixed Host

Criteria 1

Decide the replication scheme, for each table in the database that the user is authorize to view/edit.

Only one of 3 replication schemes can be used for each table

- Pull replication

  All the data from the specific table are replicated on the MH leaving the FH data intact. This means that the data in the MH are overwritten each time synchronization occurs.

- Push replication

  All the data from the specific table are replicated on the FH leaving the MH data intact. This means that no other user is allowed to change the data and push them to the FH. In case of an error, all other changes must be discarded.

- Merge replication

  Both the MH and FH can alter the data, and each time the MH requests synchronization a resolver module is responsible for any conflicts that may occur.

Criterion 2

Decide the authorization level, for each conflicting table in the database that the user is authorized to view/edit.

It is important to know whether the user is granted greater authorization access on some records than other users.

Criterion 3

Decide the resolver mechanism scheme, for each conflicting table in the database that the user is authorized to view/edit.

The resolver can be adjusted to perform one of the default actions:

The criteria described above are presented in the workflow diagram below:

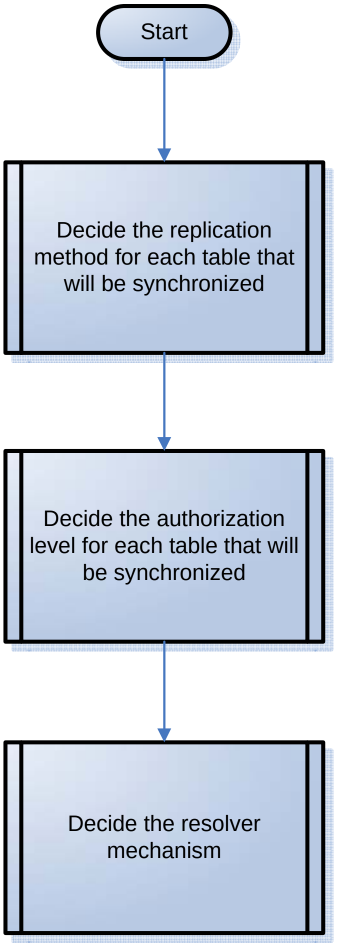

Start
Decide the replication method for each table that will be synchronized
Decide the authorization level for each table that will be synchronized
Decide the resolver mechanism

Figure 19: Synchronization methodology

1.17 Proposed synchronization type

Having studied all major commercial synchronization software and synchronization types we are now in the position to propose our own synchronization method that we expect will fuse all the advantages into one solution.

To illustrate our proposed synchronisation scheme, we describe the actions performed for each record in the mobile device. Afterwards, we explain what happens when a conflict occurs. Finally we present the sequence diagram for this synchronization scheme as well as further explanation for the specific modules.

These are the actions that will be performed when synchronization occurs, according to the state of each record.

RN: Record not changed since last synchronization date

MD: Mobile Device

SR: Server

MD	SR	Action
New record	<Not relevant>	SR inserts new record
<Not relevant>	New record	MD inserts new record
Delete record	RN	SR deletes record
RN	Delete record	MD deletes record
Update record	RN	SR updates record
RN	Update record	MD updates record
Update record	Update record	Conflict resolution
Delete record	Update record	Conflict resolution
Update record	Delete record	Conflict resolution
Not present	Update Record	MD inserts new record

Conflict resolution

The conflict resolution algorithm must take under consideration if a policy exists in the organization that can automatically resolve the conflict. This policy may be defined by the organization or the user. This is a novel feature that is not provided by any commercial synchronization software that was studied in Chapter 1.15. Indeed, if any policies or rules exist in the organization for record management, why not employ them in the synchronization process and as a result do not bother about conflicts that should be automatically resolved. Business policies of the organization may include rules that apply for each table. A rule for example could be the following:

"if a user is an administrator then his actions then his actions precede the actions of users who had been granted a lower access level"

As a result, if the conflict resolution module receives a call with the parameters

Mobile Device (MD) **Normal user**	**Server (SR)** **Administrator**	**Action**
Update record	Delete record	Conflict resolution

then the default action should be to delete the record at the MD and preserve the record at the SR.

Of course there are situations that no police or rule applies. Consider the simple example similar to the case above but the users are both administrators.

Mobile Device (MD) **Administrator**	**Server (SR)** **Administrator**	**Action**
Update record	Delete record	Conflict resolution

The resolution for a conflict that no rule or policy applies, must give the user the choice of what action must be taken in order to proceed. The choice could be in the form of a dialog box that requests the user to point the right action. In our example, the choice should be:

- delete the record at the MD and leave the SR record intact
- update the record at the SR (in this case recreate it) and leave the MD record intact

The update-update conflict resolution is a little more complex and we will explain why. Given that two users have the same access rights to update a record, then the following question arises: What should be done in case both users update different columns on that record? Should one update overwrite the other, or a merging of the two updates should occur? Again, there could be predefined rules and policies that dictate the action to the conflict resolution module and a default action is taken.

For example, in the case of different user types, the default action could be to overwrite the update record whose editor is the user with the lower access level. In another case, the default action could be to preserve the most up-to-date record and update the old one.

Having our discussion above in mind, we propose a new synchronization engine whose architecture is shown in the schema below.

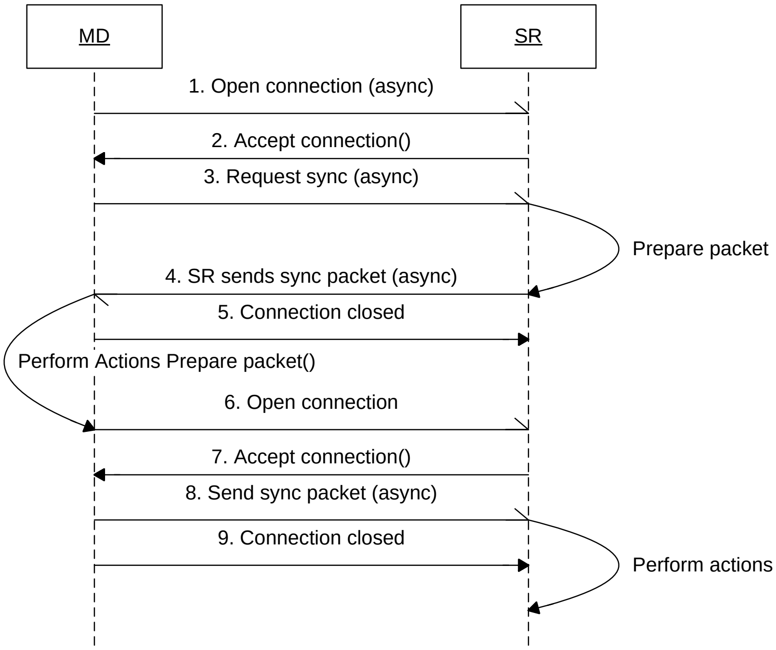


Figure 20: Synchronization Sequence Diagram UML

- Step 1

  MD requests connection to SR.

- Step 2

  SR accepts connection.

- Step 3

  MD requests synchronization packet from SR by sending his last synchronization date and other necessary information.

  This information may include the modules/tables that the MD wants to synchronize in the form of a bit table.

- Step 4

  SR receives the request and queries the database for the transaction log. The log is filtered according to the last synchronization date, the user profile and access levels. It then generates an XML file with the filtered log and sends it to the MD.

- Step 5

MD receives the packet and closes the connection with SR. It then synchronizes its database according to the log and produces an SR synchronization package that contains all the updates that the server must perform.

- Step 6

  MD requests connection to SR

- Step 7

  SR accepts connection.

- Step 8

  MD transmits synchronization packet to SR. Note that the server must execute this package according to the access level of the MD. If conflicts occur then conflict resolution is performed.

- Step Perform actions

  Conflict resolution is performed for any conflicts both on MD and SR side

- Step 9

  Finally the connection is closed and MD goes offline while SR waits for other requests.

1.18 Synchronization process (record management)

Now, after analyzing the sequence that each process must undertake in order to complete the synchronization process, we have to dig a little deeper and analyze what will happen at the MD and SR side.

As mentioned earlier, the SR will send the synchronization packet to the MD and afterwards, when all actions are performed on the MD, a new synchronization packet will be generated and transmitted to the SR.

In this section we analyze the action that will be performed on the SR and MD side as well as the structure of the packets.

1.18.1Generation of the sync packet at the (SR)

The first packet that will be generated on the server side will include all the updates (new records, updates, deletes) that occurred between the last synchronization date (provided by the MD) and that current date.

This packet will include the records, in XML format, ordered by the type of transaction. For example, new records, that must be created on the MD side, must appear first in the list, so that they will precede the update records. This will ensure consistency in the case that a newly created record has also been updated.

To give a clearer view to the reader, we present a simple example below. Let's assume that there is a PERSON table in the database for holding demographics information for every individual in an organization. This PERSON table has the following columns:

COLUMN NAME	TYPE
PERSON_ID	INT
FIRSTNAME	NVARCHAR(30)
LASTNAME	NVARCHAR(30)
DATE_OF_BIRTH	DATETIME

Let's also assume that a synchronization process has taken place at time T0 and the MD has downloaded all the records from the PERSON table (PERSON_ID is in 1-10).

The following transactions have occurred at a later time:

1. T1: Insert new person with PERSON_ID = 11 (11, Panic, Andreou, 11/11/1978)
2. T2: Insert new person with PERSON_ID = 12 (12, Spyros, Georgiou, 19/06/1980)
3. T3: Update person with PERSON_ID = 12 (Georgiou updated to Andreou)
4. T4: Insert new person with PERSON_ID = 13 (13, Dimos, Dimou, 01/01/1970)
5. T5: Update person with PERSON_ID = 13 (Dimos updated to Dimosthenis)
6. T6: Delete person with PERSON_ID = 13

The MD needs not know all the transaction history to be at a consistent state with the SR. It only needs to know the most "current" records. As a result, some of the transactions above should be omitted.

Let's assume that the SR will add an ACTION column to each record that send to the MD. The ACTION column values will determine the action that must be taken at the MD for each record. Let's assume that these values are the following:

Value	Action
1	Insert new record
2	Update record

3	Delete record

Applying this, to transactions T1-T6, gives us the following transaction log.

ACTION	PERSON_ID	FIRSTNAME	LASTNAME	DATE_OF_BIRTH
1	11	Panic	Andreou	11/11/1978
1	12	Spyros	Georgiou	19/06/1980
2	12	Spyros	Andreou	19/06/1980
1	13	Dimos	Dimou	01/01/1970
2	13	Dimosthenis	Dimou	01/01/1970
3	13	Dimosthenis	Dimou	01/01/1970

We have mentioned before that for the database to be consistent we have to reorder this log according to the ACTION column. This results in the following table:

ACTION	PERSON_ID	FIRSTNAME	LASTNAME	DATE_OF_BIRTH
1	11	Panic	Andreou	11/11/1978
1	12	Spyros	Georgiou	19/06/1980
1	13	Dimos	Dimou	01/01/1970
2	12	Spyros	Andreou	19/06/1980
2	13	Dimosthenis	Dimou	01/01/1970
3	13	Dimosthenis	Dimou	01/01/1970

One can notice that record with PERSON_ID=13 is firstly inserted in the SR database, then updated and the deleted. It is clear that this record should not be sent to the MD since it will make no difference. At the end of the execution the delete statement will delete this record. As a result, the following rows should be omitted from the generated transaction log.

ACTION	PERSON_ID	FIRSTNAME	LASTNAME	DATE_OF_BIRTH
1	13	Dimos	Dimou	01/01/1970
2	13	Dimosthenis	Dimou	01/01/1970
3	13	Dimosthenis	Dimou	01/01/1970

This leaves us with the following records in the transaction log

ACTION	PERSON_ID	FIRSTNAME	LASTNAME	DATE_OF_BIRTH
1	11	Panic	Andreou	11/11/1978
1	12	Spyros	Georgiou	19/06/1980
1	13	Dimos	Dimou	01/01/1970

2	12	Spyros	Andreou	19/06/1980
2	13	Dimosthenis	Dimou	01/01/1970
3	13	Dimosthenis	Dimou	01/01/1970

Having identified the different actions, we can now analyze what should be transmitted to the MD for each action.

Firstly new records (ACTION=1) should be sent as is. By that we mean that all the information (all the column values) has to be sent to the MD because a new row in a table may not be able to execute if not all column values are provided.

Translated to XML, an inner node for synchronizing the PERSON table should be:

ACTION	**PERSON_ID**	**FIRSTNAME**	**LASTNAME**	**DATE_OF_BIRTH**
1	11	Panic	Andreou	11/11/1978
1	12	Spyros	Georgiou	19/06/1980

```
…
<PERSON>
      <INSERT>
            <row  PERSON_ID="11"
                  FIRSTNAME="Panic"
                  LASTNAME="Andreou"
                  DATE_OF_BIRTH="11/11/1978"/>

            <row  PERSON_ID="12"
                  FIRSTNAME="Spyros"
                  LASTNAME="Georgiou"
                  DATE_OF_BIRTH="19/06/1980"/>
      </INSERT>
</PERSON>
…
```

Deleted records should be deleted from the mobile device base on some key if it exists. In our example the PERSON table key is the PERSON_ID. In order to delete a record from this table you should only query the database by this PERSON_ID key and delete the record at the position, if it exists.

Let's assume that the transaction log contains another deletion from the PERSON table for record with PERSON_ID=9. Modifying our generated XML file we get the following:

…

```
<PERSON>
      <INSERT>
            <row  PERSON_ID="11"
                  FIRSTNAME="Panic"
                  LASTNAME="Andreou"
                  DATE_OF_BIRTH="11/11/1978"/>

            <row  PERSON_ID="12"
                  FIRSTNAME="Spyros"
                  LASTNAME="Georgiou"
                  DATE_OF_BIRTH="19/06/1980"/>
      </INSERT>
      <DELETE>
            <row PERSON_ID="9"/>
      </DELETE>
</PERSON>
…
```

Finally we have to deal with updates. As mentioned in section 1.5.3 we do not have to transmit a separate update row for each update transaction on a single record. Let's assume that the transaction log contains another update for the record with PERSON_ID=12 which sets the FIRSTNAME from "Spyros" to "Spyro".

ACTION	PERSON_ID	FIRSTNAME	LASTNAME	DATE_OF_BIRTH
2	12	Spyro	Andreou	19/06/1980

The MD does not need to "know" about both updates on the same record. It only needs to know the columns that have been updated, that are the FIRSTNAME from "Spyros" to "Spyro"and the LASTNAME from "Georgiou" to "Andreou".

Modifying our generated XML file we get the following:

```
…
<PERSON>
      <INSERT>
            <row  PERSON_ID="11"
                  FIRSTNAME="Panic"
                  LASTNAME="Andreou"
                  DATE_OF_BIRTH="11/11/1978"/>

            <row  PERSON_ID="12"
                  FIRSTNAME="Spyros"
                  LASTNAME="Georgiou"
                  DATE_OF_BIRTH="19/06/1980"/>
      </INSERT>
      <UPDATE>
```

```
            <row PERSON_ID="12" FIRSTNAME="Spyro" LASTNAME="Andreou"/>
      </UPDATE>
      <DELETE>
            <row PERSON_ID="9"/>
      </DELETE>
</PERSON>
…
```

If for example, another update was in the generated transaction log that updated record with PERSON_D=11 setting the DATE_OF_BIRTH column from “11/11/1978” to “12/11/1978” then the resulting XML file should be:

```
…
<PERSON>
      <INSERT>
            <row  PERSON_ID="11"
                  FIRSTNAME="Panic"
                  LASTNAME="Andreou"
                  DATE_OF_BIRTH="11/11/1978"/>

            <row  PERSON_ID="12"
                  FIRSTNAME="Spyros"
                  LASTNAME="Georgiou"
                  DATE_OF_BIRTH="19/06/1980"/>
      </INSERT>
      <UPDATE>
            <row PERSON_ID="12" FIRSTNAME="Spyro" LASTNAME="Andreou"/>
            <row PERSON_ID="11" DATE_OF_BIRTH="12/11/1978"/>
      </UPDATE>
      <DELETE>
            <row PERSON_ID="9"/>
      </DELETE>
</PERSON>
…
```

This is our final xml transaction log. Of course, depending on the implementation of the application the resulting XML could have another form, without INSERT/, UPDATE and DELETE segments. Another example is setting the ACTION column for each row. This will result in the following XML.

```
…
<PERSON>
      <row  ACTION="1" PERSON_ID="11" FIRSTNAME="Panic" LASTNAME="Andreou"
DATE_OF_BIRTH="11/11/1978"/>
```

```
      <row  ACTION="1" PERSON_ID="12" FIRSTNAME="Spyros"
LASTNAME="Georgiou" DATE_OF_BIRTH="19/06/1980"/>
      <row ACTION="2" PERSON_ID="12" FIRSTNAME="Spyro" LASTNAME="Andreou"/>
      <row ACTION="2" PERSON_ID="11" DATE_OF_BIRTH="12/11/1978"/>
      <row ACTION="3" PERSON_ID="9"/>
</PERSON>
…
```

In this XML file the ACTION column value dictates the action that must be taken in database terms, that is 1=insert new record, 2=update record and 3 = delete record.

1.18.2Synchronization process at the Client (MD)

As soon as the synchronization packet is generated and transmitted from the SR, the MD must execute each action in the package as long as there are no conflicts. As we mentioned earlier in section 1.17 the conflict resolution engine that is generated from the rules and policies of the organization will be used to resolve any conflicts that may appear.

The MD will then prepare a new synchronization packet that will be sent to the SR for updates. The synchronization process is summarized in the figure below:

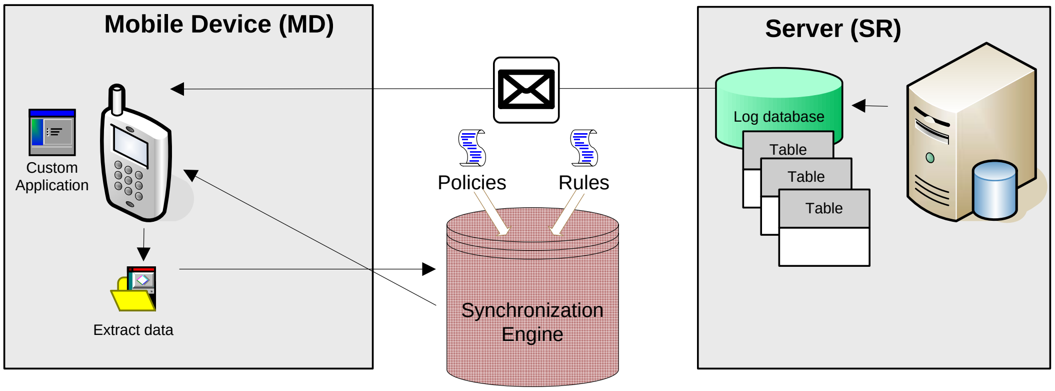


Figure 21: Synchronization process MD

1.18.3Synchronization process at the Server (SR)

As soon as the synchronization packet is generated and transmitted from the MD, the SR must execute each action in the package as long as there are no conflicts. As we mentioned earlier in section 1.17 the conflict resolution engine that is generated from the rules and policies of the organization will be used to resolve any conflicts that may appear.

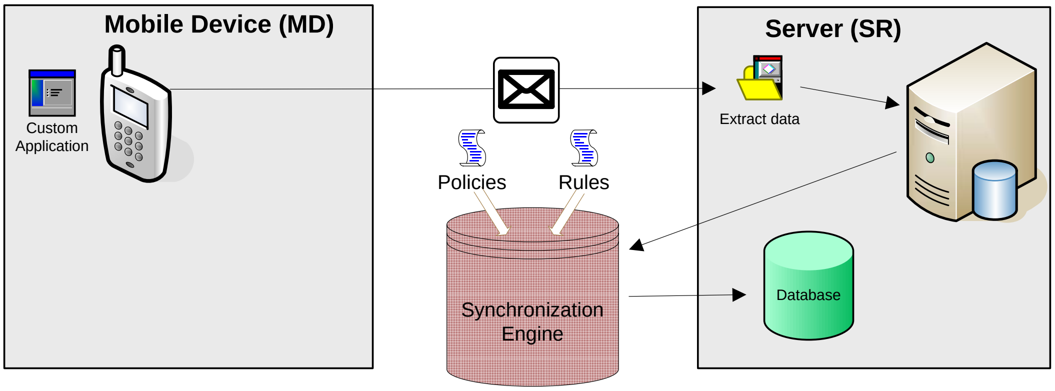


Figure 22: Synchronization process SR

1.19 Synchronization Platform

Summarizing what is proposed in section 1.15 to 1.18 we present the new synchronization platform architecture.

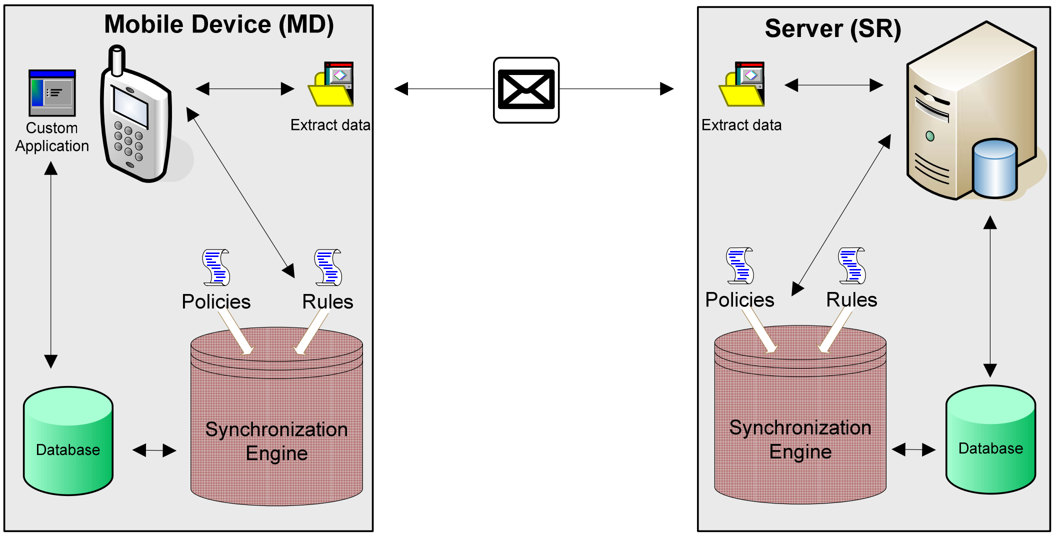


Figure 23: Synchronization platform

1.20 Limitations

Given the fact that every data transmitted, using the proposed approach, is in XML format, the synchronization process may not be efficient in the case of numerous records. If, for example, we had the case of 10000 records then the response time would be over 600 seconds which is not sufficient for most applications.

In addition this synchronization approach does not guarantee overall record consistency. In fact, because the synchronization process is triggered by each mobile device and not by the server, two mobile devices with the same record state may not be consistent after a synchronization process. This of course is the case for most commercial synchronization software since not every mobile device can participate in the synchronization process at any time. Network disconnections or low bandwidth may not allow a connection between the mobile device and the server.

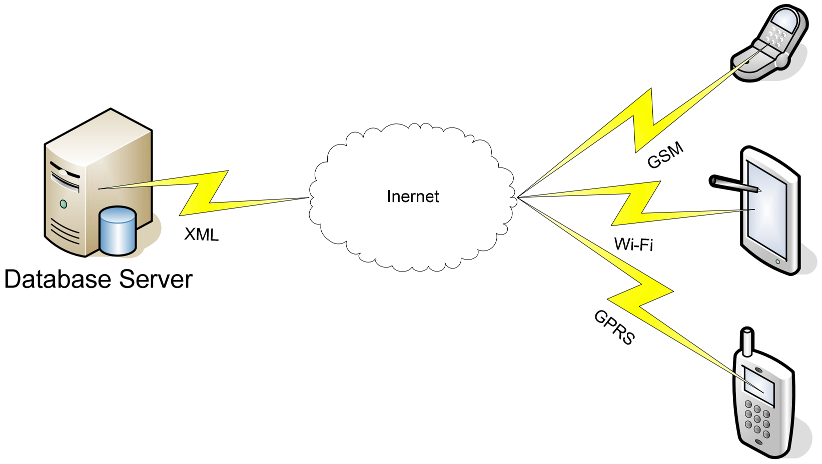
ENHANCING THE SYNCHRONIZATION PLATFORM

1.21 Introduction

Having completed the synchronization scheme in Chapter 5, we employ the technologies discussed in Chapter 3 in order to make our solution more efficient. The following sections provide a step by step integration of these technologies as well as specifics on their implementation.

1.22 Platform Architecture

Figure 24: Initial architecture, represents the initial platform architecture of the system. This architecture can be used for a small data-load of <100 records per request. Using any network connection besides Wi-Fi will render this solution ineffective[‡] because of the XML's lengthy format.



[‡] Of course, this statement depends on the number of columns and data types.

Figure 24: Initial architecture

This platform is composed of the following members. A Database Server that has the ability of transforming the requests in XML format and transmits the XML results using an available network connection.

On the other end, mobile devices use the best available connection to the internet, according to their hardware specifications, and retrieve the result in xml format.

1.23 Implementation overview

In the following sections we will progress the initial architecture so that it becomes more efficient and secure. To do this, we need to add more layers to the architecture. Firstly, we incorporate web services at the server side and divert the requests of the end clients from the database server. In the next step, we add a compression layer so that the any request will be compressed and downloaded faster from the server. This step, of course, adds a new decompression layer at the client side. Finally, to make the solution more secure a new encryption/decryption layer is added to both server and client.

1.23.1Integrating Web Services

This first step involves adding a new layer to our solution which will house the web services. These web services will accept the requests from the clients and perform any necessary requests to the users.

The web service encapsulates all the objects and methods that the client needs to perform the request to the database server. As a result the client will only need to invoke the web service by name and pass all the necessary parameters to it. Figure 25: Integrating Web Services encapsulates the main idea behind the integration of web services.

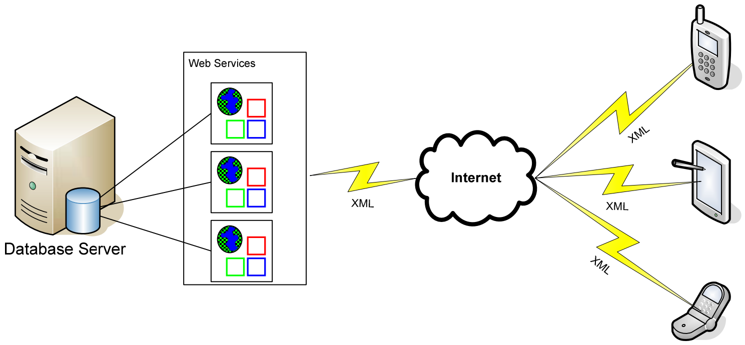


Figure 25: Integrating Web Services

Firstly the client issues a new request by invoking the web service by name and passing all the necessary parameters. The web service encapsulates the knowledge of connecting to the database server and proceeds with a new request to the database server. The database server returns the results to the web service which in its turn produces an XML file with the results and returns it to the client.

The transmission of XML files though can be a slow process because of their lengthy size. This is a big problem, especially when dealing with tables with a lot of columns (>10) and a big number of records (>1000). The next section enables compression of these XML files in to minimize the size of the file and thus improve transmission time.

1.23.2Enabling Compression

The second step towards our final solution is enabling compression of the results issued by the web services. In this scenario, the client issues again the request by invoking the web service by name and passing all the necessary parameters. The web service then, compresses the results and returns it to the client. This has a direct impact on the download time of the results and thus a considerable decrement on the cost. Figure 26: Enabling Compression, encapsulates the main idea behind enabling compression.

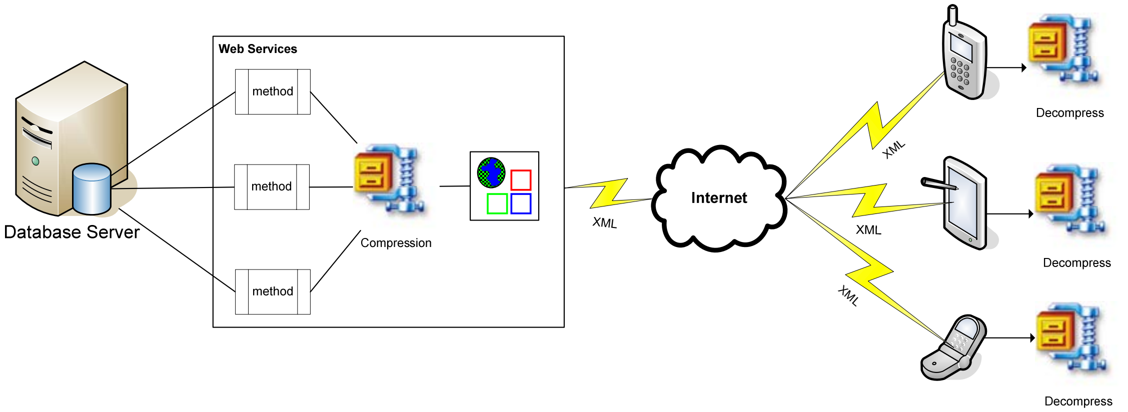


Figure 26: Enabling Compression

Firstly the client issues a new request by invoking the web service by name and passing all the necessary parameters. The web service encapsulates the knowledge of connecting to the database server along with compressing the xml result and proceeds with a new request to the database server. The database server returns the results to the web service which in its turn produces an XML file with the results. Afterwards, it applies a compression algorithm and produces a byte code, which is considerably less in size, and returns it to the client. On the other end, the client downloads the xml file and employs a decompression algorithm that will generate the final results.

This solution, though it increases the system's overall performance, it lacks in the aspect of security. These xml files that are transmitted through the wireless medium can be easily intercepted and downloaded. A hacker can apply numerous decompression algorithms and retrieve the original xml file. To address this problem, Section 1.23.3, enhances the security aspect by incorporating encryption in the compression mechanism.

1.23.3Enhancing Security

The third and final step of the proposed architecture is the enhancement of security by using an encryption mechanism embedded in the compression process. In this way, the original files will not be easily recovered in case of any interception. Figure 27: Enhancing Security, encapsulates the main idea behind enabling compression.

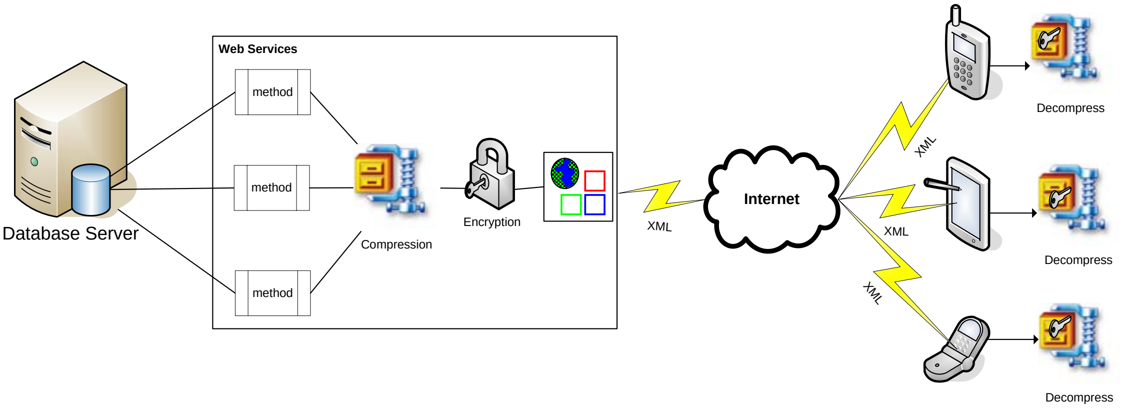


Figure 27: Enhancing Security

Firstly the client issues a new request by invoking the web service by name and passing all the necessary parameters. The web service encapsulates the knowledge of connecting to the database server along with compressing the xml result and proceeds with a new request to the database server. The database server returns the results to the web service which in its turn produces an XML file with the results. Afterwards, it applies a compression algorithm using the username and password of the client's request and produces an encrypted byte code, which is considerably lesser in size, and returns it to the client. On the other end, the client downloads the xml file and employs a decompression, decryption algorithm that will generate the final results.

CASE STUDY – DITISv2 MOBILE SYSTEM

1.24 Introduction

This case study follows the methodology established in the previous sections. It illustrates the various steps described in section 4 and 5 and how they are employed in an already existing system, the DITIS system. While this case study follows the methodology outlined in this paper, it is only one example of how the platform architecture can be integrated into an already existing running system. This case study will demonstrate the potential benefits of employing the platform architecture in the DITIS system as well as possible drawbacks. In the following sections we will describe the DITIS system afterwards we will employ the platform architecture to it and discuss the results.

1.24.1What is DITIS?

DITIS[21,22,23,24] was initiated in 1999, supporting the activities of the home healthcare service of the Cyprus Association of Cancer Patients and Friends (PASYKAF) in the district of Larnaca. Cancer patients, like other chronic illness patients, demand the use of specialized treatment that is coordinated by a multi-person team composed of health care professionals (health care team). This health care team operates at the patient's habitat, due to lack of hospital based treatment because of the incapacity of hospitals to accept large numbers of patients. The need for a health care team arises from the fact that each stage of the chronic illness must be dealt with different types of care, medication, therapies and so on. Unfortunately, having all health care professionals being physically present at the patient's habitat is often impossible.

DITIS, is a system that supports the dynamic creation and management of Virtual Collaborative health care teams that deal with the patient at his home. Treating the patient at his house offers increased psychological support from the patient's family and relatives and is of course more cost-effective.

DITIS uses a patient centric philosophy, focused on home-based rather than facility-based care, receiving governmental and industry grants. It deploys a novel networked system for Tele-collaboration in the area of patient care at the home by a virtual team of medical and paramedical professionals, implemented using existing networking and computing components, initially addressing cancer patients (A.Pitsillides et al., Mar 1997; M.Dikaiakos et al., Mar 1998; A.Pitsillides et al., Apr 1999; A.Pitsillides et al., Dec 1999; B.Pitsillides., Dec 1999; B.Pitsillides et al., Apr 2003). DITIS was originally developed with a view to address the difficulties of communication and continuity of care between the home health care multidisciplinary team (PASYKAF) and between the team and the oncologist often located over 100kms away.

Through its databases, DITIS offers possibility of access via mobile or wire line (computers) offered much more than improved communication. Its flexibility of communication and access to the patient's history and daily record at all times and from anywhere (e.g. home, outpatients, or even during emergency admission) has offered the team a continuous overall assessment and history of each symptom. DITIS has thus offered improved quality of life to the patient, for example by offering the nurses the possibility of immediate authorization to change prescription via mobile devices and the oncologist the possibility of assessment and symptom control without necessarily having to see the patient. It has offered the home care service the opportunity to plan future services and lobby for funding by offering audit, statistics, and performance evaluation and also the possibility for research. The pilot initial results indicated the usefulness of the proposed system, as well as highlighted practical, at times frustrating, problems. DITIS is currently being extended to island-wide implementation to support the PASYKAF home care service.

1.24.2DITIS Architecture

DITIS architecture is based on the server-client paradigm. A database server resides at the CYprus Telecommunication Authority (CYTA) headquarters. This database server is supported by a backup server that performs all maintenance options for DITIS databases. The database is accessible through the internet through a web server running IIS. Note that DITIS is a program divided into two major components. The web component which is in fact a web application and the desktop component which is a program designed for windows operating system enabled hosts. Authentication to DITIS is provided by a simple username/password authentication scheme as well as a sophisticated PKI infrastructure that enable encryption through digital certificates.

A replica of these servers is running also at the University Of Cyprus and serves as the fail-safe plan in the case of any failure of the primary server.

In DITIS, users are divided into three major groups, office users, home users and mobile users. Office users, as the name depicts, reside in the office and are using a high bandwidth enabled connection through an ADSL line. These users are using the desktop component of DITIS and are provided with the full functionality of the system. Of course they can use the web component of DITIS. Home users have the ability to connect through any type of connection they possess (modem, ADSL) and most of the times they use the desktop component. They too can use the web component. Mobile users use their mobile device's browser to connect to DITIS web component. These users lack in bandwidth efficiency as they are depending on the networks connectivity.

A stand-alone mobile component of DITIS is also under development that aims in providing the mobile users more services and functionality. In this case study we will develop the synchronization scheme for the mobile component and thus ensure that the users have full functionality on their mobile device. DITIS architecture is illustrated in the figure below.

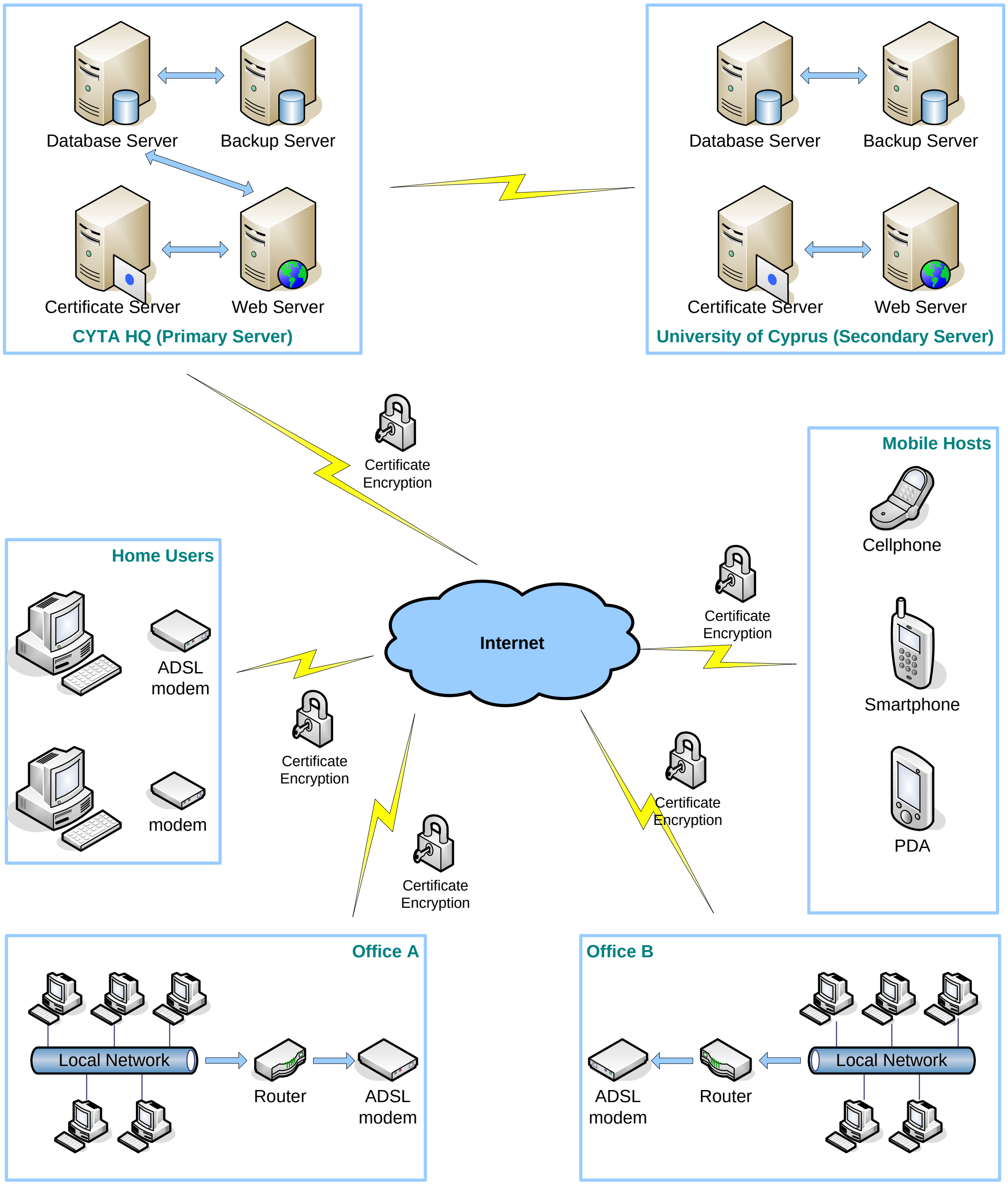


Figure 28: DITIS architecture

1.25 The need for synchronization

Although DITIS benefits are compensational, they do not take into account the network disconnections and weak connectivity. As a result the need for a stand alone mobile application arises. This stand alone mobile application will ensure that the user can access the patient's electronic health record without having to connect to the database continuously. This fact leads to the conclusion that a synchronization mechanism is needed, which will provide the opportunity to the user to synchronize his/her, data with the mobile device when a connection is available and afterwards the user will continue offline querying his local database.

Synchronizing the data of a virtual team member is not a trivial task though. One must take into consideration every record that the user can update or not according to its role. In DITIS, each user is defined by a database role which grants or denies permission to execute specific requests. In this way, a high degree authorization is ensured. The following schema represents an abstract representation of DITIS role based authorization.

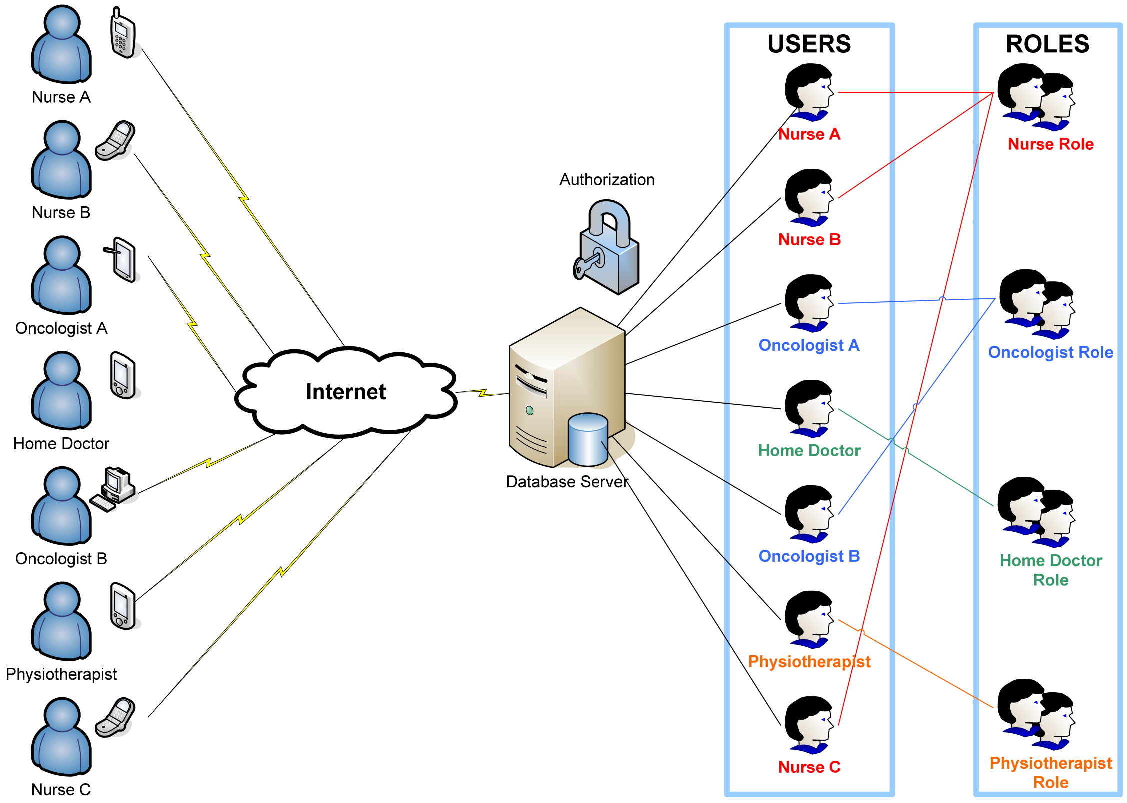


Figure 29: DITIS role based authorization

We can see from Figure 29 that each user is able to connect to DITIS master database using an available internet connection to him. Each time he connects to the database his credentials are authenticated and he is mapped onto a user account. Every user account belongs to a predefined role which tells it what it can or can not do that is which tables or services he can access in the master database. Having said all that we need to clearly identify the hierarchy of the roles and their access rights in order to “feed” them to our synchronization engine.

1.26 Employing the framework to DITIS system

1.26.1Building DITIS's synchronization engine

Following our synchronization methodology described in section 1.17-1.18 we need to identify what kind of replication will be used for each table that will be synchronized, keeping in mind that not all tables are going to be synchronized between mobile device and server. Below we list the tables that are going to synchronized:

1. Appointments
2. Services
3. Appointment Care
4. Care
5. Medication
6. Drugs
7. Medication Type
8. Frequency
9. Route
10. Virtual Teams
11. Virtual Team Members
12. Patient History
13. Symptom Diary
14. Patient Symptoms
15. Diagnosis
16. Equipment
17. Person
18. Profession
19. Countries

Following our methodology we are going to present our steps in the following sections.

1.26.1.1 Identifying replication types for each table

Our first step involves deciding the replication type for each table in the synchronization process. In the table below we are going to divide our tables into three groups, namely Pull Replication, Push Replication, and Merge Replication.

Push Replication	Pull Replication	Merge Replication

Symptom Diary	Services	Appointments
Patient Symptoms	Care	Appointment Care
	Drugs	Medication
	Medication Type	Virtual Teams
	Frequency	Virtual Team Members
	Route	Person
	Patient History	
	Diagnosis	
	Equipment	
	Profession	
	Countries	

In common terms, we have just defined which tables will be retrieved from the server without involving any process (Pull Replication), which tables will be sent as is from the mobile device to the server without involving any process (Push Replication) and which tables may require conflict resolution. Since the two former cases are not complex we move on to the next section where we will identify the rules and policies of the organization running DITIS for the conflict resolution engine.

1.26.1.2 Identifying rules and polices of DITIS

Having distinguished the tables that may require conflict resolution in section 1.26.1.1 we are now going to include the rules dictated by the organization that runs DITIS in our conflict resolution engine. We present again the tables that will be analyzed here.

Merge Replication
Appointments
Appointment Care
Medication
Virtual Teams
Virtual Team Members
Person

Rule 1

Since each patient in the organization is managed by a virtual team, virtual team members have higher access in the hierarchy of roles than other users. This means that if a conflict occurs, then if the two conflicting records are edited by two members of the organization and the first one is member of the virtual team of the patient and the second one is not then the former's edit supersedes the latter's.

Rule 2

If a conflict occurs for two Appointment records and the conflict is resolved by selecting the first of the two records then the Appointment Care records related to the selected record supersede the ones of the Appointment record which is not selected.

Rule 3

If a conflict occurs for two Medication records and one of the editing members belongs to the Oncologist, Home Care Doctor, or Psychiatrist role and the other one is not then the conflict is resolved by selecting the first of the two records.

Rule 4

All other cases should be handled by a conflict resolution dialog

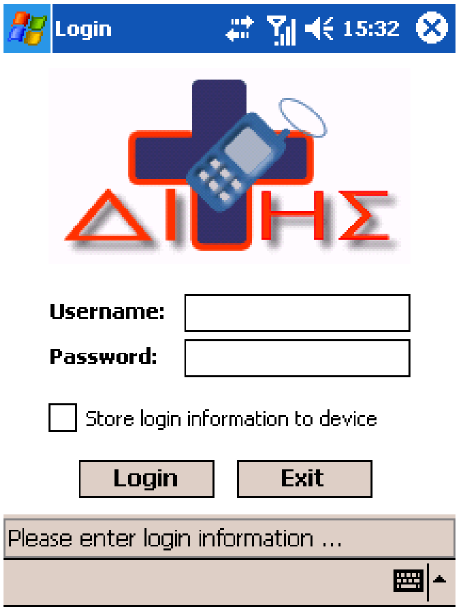
Having displayed all the rules of the organization, we now proceed to the next step, which is implementing the synchronization scheme for DITIS's mobile component.

1.27 Implementation overview

In the following sections we provide information about the implementation of the synchronization process in DITIS mobile component (DITISv2MC). We also provide the reader with some easy to understand examples of how the synchronization process works.

1.27.1Login in DITISv2MC

DITISv2MC start with the following screen:



Username:

Password:

□ Store login information to device

Please enter login information ...

Figure 30: DITISv2MC Login Screen

You can see, from Figure 30: DITISv2MC Login Screen, a checkbox labeled “Store login information to device”. If this checkbox is checked then the following process takes place:

PROCESS A

1. The user enters a username and password and presses the “Login” button
2. DITISv2MC uses a wireless or wired connection (via cradle) and connects to the master database. The user is authenticated based on his credentials at the master database
3. If the use is authenticated then the first phase of the synchronization process takes place
    a. All the data concerning the user and his organization are downloaded using no filters to the mobile device. These data are encrypted and compressed
    b. As soon as that is finished the user’s profile is downloaded and saved in an encrypted form using the username and password as the encryption key
    c. Finally the program proceeds to the “Main Menu” screen
4. If the user is not authenticated then DITISv2MC requests the credentials to be entered again.

If the checkbox “Store login information to device” is not checked then the following process takes place:

PROCESS B

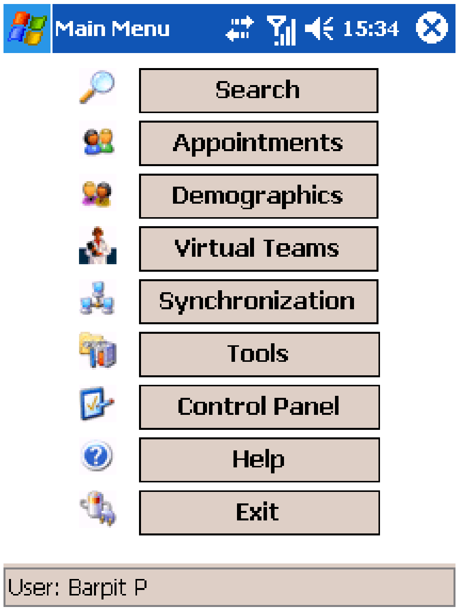
1. The user enters a username and password and presses the “Login” button
2. DITISv2MC tries to find the profile information on the mobile device

3. If a profile information file is not found then PROCESS A is repeated
4. If a profile information file is found then DITISv2MC tries to decrypt the file using the username and password provided
    a. If it succeeds then this means that the user is authenticated and all the data previously stored for DITISv2MC are loaded into memory and the application proceeds to the Main Menu screen
    b. If it does not succeed then the user is not authenticated and the program asks the credentials to be entered again

Note that using this process the user does not have to connect to the master database to be authenticated. By entering the username and password, in fact he has entered the candidate encryption key for decrypting the profile information. This key is stored for decrypting or encrypting the data used in DITISv2MC at a later point.

1.27.2DITISv2MC Basic Functionality

As soon as the user is authenticated, DITISv2MC proceeds to the "Main Menu" screen.



User: Barpit P

Figure 31: DITISv2MC Main Menu screen

Here all the functionality of DITISv2 is presented. The user can perform the following operations:

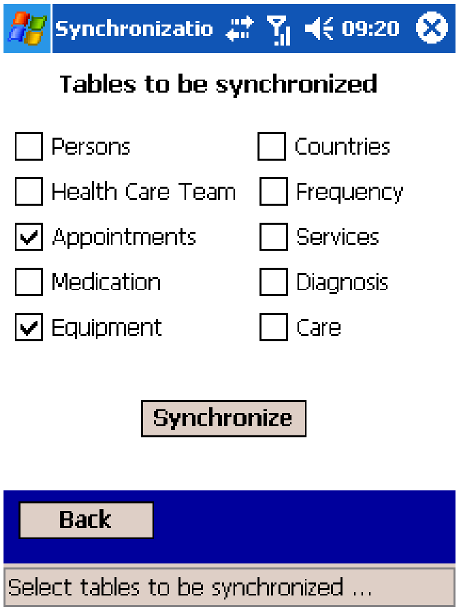
- Search for a person of the organization (personnel and patients)
- Check his appointment status and/or create/update/delete an appointment
- Check the demographics for a person of the organization (personnel and patients)
- Check the Virtual Teams for the patients that the user is part of their Virtual Team

- Perform Synchronization
- Use some HealthCare tools or enter the control panel options
- View the help manual
- Exit the application

1.27.3DITISv2MC Synchronization

In this section we will present all the implementation details of the synchronization module. Note that synchronization can only be performed when a full synchronization has occurred at the first authenticated login process of the user.

The synchronization screen appears as soon as you press on the "Synchronization" button on the "Main Menu" screen.



Tables to be synchronized

☐ Persons
☐ Health Care Team
☑ Appointments
☐ Medication
☑ Equipment
☐ Countries
☐ Frequency
☐ Services
☐ Diagnosis
☐ Care

Select tables to be synchronized ...

Figure 32: DITISv2MC Synchronization screen

As it can be easily concluded from Figure 32: DITISv2MC Synchronization screen, the user can select any tables he wants to synchronize. This is not the case for most commercial mobile synchronization software. As a result the synchronization process is performed faster and the user is not obligated to wait for all the tables to be synchronized in order to continue. This is especially useful in the case of network synchronization.

After the user has selected the tables he wants to synchronize, he presses the synchronization button. The synchronization process takes place in two steps as it is described in section 1.17 (Proposed synchronization type). Keep in mind that this is the case for every table in the database.

We present below a thorough example of synchronizing the Appointments tables.

Let's assume that the following updates are performed on the mobile device and server:

Mobile Device

1. Created new appointment with APPOINTMENT_ID=-1[§].
2. Updated appointment with APPOINTMENT_ID=-1
3. Deleted appointment with APPOINTMENT_ID=15
4. Updated appointment with APPOINTMENT_ID=14
5. Updated appointment with APPOINTMENT_ID=13
6. Updated appointment with APPOINTMENT_ID=12

Server

1. Created new appointment with APPOINTMENT_ID=20.
2. Created new appointment with APPOINTMENT_ID=21.
3. Created new appointment with APPOINTMENT_ID=22.
4. Updated appointment with APPOINTMENT_ID=21
5. Updated appointment with APPOINTMENT_ID=20
6. Deleted appointment with APPOINTMENT_ID=21
7. Updated appointment with APPOINTMENT_ID=14

   This record is updated by a member of the patient's virtual team
8. Updated appointment with APPOINTMENT_ID=12
9. Updated appointment with APPOINTMENT_ID=9

   This record does not exist on the mobile device

1.27.3.1 DITISv2MC Synchronization – Server Side

In the first step of the synchronization process, the mobile device requests the sync packet from the server.

The server will return the following XML file. (Note that most attributes are not included for easy comprehension)

```
…
<APPOINTMENTS>
      <row  ACTION="1" APPOINTMENT_ID="20" …/>
      <row  ACTION="1" APPOINTMENT_ID="21" …/>
      <row  ACTION="1" APPOINTMENT_ID="22" …/>
      <row  ACTION="2" APPOINTMENT_ID="21" …/>
      <row  ACTION="2" APPOINTMENT_ID="20 …/>
      <row  ACTION="2" APPOINTMENT_ID="14" …/>
      <row  ACTION="2" APPOINTMENT_ID="12" …/>
```

[§] Negative values are used for new records on the mobile device so that there are no conflicts with any new records on the server. When a new record is pushed to the server, the server creates a new record id and assigns it to the record. It then returns it to the mobile device which in its turn updates the negative value.

```
      <row  ACTION="2" APPOINTMENT_ID="9" …/>
      <row  ACTION="3" APPOINTMENT_ID="21" …/>
</APPOINTMENTS>
…
```

This is not the final version of the sync packet. The server is "smart" to discard all the rows that will not have any effect on the client, that is

```
      <row  ACTION="1" APPOINTMENT_ID="21" …/>
      <row  ACTION="2" APPOINTMENT_ID="21" …/>
```

since the row

```
      <row  ACTION="3" APPOINTMENT_ID="21" …/>
```

Will delete the previous rows

The final version of the sync packet is the following:

```
…
<APPOINTMENTS>
      <row  ACTION="1" APPOINTMENT_ID="20" …/>
      <row  ACTION="1" APPOINTMENT_ID="22" …/>
      <row  ACTION="2" APPOINTMENT_ID="20 …/>
      <row  ACTION="2" APPOINTMENT_ID="14" …/>
      <row  ACTION="2" APPOINTMENT_ID="12" …/>
      <row  ACTION="2" APPOINTMENT_ID="9" …/>
</APPOINTMENTS>
…
```

Finally the sync packet is sent to the mobile host.

1.27.3.2 DITISv2MC Synchronization – Mobile Device Side

In the second of the synchronization process, the mobile device receives the sync packet from the server. The sync packet has the following rows included:

```
…
<APPOINTMENTS>
      <row  ACTION="1" APPOINTMENT_ID="20" …/>
      <row  ACTION="1" APPOINTMENT_ID="22" …/>
      <row  ACTION="2" APPOINTMENT_ID="20 …/>
      <row  ACTION="2" APPOINTMENT_ID="14" …/>
      <row  ACTION="2" APPOINTMENT_ID="12" …/>
      <row  ACTION="2" APPOINTMENT_ID="9" …/>
</APPOINTMENTS>
…
```

The synchronization process at the server will first create all the new records, then perform any updates and then delete any records.

The first two rows

```
<row  ACTION="1" APPOINTMENT_ID="20" …/>
<row  ACTION="1" APPOINTMENT_ID="22" …/>
```

are executed with no constraints and two new records are created with APPOINTMENT_IDs 20 and 22 respectively.

Next the updates are going to be executed. Because of any conflicts that may appear we will provide a step by step description for each row:

-
  ```
  <row  ACTION="2" APPOINTMENT_ID="20 …/>
  ```
  This record did not exist on the device prior to synchronization, so there is no updated date recorded. Therefore it is handled as a pre-existing record in the mobile device database that was not updated after the last synchronization date. As a result, no conflict exists and thus the record is updated

-
  ```
  <row  ACTION="2" APPOINTMENT_ID="14" …/>
  ```
  This record is both updated by the server and mobile device. As a result the resolver is called for this record. The resolver has the predefined rule (Rule 1) that if a conflict exists and one of the conflicting records is updated by a virtual team member whereas the other one is not, then the access level of the former supersedes the latter's. As a result the record with APPOINTMENT_ID=14 is automatically updated with no questions to the user.

-
  ```
  <row  ACTION="2" APPOINTMENT_ID="12" …/>
  ```
  This record is both updated by the server and mobile device. As a result the resolver is called for this record. The resolver can not match any predefined rule to this case and as a result a dialog box is generated and displayed to the user containing the two records information, with the question to choose which the correct one is. According to the user's choice the record is updated or left as is.

-
  ```
  <row  ACTION="2" APPOINTMENT_ID="9" …/>
  ```
  This record is updated only by the server and does not exist on the mobile device. This means that when the full synchronization was executed, the record was not transferred for some reason and therefore it should be created. As a result the record is created on the device with APPOINTMENT_ID=9.

1.27.3.3 DITISv2MC Synchronization – Server Side

In the third and final step of the synchronization process, the mobile device sends the sync packet to the server. This packet will include all the records updated on the device as well as the decisions of the conflict resolution engine. Carrying on with our example, the following sync packet is generated.

```
…
<APPOINTMENTS>
      <row  ACTION="1" APPOINTMENT_ID="-1" …/>
      <row  ACTION="2" APPOINTMENT_ID="-1" …/>
      <row  ACTION="2" APPOINTMENT_ID="14" …/>
      <row  ACTION="2" APPOINTMENT_ID="13" …/>
      <row  ACTION="2" APPOINTMENT_ID="12" …/>
      <row  ACTION="3" APPOINTMENT_ID="15" …/>
</APPOINTMENTS>
…
```

This packet is sent to server and since all the conflicts are resolved the server will simply execute the packet changes

We have presented a simple example that illustrates the implementation of the synchronization process as well as the conflict resolution. It is obvious (IS IT? HOW? WELL KNOWN? OR YOU DID A CONSISTENCY TEST???) that the use of the proposed synchronization process will maintain consistency between mobile device and server. In the next section we present the performance benefits of our solution. …

1.28 Comparison and evaluation of the new system

In this section we present an evaluation of our system as well as a comparison of the new system with the old one.

Using our synchronization methodology the system is able to keep an up-to-date database state according to the user selection of the tables that need to be synchronized. The test cases that were performed have shown that all the cases were successful and the mobile device and server’s records were consistent.

To evaluate the compression mechanism and network efficiency we present some benchmarks for the tables provided in section 1.8.4 - Data load.

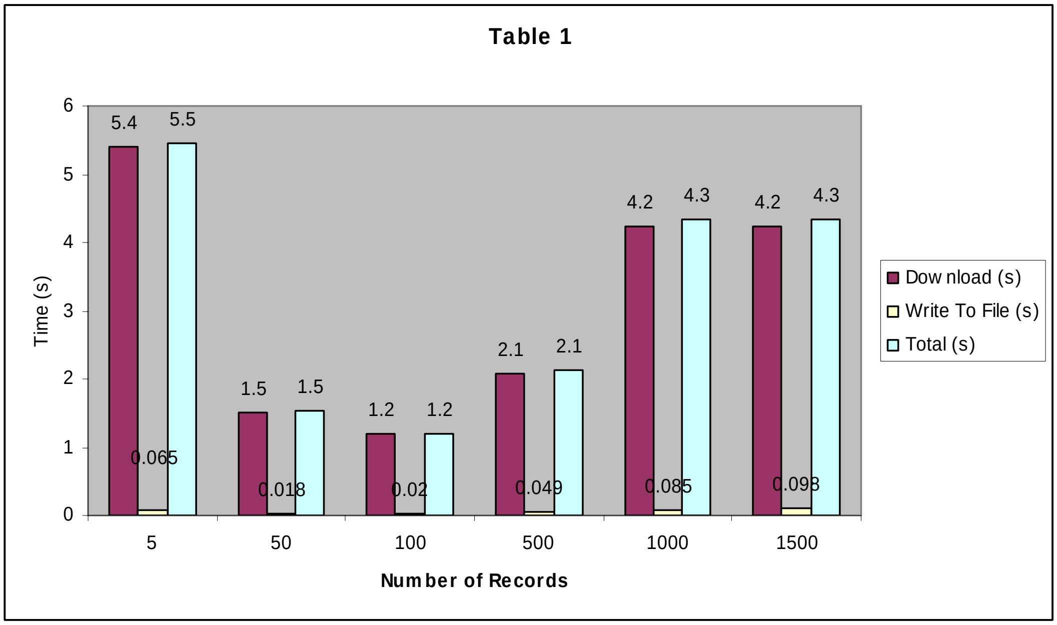


Figure 33: Compressed XML data loading times with different number of records

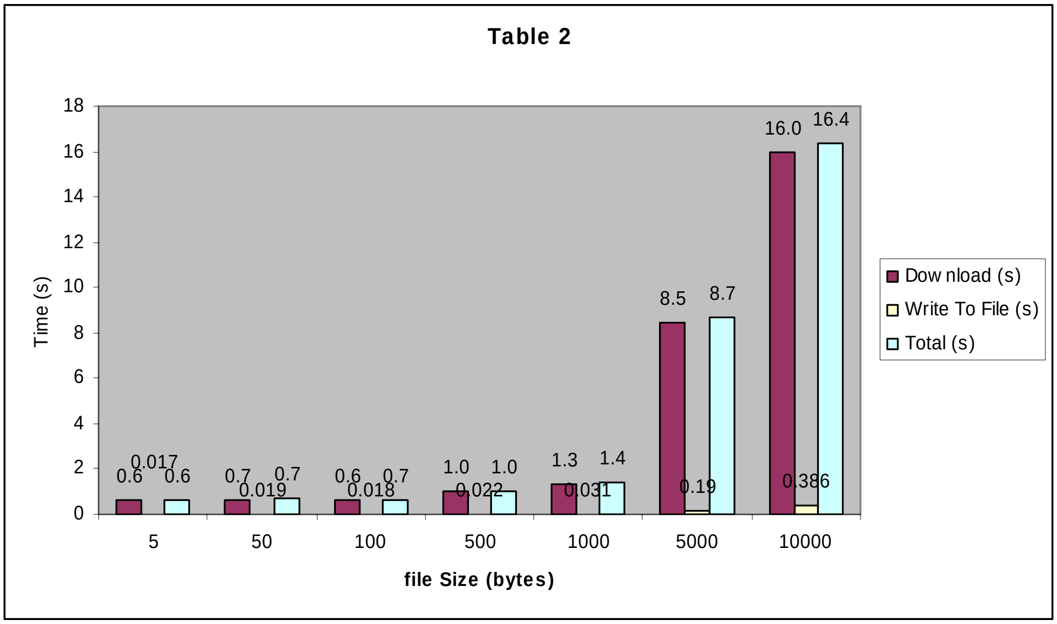


Figure 34: Compressed XML data loading times with different number of records

As it can be easily concluded from Figure 33 and Figure 34 the download times have dropped exponentially. Just note that the download time for the 10000 records example is now just 16.4 seconds in comparison to the 631.3 that was earlier. This offers a significant reduce in cost.

Below, we present the comparison of the two approaches for Table 1 and Table 2.

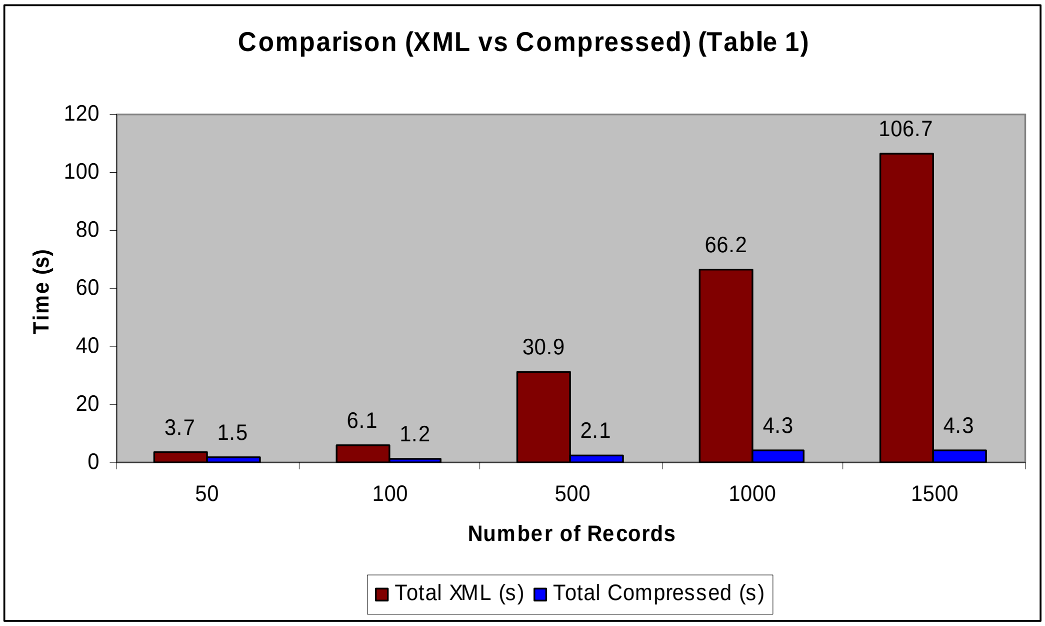


Figure 35: XML vs Compressed XML for Table 1

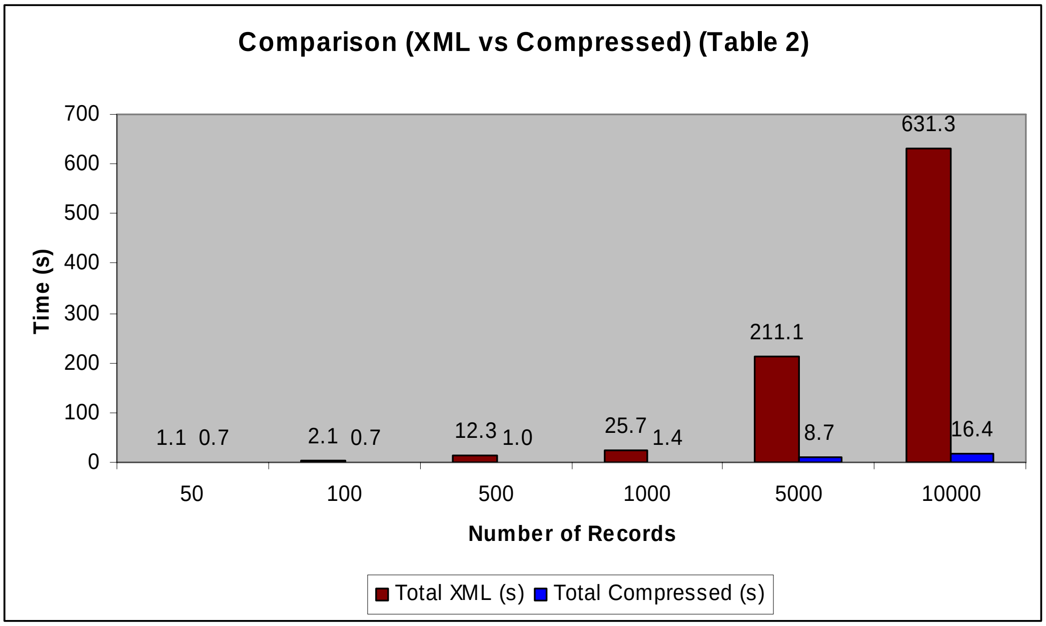


Figure 36: XML vs. Compressed XML for Table 1

We will now proceed to present our overall conclusions as well as possible future work.

CONCLUSIONS

Having presented our synchronization methodology as well as a concrete example of how it can be used, we are now in the position to list the benefits of our solution as well as possible limitations.

Using our synchronization methodology a system can benefit in the following areas.

- Record consistency

  Using our synchronization methodology and algorithm, any application that has the ability to parse and process XML data can achieve consistency with the master database. Of course, we assume that the master database management system has the ability to export data in XML format. That is not a lazy assumption since the vast majority of commercial database management systems have interfaces and automated process for exporting and importing XML data.

  The synchronization process is relatively simple to implement and can overcome may of the problems that were listed in section 1.15 were most commercial mobile systems were analyzed.

- Synchronization optimization

  Rules and policies of the organization can be configured at the start or at a later point in the synchronization engine is a key novelty of our approach. In this way conflicts are resolved automatically if they "match" a predefined rule. This fact makes our approach more appealing since the user is not bothered to resolve any conflicts that should be resolved automatically

- Network efficiency and decreased cost

  By employing compression mechanisms to our approach we have decreased the network cost for synchronization to a very high degree. In some of our examples the download size was decreased by a factor of 40.

  This also enables users that are connected to the internet using a mobile/wireless connection other than WiFi to not overload the cell they are connected and therefore not to lose any packets.

  Compression and decompression of course, have a negative effect in the performance of both the mobile and the fixed host. In the case where compression is used and the overall performance of the application is not altered positively nor negatively then compression should be used since it decreases network cost.

- Overall performance

  The systems overall performance is increased except in the cases where the number of records that are synchronized is small.

- Security

  By implementing encryption based on the username and password of the user we have managed to secure the user's profile and data on the device so that even in the case that the mobile device was lost or stolen, the encrypted data can not be intercepted or retrieved in a correct format.

  The encryption algorithm that is implemented, XTEA, is a footprint encryption algorithm that minimizes the performance inflict to a minimum.

The limitations of our solution are summarized in the point below:

- Although the synchronization process produces the desires results it could perform faster if we had added column level support, that is retrieve only the columns that were updated after the last synchronization date
- Compression should not be used when the number of records is small. A smart decision synchronization process, that would be able to decide when to use compression or not, should increase the performance of the application

FUTURE WORK

Although our synchronization methodology offers significant benefits to the overall performance of the synchronization process, it lacks in some specific areas.

In the case of record transfer in conflict resolution, it is obvious that some optimizations are missing. If an update occurs, then not all columns are required to achieve consistency in the conflicting records. In fact, only the updated columns are needed. This could boost the performance of an application that has to deal with large amounts of data. Just imagine a table with 10000 records that need to be synchronized but only one column is changed in each record. The transfer of each record could "hurt" badly the performance of the download process whereas our suggestion will increase the performance by a ration of $\frac{(n-1)}{n}$ where $n : columns$

Another important topic that could be studied is when to use the compression mechanisms. Have in mind that we consider the security aspect essential and thus can not be removed from any implementation. When should compression be used though? Our results have shown that when dealing with less than 10 records (of course this depends on the columns and data types of a table) it is faster to download the encrypted data rather than compressing the encrypted data and downloading them at the mobile host. A "smart decision" synchronization engine should be able to decide when to use the compression mechanism, using some evaluation heuristic algorithm and act accordingly.

Also, a formal verification of the correctness of the algorithm could be undertaken.

Merging two conflicting updated records is also a topic that can be investigated thoroughly. In the case that two updates on one record are both valid and update different columns a synchronization algorithm could merge the updates into one which would satisfy both parties.

Bibliography

[1] Intel Co., http://www.intel.com/design/pca/applicationsprocessors/linecard/index.htm

[2] Intel Co.,http://www.intel.com/design/pca/applicationsprocessors/index.htm

[3] http://www.samsung.com/Products/Semiconductor/common/product_list.aspx?family_cd=MSC0102

[4] http://www.adriansrojakpot.com/Reviews/Pocket_PC_Performance/GAPI_2.0/GAPI_2.0.htm

[5] George T. Heineman, William T. Councill, “Component Based Software Engineering: Putting the Pieces Together”, Chapter 1, Page 5

[6] Clemens Szyperski, “Component Software: Beyond Object-Oriented Programming”, Chapter 1, Page 10-11

[7] The Universal Description, Discovery and Integration of web services, http://www.uddi.org/

[8] David A. Huffman, “A Method for the Construction of Minimum-Redundancy Codes”, Proceedings of the I.R.E., 1952

[9] Ziv J., Lempel. A., “Compression of individual sequences via variable-rate coding”, IEEE, Volume:24, Issue:5, Pages 530-536, Sep 1978

[10] Welch, T. A., “A technique for high-performance data compression”, Computer Vol. 17, pp. 8-19, June 1984

[11] Ziv J., Lempel. A., “A universal algorithm for sequential data compression”, IEEE, Volume:3,Issue:3,Pages 337-343, May 1977

[12] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/sqlce/htm/_lce_repl_intro_how_replication_works.asp

[13] http://www.microsoft.com/sql/editions/sqlmobile/sqlmobile.mspx#EOD

[14] SyncML White Paper, “Building an Industry-Wide Mobile Data Synchronization Protocol”, OMA

[15] James Jennings, Ph.D., of IBM, and SyncML DM Chair, “SyncML DM: A SyncML Protocol for Device Management”, SyncML World Congress, January 28, 2002

[16] HotSync Conduits, http://kb.palmone.com/SRVS/CGI-BIN/WEBCGI.EXE?New,Kb=PalmSupportKB,ts=Palm_External2001,Case=obj(32839)

[17] Understanding the HotSync process, http://www.palmos.com/dev/support/docs/conduits/win/Intro_HotSyncProc.html

[18] Understanding HotSync conduits, http://kb.palmone.com/SRVS/CGI-BIN/WEBCGI.EXE?New,Kb=PalmSupportKB,ts=Palm_External2001,Case=obj(32839)

[19] DB2 family, http://www-306.ibm.com/software/data/db2/everyplace/

[20] IBM DB2 Everyplace Sync Server Administration Guide version 8.2.3, SC18-7186-06

[21] **A. Pitsillides**, G. Samaras, B. Pitsillides, D. Georgiades, P. Andreou, E. Christodoulou, Virtual Collaborative Healthcare Teams for Home Healthcare, Journal of Mobile Multimedia (JMM), special issue on Advanced Mobile Technologies for Health Care Applications, accepted, 2006

[22] A. Pitsillides, B. Pitsillides, G. Samaras, M. Dikaiakos, E. Christodoulou, P. Andreou, and D. Georgiades, A Collaborative Virtual Medical Team for Home Healthcare of Cancer Patients, Book Chapter, M-Health: Emerging Mobile Health Systems, (R. H. Istepanian, S. Laxminarayan, C. S. Pattichis, Editors), Kluwer Academic/Plenum Publishers, Springer Science, pp. 247-266, 2005

[23] B. Pitsillides, A. Pitsillides, G. Samaras, E. Christodoulou, N. Panteli, P. Andeou, D. Georgiades, User Perspective Of Ditis: Virtual Collaborative Teams For Home-Healthcare, IOS press book series "Studies on Health Technology and Informatics", 100th "Golden" volume, Current Situation and Examples of Implemented and Beneficial E-Health Applications, 2004, hardcover ISBN: 1 58603 448 0, p. 205-216

[24] B. Pitsillides, A. Pitsillides, G. Samaras, D. Georgiades, P. Andreou, N. Panteli: DITIS: A Collaborative System to Support Home Care by a Virtual Multidisciplinary Team, 8th Congress of the European Association for Palliative Care (EAPC-2003), The Hague, Netherlands, April 2-5, 2003