# Peer-to-Peer Applications on Mobile Devices: A Case Study with Compact .NET on Smartphone 2003

Fabio De Rosa
Università di Roma "La Sapienza"
Dipartimento di Informatica e Sistemistica
Via Salaria 113 (2nd floor, lab C4)
00198 Roma, Italy
derosa@dis.uniroma1.it

Massimo Mecella
Università di Roma "La Sapienza"
Dipartimento di Informatica e Sistemistica
Via Salaria 113 (2nd floor, room 231)
00198 Roma, Italy
mecella@dis.uniroma1.it

## ABSTRACT
The traditional approach to information systems, accessed by users by means of powerful devices (such as desktops and laptops) with known features, will not be anymore significant in the future years. Indeed, the current trend suggests that it will be possible to offer continuous access to all information sources, from all locations and through various kinds of devices, mainly small and mobile (e.g., palmtops and PDAs, cellular phones). Therefore, the need emerges for the design of applications for smart devices, which are highly flexible, capable of exploiting in an optimal way the resources. This experience paper analyzes the opportunity to design, develop and deploy interactive applications running on smart cellular phones (commonly referred to as smartphones), based on a peer-to-peer communication model and GPRS technology. A case study is presented to verify whether current development tools and technologies for small devices require a radical different approach with respect to more traditional application development. As a development platform, Window Mobile for Smartphone 2003 with Compact .NET has been used, which is currently available, at least in Europe, only on prototype devices.

## Keywords
Peer-to-Peer – Mobile Device – Compact Framework .NET – Smartphone 2003.

## 1. INTRODUCTION
The traditional approach to information systems, accessed by users by means of powerful devices (such as desktops and laptops) with known features, will not be anymore significant in the next years. The current trend suggests that it will be possible to offer continuous access to all information sources, from all locations and through various kinds of devices, either powerful but mainly static (e.g., PCs, laptops) or small but mobile (e.g., palmtops and PDAs, cellular phones). Moreover, users are interested in a wider and wider variety of applications, beyond traditional vocal interaction access to data of any kind and complex interactive applications, also with transactional properties.

Telecommunication networks, indeed, are continuously evolving and diversifying; each kind of network has its own features, in terms of capacity, reliability, quality of service security, availability, cost. Such features change significantly with the various applications that make use of the network services. However, the user is not interested in technical details: he/she wants to access the end services from the current location and with the best possible performances. Therefore, the need emerges for the design of applications which are highly flexible, capable of exploiting the resources in an optimal way. Finally, traditional client/server computing, based on the availability of centralized or clustered servers offering services and applications to clients, is leaving the place to more decentralized paradigms, such as peer-to-peer computing, in which loosely coupled devices (i.e., the peers) interact with each other without previously established mutual agreements and knowledge. The goal of the "MAIS"[1]

***.NET Technologies'2004*** *workshop proceedings,*
***ISBN 80-903100-4-4***
Copyright UNION Agency – Science Press, Plzen, Czech Republic

[1] Multi-channel Adaptive Information Systems – MAIS – is an Italian research projects, jointly carried

project is the development of models, methods and tools that allow the implementation of adaptive information systems able to provide services with respect to different types of networks and access devices.

The work presented in this experience paper, with respect to the MAIS project sphere, is centered on mobility and small device application development, specifically on design and implementation issues related to the development of distributed applications running on cellular phones.

This work aims at experimenting the opportunity to design, develop and deploy interactive applications running on smart cellular phones (commonly referred to as smartphones), based on a peer-to-peer communication model. Current services offered by telecommunication operators are mainly based on a centralized paradigm, in which cellular phones download simple applications from services available on the Web, and, if communication with other users/devices is needed, it is obtained through centralized services. Apart from vocal interaction and exchange of SMS/MMS[2], no other end-user application is currently designed and deployed assuming a direct and peer-to-peer interaction between users/devices. Even if this can be due to commercial and exploitation considerations, we argue that peer-to-peer interactions between devices should be considered, as a possible alternative for the future, on which to base future commercial and exploitation strategies[3]. In depth, the work presents the development of a peer-to-peer application running on smartphones, in which all communication is based on GPRS technology: as development platform, Window Mobile for Smartphone 2003 with .NET Compact Framework has been used, which is currently available, at least in Europe, only on prototype devices. From a practical software engineering point of view, the aim was also to verify whether current development tools for small devices require a radically different approach with respect to more traditional application development.

The paper is organized as follows. In Section 2 relevant background is presented, focusing on the peer-to-peer computing model and on the technologies for application development on smartphones. In Sections 3 and 4, the application used as case study is presented, whereas in Section 5 a discussion on the gained experience and some insights are presented.

## 2. BACKGROUND

In this section we give a brief overview and state of art on peer-to-peer (P2P) systems and architectures, as well as on technologies and tools commonly used to realize smart device applications.

### Peer-to-Peer Systems and Protocols

The interest for peer-to-peer (P2P) systems has been considerably growing during the last years. Although it is considered a revolution in network based environments, it is actually only an evolution of the original Internet model, that enables packet exchanges among nodes with interchangeable roles. The P2P acronym refers to each distributed system in which nodes can be both clients and servers. In other words, all nodes provide access to some of the resources they own; in the context of this paper, the resources are services provided/accessed from the peers (i.e., mobile devices), enabling a basic form of cooperation among them. An interesting classification of P2P systems can be found in [1], in which the following three models are introduced: **Decentralized Model, Centralized Model,** and **Hierarchical Model**. With respect to such a classification, the application presented in Section 3 has been developed according to the decentralized model. Example of P2P software architectures and systems are [2, 3, 4] and Gnutella [5], the first system implementing a fully distributed file search. All such systems and protocols have been thought for wired networks, that is, networks in which the connection between two peers remains established as long as peers dwell in the system (static connections). Works that take into account mobility scenarios (i.e., dynamic connections) can be found in [6] and [7], respectively. In the former, a mobile P2P architecture and platform is proposed; in the latter, instead, a special-purpose P2P file sharing tailored to Mobile Networks, denoted Optimized Routing Independent Overlay Network (ORION), is presented.

out by about 10 subjects, including Universities and enterprises. The interested reader can refer to http://black.elet.polimi.it/mais/index.php.

[2] In this work, when we refer to interaction, we consider it at the application level, not at the network one. Of course also vocal interaction and SMS/MMS exchange run through centralized servers (e.g., the SMS dispatch center), but users perceive such communication as direct with the others. Conversely, current applications, such as the recently appeared distributed games, require that all application-level communication is collected through a centralized service, and users/devices do not communicate directly.

[3] New computing paradigms for cellular phones could foster GPRS and UMTS technologies in a similar way that Napster/Gnutella-based systems made the Internet popular among teenagers.

**Table 1. Differences among .NET CF and J2ME (CLDC/CDC) platforms**

| | .NET CF | J2ME |
|---|---|---|
| Supported Devices | Devices with Windows Mobile (in Europe cellular phones not yet available) | Java-enabled devices (for MIDP 2.0 only high-end phones) |
| Language Support | C#, Visual Basic .net, C++ | Only Java |
| Virtual Machine | Unique CLR Virtual Machine | Different versions: CDC and CLDC Virtual Machines |
| Byte Code Compatibility | Standard .net CLR | No compatibility with J2SE, and between CDC and CLDC |
| API Compatibility | Between all platforms supporting .net CF (currently only Windows Mobile) | Partial compatibility between CDC, CLCD, and J2SE |
| Development Tools | Visual Studio .net 2003 | Several tools (by SUN and different vendors, not completely integrated) |
| Testing | Emulators in Visual Studio .net 2003 | Various emulators (provided by SUN and by device vendors) |
| Client Installation | ActiveSync or through Internet Explorer download | Device synchronization mechanisms and OTA (Over The Air) download |

The convergence of Instant Messaging [8] with the *conference* peer-to-peer protocols (commonly referred to as call protocols between two or more peers) considers issues similar to the ones addressed by our simple application. In particular, both the ITU (International Telecommunications Union) standard H.323 [9,10] and SIP (Session Initiation Protocol, [11]) are protocols for multimedia conference calls over IP, that can be used to establish, maintain and terminate calls between two or more peers. A current effort to combine Instant Messaging and SIP is made by the SIMPLE (SIP for Instant Messaging and Presence Leveraging Extensions) working group [12]: its goals are of applying SIP to the instant messaging and presence (IMP) suite of services, thus enabling the development of distributed multimedia applications between different peers communicating through messages (and not requiring continuous connections as in SIP). To the best of our knowledge, currently there is not yet a complete specification and no implementation is available. The purpose of our application is very similar to the one of SIMPLE (even if much narrower in generality), and SIMPLE availability would have considerably reduced the development issues.

## Smartphone Application Development

Nowadays, two of the most promising technologies for developing and running applications on smart cellular phones are Java 2 Micro Edition (J2ME, [14]) and the Compact .net Framework (.net CF) [13]. Both J2ME and .net CF are platforms that, differently from micro-browser technologies such as WAP (Wireless Application Protocol), support rich user interfaces, leverage device extensions (e.g., GPS – Global Positioning System - and barcode scanners), and security protocols. Furthermore, compared with smartphone native platforms (e.g., eMbedded Visual C++, C++ SDKs for the Symbian OS), both those technologies have managed environments enabling component-based application development, thus improving developer productivity and application reliability. Table 1 summarizes the differences among the .NET CF and J2ME (CLDC/CDC) platforms, with respect to the designer's/developer's point of view.

## .NET Compact Framework

.NET Compact Framework (.net CF) [13] is a subset of the desktop .NET Framework. It has two main components, namely the Common Language Runtime (CLR) and the Base Class Library. The Common Language Runtime is responsible for managing code during execution: it provides core services such as memory and thread management, designed to enhance performance. Just-In-Time (JIT) compilers enable the generated code to run in the native machine language of the target platform.

The Base Class Library is a collection of reusable classes that are used to develop applications; they provide common and reusable programming tasks such as string management, data collection, database connectivity, user interface, etc. The classes included in the .NET CF provide an identical interface to their counterparts in the workstation/server .NET Framework; some functionalities are not supported due to size constraints, performance issues, or limitations in the target operating system (e.g., printing, Multiple Document Interface forms, Drag-and-Drop functionalities, etc). Class behaviors, properties, methods, and enumeration values are the same under both versions of the .net Framework.

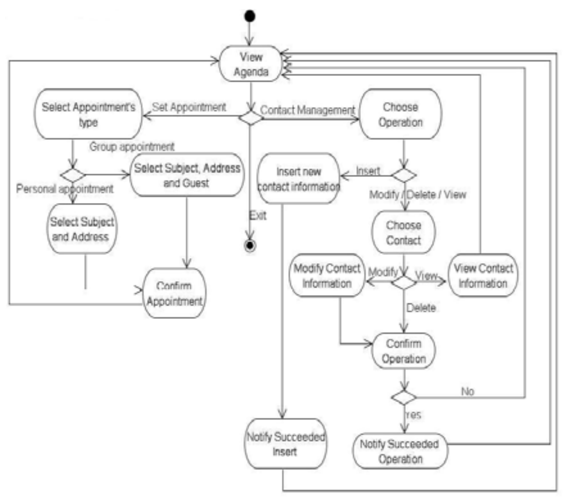


**Figure 1. Activity Diagram for the Appointment and Contact Management.**

Such libraries include classes for connecting to a remote data source, submitting queries, and processing results; frequently, they are used as a robust, hierarchical, disconnected data cache to off-line (i.e., during disconnections) work with remote data. XML APIs, instead, are the same classes provided by the .NET Framework, and used to develop applications manipulating XML structures.

## 3. THE INTERACT-AGENDA APPLICATION

In this section we describe the design of our SMS- and GPRS-based application (called Interact-Agenda), as well as the proposed peer-to-peer architecture and protocol on top of which the application is based. In next section implementation details are provided.

### Application Requirements

The Interact-Agenda application is an interactive agenda for smartphone devices, which allows users to automatically organize appointments between several persons. The application offers to users the following functionalities:

- **visualization**, to view details of one or more appointments, also in the mode “all of the week";
- **appointment management**, for inserting, deleting and modifying both personal and group appointments;
- **contact management**, for inserting, deleting and modifying contacts in the personal book.

The novelty of the application with respect to already existing ones (e.g., Pocket Outlook) providing the same functionalities, is that the appointment management is carried out in a (semi-)automatic way, on the basis of the protocol described in Section 3.3. Currently, a user willing to organize an appointment with several persons: *(i)* decides a candidate time slot (on the basis of its agenda); *(ii)* manually contacts all involved persons (by calling them, by sending them an SMS, by writing an e-mail, etc.) and waits for their reply; *(iii)* if all invited persons agree upon the time slot, he sends them a confirmation, else *(iii ')* he/she chooses a new time slot and begins the process again. All such activities are carried out manually by the proposing user, and they are a serious burden for very busy persons.

The idea of our Interact-Agenda is to provide an application, running on user smartphones, that carries out all the negotiation automatically, and only at the end (i.e., after finding a suitable time slot) asks all involved users for confirmations[4].

When a user creates a new **personal** appointment in his/her agenda, all the details are stored (as it normally happens when the user takes an appointment in Outlook): conversely, when the user creates a new group appointment, a negotiation procedure with all involved persons is required.

In Figure 1 we report the complete Activity Diagram for the application. When a user selects the Interact-Agenda menu, he/she can choose between the following options: to enter in the appointment management section; to enter in the contact management section; or to exit from the application. On the basis of the choice, the user can do several tasks; for example, if he/she has chosen the appointment management section, the user can view, modify, delete or create an appointment.

In Figure 2 we report an example of the Sequence Diagram for the negotiation phase between three users (and their smartphone devices), carried out in order to establish a group appointment proposed by **user_1**. After the proposer (i.e., **user_1**) has entered all appointment information (through an appropriate sequence of windows), the peer instance of Interact-Agenda running on its smartphone communicates with the peer instances of Interact-Agenda running on the smartphones of **user_2** and **user_3**, in order to verify the availability near the proposed date, and if not possible, asking the user a new time slot and conducting a new negotiation round[5].

[4] As users, the authors would not like a smartphone taking appointment without at least letting them know !!

[5] The Interact-Agenda instance of the proposer asks the others for the availability of time slots near the initially

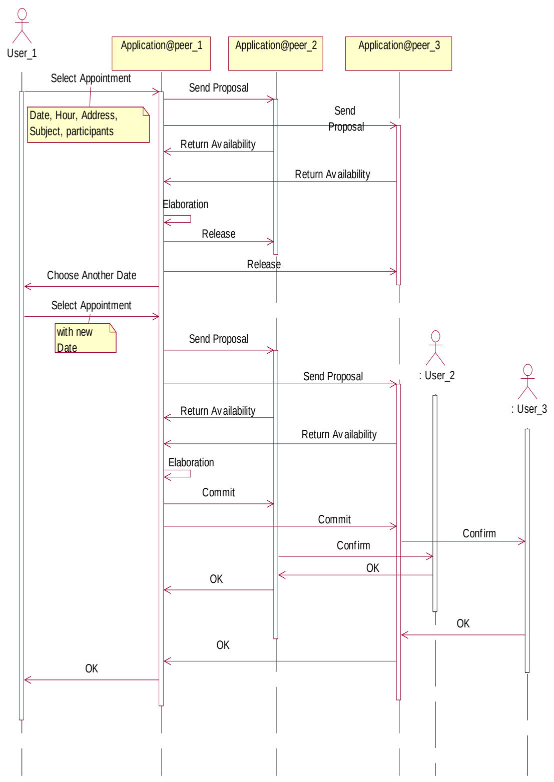


**Figure 2. Sequence Diagram of the Appointment Negotiation.**

All this process happens silently and without (or minimal) intervention by the users. At the end, when a time slot has been "agreed upon" by the peer instances of the Interact-Agenda, a message is displayed to the respective users in order to gain confirmation. That negotiation process runs on top of SMS messages and GPRS network connections, as detailed in Section 3.3.

## Application Design

The Interact-Agenda is developed and deployed as a peer-to-peer smartphone application: each device hosts an instance of the application. In Figure 3, the structure of (an instance/copy /peer of) the Interact-Agenda application, is shown. The application consists of:

- the `User Interface` package, in which all the user-interface functionalities are managed;
- the `Application Logic` package, in which all the negotiation logic is contained;
- the `Logical Data` and `Database` are the packages managing the local database (storing appointment records and contact records). Logical Data is an abstraction of simple DBMS functionalities (table creation, tuple insertion, deletion, modification and selection, etc.), while Database is a concrete implementation of it (in our application it implements that operations on .txt files).
- the `Peer-to-Peer` package, which implements the peer-to-peer protocol (see Section 3.3), and the Networking package that manages SMS-based communications and GPRS connections.

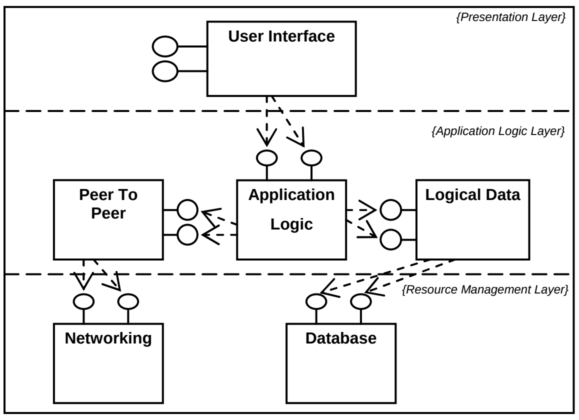


**Figure 3. Structure of the Application.**

## Communication Protocol

The communications between the smartphone devices are conducted over GPRS and SMS. Indeed, each device (peer) may be both client and server at the same time, sending and receiving messages (by SMS sockets) to/from another peer and exchanging information by GPRS sockets. A device receives and replies to an appointment request incoming from another peer, which is the *negotiation initiator*; in turn, the device can initiate an appointment negotiation phase (its user wants to establish an appointment with other persons).

In Figure 4 we illustrate the whole protocol. It may be split into two phases: in the first phase the initiator communicates to all members its IP address by a SMS with a particular header; each active destination, after receiving the message, replies with an "I-am-alive" message; if the initiator receives all the responses from all the clients, then it can proceed with the second phase, that is, the exchange of appointment information (month, day, hour, etc.)

---

proposed one, in order to find a match; near means a given number N of time slot before or after the originally proposed one, and it can be configured. If a match cannot be found, the details of a new appointment are requested from the proposer; the number of repetition of the negotiation process is therefore directly driven by the proposing user.

with all participants, through sockets on GPRS. Otherwise, i.e. if at least one device has not replied to the initial message, the initiator terminates the negotiation phase and closes all opened connections with the others clients. Then the initiator (which acts as a server of the different sockets) exchanges information with the other devices (acting as clients) until the agreement has been reached or the proposing user decides to abort the appointment. Therefore the second phase of the protocol is straightforwardly derived from Figure 2. During all the protocol, the initiator sets timeouts: if not all replies are collected before their expiry, the initiator aborts everything. Different timeouts are set for the first and second phase (longer for SMS-based communications, lasting hours, and shorter for socket-based communications, as in usual network programming practice). The choice of two different technologies (SMS during the first phase and GPRS connections in the following) is needed because of:

- GPRS connections assign an IP address dynamically (i.e., each time the device connects), therefore it is not possible to statically store information such as < user, telephone number, IP address of its device >. Conversely, the initiator can discover its IP address (as described in Section 4.2), then it can send such an IP address to all other devices (through SMS messages, as the couple < user, telephone number > is static), listening to a specific GPRS sockets, and finally all contacted devices can connect to such an initiator's socket through GPRS.

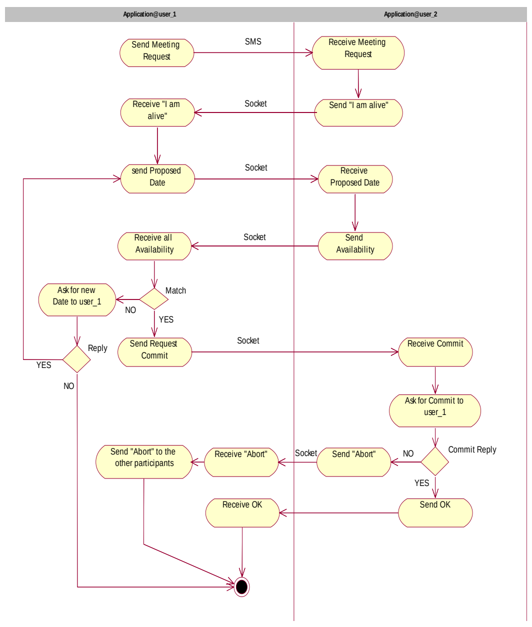


**Figure 4. Communication Protocol.**

- Currently (at least in Italy), an SMS is more expensive than the GPRS transmission cost. The former is about 20 cent of Euro, the latter is about 0.6 cent of Euro per transmitted kilobyte. As the dimension of data exchanged between peers is low (few bytes), it is more convenient to use a GPRS transmission rather than a SMS communication. Indeed, the proposed protocol has been thought to reduce the number of exchanged SMS messages; to establish an appointment among *N* peers, it needs no more than *N* SMS: the peer that initiates the negotiation sends its IP to all participants (*N* - 1).

## 4. IMPLEMENTATION FEATURES

In the current section we provide some implementation insights of our Interact-Agenda application. We will concentrate only on those packages in which .NET CF has been heavily used, providing specific functionalities to be considered during the development of smartphone application, i.e., networking, data storage and data access, user interface logic (windows, forms, listbox, etc.) and device interactions (keypad, joypad, home button, record button, soft keys, etc.). The development of the other packages, being pure business logic, does not present peculiarities due to smartphone device and .net CF (it is "normal" C# code). We will stress the possibility and the simplicity for an programmer to implement powerful and graceful application running on smartphone devices, both in terms of used libraries and generated source code lines.

### User Interface Development

The Base Class Library provides a sufficient subset of the components/widgets provided in the workstation/server .NET Framework; therefore no further training time is needed by a programmer in order to develop the user interface. In order to build graceful forms, we use the `System.Windows.Forms` and the `System.EventHandler` packages. The produced code is similar to that would have been produced for workstation applications.

### Networking

The networking logic has been realized by combining the .NET CF library `System.Net` and the .NET CF mechanism for external procedure call, i.e., the method provided by the .NET CF technology (more in general by the .net Framework) to invoke procedures contained in external libraries (`.dll`). Indeed, in order to use both SMS and GPRS communication, we had to consider external native libraries, and embed calls to these `.dll` in the source code managing our peer-to-peer protocol. Then, in order to establish a connection between two peers over the GPRS protocol, we used the system library `System.Net`, that provides all functions needed to manage socket-oriented connections over TCP/IP networks.

The code shown in the following is the one executed by the initiator for sending an SMS with its IP address and creating a specific socket; contacted device has to reply with an “I-am-alive” message on that socket (see class `Receiver`). Again, it is not very different from the one to be used in workstation/server scenarios for managing socket-oriented communications.

```
public class Initiator{
 .............
 /* GPRS Connection Management */
   private System.Net.Sockets.Socket
      getBindSocket() {
      GPRS.DataCall();
      /* Getting local device IP */
      IPHostEntry ipHostInfo =
      Dns.Resolve(Dns.GetHostName());
      IPAddress ipAddress =
ipHostInfo.AddressList[0];
     ..........
      /* Opening TCP/IP socket */
      System.Net.Sockets.Socket sock = new
System.Net.Sockets.Socket(System.Net.Sockets.Addres
sFamily.InterNetwork,System.Net.Sockets.SocketType.
Stream,System.Net.Sockets.ProtocolType.Tcp);
      try {
         sock.Bind(localEndPoint);
         sock.Listen(10);
      }
      catch(Exception e) {
         MessageBox.Show(e.ToString());
         throw(e);
      }
      return sock;
   }
   /* Message sending */
   private void contact() {
      for(int i = 0; i < guest.Count; i++) {
      try {
SMS.SendMessage(guest[i].phoneNumber,this.myIP);
      }
      catch(Exception sendSms){}
      }
}
   .........
}

public class Peer
{    ..............
   private void sendMessage(Message msg) {
      netStream = new
        System.Net.Sockets.NetworkStream(getConnec
        tSocket());
      writerClient = new StreamWriter(netStream);
      ..........
   }
}
```

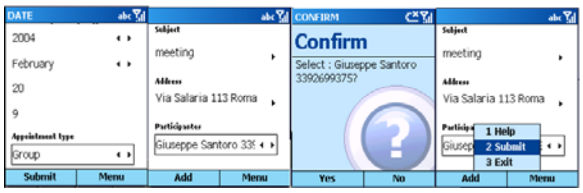
## Database

In the Smartphone 2003 SDK, up to now, there are no libraries and tools to manage local relational database. Therefore the Database package has been realized on the basis of the `FileSystem`. Specifically, all the $\mu$-databases used in the Interact-Agenda consist of collections of text files stored on an external SD-CARD memory. Through the file management, the package provides a very simple relational-like interface, allowing upper layer to create tables, columns simple constraints and primary keys. Simple querying capabilities (specifically *select* but not *join*) have been provided. Specifically, the .NET CF libraries `System.IO` and `System.Data` have been used for writing and reading operations on files and for table and column manipulation, respectively. In the following code sample we report database management, in particular how tables are stored into text files.

```
public class DataBase{
   ..........
   /* Writing table to file.txt */
   public static bool WriteTable(DataTable tab) {
      System.Data.DataTable table = tab;
      System.String fileTable = table.TableName;
      ForeignKeyConstraint key = null;
      UniqueConstraint pKey = null;
      /* openig table file.txt */
      System.IO.StreamWriter fileWriter =
        System.IO.File.CreateText("\\"+fileTable+"
.txt");
/*Writing Table information: Name, column number */
      fileWriter.WriteLine(
      table.TableName.ToString());
      fileWriter.WriteLine(
      table.Columns.Count.ToString());

      /* Writing column's name and type */
      for(int i = 0; i < table.Columns.Count; i++)
        fileWriter.WriteLine(table.Columns[i].Colu
mnName);
        fileWriter.WriteLine(table.Columns[i].Data
Type.ToString());
      }
. . . .
}
```

Finally, Figure 5 shows the sequence of windows presented to the initiator user when establishing a group appointment.



2004
February
20
9
Group
meeting
Via Salaria 113 Roma
Giuseppe Santoro 33

Select : Giuseppe Santoro 3392699375?

**Figure 5. Task to Establish a Group Appointment.**

## 5. CONCLUSION AND DISCUSSION

The principal goal of the present work has been to investigate how to develop and design distributed applications for small devices (PDAs, smartphones, etc.), using current technologies such as .net CF platform. Our main conclusion, supported by the analysis of the produced code, is that concepts, patterns and development methods that are used in traditional software construction can be seamlessly applied to smart device application development. Clearly this is due to the availability of a framework that is similar to desktops/workstations/servers. In particular, the network programming interfaces are powerful enough for developing peer-to-peer applications on cellular phones, that was the second main objective of this work.

Some observations and recommendations can be made; specifically, when we design applications for smart devices, we must consider some factors such as transmission cost, security, privacy, performances and development platforms. With respect to them, smartphone applications must be able to use several transmission channels (our application uses two distinct channels: SMS and GPRS), and their protocols must be able to support different communications. In the future work we would like to apply the gained knowledge about design, development, and deployment of mobile applications in particular scenarios, such as the one of MANET (Mobile or Multi-hop ad hoc NETwork), in which mobile devices communicate on the basis of a non-fixed network [15].

## 6. ACKNOWLEDGMENTS

This work is supported by MIUR through the FIRB 2001 Project MAIS. Thanks to Microsoft Research - University Relationship Department and Marco Combetto for providing the prototype devices.

## 7. ADDITIONAL AUTHORS

Angelo Ritucci and Giuseppe Santoro, bachelor students of the Faculty of Computer Engineering, Università di Roma “La Sapienza”.

## 8. REFERENCES

[1] K. Aberer and Z. Despotovic. Managing Trust in a Peer-2-Peer Information System. In Proceedings of the 10th International Conference on Information and Knowledge Management, Atlanta, GA, USA, 2001.

[2] KaZaA. http://www.kazaa.com.

[3] Napster. http://www.napster.com.

[4] M.C. Fauvet, M. Dumas, B. Benatallah, and H.Y. Paik. Peer-to-Peer Traced Execution of Composite Services. In Proceedings of the 2nd VLDB International Workshop on Technologies for e-Services (VLDB-TES 2001), Rome, Italy, 2001.

[5] Gnutella. The Gnutella Protocol Specification (version 0.4). http://www9.limewire.com/developer/gnutella_protocol_l 0.4.pdf, June 2001.

[6] T. Kato, N. Ishikawa, H. Sumino, J. Hjelm, Y. Yu, and S. Murakami. A Platform and Applications for Mobile Peer-to-Peer Communications. NTT DoCoMo & Ericsson Research Document, 2003, http://www.research.att.com/~rjana/Takeshi Kato.pdf.

[7] A. Klemm, C. Lindemann, and O.P. Waldhorst. A Special-Purpose Peer-to-Peer File Sharing System for Mobile Ad Hoc Networks. In Proceedings IEEE Semiannual Vehicular Technology Conference (VTC2003-Fall), Orlando, FL, October 2003.

[8] RFC 2779 - Instant Messaging / Presence Protocol Requirements. http://www.faqs.org/rfcs/rfc2779.html.

[9] International Telecommunications Union Standard H.323. http://http://www.itu.int/.

[10] The OpenH323 Project. http://www.openh323.org/

[11] SIP - Session Initiation Protocol. http://rfc.sunsite.dk/rfc/rfc3261.html.

[12] SIMPLE - SIP for Instant Messaging and Presence Leveraging Extensions. http://www.ietf.org/html.charters/simple-charter.html.

[13] .NET Compact Framework. http://msdn.microsoft.com/mobility/prodtechinfo/devtools/net cf/.

[14] Java 2 Micro Edition. http://java.sun.com/j2me/.

[15] F. De Rosa, V. Di Martino, L. Paglione, and M. Mecella. Mobile Adaptive Information Systems on MANET: What We Need as Basic Layer? In Proceedings of the 1st Workshop on Multichannel and Mobile Information Systems (MMIS'03), Rome, Italy, December 2003, IEEE.