High-Performance Sharing Consistent Data in Ad-Hoc Networks

F.J. Knaesel and M.A.R. Dantas
University of Santa Catarina (UFSC)
Department of Informatics and Statistics (INE)
88040-900 – Florianopolis – Brazil
{fknaesel,mario}@inf.ufsc.br

Abstract

In mobile wireless environments, personal digital assistants may cache data to work even when the mobile client is far from the structured network. However, the same cached data may be shared by two or more users and this can be done in an ad-hoc wireless network. In this case, it is necessary to have always the most updated data. To solve this problem, we developed a software that share data between the server and mobile devices using a timestamp protocol to have always the most updated data. The experimental environment has one server in a structured network and two mobile clients getting and sending data between client and server (when it is covered by the structured wireless network) and between the mobile clients (when a structured wireless network is absent) using an ad-hoc network. In this paper, we describe the experimental environment and detail what and how we have done this software.

1. Introduction

Mobile applications are applications that executes on handheld devices, such as personal digital assistants (PDA), smart phones, tablet PCs and cell phones. There are also three kinds of applications that can run inside these devices: (a) applications that runs without communication with other devices; (b) applications that runs with intermittently communication trough wired or wireless network; (c) applications that runs with permanently connection trough a wireless network. In this article, we are interested in the second type of applications.

The main component of this kind of mobile application is the persistent data storage. This technology permits that mobile clients to work with data previously stored in the device (got trough a wireless or wired connection), without frequent necessity of connection to the structured wireless networks. To make this persistent data storage happens, we can use the file system from the device to store the data, purchase a commercial database solution or build your own solution for that [12].

Therefore, in wireless mobile computing environments, mobile clients can use local cached data to work even the structured wireless network is absent. This can be made by a replication process of creating several local copies of the data called replicas. The reconciliation is the process where a client sends back to a server the updated data in the mobile application [13]. Usually the replication process starts from one (or more) servers to clients and the reconciliation process starts from a client to a server. After the reconciliation process, the servers needs to refresh the data to other clients. A reasons to do that, is to reduce the use of the network and consequently, saving battery time, and because of the instability of the wireless network signal that can cause frequent disconnections.

Some researchers[1,2,3] mention that the local data storage, is an interesting method to reduce the amount of data transfer process, as the latency to the data access and the limited bandwidth available to mobile devices. This local data storage just reduce the time to access the data if clients find their necessary data in side local cache.

However, the network stability and the high bandwidth network required in existing mechanisms on cache management in distributed database applications get in conflict with the mobility of the mobile devices as the very limited bandwidth available and the instability of the wireless networks. Thus, the development of strategies to data consistency sharing becomes a challenge [3].

As mentioned in [4] and [5], the increasing number of users getting and modifying data, force a carefully planning of distributed database management systems.
This occurs, to guarantee high levels of reliability and availability in the process of data update.

Methods for querying and updating data in mobile environments differ from a traditional client/server architecture [6]. This fact occurs mainly because of high mobility in wireless networks, requirements from users in terms of exactness of the data and how much users will invest in the wireless communication infrastructure.

In this paper, the problem of the consistency of data based in a timestamp protocol in the mobile clients will be treated, as the communication and updating data between a server and mobile clients, and also between mobile clients itself.

The paper is organized as follows: the first section is a general introduction for mobile database, and its problems or limitations with network disconnections and battery lifetime. Section 2, presents relevant aspects related to wireless computing. This section shows some characteristics of mobile databases, the cache coherency model and the timestamp protocol used in database management systems to solve the concurrency control and update of data. In section 3 we present our architecture and prototype software for sharing consistent data between client and server and the clients itself. In section 4 we show some related work. In Section 5 we present conclusions and future work.

2. Wireless Computing

In this section we show some theoretical aspects related to wireless computing. Those elements are important to the design of our proposal.

2.1 Wireless Network and Mobile Databases

A structured wireless network is characterized as a cabled backbone network, in which there can be control stations, commonly known as access points. Each control station coordinates the communication of one or more mobile devices in the same cell or in another cell or even to a computer in the wired network. In such environment the data can reside on the fixed network or in mobile devices.

There are different kinds of mobile devices, depending on its characteristics and how much a user wants to spend on it. On of these types are very simple devices with limited resources. In this case, the data is usually stored in computers in the wired network and the mobile devices download data when necessary. This scenario is realistic for some applications and it is actually the most common. However, the management of the distributed data is not hardly affected by mobility, because the data usually reside on computers inside the wired network. An interesting environment is characterized by mobile stations storing native data and it is possible to share this data among other mobile stations. Some researchers (e.g. [14]) called these stations as walkstation.

Mobile computing environments characterize itself in three questions: communication, mobility and portability. The communication is conducted by wireless network concern to disconnections, noise, echo and low bandwidth. The mobility of some devices transform static data in static networks in dynamic and volatile data in wireless networks. We can add to these characteristics the limited battery lifetime or autonomy, which also can cause disconnections or large amount of time turned off.

These problems can make appear interesting questions to database management systems project. In other words, because these issues cause significant changes in the infrastructure on which such systems are build. As [10] mention, well-known changes are directory management, diffusion data, query processing and optimization, transactions management.

2.2. Cache Coherency

In the mobile environment context, multiple devices requesting the same item of data can cause conflicts. Moreover, mobile devices can remain detached from the structured wireless network for long periods of time. These disconnections occur due to battery lifetime, or quit from the network covering area. Mobile devices can also be reallocated in different cells to be communicated with different servers of data, or to run different applications. In this way, an interesting method for cache management in wireless environments should be capable to tackle problems such as limited resources and its frequent disconnections [7].

The local storage of the information frequently accessed is an important technique to reduce the dispute in the communication channel between the client and the server[8,9]. In addition, this storage can be a accomplishes way to save energy, a time that is not necessary to expend energy for the transmission and reception of the data. The storage of the data in the mobile devices can still represent economy of costs, depending on the cost of creation and maintenance of the structured wireless network.

However, considering that the data are stored locally in the mobile devices, the use of mechanisms
becomes necessary to provide the consistency of these data. The use of these mechanisms are applied so that it is possible to certify that these data that had been talked back for the customers are consistent with the data stored in the server. As mobile devices can remain detached, or to be without energy, for long periods of time, this verification of the consistency becomes more complex.

Depending on the application, to maintain the cache coherence in the mobile device effective and reliable, there are two techniques commonly used in mobile applications that are usually read-only and the data comes all the time from a stationary server: (a) the notification of invalidation of records and (b) the notification of validation of records. The first is used when less than 50% of the data in the server was changed while the other method is used when more than 50% of the data was changed, to reduce the amount of data transferred through the wireless network [4].

In the present research work, the data can come from the server, but it can come from other client acting also as a server. As we are using a pessimistic approach and servers do not know what data are stored in our mobile device, all data requested are send through the network. Thus, we used a timestamp protocol to maintain the consistency and freshness of the locally stored data, since it is widely used in distributed database management systems [10].

2.3. Timestamp Protocol

In database management systems we have two methods to implement concurrency control: the blocking protocol and the timestamp protocol. In mobile computing we can not use the blocking protocol because we do not have access to the database server full time. We can use it assuming that our environment is optimistic and the devices are always connected through a wireless network.

Different from algorithms based on blocking protocols, the timestamp protocol does not try to maintain the serializability by mutual exclusion. The implementation of the database system establishes the serialization order by the timestamp, which is a simple identifier that contains the date and time some data has been updated. An example is 2007-01-12 23:59:59 (the milliseconds information may not be include, depending on them implementation of the DBMS). Thus, through this timestamp, we can know whether the incoming data is newer or older than the data we have (or not).

There are a lot of options to assign timestamps to transactions or data: one is to use a global timestamp generator in the network covered area. But, the synchronization and maintenance of global timestamps generators is a challenge in distributed systems. In the other hand, we can just assume that the clocks of the devices are synchronized, and each device can generate its own timestamp. To guarantee a more synchronized timestamp generation in the network, we can use the Network Time Protocol (NTP). This protocol is available in TCP/IP networks, to synchronize the watches of the timestamps generators to achieve a high level of consistency of updated data.

3. Experimental Environment

The main goal of this research work, is to have a lightweight application that can share and update consistent data between mobile devices and stationary servers. In addition, to consider data exchange between mobile devices itself in a structured wireless network or in wireless ad-hoc networks. It is our objective to use simple formats for exchanging and storing data to consume less wireless network bandwidth and less available storage space on the mobile client.

A number of algorithms have been proposed to conventional client-server architectures [15]. In the same way, there is a tendency on utilization of these client-server architectures on mobile environments [16]. In our prototype, we used the client-server model to communication between mobile client and a server in the wired network through control stations and between the mobile devices we also used the client-server approach on which on mobile device acts as server and the other acts as client.

The prototype represents an information system application, where professors can store presences and absences from students in many classes. Teacher can also share a course, but not at the same time. Therefore, a class is shared among one or more professors. This application was used as empirical case study in an university in the south of Brazil, in the State of Santa Catarina. The Figure 1 shows the main architecture of the experimental environment developed.

To implement the server and the client application (which can also act as a mobile server), we used the Java Language Programming with the J2SE API (for the server only application) and J2ME API (for the mobile application). The Java virtual machine version used in the development was the 1.5.0_07, also known as 5.0. We used datagram and socket functions to make the communication between them.
The hardware and software configuration used in this experimental environment was: (a) stationary server is an AMD Duron 2400 Machine with 512Mb of RAM, running the server Java application with the , accessing a MySQL Database through the Java MySQL Connector (available from the MySQL Team Website); (b) two Palm Tungsten C with 400MHz Intel Processor, 64Mb of Internal Memory, PalmOS 5.2.1, Wi-Fi Interface (802.11b) and the JVM used in it was the IBM WebSphere Everyplace MicroEnvironment 5.7. For the development of the application the Eclipse SDK was used with the EclipseME plugin and the Sun Java Wireless Toolkit.

The datagram function was used in the mobile application to discover who were in my wireless network covered area because it is connectionless. The mobile application sends broadcasts datagrams asking: “are you alive?”. On the other side, the application running in other mobile device is listening for datagrams in a specific port. If it receives the question above, the application sends back another datagram with a message like “Yes, I am”. The application that has send the questions then store the IP address of the receiver in a memory array. The IP address of the receivers are not stored in the persistent data storage of the mobile device because they can change frequently.

In addition, not just the questions are made through the network. The application also sends alive notification when it starts. This was done to notify other mobile devices in the covered area of the structured wireless network or ad-hoc network that he is alive in case that it was tuned off when the other mobile device has send the alive requests and in that moment the mobile device was unable to send the response. The Figure 2 shows the notification sending process happening.

After that, a menu is present to the user with the following options: (a) get data, (b) send data, (c) clear data. If the first option is set, it is presented a screen with the hosts discovered on the previous process. After the host is chosen, the application sends another datagram asking again: “Are you still alive?”. If it really is, sends back another datagram saying “Yes, I’m”. Then the application make a socket connection with the host sending the request “I want that data”, and the mobile server sends the requested data through that socket connection. At the same time, the stationary server is also listening for these datagrams and requests, because the data may come from a mobile device or from the stationary server.

When the data comes to the mobile device (from the stationary server of from another client, if is has the requested data), then the application starts the process to store the data at the persistent data storage of the mobile device using the Record Management System class from the J2ME API. Each record that comes through the socket connection is verified if already exists in the persistent data storage. If not, then the application inserts the record. If already exists, then the application compares both records. In case that the record is the same, don’t need update. Otherwise, the

Figure 1 – Architecture of the project

Figure 2 – Notification sending process
application compare the timestamps (available in the persistent data storage and the record has come). If the incoming record is newer, then the application updates the record.

Also during the process of receiving data, it interesting to mention that for the user, it is interesting to have ordered data. For example, when the teacher marks the presences/absences, the students are alphabetically ordered. This was made using a simple recursively implementation of the quicksort ordering algorithm. The Figure 3 shows the process of getting data happening.

The same process of comparing incoming records and its timestamps occurs when the application requests data from the stationary server.

The datagram sending and receiving process, as the socket connection process for incoming and outgoing of data can be made in a wireless network with control station (access points) or in an wireless ad-hoc network. This is important because of the connectionless characteristic of the mobile devices, and because of the covered area from the control station. The communication process between mobile device and the stationary server must happens with a control station because the connectivity on the server is wired.

The implemented application has also an user interface to interact with the data stored locally. In our application, it is not possible to insert new records, because they come from another application, that the Academic Department of the University maintain.

Then, after the first incoming of data, the user/teacher can choose the course he teaches. Then he is presented with a form on which he can choose the class by an integer number, for example: 1 (first class), 2 (second class), and so on. After he chooses the class, he is presented with a new form on which he can fill in the date of the class, and below a list of the students he teaches. In this from he can mark which students are absent. This was chosen because normally, the number of students that are present in the class is higher than the number of absent students. The Figure 4, shows the user interface for updating data.

![Figure 3 – The data getting process](image)

![Figure 4 – The user interface](image)

After filling this form he can save the data locally. The process that saves de local data gets the records from the form and locates the correspondent record in the persistent data storage facility. After located, it is updated with the new local generated timestamp. After this process of maintenance of the local data, the user can sends the new updated data to the stationary server, to the client who shares data with him, or to another clients, for high availability, because a problem can happen with the mobile device and he can loose all the data. Through this, he guarantees that
the data he has updated is safe on the server or with another mobile device.

Is it also important to mention that the data store locally in the mobile device is in a CSV (comma separated values), to reduce the amount of space necessary on the mobile device to store the data.

4. Related Work

Berkenbrock [4] has implemented a similar application that treat of the consistency of the local cache using timestamp protocol, but the application does not update records. It only reads and inserts new data. And the inserted data is not treated by the consistency mechanism.

Cunha [5] has implemented a mechanism for replication and reconciliation between mobile client and server, but does not permit communication between mobile clients itself.

In [17], the team developed a similar application but it uses XML format for exchanging data, which consumes more wireless network bandwidth and more local available space to store persistent data.

The Bayou Architecture [11], is based on a traditional peer-to-peer communication but they typically scale poorly in the number of replicas. Since mobility increases replication factors, the restriction on scale is equally a hindrance to mobility.

Roam [18], is a replication system for mobile computing but it uses the optimistic replication, while in this study, we use the pessimistic approach.

5. Conclusion and Future Work

In this article we present a research work which goal was to develop an application that could share consistent data between mobile clients and fixed servers and between mobile clients itself. We also could share the data using a wireless ad-hoc network, not just a conventional structured wireless network with control stations. The mobile device in this approach can act as an a client or as an a server. We also used simple formats and a lightweight protocol for exchanging and storing data.

There are many related work on sharing data in sharing data in mobile environments, but none of them were exactly what we conceived. The majority of the related work analyzed treats about read only applications or to traditional client server environments and do not permit communication between the mobile devices itself.

Another differential of the present work is that it is not a simulation, it was real implemented and tested at an university considering a large number of professors, and it worked satisfactorily.

For future work we can mention the using a commercial mobile database management system, the use of broadcast datagrams to share data in a connectionless way and increase the number of facilities to the professors that used the application.

References


