ToothAgent: A Multi-Agent System for Virtual Communities Support

Volha Bryl [*]
Department of Information and Communication Technology
University of Trento, Italy
volha.bryl@unitn.it

Paolo Giorgini
Department of Information and Communication Technology
University of Trento, Italy
paolo.giorgini@dit.unitn.it

Stefano Fante
ArsLogica Lab
Mezzolombardo (TN), Italy
stefano.fante@arslogica.it

ABSTRACT

People tend to form social networks within geographical areas. This can be explained by the fact that generally geographical localities correspond to common interests (e.g. students located in a university could be interested to buy or sell textbooks adopted for a specific course, to share notes, or just to meet together to play basketball). Cellular phones and more in general mobile devices are currently widely used and represent a big opportunity to support social communities. In this paper, we present a general architecture for multi-agent systems accessible via mobile devices (cellular phones and PDAs), where Bluetooth technology has been adopted to reflect users locality. We illustrate ToothAgent, an implemented prototype of the proposed architecture, and discuss the opportunities offered by the system.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*multiagent systems*; H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Multi-agent systems, mobile devices, virtual communities, Bluetooth

1. INTRODUCTION

Being widespread and ubiquitous, cellular phones are recently used not only as means of traditional communication. They are also supposed to satisfy the information needs of their users, e.g. to support information search and filtering or electronic data exchange [13, 1, 15, 10]. Users equipped with mobile devices, such as cellular phones or PDAs, can form so called mobile virtual communities [14], which make possible the interaction and the information exchange between their geographically distributed members. Such communities are inherently open, new users can join and existing ones can leave anytime.

A number of multi-agent applications to mobile device environments have been proposed in literature. [10] presents a multi-agent system named KORE where a personal electronic museum guide provides to visitors (with Java-enabled mobile devices) information about artistic objects they are currently looking at. Information is filtered and adapted to the user profile. Bluetooth technology [2] is used to detect the user position. Bluetooth is a cheap and a widely used wireless communication technology able to connect Bluetooth-enabled devices located in a range of 100 meters. [15] proposes MobiAgent, an agent-based framework that allows users to access various types of services (from web search to remote applications control) directly from their cellular phones or PDAs. Once the user sends a request for a specific service, an agent starts to work on her behalf on a centralized server. The user can disconnect from the network and the agent will continue to work for her. When the request has been processed, the user is informed via Short Message Service and she can decide to reconnect the network and download the results. MIA information system [1] is another example that provides personalized and localized information to users via mobile devices.

What is still missing in these systems is the interaction between the members of the virtual community. Just few proposals in literature introduce domain-specific environments where interacting agents act on the behalf of their users. For instance, [11] describes a context-aware multi-agent system for agenda management where scheduling agents can execute on PCs or PDAs and assist their users in building the meeting agenda by negotiating with the other agents. ADOMO [12] is an agent-based system where agents running on mobile devices sell the space on the device screen to commercial agents for their advertisements. Agents on behalf of their users negotiate and establish contracts with neighbors via Bluetooth.

There exist a number of multi-agent platforms that can be used on mobile devices. Taking into account the limited computational and memory resources, it could be very problematic to run a multi-agent platform on such mobile devices as cellular phones. A possible solution is either to avoid running multi-agent platform on mobile devices, as for example in [12], or to use portal multi-agent platforms [13] where

[*]Ph.D. Student

agents are executed not on the device itself but on the external host.

In this paper we present a general architecture based on this last option. The architecture proposes independent servers where multi-agent platforms can be installed and where agents can act on behalf of their users. Each server proposes one or more specific services related to the geographical area in which it is located (e.g. a server inside the university could offer the service of selling and buying textbooks, renting an apartment, etc.), and users can contact their personal agents using their Bluetooth-enabled mobile phones or PDAs. The main advantage of the proposed framework with respect to the above described architectures is that the system is domain independent (it does not depend on the specific services offered by the servers) and independent from the multi-agent technology adopted (we can use different technologies on each server).

The paper is organized as follows. Section 2 describes a motivating example for our system. The general architecture of the system is introduced in Section 3, while Section 4 provides some architectural details and describes ToothAgent, the implemented prototype. Section 5 concludes the paper and provides some future work directions.

2. MOTIVATING EXAMPLE

Let's consider three places in a town: university, railway station and bar. People spending some time in one of these places may have some common interests and needs. For instance, students at the university might want to buy or to sell secondhand textbooks, find a roommate or form study groups. People at the bar could be interested in the latest sport news (especially in Italian bars), or they could just be looking for someone to chat with. Passengers waiting at the railway station may want to know some details about the trip they are going to have — what cities their train goes through, or what the weather is like at the destination point. They may want also to find someone with common interests to chat with during the trip.

Let's suppose also that people cannot or do not want to spend their time on examining announcements on the bulletin boards, questioning people around them, or searching for the information office. They would prefer to ask their mobile phones and wait for the list of available proposals.

To support interests and needs of such groups of co-localized users, a server is placed at each of the three meeting points. Servers can provide a certain number of services to people equipped with mobile phones or PDAs (hereinafter referred as users). A user can have access to the services when she is close enough (depending on her Bluetooth device) to one of the three servers — at the bar, in the waiting room of the station, or at the main hall of the university.

Let's suppose that among the available services we have the following ones. At the university users can buy/sell their secondhand books and search for roommates. At the bar, users can access a sport news service or search for "interesting" people. Finally, at the railway station, users can receive information about their trips (including touristic information).

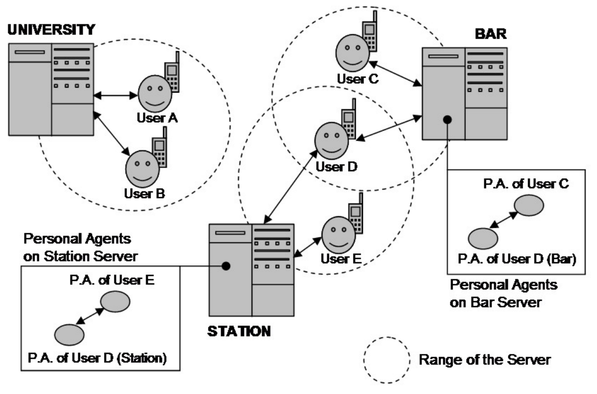


Figure 1: Users, servers, and virtual communities of personal agents.

User interaction is often essential for the satisfaction of their needs. To sell a secondhand textbook, one should find a buyer and agree on the price. To find someone in the bar to chat with, one should look for the person with similar interests or preferences. Each server recreates the group of co-localized human users in a virtual community of personal agents (Figure 1) able to interact with one another. Users formulate their requests and forward them to their personal agents.

Personal agents interact with the other available agents (the interaction might include negotiation as in the case of selling or buying books), and produce results that will be sent back to the users. The main idea is to have a distributed system composed of a number of open virtual communities that evolve and act autonomously on the behalf of human communities.

3. SYSTEM ARCHITECTURE

In this section we describe the general architecture of the system. We start from the requirements and then we illustrate the various sub-components and their interaction.

3.1 System Requirements

We can summarize the requirements of the whole system in the following objectives.

- The system should allow the user to express her interests and choose the services she wants to access. It means that the user should be able to search for available servers and services by location, category, keywords, etc. The user should be provided with an interface to select services she is interested in and to customize them, i.e. to specify parameters of the requests to these services (e.g. book title and price for "buy/sell books" service). The system should allow to view and edit customized requests, and transfer the list of requests to a mobile device.
- The system should provide access to the requested services when the mobile device and the appropriate server are co-localized (i.e. the Bluetooth connection is feasible). This means the system should be able to support the search for servers (and corresponding services)

in the neighborhood of the mobile device, and the verification of the mobile device by the server. Then user's requests should be transferred to the server, where it should be possible to find correspondence between the user and her personal agent. The system should also support interaction among the personal agents, store the results and let users access these results from their mobile devices.

- The system should allow the user to retrieve pending results. Results should be accessible both in case the user is still in the Bluetooth range of the server and when she is out of the range. In the second case the system should allow the user to access pending results from her mobile device by connecting to any server of the system, or from her PC via a dedicated interface. To do this, the system should support interaction between different servers and between the server and the PC. The system should keep track of all servers visited by the mobile devices and transfer these information to the PC or to the connected server.

3.2 System Components

The architecture of the system includes four main types of components: mobile device, PC, server and services database.

The *PC component* provides an interface for the user registration, to get and to choose available services, and to build requests for the chosen services. Also the pending results can be retrieved via PC. The *mobile device* is used to send the user's requests to the servers and to get back the results. Each *server* within the system provides a list of predefined services. The server runs a multi-agent platform with personal agents representing single users, a database where results are archived, and an interface responsible for establishing connections with mobile devices and PCs, and for redirecting the users' requests to the corresponding personal agents. The central *services database*, accessible via web, contains information about all the servers and their properties, such as name, location, etc. The database also provides a description of available services on each server, and stores the information about users registered to the system.

Figure 2 illustrates the general architecture of the system and the interaction among its components. Connection between the mobile device and the PC, and between the mobile device and the server is established via Bluetooth wireless communication technology.

3.3 Getting Access to the Services

In the following we describe how the process of getting access to the services is organized (Figure 3).

The software running on the PC allows the user to search and discover the servers and the services registered to the services database (steps 1–3). The user selects one or more services and provides information (i.e. requests) related to the use of such services (step 4). For example, using the service "Buy/sell secondhand books", the user could request to "Sell the copy of *Thinking in Java* by Bruce Eckel, printed in 1995, for the price not less than 20 euros". All the user's

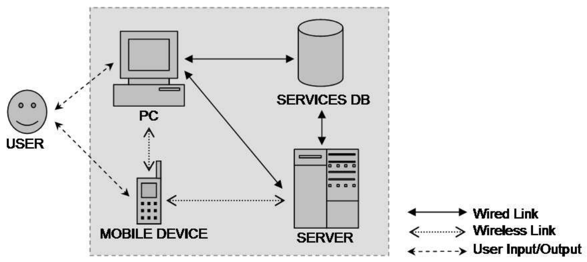


Figure 2: Interaction of system components.

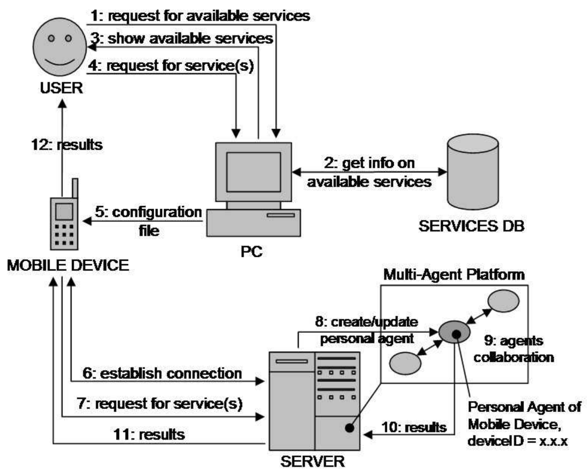

1: request for available services
3: show available services
4: request for service(s)
USER
12: results
2: get info on available services
5: configuration file
SERVICES DB
MOBILE DEVICE
PC
Multi-Agent Platform
8: create/update personal agent
9: agents collaboration
6: establish connection
7: request for service(s)
11: results
10: results
Personal Agent of Mobile Device, deviceID = x.x.x
SERVER

Figure 3: Getting access to services.

requests are stored in the configuration file, which is downloaded onto the mobile device via Bluetooth (step 5).

When the user with her mobile device approaches one of the servers, the software on the device establishes a connection with the server (step 6) and sends requests related to the available services (step 7). The requests are built on the base of the configuration file of the mobile device. In other words, the mobile device checks in the configuration file if the user is interested in the services provided by the server and then builds and sends the requests to the server. The requests are processed on the server, and the results are sent back to the user (steps 8–12) The mobile device stores the server address to keep track of the contacted servers. It stores the address even if there are no relevant services on the server. This allows the user to check later the list of all visited servers and associated services, and decide to update her preferences by including new servers/services in the configuration file.

3.4 Retrieving Pending Results

We describe now how the process of retrieving the pending results is organized (Figure 4).

The user has basically two options to get back the results of her requests. The first one is to receive them directly on her mobile device. However, this is not always possible. The user could leave the Bluetooth area or the mobile device

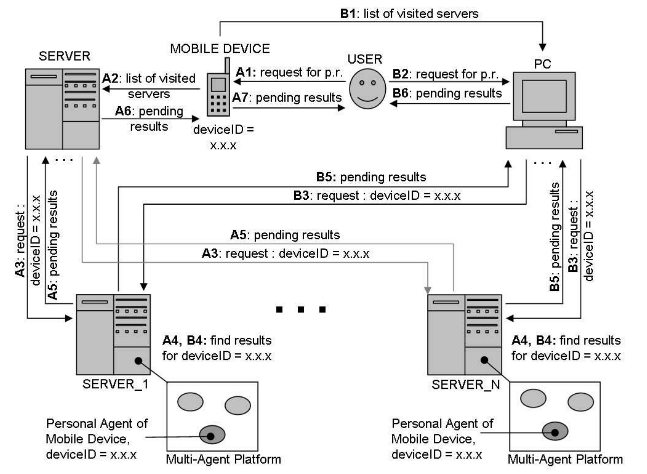

B1: list of visited servers
MOBILE DEVICE
SERVER
USER
PC
A2: list of visited servers
A6: pending results
A1: request for p.r.
A7: pending results
B2: request for p.r.
B6: pending results
deviceID = x.x.x
B5: pending results
B3: request : deviceID = x.x.x
A5: pending results
A3: request : deviceID = x.x.x
A3: request : deviceID = x.x.x
A5: pending results
B5: pending results
B3: request : deviceID = x.x.x
A4, B4: find results for deviceID = x.x.x
SERVER_1
SERVER_N
Personal Agent of Mobile Device, deviceID = x.x.x
Multi-Agent Platform

Figure 4: Retrieving pending results from mobile device (A) and from PC (B).

may not have enough memory or computational power to manage the answers (e.g. in the case the answers are a number of big files). Thus the second option is to get back the results later when the connection with the server they were requested from is closed.

Pending results can be retrieved both from the mobile device (steps A.x) and from the PC (steps B.x). In the first case the mobile device has to be configured to get the pending results and has to be in the Bluetooth range of some server. For example, if a student is going to spend a whole hour in the main hall of the university waiting for the next lecture, namely she will have enough time to download the results of her requests sent in the morning to the railway station server (where she bought her train ticket before going to the university). She switches on the option "get pending results" on her mobile phone (step A1), and waits for results. The mobile device sends to the university server the list of addresses of the servers the user has visited (step A2). The server establishes a connection with each server in the list, and sends the information that identifies the mobile device (e.g. its Bluetooth address) as a request for the pending results (step A3). The obtained information is sent back to the mobile device (steps A5–A6).

In the second case the user receives pending results through the PC. The student goes back home and runs the PC software that collects all the pending results obtained from the visited servers (steps B3–B5), after the list of the visited servers and their addresses is transferred from the mobile device to the PC (step B1).

3.5 Agent Platform

Each server runs a multi-agent platform, where agents correspond to mobile devices and receive and process requests obtained from the users. There is a one-to-one correspondence between agents and mobile devices (users). An agent is identified by a unique Bluetooth address of the corresponding mobile device. The same device can have many personal agents within different platforms on different servers.

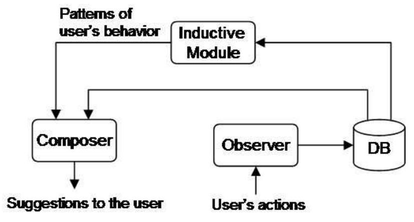


Figure 5: General architecture of SICS.

When the server receives the request from the mobile device, it checks if the personal agent of this device exists within the platform. If not, a new personal agent is created. Each personal agent communicates and interacts with other agents in order to find "partners" which will satisfy its request. Interaction protocols mechanisms are domain (services) dependent.

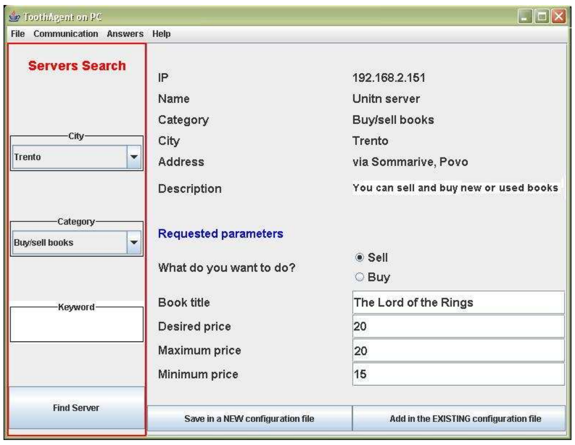
Multi-agent platforms on the server are based on the Implicit Culture [9] framework. In short, Implicit Culture allows new members of a community to behave in accordance with the culture of the community. For example, a new student may not know which textbooks can be helpful for the Programming Languages course and starts to search for *Textbook on Programming Languages*. The idea of the Implicit Culture framework is that the system suggests the student the items that are usually used by the other members of the community. So for example, the system could suggest the student to search for the book *Thinking in Java*.

Multi-agent systems with the Implicit Culture support are used for example for searching the web. See [8] for the description of Implicit, an agent-based recommendation system for the web search, which improves the search for information for a community of users with similar interests. When a user submits a query, Implicit looks for the relevant information, exploiting observations of the behavior of other users when they have issued similar queries.

To follow the Implicit Culture concept the agent on the platform should contain System for Implicit Culture Support (SICS) [9]. SICS consists of three basic components (their interaction is illustrated in Figure 5):

- *Observer*, which stores in a database the information about user actions (observations).
- *Inductive module*, which analyzes the obtained observations and induces behavioral patterns of the community using Data Mining techniques.
- *Composer*, which produces suggestions on the base of the information from the Observer and Inductive module.

More details about the Implicit Culture framework are available at [3].


ToothAgent on PC
File Communication Answers Help
Servers Search
City
Category
Keyword
Find Server
Save in a NEW configuration file
Add in the EXISTING configuration file

City: Trento
Category: Buy/sell books

IP	192.168.2.151
Name	Unitn server
Category	Buy/sell books
City	Trento
Address	via Sommarive, Povo
Description	You can sell and buy new or used books

Requested parameters

What do you want to do?	[✓ Sell] Buy
Book title	The Lord of the Rings
Desired price	20
Maximum price	20
Minimum price	15

Figure 6: Request input form.

4. IMPLEMENTATION ISSUES

In this section we present the details of ToothAgent, the implemented prototype of the proposed architecture. Basically, the system is a first implementation of the architecture presented in Section 3 and focuses on a number of servers spread around the university campus (faculties, libraries, and departments). Each server offers only the service for selling and buying books. We are currently working on a number of other services including ones available on servers located outside the university campus (e.g. train station, museums and places close to touristic attractions).

We tested the system using Nokia 6260 cellular phones and PC/Server equipped with Tecom Bluetooth adapter. Bluetooth communication has been implemented using Blue Cove [4] which is an open source implementation of the JSR-82 Bluetooth API for Java.

4.1 Online Registration and Service Selection

To start working with the system, the user has to register. To do this she should fill the online registration form where she needs to put her personal information such as name, birth date, e-mail, Bluetooth address and phone number of her mobile device, and password. The registration, basically, allows the system to identify the user and the mobile device she is going to use. Password is used to access the information about servers and related services, and to upload/update the user information (e.g. the user can decide to use different mobile device or just to change her data such as telephone number or e-mail address). Also the password is needed to access servers and their services via mobile device (for this purpose user has to input the password while configuring the application on her mobile device). All the information about the user is stored in the services database. Registered users obtain the rights to download the software for PC and mobile device components (which are two jar files), and the XML file containing information about all available servers with corresponding services.

After the registration (or login), the user can start selecting services to use. Using the Java GUI interface shown in Figure 6, she can explore all the available services using filtering criteria such as server location (e.g. we can have servers located in different cities or in different places in the same city), type or category of the service (e.g. buy/sell books, exchange course notes, or meet people), and keywords (e.g. books, course, etc.). The list of the selected services is managed by the PC component that allows the user to customize these services with the specific requests (e.g. title of the book to buy or to sell, the desired price, minimal or maximal price).

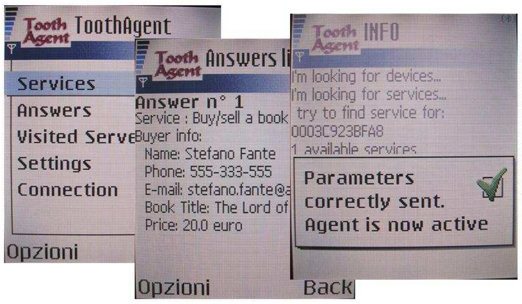
The list of customized services (with related servers addresses) is stored in an XML configuration file, which is uploaded via Bluetooth in the mobile device. Figure 7 shows an example for the "sell/buy books" service. Note that the file format does not depend on what services it describes, i.e. it is domain independent.

4.2 Accessing the Services

To access the services, the user needs to run the Bluetooth application on her mobile device (Figure 8). The application is written in Java and uses JSR-82 [5], which is Bluetooth API for Java. The application starts a continuous search for Bluetooth-enabled devices in the neighborhood, and whenever it finds a server with the services specified in the configuration file, the mobile device sends the user requests to the server. Figure 9 shows the protocol we use for the interaction among the different components.

```
<server>
  <ip>       192.168.2.151                </ip>
  <name>     UnitnServer                  </name>
  <location> Trento, via Sommarive, Povo </location>
  <service>
    <description> Buy/sell new and used books </description>
    <parameters>
      <param>
        <paramname>  Book title         </paramname>
        <paramtype>  String             </paramtype>
        <paramvalue> Lord of the Rings  </paramvalue>
      </param>
      <param>
        <paramname>  Desired price </paramname>
        <paramtype>  int           </paramtype>
        <paramvalue> 15            </paramvalue>
      </param>
      <param>
        <paramname>  Maximum price </paramname>
        <paramtype>  int           </paramtype>
        <paramvalue> 20            </paramvalue>
      </param>
    </parameters>
  </service>
</server>
```

Figure 7: Configuration file.



Answer n° 1
Service : Buy/sell a book
Buyer info:
Name: Stefano Fante
Phone: 555-333-555
E-mail: stefano.fante@a
Book Title: The Lord of
Price: 20.0 euro

I'm looking for devices...
I'm looking for services...
try to find service for:
0003C923BFA8
1 available services

Parameters correctly sent. Agent is now active

Figure 8: ToothAgent application running on the mobile phone.

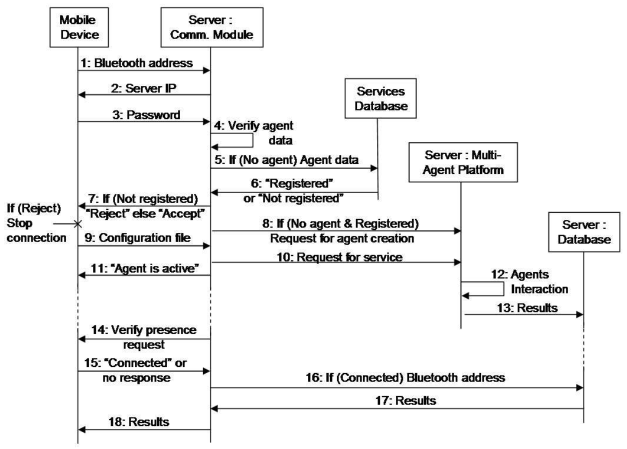

Mobile Device
Server : Comm. Module
Services Database
Server : Multi-Agent Platform
Server : Database
1: Bluetooth address
2: Server IP
3: Password
4: Verify agent data
5: If (No agent) Agent data
6: "Registered" or "Not registered"
7: If (Not registered) "Reject" else "Accept"
If (Reject) Stop connection
8: If (No agent & Registered) Request for agent creation
9: Configuration file
10: Request for service
11: "Agent is active"
12: Agents Interaction
13: Results
14: Verify presence request
15: "Connected" or no response
16: If (Connected) Bluetooth address
17: Results
18: Results

Figure 9: Getting access to services.

A specific communication module on the server is responsible for managing the interaction with the mobile device. It receives the Bluetooth address and the encrypted password from the mobile device (steps 1 and 3) and checks whether in the platform running on the server a personal agent assigned to that mobile device already exists (step 4), the Bluetooth address is used to map the mobile device with the personal agent. If there is no personal agent for the user, the communication module connects to the central services database and verify whether the user is registered to the system (steps 5–6) by matching the Bluetooth address of the device and the password. Only in case of a positive answer, it creates a new agent and assigns it to the mobile device user (step 8). Then, the mobile device sends the configuration file to the communication module (step 9), which forwards all the user requests to the personal agent (step 10).

Now, the personal agent starts interacting with other agents on the platform trying to satisfy all the user requests (step 12). In our example a personal agent receives one or more requests for buying and/or selling books (with specified title, desired price, maximum and minimum prices, etc.). If the agent reaches an agreement with another agent about their users requests it stores the results locally in the server database (step 13). Later the results could be sent back to the user (steps 14–18) or left on the server, depending on the retrieval modality that the user has defined in the configuration file.

4.3 Results Retrieval

Whenever a new connection between a server and a mobile device is established, the communication module sends to the mobile device the IP-address of the server (step 2 on Figure 9). The mobile device stores the IP addresses of all the visited servers in an XML list, that is used later to retrieve all pending results. The format of the results produced by the personal agent is shown in Figure 10. It may contain the request identifier, contacts (e.g. phone number) of the user interested to buy or sell the book, the actual agreed price, etc.

As discussed in Section 3, the user has three different modalities to retrieve results: get the results immediately, get

```
<responses>
  <response>
    <ip>          192.168.2.151               </ip>
    <name>        UnitnServer                 </name>
    <description> Buy a book </description>
    <parameters>
      <param>
        <paramname>  Book title         </paramname>
        <paramtype>  String             </paramtype>
        <paramvalue> Lord of the Rings  </paramvalue>
      </param>
      <param>
        <paramname>  Price      </paramname>
        <paramtype>  int        </paramtype>
        <paramvalue> 20         </paramvalue>
      </param>
      <param>
        <paramname>  Buyer name         </paramname>
        <paramtype>  String             </paramtype>
        <paramvalue> Stefano Fante      </paramvalue>
      </param>
      <param>
        <paramname>  Buyer phone number  </paramname>
        <paramtype>  String              </paramtype>
        <paramvalue> 555-321-555         </paramvalue>
      </param>
    </parameters>
  </response>
  ...
</responses>
```

Figure 10: List of responses.

pending results using the mobile device, and get pending results using the PC. Each of these modalities has to be defined in advance by the user and can be changed at runtime by means of the mobile device application.

Choosing the first option, the user can receive the results immediately in her mobile device. Of course, she can receive the results if and only if she is still at a Bluetooth distance from the server. The communication module checks the availability of the mobile device and sends to it the results stored in the internal server database by the corresponding personal agent (see Figure 9, steps 14–18).

Figure 11 shows the interaction protocol of retrieving pending results via mobile device. Consider for example a situation in which a user is near to the server of the central library. After the connection has been established, the mobile device sends the list of IP-addresses of all previously visited servers (e.g. faculty servers, departments servers, etc.) to the library server (step 2). The communication module of the server sends then the Bluetooth address of the mobile device to all listed servers (step 3). In turn, the communication module of each server extracts from the internal database all the stored results related to that user and sends them back to the requester server (steps 4–7). All the results are collected by the communication module and finally sent to the mobile device (steps 8–10). If the mobile device is no longer connected to the server (e.g. the user has left the library), the retrieval process will fail and the results will be cancelled (they are still available on the original servers).

Figure 12 shows the interaction protocol of retrieving pending results via PC. A user connects her mobile device to the PC via Bluetooth and sends the list of all visited servers to the PC component (step 2). Now, the user can decide either to retrieve the results from all the servers or just to

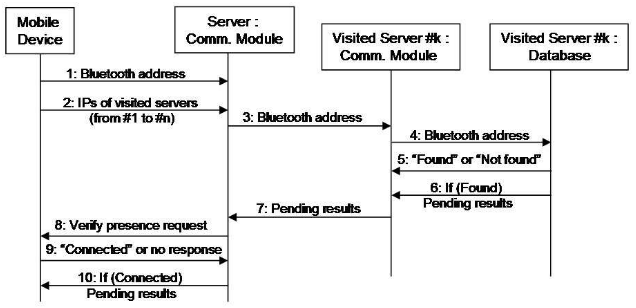

Mobile Device
Server : Comm. Module
Visited Server #k : Comm. Module
Visited Server #k : Database
1: Bluetooth address
2: IPs of visited servers (from #1 to #n)
3: Bluetooth address
4: Bluetooth address
5: "Found" or "Not found"
6: If (Found) Pending results
7: Pending results
8: Verify presence request
9: "Connected" or no response
10: If (Connected) Pending results

Figure 11: Pending results from the mobile device.

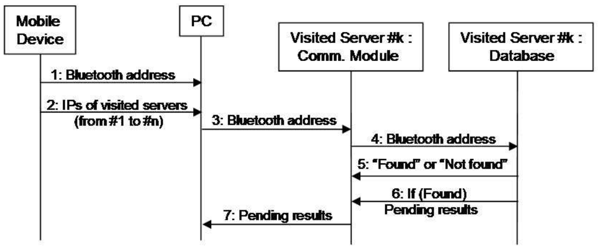

Mobile Device
PC
Visited Server #k : Comm. Module
Visited Server #k : Database
1: Bluetooth address
2: IPs of visited servers (from #1 to #n)
3: Bluetooth address
4: Bluetooth address
5: "Found" or "Not found"
6: If (Found) Pending results
7: Pending results

Figure 12: Pending results from PC.

select some of them. An interface on the PC allows the user to connect to the servers and then to view or download the pending results (steps 3–7).

4.4 Agents Interaction

As we said, in this first prototype we implemented just one kind of service, namely the "buy/sell books" service. The multi-agent system has been implemented in JADE (Java Agent DEvelopment framework) [6], FIPA-compliant [7] framework for multi-agent systems development. The agent interaction includes two phases: elaboration of user request and agent negotiation.

During the first phase the request of buying/selling a book is elaborated and detailed. For example, the request of "Buy a textbook on Java for the price from 10 to 20 euros" is incomplete as the exact title is not specified. The personal agent makes the request more clear exploiting the information about what textbooks on Java other users were recently interested in and what they have finally bought, at what prices, etc. Another example of request that needs to be elaborated could be "Buy *Thinking in Java* for the price less than 10 euros". It is unlikely that this request will be satisfied as all copies of *Thinking in Java* currently available, or sold so far, cost at least 20 euros. The user has clearly underestimated the price. In this case we want the personal agent to extend the price range when starting to search for a copy of the book.

Figure 13 presents the interaction protocol used by agents during the request elaboration phase. On each platform there is a dedicated agent, called Expert Agent (EA), which contains the System for Implicit Culture Support (SICS). After a personal agent receives its user's request (step 1), it sends it to the Expert Agent (step 2). On the EA side

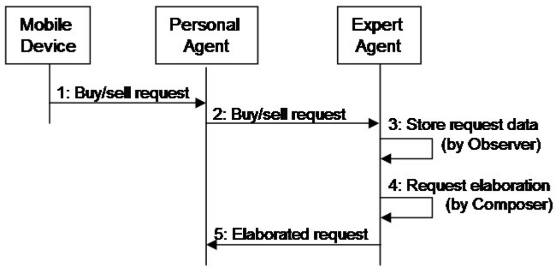


Figure 13: User request elaboration.

Observer component of the SICS extracts data from the request and stores it in the database of observed user behaviors (step 3). Composer component estimates the real price for the requested book and/or suggests the title of the book if the input was incomplete (step 4). For the elaboration process Composer uses the information about the past user actions, obtained from Observer and analyzed by Inductive module. At the end the user's personal agent gets back the elaborated request (step 5), which it will process during the second phase.

As it was explained in Section 3.5, SICS needs to gather information about user behavior. In the described prototype to observe the user behavior Expert Agent extracts data from the requests it gets from personal agents. Two other additional sources of observations could be added. The first one is the database where results of agent negotiations are stored. Each time two personal agents agree on buying/selling a book and send their proposals to the database, Expert Agents extracts necessary information (e.g. book title and the price) from the proposals and stores it in its internal database. The second source is the direct user feedback. When the user views the list of proposals on her mobile device, she can choose to make a phone call or to send an SMS to the other user whose contacts are in the proposal. When the proposals are viewed on the PC, the user can choose to write an e-mail to her potential partner. For the purpose of feedback the system records the information about these phone calls/SMSs/e-mails assuming that if the other user of the proposal (the potential partner) was contacted then the feedback is positive, otherwise negative. The feedback information is sent to the Expert Agent as soon as the user establishes connection with the corresponding server via her mobile device or the PC.

On the second phase — agent negotiation — the interaction mechanism is very simple. Figure 14 presents the implemented agent interaction "from the point of view" of the agent which is buying a book. First, the buyer's personal agent broadcasts the request of looking for a specific book (step 1), information about title, desired price, etc. is specified in the message. If in the platform there is another agent that is selling the requested book, it responds to the buyer with the price it wants for the book (step 2). If the price is greater than the maximum price specified by the buyer, the interaction continues with the request for discount from the buyer agent (step 3). The seller responds either with the discounted price or with the initially proposed price (step 4) in case it does not want to give the discount. If this price is

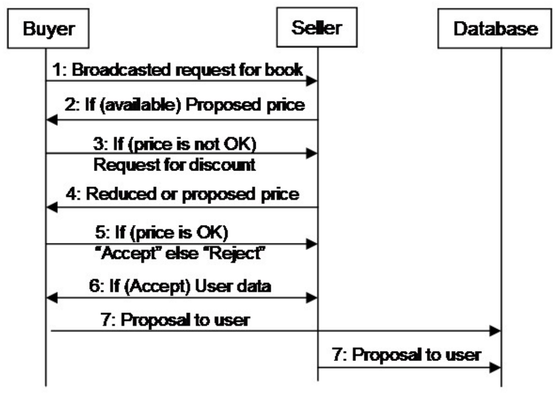

Buyer
Seller
Database
1: Broadcasted request for book
2: If (available) Proposed price
3: If (price is not OK) Request for discount
4: Reduced or proposed price
5: If (price is OK) "Accept" else "Reject"
6: If (Accept) User data
7: Proposal to user
7: Proposal to user

Figure 14: Agent negotiation.

less than maximum price for the buyer, it accepts the deal (step 5). After that, the buyer and seller personal agents exchange their users' data (step 6), form the agreed proposals and send them to the server database (step 7). The proposals are then forwarded either to mobile device, or to the PC as described in Section 4.3.

5. CONCLUSIONS

In this paper we have presented an implemented prototype where multi-agent systems and Bluetooth wireless communication technology are combined together to support co-localized communities of users. We have discussed the general architecture of the system, and presented some implementation issues related to ToothAgent, the prototype we have built.

We are currently working with ArsLogica s.r.l. on the development of a real-life scenario where many different services will be available in the city center of Trento (including the university campus). Of course a lot of work has to be done before the real use of the system. In particular, we need to verify the scalability of the system and test its performance for a considerably high number of users. Some preliminary tests have, however, shown the effectiveness of the system in supporting co-localized community of users.

6. ACKNOWLEDGEMENTS

We thank ArsLogica s.r.l. for the collaboration and the support to this project. This work has been partially funded by EU Commission, through the SENSORIA and SERENITY projects, by the FIRB programme of MIUR under the ASTRO project, also by the Provincial Authority of Trentino, through the MOSTRO project. We also thank people working on Implicit Culture, in particular Aliaksandr Birukou and Enrico Blanzieri for their support in using the Implicit Culture framework.

7. REFERENCES

[1] MIA project — http://www.uni-koblenz.de/∼bthomas/MIA_HTML.

[2] The official Bluetooth website — http://www.bluetooth.com/.

[3] Implicit Culture website — http://dit.unitn.it/∼implicit/.

[4] Blue Cove project — http://sourceforge.net/projects/bluecove/.

[5] JSR-82: Java APIs for Bluetooth — http://www.jcp.org/en/jsr/detail?id=82.

[6] JADE: Java Agent DEvelopment Framework website — http://jade.tilab.com/.

[7] FIPA: Foundation for Intelligent Physical Agents — http://www.fipa.org/.

[8] A. Birukov, E. Blanzieri, and P. Giorgini. Implicit: An agent-based recommendation system for web search. In *Proceedings of the 4th International Conference on Autonomous Agents and Multi-Agent Systems*, pages 618–624. ACM Press, 2005.

[9] E. Blanzieri, P. Giorgini, P. Massa, and S. Recla. Implicit culture for multi-agent interaction support. In *CooplS '01: Proceedings of the 9th International Conference on Cooperative Information Systems*, pages 27–39, London, UK, 2001. Springer-Verlag.

[10] M. Bombara, D. Calì, and C. Santoro. Kore: A multi-agent system to assist museum visitors. In *Proceedings of the Workshop on Objects and Agents (WOA2003), Cagliari, Italy*, pages 175–178, 2003.

[11] O. Bucur, P. Beaune, and O. Boissier. Representing context in an agent architecture for context-based decision making. In *Proceedings of the Workshop on Context Representation and Reasoning (CRR'05), Paris, France*, 2005.

[12] C. Carabelea and M. Berger. Agent negotiation in ad-hoc networks. In *Proceedings of the Ambient Intelligence Workshop at AAMAS'05 Conference, Utrecht, The Netherlands*, pages 5–16, 2005.

[13] C. Carabelea and O. Boissier. Multi-agent platforms on smart devices : Dream or reality? In *Proceedings of the Smart Objects Conference (SOC03), Grenoble, France*, pages 126–129, 2003.

[14] A. Rakotonirainy, S. W. Loke, and A. Zaslavsky. Multi-agent support for open mobile virtual communities. In *Proceedings of the International Conference on Artificial Intelligence (IC-AI 2000) (Vol I), Las Vegas, Nevada, USA*, pages 127–133, 2000.

[15] L. Vasiu and Q. H. Mahmoud. Mobile agents in wireless devices. *Computer*, 37(2):104–105, February 2004.