Workshop Proceedings of the 9th International Conference on Intelligent Environments
J.A. Botía and D. Charitos (Eds.)

Creating and optimizing client-server applications on mobile devices

Ricardo ANACLETO [a], Nuno LUZ [a], Ana ALMEIDA [a,1], Lino FIGUEIREDO [a] and Paulo NOVAIS [b]

[a] *GECAD - Knowledge Engineering and Decision Support Group*
Instituto Superior de Engenharia do Porto
R. Dr. António Bernardino de Almeida, 431
4200-072 Porto, Portugal
[b] *CCTC - Computer Science and Technology Center*
Universidade do Minho - Campus of Gualtar, Braga, Portugal

Abstract. Mobile devices are embedded systems with very limited capacities that need to be considered when developing a client-server application, mainly due to technical, ergonomic and economic implications to the mobile user. With the increasing popularity of mobile computing, many developers have faced problems due to low performance of devices. In this paper, we discuss how to optimize and create client-server applications for in wireless/mobile environments, presenting techniques to improve overall performance.

Keywords. Mobile applications, distributed environments, communications

Introduction

In the age of technology and information sharing, recent advances in wireless data networking and portable information appliances have engendered a new paradigm of computing, called mobile computing, in which users carrying portable devices have access to data and information services regardless of their physical location or movement behavior. Nowadays, it is essential to have solutions that can support mobility and that leverage business data and processes providing access anytime, anywhere. Todays' busy mobile workers spend more time on the road than at their desks. With business applications inside and workers outside, mobility is imperative. In this sense, extending a business applications to the mobile worker using any mobile device is imperative. Furthermore, employee productivity decreases when vital data and business processes are not easily accessible. For optimal performance on the job, todays' mobile workforce needs business-driven data and tools at their fingertips.

Besides these business applications, there are many other applications that use location and context aware features (e.g., for leisure, community games). Taking as example a leisure application, we talk about a tourism planning support system [1], that takes

[1] Corresponding Author

many advantages from mobile devices, like their small size and simplicity. They are easy to carry around and use in all kinds of environments.

When we go on holidays, and because cities are large information spaces, we need numerous guide books and maps that provide large amounts and complicated information in order to navigate these spaces. With a mobile phone we can access all the important information we want about a place. Still, communications with the server-side application are often required.

Mobile devices have the capability of connecting to the Internet through a mobile network. These connections, however, can be significantly slower compared to fixed or wired data connections and often have a measurably higher latency. This can lead to long retrieval times, especially for lengthy content. Also, many connection failures occur because of transmission interference (e.g., weather and terrain obstacles) and noise. Furthermore, mobile data transfers are often expensive and require additional power consumption. All of these limitations become important challenges in a mobile application that requires persistent communications with a server. Therefore, in order to provide acceptable quality of service to users, applications have to be context-aware, which requires them to adapt to context changes, such as exhaustion of battery power or constants failures to internet connectivity.

In this article, we suggest a middleware for client-server mobile applications. We assume that the middleware at the server-side has no resource restrictions, whereas at the client side it is light and does not incur much overhead to the applications. The applications running on a mobile device require a network connection and a communication protocol to access to the remote server and obtain some information. For this type of applications the accurate and fast response from the server is also a performance key actor. Besides, the connection method used to access the network can also affect the application efficiency. We will discuss what might be the best methods of communication, conjugating low traffic over the network and the simplicity to implement the whole system.

1. Challenges

New mobile computing devices (also known as pervasive or hand-held devices) are appearing everywhere. Despite the fact that growth in mobile devices and emergence of pertinent technologies paves the way for rapid growth in mobile wireless communications, the adequacy of the provided services is not the best. Poor reliability in todays' wireless networks greatly inhibits (sometimes, to the point of infeasibility) supporting applications that involve efficient access to the WWW and Web Services.

Mobile solutions can be built in many different ways, used on many different devices, operate over many different networks, and integrate with many different back-end systems. The task of building a mobile solution can often be daunting given the many technology choices and implementation approaches. As we move from large, centralized systems to client/server systems that use a mixed bag of LAN-connected hardware, we need to optimize our programs for the new environment. This means we need to consume less network traffic in communications between clients and servers, to combine on less costs and better performance.

We have based our work only on client-server applications, where there is a need for a distributed environment and the communication between the distributed entities is

crucial. Here, we evaluate application speed, which is highly related to network speed, server and mobile performance. Additionally, there are many business factors that need to be considered. For example, who are the target users? How critical is it to have the latest data? Are there restrictions for storing data on the device? What countermeasures are there in case of no network connectivity? Is it a critical application?

In this sense, the following challenges must be addressed [2]:

- Interfacing Disparate Technologies - when different technologies are used (e.g., Microsoft .NET, Java), they must communicate seamlessly with each other;
- Device Configuration - since different manufacturers use proprietary methods for loading applications and configuration settings, the application should provide a menu for configuring all device settings required to the application;
- Software Deployment And Upgrades - deployment and remote configuration in several mobile devices should be available (e.g., automatic updates);
- User Interface Design - the graphical user interface (GUI) on a mobile device presents challenges due to small screen size and the difficult data entry interface;
- Performance - by comparison to desktop computers, many mobile devices are very slow, which implies special care in the application design and when dealing with complex algorithms and interfaces;
- Memory Management - since most mobile devices come with a fixed amount of memory that cannot be upgraded, an efficient handling of data is imperative;
- Security - loosing the device and allowing a stranger to access sensitive business data are some of the security issues introduced in mobile system.

Besides all the previously enumerated challenges, when selecting the platform for the device, we can see three different types of applications [4]:

- Online Applications (also known as a thin client). This is client software, normally a browser, used when connectivity can be guaranteed. Without a connection, the mobile application does not work;
- Offline Applications (also known as a thick client). This is client software installed locally to the device that holds all required data for the duration of most operations, and synchronizes at the end of each day or after a preconfigured period of time;
- Occasionally Connected Applications (also known as a smart client). This is client software installed locally, similar to the offline model, but where the application can update and refresh data at any point in time. The frequency of the data refresh depends on the criticality of the application.

The continuous need of advanced services on mobile phones makes the traditional mobile terminals, based on closed software platforms, not appropriate to deliver new applications for supporting advanced services. For this reason, the mobile terminal software stack becomes always more open to host new applications as happened with personal computers. Due to this trend, also the programming environments and paradigms became closer to that of personal computers as well as the understanding operating system features [5].

Summarizing, the technological choice for programming a distributed application has different degrees of freedom: (i) application framework/OS, (ii) required computational resources, (iii) distributed programming paradigm and environment, (iv) communication among application fragments, (v) security.

2. Proposed Solution

We propose a solution that has been implemented in a system that aids the tourist in planning his travel and staying, according to his objectives, preferences, knowledge, budget and staying period. Such a system replaces the need to look for guide prospects/bulletins which sometimes can be quite confusing, or having to follow a standardized plan which does not fulfill his/her needs. It focuses on personalized sightseeing tour recommendation, based on tourism profile and recommendation techniques integration creating a personalized itinerary for holidays, based on user preferences and tastes. It recommends sites to visit, restaurants to eat and hotels to rest. Also, if necessary, the system provides a detailed and scheduled itinerary.

Because the worker can't always carrying his computer, he needs a tool that can be at his pocket, and ready to help and assist him when he needs. In this case we have a smartphone running the Android OS. The problem is that Android uses Java technology. There are two different modules, implemented with different technologies that need to communicate. Another issue is the low RAM memory capacity: only 288MB to the whole system, so we need to be very careful with the mobile application development. On the other hand, this PDA is equipped with HSDPA/WCDMA interface that allows up to 2 Mbps up-link and 7.2 Mbps down-links speeds.

Since the system uses Microsoft server-side technology and our mobile device uses the Android OS (Java technology), we need to specify a communication protocol for the mobile device to connect to the system. The objective is to give real time support to the user, indicating what to see or to do. For this system, that has maps, to show location and context aware information to the user, and involves big amounts of data in client-server communications, we suggest a solution where all the referred issues are taken into account.

This mobile device will run the Mobile Client Application. There are many considerations at this tier, including data availability, communication with middleware, local resource utilization, and local data storage. In addition, many business factors need to be considered.

Since we have two different technologies communicating with each other, and the base system is already implemented, we must implement a middleware application that bridges communications between these applications. This means that the mobile middleware will play a crucial role on the system.

Some of the important features of this tier include security, data synchronization, device management, and the necessary support for multiple devices.

When extending an application on to mobile devices, the challenges mentioned in the previous section need to be effectively addressed. The architecture needs to consider components that work in tandem to address these challenges.

For the communication protocol we will use sockets to transmit HTTP requests and responses when internet connection is available. Because this will be an occasionally connected application, a temporary database is used on the mobile device to permit access to parts of the data without constant traffic consumption over the network, and to allow the application to work without internet connection (with multiple limitations, of course). The middleware is in charge of all of the data synchronization, between the system and the mobile.

The architecture, presented on figure 1, can be summarized saying:

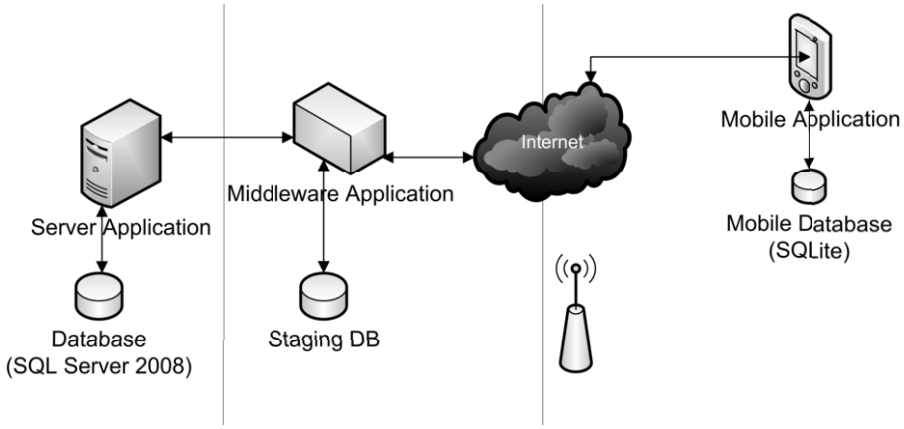

Figure 1. Mobile Architecture Description

- That the existing system will not be changed;
- Middleware application is a component that will reside on the enterprise end and will be developed on .NET Framework, with directives to permit the communications between the existing system and mobile application; a user authentication at the system; a data synchronization service between the system and the mobile device and a data access manager with a staging database; and a device manager, to permit updates of the application on the remote devices;
- Mobile application runs on Android devices and is used to capture/send data from/to the field. The application also has a synchronization component to synchronize the handheld data with the server database. Also, it contains a communication manager that is responsible for the communication protocol with the middleware application, accepting responses and making all the requests; a temporary database is used to enhance the application performance, besides the other necessary layers;
- Internet connection is used to retrieve/update itinerary information, costumers information, personal preferences and to get maps for orientation on site, based on location by GPS;
- Data is uploaded and downloaded automatically without user intervention; latest data is available to the user via background synchronization.

3. Details

We have presented a general overview of system architecture. Now, we present some application details, namely, why we choose some techniques instead of others, and a list of best practices to adopt when developing this type of applications.

To implement a good solution, it is useful to have a list of best development practices. Here is the list that we follow in our case study [4]:

1. Use Database Stored Procedures to write wrapper code for faster data access;
2. For ad-hoc data, the data should be populated using database views for faster output to the device;

3. The staging database infrastructure could be part of the main database server for faster response to mobile devices (the benefit is dependent on the number of users and the server load at any point of time);
4. While extending data from the back end to the staging database, include only those columns and fields that are necessary on the mobile device as the same is to be extended on to the device. This will help in adhering to size constraints on the device;
5. The staging database should only have data for a limited period (two months, for example) with regular scheduled archives; constraining the size of the database will reduce seek time;
6. Use the record version number to easily track records for delta updates during synchronization;
7. Use mapping tables in the staging database to track record version to facilitate conflict resolution; for example, to impose a conflict rule, overriding a transaction record with a server-side change even when multiple changes are done on the client end (A mapping table is a table in the staging database which contains the primary key of the back-end database table and the primary key of the record on the device database.);
8. The Data Exchange Service should be a recurring process and should be configurable in the middleware console to handle continuous changes on the back-end system and staging database (triggered from client), creating an asynchronous method of working;
9. Maintain only necessary user details on the middleware and link to the enterprise directory service for authentication and other user data. This will reduce out-of-sync issues for user information between the enterprise directory and the middleware;
10. Do not store passwords in the staging database; instead, query the enterprise directory service during authentication. This eliminates out-of-sync issues caused due to non-update of the server-side password in the middleware;
11. During synchronization, the client application should first check for application updates by sending its current version and downloading the latest version if applicable; this is an optimized mechanism for application version management;
12. Store the user device profile (device platforms and OS versions) in the user database and push version updates to the device accordingly, sending different builds to different users;
13. Maintain three tables: in-queue, out-queue, and user-wise out-queue for synchronization management, simplifying queue management and optimizing the synchronization process;
14. The communication manager can be made to try alternative types of connectivity when the primary method is not available, so as to use the most efficient available network connectivity option. For example, when wireless LAN is not available, the application tries General Packet Radio Service (GPRS) network; if GPRS is not available, the client does not synchronize;
15. The background sync interval should take into consideration the number of users and the number of concurrent users the server can support. These considerations will assist in reducing the load on the server supporting the maximum number of mobile users;

16. The Device Synchronization Manager just needs to send the username, device application version, sync interval time, and the delta updates during synchronization. To reduce the number of concurrent synchronizations, the middleware should return whether an update is available and the next time for synchronization if no update is required;
17. The record state column should be maintained at record level for faster composing of data during synchronization;
18. The applications should be designed in such a way that when the battery power is low, background thread priority should be set to low, reducing CPU usage and extending battery life;
19. Where the application consists of multiple screens having common UI parts and functionality, design a base form that contains the common elements;
20. Use frames instead of multiple forms wherever possible for faster user interface response;
21. Messages (such as error messages and alerts) should be configured in the middleware and should flow to the devices. Other configuration files should also be configurable on middleware and then pushed to the device application for use;
22. Database-specific queries should not be hard-coded; instead, the queries should be fetched from the middleware via a configuration file.

All these strategies force the applications to execute following some necessary principles in order to guarantee some properties. The following list contains some of the main advantages of our model.

- Scalability: the client side of a middleware is light and introduces a minimum overhead to applications and data communication network. It is possible to distribute the computation on the server side among different platforms;
- Openness: applications are ”modules” that can be inserted and removed at any instant, even during runtime. Authentication, persistence and localization modules can be easily updated;
- Heterogeneity: the client side is robust and can execute on different platforms of mobile devices such as PDAs and smart phones. On the server side, the middleware can also execute on different platforms. The only requirement is to have a TCP/IP connection between those platforms and only the gateway needs to have a wireless link to communicate with the clients;
- Fault tolerance: the server side can be distributed, in order to overcome machine failures. In case the gateway fails, another entity can be configured to assume that same role. Also, redundancy can be obtained with multiple instances of the same server application. The client side deals with disconnections and reconnections in a transparent way;
- Resource sharing: multiple mobile clients can use storage of application servers, which can access a common database provided by the persistence module. Server applications share the gateway data communication link, which possibly has a larger capacity.

With these tips, we designed the architecture presented on figure 2. Here we give a detailed description of what each module can do. Starting with the middleware component we have the following modules (we give the name of the module followed by its functionalities):

- Data Access Manager - Consists of the piece of code that communicates with API's of the recommendation system, to insert, update and delete data and have preconfigured methods that return ad-hoc real-time data to the mobile device;
- Staging Database Management - This module manages the data that is at the staging database, where a replica of all transaction tables with mobile users is stored. It handles in-queue, out-queue and data archiving. Because data conflicts may arise, this module is responsible for gracefully manage and handle those conflicts. Database synchronization is also possible through this module;
- Data Exchange Service - It's the most important piece of software at middleware component, this service is responsible for bridging communications between applications. Compose messages to be sent from the back end to the mobile device and vice-versa, sending data to the out-queue, picking it from the in-queue, and use the received messages from the mobile device to call the Data Access Manager. To have effective data exchanges data optimization and a security to data introduction is necessary. All the transferred data is compressed, to consume less time on the data transmission between middleware and the mobile device. After the data is compressed it is encrypted for security;
- Middleware Console - Contains business logic to do specific activities that are not part of the middleware core, but part of the mobile solution. In this case it gets location updates captured from the device and uses them to retrieve maps to help the tourist. It also has a user interface for configuring middleware, allowing the configuration of modules such as user management, data sub setting, synchronization management, device management and so on;
- Device Management - It helps the management of the devices remotely; sends new application updates to the mobile devices; allows viewing application logs; explores device-related issues, to help solve problems remotely;
- Data Synchronization Service This is a core component of the middleware. Incoming data from the mobile device is received into the in-queue and the outgoing data is pushed to the mobile device from out-queue. It has background synchronization from the device and maintains a user-wise queue and checks for new records to be sent to the mobile device;
- Authentication Service - Manages mobile device users through a user list that is linked with the main system for users to use the same authentication on mobile devices. Also, it authenticates the mobile user during login process and synchronization, and has the capacity to handle multiple connections at the same time: from mobile users up to a maximum number of concurrent users.

On mobile side, we have the following implemented modules:

- Communication Manager - Establishes connection to the network and optimizes the way data is sent to the middleware (Compressing and encrypting the data). It is also responsible for authenticating a user with the middleware if there is network connectivity, otherwise authenticates with credentials available locally;
- Device Synchronization Manager - Sends and receives data from the synchronization service in the middleware, downloads application updates. Schedules background synchronization from mobile device sending data to the server at a preconfigured interval without user intervention (automatically), ensuring that the mobile client back end are in sync;

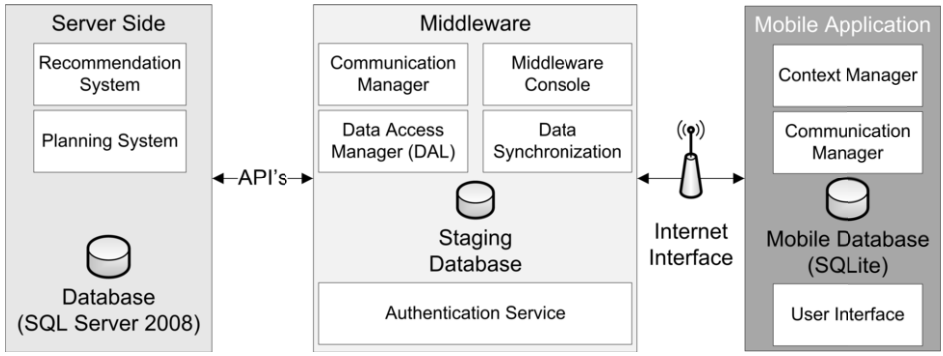

Figure 2. Detailed Solution Architecture

- Remote Data Access - Calls the methods defined in the Data Request Manager to have real-time data on the mobile device; if connectivity is not available then data existing in the device is used;
- Local Database Manager - Manages data in the device database, applying and composing the data. It is responsible to clean up temporary data from the database;
- Device Management - Is the complement of the Device Management module at middleware side, it executes commands from middleware, applies the application updates. Send application logs to middleware and if necessary it is capable of locking the application if the user enters wrong password for a specific number of times;
- Device Application - It is the face of the mobile solution it validates the user input with some business rules and interacts with external hardware interfaces attached with the device.

One of the most important aspects that must have a good performance is the data transfer. Here is where all the communications will stand. If data is carried faster, it means a more responsive system and less costs of internet traffic. It plays a significant role in mobile application architecture because of the number of 'hops' the data must make. The methods and protocols should be carefully considered during system design - see figure 3. There are several considerations to take in account regarding security. If the application is run on a secured LAN, then security restrictions may be little, otherwise restricting access to critical resources is a must. If cost and ease of implementation are imperative factors, then a less reliable method may be used. If reliability must be guaranteed, then a more expensive communication mechanism should be considered.

There are several choices for network communication. We will give a list and discuss the advantages and disadvantages of some of them, starting with the most recent communication protocol, Web Services. Web Services are relatively easy to implement and, if well implemented, are platform independent so they can be used between most platforms and technologies. They are not well suited for large, binary data files. Security is available, but may be more difficult to implement between heterogeneous technologies. It offers maximum portability and readability, but performs extremely poorly, particularly with mobile devices due to the limited bandwidth and processing power that they are able to provide.

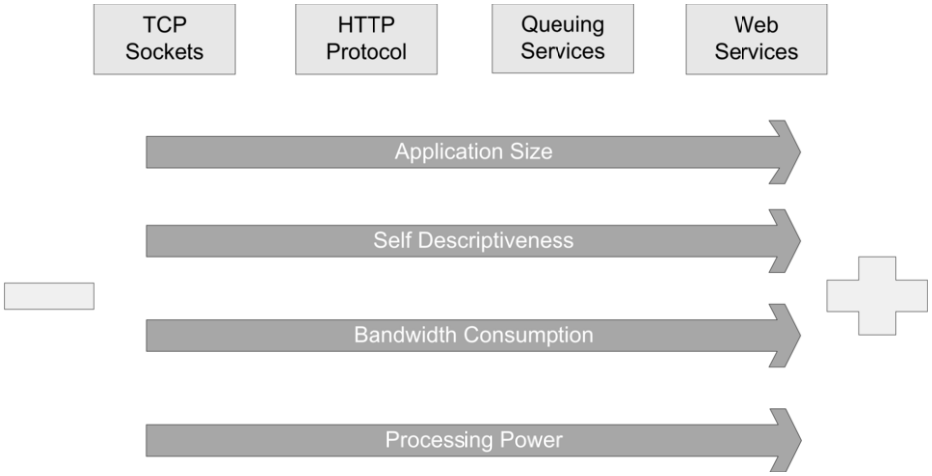

Figure 3. Comparison chart of different communication protocols

The Web Services protocol is very flexible, which is why it has established itself as the de facto standard for remote procedure calls in XML over HTTP. Unfortunately, the inherent disadvantage for the mobile segment in particular, is that it comes with plenty of overhead.

Bringing some major advantages, there are the Queuing protocols. These services guarantee data delivery. They may also be configured for store-and-forward capability. There are few queuing packages available for mobile devices and may be cost prohibitive. Queuing technologies for servers are well established and several good choices exist.

Another well-known protocol is the HTTP, which can provide an easy way to send and receive data. SSL can easily be used to provide security and the implementation is not complicated and very standardized.

Going further into some low level protocols there are the raw TCP/IP sockets. Transferring data over a raw network socket offers the greatest speed of all methods. If security is required it will need to be implemented in code, usually with an encryption library. Implementation can be quick if only raw file transfer is required, but there are many issues that need to be addressed. Multithreading will be required on the server so that multiple devices can connect simultaneously. If something far more complex than simple file transfer is required, another method (such as web services) should be considered since the specification of a new custom protocol over TCP/IP might be time consuming. The major advantage of this form of communication is its high transmission efficiency and the compact payload size. Its disadvantage, on the other hand, is that it is not self-descriptive, and also prior knowledge of the format in both the client and the server is required before the application development can even be initiated. This naturally results in a non-flexible implementation where any changes made to the message format must be consistent between the client and the server. Besides, with the increasing number of dissimilar messages that the server needs to handle, the program code becomes increasingly complex. Bringing some major advantages, there are the Queuing protocols. These services guarantee data delivery. They may also be configured for store-and-forward capability. There are few queuing packages available for mobile devices and may be cost pro-

hibitive. Queuing technologies for servers are well established and several good choices exist.

Another well-known protocol is the HTTP, which can provide an easy way to send and receive data. SSL can easily be used to provide security and the implementation is not complicated and very standardized.

Going further into some low level protocols there are the raw TCP/IP sockets. Transferring data over a raw network socket offers the greatest speed of all methods. If security is required it will need to be implemented in code, usually with an encryption library. Implementation can be quick if only raw file transfer is required, but there are many issues that need to be addressed. Multithreading will be required on the server so that multiple devices can connect simultaneously. If something far more complex than simple file transfer is required, another method (such as web services) should be considered since the specification of a new custom protocol over TCP/IP might be time consuming. The major advantage of this form of communication is its high transmission efficiency and the compact payload size. Its disadvantage, on the other hand, is that it is not self-descriptive, and also prior knowledge of the format in both the client and the server is required before the application development can even be initiated. This naturally results in a non-flexible implementation where any changes made to the message format must be consistent between the client and the server. Besides, with the increasing number of dissimilar messages that the server needs to handle, the program code becomes increasingly complex.

For our implementation we will use TCP/IP sockets (because of their performance boost), with data compressed and encrypted and a checksum algorithm to confirm the integrity of the data. The big deal will be the descriptiveness of the messages. One possibility is to send an XML file instead of pure binary text, but it requires more processing power at the client side (mobile device), and we really want to avoid that. We can say that the specification of the custom protocol will be time consuming, but will bring performance advantages in the future.

A custom protocol will be specified where the client application can invoke remote services provided by the middleware. A queuing service will also be implemented on the middleware, to manage the incoming and outgoing messages. In a distributed server-side environment, the middleware can communicate with the other server modules using web services. This way the mobile device communicates with the middleware using TCP/IP raw sockets, while the server-side distributed communications use web services. This kind of approach turns the middleware into some kind of enhanced proxy.

4. Related Work

Research in the area has been growing and there are several studies [2], [3], [4] and [6] on how to complement a business application with a mobile part. There are development tools that provide specific support for mobile device development like Eclipse and IBM WebSphere Framework [3]. We took some ideas from previous research and best practices to implement the middleware in our case study. Still, public research on the communication aspects involved between mobile devices and a middleware or business layer is still lacking. Knowledge about what is the best protocol to transport information, in an easy and effective way is the concern of many modern software enterprises that are cur-

rently developing for mobile devices. Having in mind the importance of such knowledge, we give our own contribution to the mobile community.

5. Conclusions and Further Work

In order to provide mobile capabilities to the worker, outside the office, while traveling or at home, delivering him all available business knowledge, under the form of a recommendation to a specify situation, turning business decision an easy process we have proposed a system to connect the client (mobile) application with the currently developed server-side. The final solution contains three parts: the actual server side system, a middleware component and the mobile application.

We examine aspects of mobility that distinguish mobile client-server interaction from its traditional counterpart. We also provide a comprehensive analysis of new paradigms for mobile client-server computing, including mobile adaptation, extended client-server model, and mobile data access. A comparative and detailed review of major research prototypes for mobile information access was also presented.

For this work, performance was one of our biggest concerns, as well as the reduction of traffic over internet cost. In further work we intend to implement and test the multiple communication protocols that we have presented, between mobile devices and middleware. This way, a more concrete analysis of the performance and effectiveness of the different protocols can be submitted. Also, flexibility, as well as implementation time and difficulty are some of the major future concerns.

References

[1] A. Almeida, Personalized Sightseeing Tours Recommendation System, The 13th World Multi-Conference on Systemics, Cybernetics and Informatics: WMSCI 2009, Florida, USA, 2009
[2] R. Schuler, Mobile Application Architecture, Enterprise Architect, 2007
[3] D. Bevis, P. LindaMay, Extending Enterprise Applications to Mobile Users, IBM Mobile Computing Solutions, 2002
[4] K. Hariharan, Extending Enterprise Applications to Mobile Devices, The Architecture Journal, 2008
[5] F. Fummi, S. Martini, G. Perbellini, F. Ricciato and M. Turolla, Embedded SW design issues for distributed applications on mobile terminals. Mobile and Ubiquitous Systems: Networking and Services, 2005, 507 – 509
[6] J. Jing, A. Sumi Helal, A. Elmagarmid, ACM Computing Surveys, Volume 31, Issue 2 (June 1999), 117–157