A Context-Aware Guide for a Museum Using Bluetooth Wireless Technology[1]

Juan-Carlos Cano, David Ferrández Bell, Pietro Manzoni

Polytechnic University of Valencia, Department of Computer Engineering, Valencia, Spain.
e-mail: {jucano, pmanzoni}@disca.upv.es

Abstract: **Bluetooth is a promising wireless technology which allowed electronic devices to be instantly interconnected into short-range ad hoc networks. This paper presents a "proof of concept" experiment relative to the use of the Bluetooth wireless technology to provide network support for context-aware applications. We develop a context-aware guide for a museum using Bluetooth as a defacto radio interface. We describe the steps taken to create a Bluetooth based context-aware application. Finally we evaluate using a small test-bed the overall performance of this technology when adopted in this specific area.**

1. Introduction

Ubiquitous computing strongly depends on context-aware applications. These applications offer a flexible, adaptable service being totally conscious of the client, what surrounds him, of its environment, or anything that might be useful in order to make a better experience for the end user. The application may change or adapt its functions, information and user interface depending on the context (by inferring or sensing it) and the client of that precise moment [1].

Much progress has gone on in the human-computer interactions area to help the definition of these types of applications, but still, context-aware computing remains an unexplored area with much to be investigated and developed. Context-aware applications require mobile or wireless network technology. Many possible wireless networking technologies are available, ranging from 3rd generation wireless networks to personal area network (PANs).

We based our proposal on the use of the Bluetooth technology [2]. Bluetooth is a versatile and flexible short-range wireless network technology with low power consumption. It is designed to be small and to keep costs low to be included in practically any device.

Devices that are close-by can find their neighbors using the *inquiry* procedure. After discovering close-by devices, a node can decide to page to them and to connect to them. A dedicated protocol called SDP (Service Discovery Protocol) is then used to interchange information about all the available services at each node. For a good overview on Bluetooth, please refer to [2].

Thanks to the inquiry procedure we are able to find devices in our neighborhood. Service discovery and browsing allows the user to find required services from those devices and all the necessary configuration and connection parameters. Therefore the user has an easy and transparent way of interconnecting devices, making context-aware applications much easier. From the service provider view, these features allow identifying and finding out what type of users are the ones nearby and giving them an according service.

Our experimental applications will provide context depended information to the visitors of a museum to enhance their experience. The system will give to visitors precise information about what they are viewing at their level of knowledge in their natural language. It will also give the possibility to have a graphical user interface (GUI) adapted to their device, e.g., mobile phone, PDA, laptop. The application also aims to help the maintainers of the museum by reducing costs in guiding their visitors, in keeping track of what are the preferred pieces of art.

The application is basically built around the Bluetooth's Inquiry process and the Service Discovery Protocol (SDP). A user, while wondering around the museum, will continuously search for new point of access using Bluetooth's Inquiry. When a new point of access has been found the user device will check if it can offer any information of interest through a SDP search. If anything interesting is found the user will be notified to whether he prefers to see the new information. In that case the user device will connect to the point of access and ask for the information. However, for the information source to know what type of information the user needs, it requires the context of the user. The user's device will send to the information source the profile that was entered in the beginning configuration process.

We evaluate our proposed application in a small test-bed not only to confirm the correct behavior of the designed application but also to acquire experimental data about how well is Bluetooth suited for context aware networking.

The rest of this paper is organized as follows: Section 2. describe ongoing projects devoted to provide networking infrastructure to ubiquitous computing projects. Section 3. describes the museum system architecture and present the details of the implementation prototype. Section 4. illustrates the evaluation of the proposal. Finally Section 5. conclude the paper and depicts a few possible extension of our work.

2. On going projects

Many groups are working on the issue of context-aware or more generally ubiquitous computing.

The Oxygen project at the MIT [3] aims to enable pervasive, human-centered computing by means of a combination of specific user and system technologies. So called *dynamic, self-configuring networks* (N21s) help devices to locate each other as well as the people, services, and resources they want to reach. They provide software that adapts to changes in the environment

[1]This work was partially supported by the Spanish CICYT under Grant TIC2003-00339 and by the *Junta de Comunidades de Castilla la Mancha*, Spain, under Grant PBC-03-001.

or in user requirements (O2S) to help the system to do what it has to do whenever it wants. Oxygen network technologies connect time-varying configurations of mobile and stationary devices to form collaborative regions. N21s support multiple communication protocols for low-power point-to-point, building-wide, and campus-wide communication.

The Connected Project [4] goal is *everything always connected*. The main objective of the projects are *spontaneous networks*. Spontaneous and ad-hoc networks are two important networking concepts for realizing seamless mobile networking. They define the concept of *proxy architectures* for supporting small networked devices as a mean to support networked appliances.

The Rutgers University's *DataMan* group is working on the Infodispensers (Digital Sprinklers) [5] project. Infodispensers are data aggregation and distribution devices that cover the environment and act as information sources. They provide local information on users as they pass nearby. They make it possible for user to passively receive information of their interest without initiating an explicit query.

Researchers at the Network Computing lab at Columbia University are developing so-called *thin-clients* to reduce infrastructure costs [6]. Thin-client computing moves user data and application management from the desktop to centralized servers in professional-managed data centers. Application service providers (ASPs) reduce the total cost of ownership because of its ability to deliver easier-to-maintain computational services.

The *Disappearing Computer* is a European Union funded initiative which belongs to the Information Society Technologies (IST) research program [7]. Its mission is to evaluate how information technology can be diffused into everyday objects and settings. It also studies how this can lead to new ways of supporting and enhancing people's lives, above and beyond what is feasible today. Many projects compose this initiative. We considered especially related to our work the "Ambient Agoras" [8] project and the "Smart-Its" project [9].

3. The Museum System Architecture

The overall network architecture is based on the cooperation of an *edge* wireless network and a *core* wireless/wired network. The edge side is solely based on the Bluetooth technology. The core network is based on the integration of a fixed Ethernet local area network and a wireless IEEE 802.11b LAN also called WiFi.

The system considers three types of nodes: *museum information clients* (MICs), museum information points (MIPs), and *bridging points* (BP). A visitor provided with a Bluetooth enabled PDA is the basic example of a MIC. There is a MIP associated to one or more pieces of art or objects. Finally, static computers will be used as BPs and might be provided with an "adequate" combination of Bluetooth, Ethernet or IEEE 802.11b devices. The adequacy of the configuration depends on the physical structure of the facilities. Figure 1 shows a pictorial representation of a possible configuration.

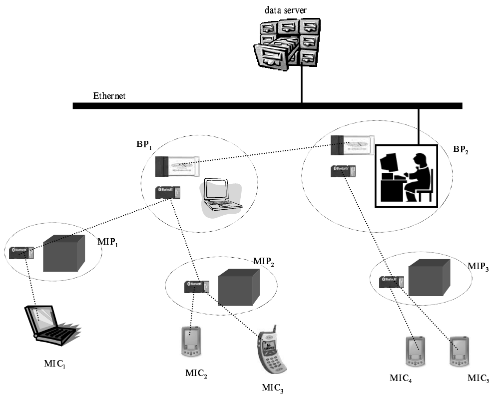

Figure 1: A pictorial representation of a possible configuration of the overall network.

The overall architecture is logically centered around the data server. The edge network integrates one or more close-by MICs. Bluetooth allows to detect devices and obtain information about them relatively quickly. From the user viewpoint, MIPs are detected without intervention, obtaining new information about what they are viewing. The application is basically built around the Bluetooth's Inquiry process and the Service Discovery Protocol (SDP). A client, while wondering around the museum, will continuously search for new devices using Bluetooth's Inquiry. When a new device is found it will then see if the peer device can offer any information of interest through a SDP search on that device, if so the user will be notified to whether he prefers to see the new information. In that case the device will connect to the information point and ask for the information. But for the information point to know what type of information the user needs, it requires the context of the user. For that the user's device will send the information point the profile that was entered in the beginning configuration process. As the information point has one Bluetooth device per object represented, the object is determined by the interface where the connection is made; therefore location data is also determined.

After the client has sent the profile, the information point has all it needs to process the petition. The information point would then add to the users profile the object the users is viewing and send it all off to the central server. There the petition would be processed, logged and returned to the information point, who would in turn send it to the client for viewing.

After viewing, the user can decide to search manually for more devices, or let the client application do it automatically. At any moment the user can change his profile if he finds that the information is too advanced or too basic, allowing futures views to be more in line with his own way of seeing things. This is the very idea of ubiquitous computing, as invisibly the system determines positioning, and without having to do nothing, the user's device gets in contact with the system for personalized information, in other words: the whole system, servers and own devices work together for the users needs.

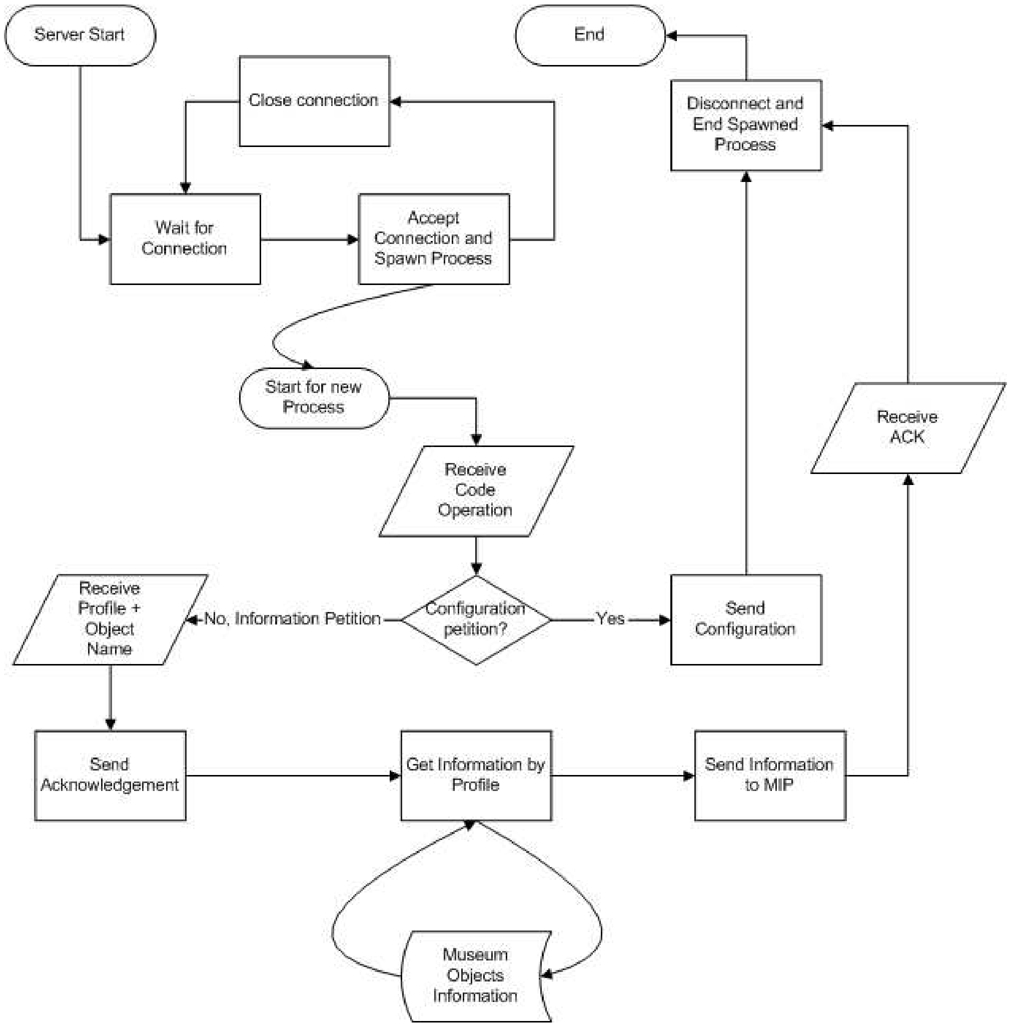

Figure 2: Museum Central Server execution Flowcharts.

3.1. The Museum Central Data Server

The operation of the museum central data server is quite straightforward: it starts and waits for a connection on the default server port. If a connection request is received on that port a new process is spawned to attend it and will receive a code operation (*codop*). Depending on the value of the *codop*, the central server will do one of two things: if the *codop* corresponds to a configuration request, the server will send to the information point the contents of the configuration file objects. This file has an object name per line. All the objects (pieces of art) names that reside in the museum and for what the service are given, will be listed in this file. Each line will correspond to information formatted for a specific devices, in a specific languages and in a given levels of detail.

The structure is replicated for each object and information is adjusted for each profile option. Once the museum information point has obtained the configuration file, the data server will disconnect, and the spawned process will end.

If the *codop* corresponds to an information request, the server will spawn another process for it to receive the profile and object name. For each profile option received the server will acknowledge it to make sure errors are controlled. Once all the profile is received, the server will log the petition for security and for statistical information. This can help the museum to know which are the preferred objects, which have had more visits and routes followed throughout the museum, helping the internal exposition and organization. After logging the petition the data server will access the directory structure taking into account each profile option. This information will be sent to the information point who in turn will resend it to its client.

A client profile consists of a language of preference, the educational level, and the device type. The profile allows information to be formatted to take into account the graphical capabilities of the client and to determine the complexity of the explanations given. On top of this, what the central server actually receives is the object name which is added to the profile by the information point. The information to be sent back to the information point is the contents which in our prototype based on a laptop are two photos, one of the actual object in view and another of the author, and three text descrip-

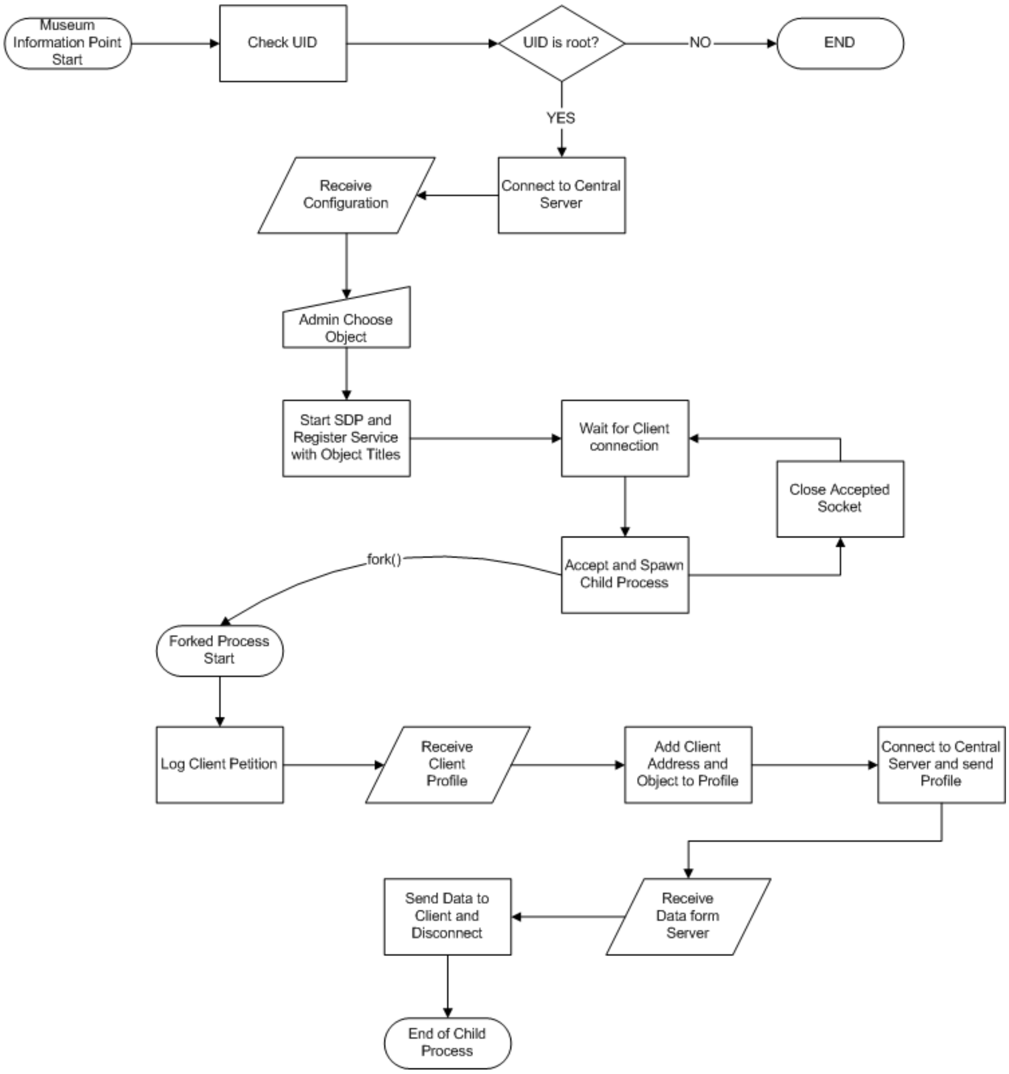

Figure 3: Museum Information Point execution Flowcharts.

tions. The text descriptions correspond to a description or interesting information of the object, another text will go over the life of the author and the third will be related to the actual position of the object in the museum, giving the client an idea of where he is and of where he can go. Once finished sending the data to the information point the central server will disconnect and the process that dealt with the communications will end.

Figure 2 shows the flowchart that resume Museum Data Server behavior.

3.2. The Museum Information Points

The information points first executes the SDP daemon to register its own services. The program will then proceed and connect to the museum server to retrieve the server's configuration file. This configuration file is presented to the information point technician who would choose an object for that information point to represent. Then the program would start the SDP daemon and register its service along with the appropriate attributes like titles in various languages and PSM values for the clients to know where to connect to. Once all the SDP registration has been done, the information point will create and bind the L2CAP layer sockets and wait for connections coming from the Bluetooth interface. When a connection comes in, the information point will spawn a child process to deal with the client's petition. The child process first mission is to log the clients Bluetooth address and date of petition. Once the logging has finished the information point will receive the clients profile. The information point would append the local object's name and clients Bluetooth address and send it off to the central Server. There it will get processed and send back the information that will be passed on to the client. Once all the data has been sent to the client it will disconnect and the child process will end. The parent process will carry on waiting for further client petitions.

Figure 3 shows the flowchart that resume Museum Information Point behavior.

3.3. Museum Information Clients

The first step for each Museum Information Clients is initialization. The user has to fill in his profile in the "Change Profile " menu, and if he prefers to change the language in the "Change Language" menu. Once all the data is filled in, the application will search for informa-

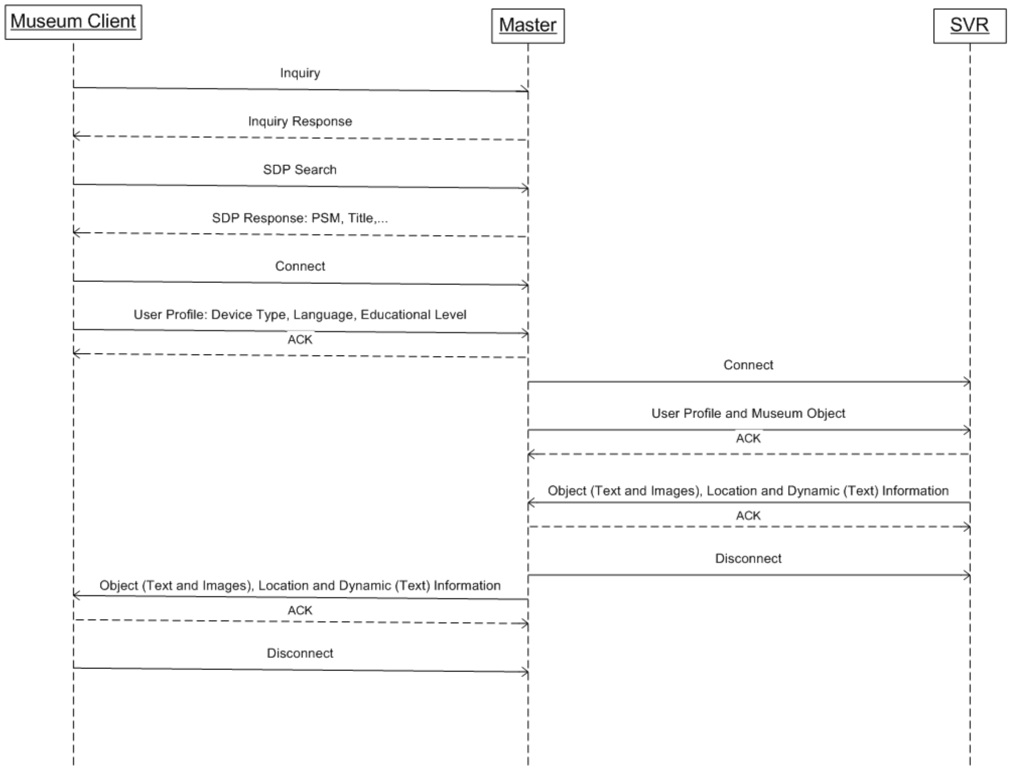

Figure 4: Basic Message passing Diagram.

tion on what he views.

To do that the user must press the "Go Get Information" button. In that moment the client application will proceed to do a Bluetooth Inquiry. If no devices with the correct Class of Device are found, it will show a message to the user to move closer to an object. If at least one device is found, it will proceed to do SDP searches on each of the devices found. If only one device was found the Client will connect to that device, if not the application will create a list of found objects with the title of the object in the users preferred language. Once the information point has been chosen, either because it was the only one or because the user had chosen it from the list, the application will connect to the device. Once connected the client will send the stored profile to the information point and receive the information. Once all the information is received the application will show it to the user. After the first retrieval of information, the entire process of inquiring, SDP searching, profile sending and information receiving is done automatically by a separate thread. To not bother the user with continuous updates, the thread will only modify the user viewed data when new servers are found or the current information is no longer available.

Figure 4 presents a diagram of how the actual message passing works between the client, the information point and the central server. This is only a basic example with one information point. Figure 5 presents a few screen capture of the final prototype related to the client set-up and MIC visualizing a data-block. The prototype application is available at http://www.grc.upv.es/software.html.

4. Performance evaluation

We evaluated our proposed application in a small test-bed. The goal of our test-bed experiment is not only to confirm the correct behavior of the designed application but also to acquire experimental data about how well is Bluetooth suited for context aware networking.

We built a test-bed where the museum data server and the museum information point run on an Intel Pentium IV 2400 Mhz standard PC based on Suse Linux 8.1 with a 3Com CREB96 Bluetooth USB Dongle card. The museum information client runs on an Intel Pentium IV 2000 Mhz TOSHIBA Satellite 1900-303 based on Suse Linux 8.1 with a 3Com CREB96 Bluetooth USB Dongle card. We used the BlueZ [10] protocol stack. BlueZ is the official Linux Bluetooth stack and it has strong support for USB dongles devices. BlueZ is part of the 2.4.20 version of the Linux kernel. The performance experiments first focussed on the Inquiry establishment delay of the Bluetooth channel. Bluetooth uses the Inquiry procedure to transparently discover nearby devices. However it not only can take a long time but also is to important from the energetic point of view because power consumption increases during the inquiry. We basically evaluate how distance affects inquiry delay. The museum information point will be waiting to be inquired by the museum information client. When the client arrives to the inquiry mode, we establish a checkpoint. Finally, to avoid consuming the inquiry timeout period of 10.24 defined in the standard, we fixed the maximum number of discovered devices equal to 1.

Using a distance of 1 m the experiments show an average inquiry delay of 2.313s ± 0.361 and a standard de-

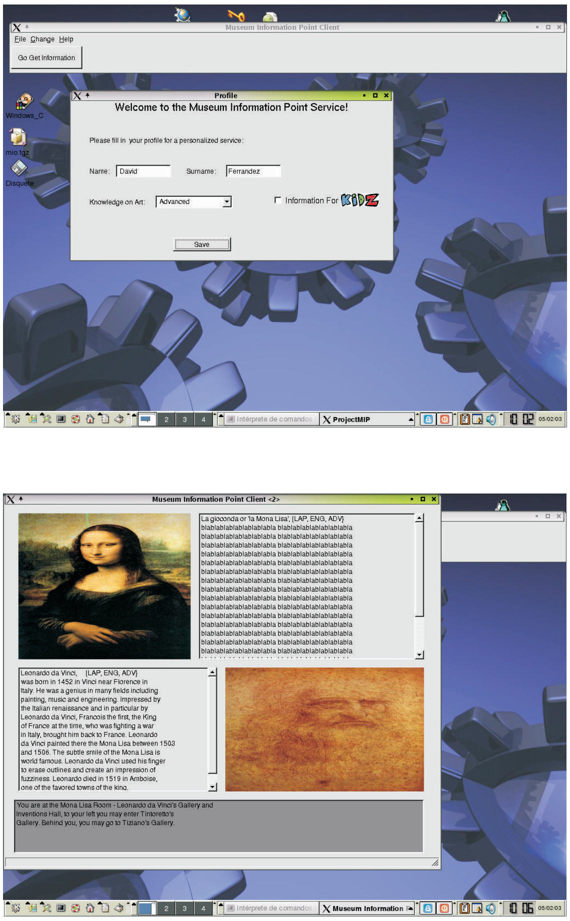

Figure 5: A few screens captures of the prototype application.

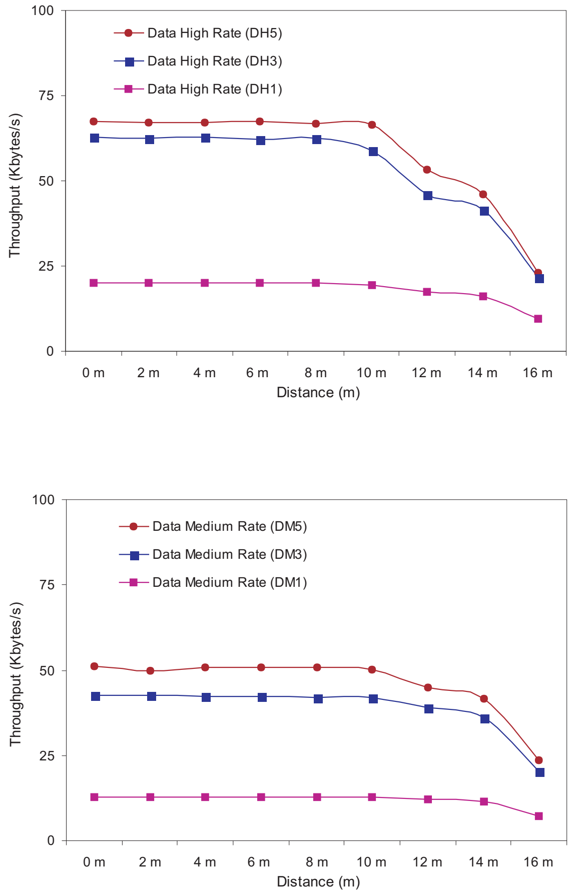

Figure 6: Obtained application throughput comparison of an ACL link using different DH and DM packets as a function of distance.

viation equal to $1,8122 \pm 0.258$ with confidence interval of 95%. As we increase the distance among nodes to 10 m we obtain an average inquiry delay of 2.364 s and a standard deviation of 2.033 with the same confidence interval. It seem that the inquiry procedure is not highly sensible to distance.

After 2.455s the museum client discovers the information point in a 50% of the tests. We also realized that only after 5.264s and 8.236s this percentage increases to 95% and 99% of the total number of tests.

We now evaluate the throughput performance. On average, our museum application data-block size averages 150 Kbytes and includes images and text. Bluetooth offers "Asynchronous ConectionLess" (ACL) links for data transmission. An ACL link is parameterized by packet size and data encoding. Each ACL link allows the use of 1, 3, and 5 slots data packet. Additionally it allows the optional use of Forward Error Correction (FEC). According to this parameters, Bluetooth offers 6 different data packets that can be classified in two main groups. Data Medium Rate packets DM packets which provide a 2/3 FEC Hamming code and Data High Rate packets DH which provide no FEC coding at all.

From the experiment we observed that Bluetooth offers a quite steady throughput below 10 m, independently of the data packet selected. Passing the 10 m limit the application can go on for further distances without a sharp performance reduction. When we select the more efficient DH data packets, we improve the observed throughput with respect to the DM packets. In the range of 10 m the DH packets improve the DM ones by 57%, 46% and 32% respectively for 1, 3, and 5 slots packets. The results confirm that although DM packets tradeoff efficiency and probability of successful transmission, the DH packets can be a candidate for more efficient transmissions. Moreover, after 10m, as distance increases, the more efficient longer packets ($DH5$ and $DH3$) suffer from a higher degradation.

5. Conclusions and further development

This paper evaluated Bluetooth as a contender network support system for context-aware application. We develop a context-aware guide for a museum which was designed to give to visitors precise information about what they are viewing at their level of knowledge in their natural language of preference and giving a possibility for the user to have a GUI adapted to their device, enhancing their experience. The application also aims to help the maintainers of the museum, reducing costs in guiding their visitors, and keeping track of what are the preferred pieces of art. We evaluated our application in a small test-bed focusing on the inquiry delay that rounds 5s and which is not to sensitive with distance. As far as throughput Bluetooth offers a quite steady behaviour below 10 m. We did not take into consideration many real implementation issues that would make the application more feasible. Many of those issues would be better solved with more high level solutions. We outline the following guidelines that we consider the most relevant in the immediate future.

- **Security.** Prudent users might want to have a certificate to testify who is actually providing him with data. A local space like a museum might be the perfect place for intruders to get access to others devices to steal information.
- **Quality of service** As the graphical possibility of the mobile devices increases the issue of providing multimedia information raises. Audio and music hasn't been included due to the lack of SCO support over USB in the current BlueZ implementation. In the future, to provide multimedia content with a sufficient level of quality, QoS techniques must be introduced.
- **Easing and generalizing the interface** Using HTTP over the L2CAP link the client could simply use a modified web browser to access a web server to retrieve the information. The dynamic pages on the server would do most of the profile formatting and adaptations.
- **Caching and persistence of information** Much network traffic would be reduced if caching techniques were introduced. This would reduce the number of transfers between the information point and the central server. It would also create a consistency information problem that can be resolved, for example, by notifying the information point.
- **More profile parameters** This would allow a much higher level of quality if the information is adapted much more to the user, also taking into account the history of a user, where he goes and seeing a pattern in his preferences could also infer a better adaptation of the information.
- **Instant messaging and employees' assistance** Give the user an opportunity to contact an employer of the museum in case of problems or any other request of information, thus allowing the user to have the information office in his hand.

REFERENCES

[1] M. Weiser, "Some computer science problems in ubiquitous computing," *Communications of the ACM*, July 1993.

[2] B. Chatschik, "An overview of the bluetooth wireless technology," *IEEE Communications Magazine*, vol. 39, no. 12, pp. 86–94, December 2001.

[3] "MIT project Oxygen," Massachusetts Institute of Technology, Cambridge, MA, USA. Web page: http://oxygen.lcs.mit.edu/.

[4] "The connected project," Swedish Institute of Computer Science (SICS), Uppsala University. Web page: http://www.sics.se/cna/connected/.

[5] "Infodispensers (digital sprinklers)," Dataman Mobile COmputing Laboratory at Division of Computer and Information Sciences, Rutgers University, NJ, USA. Web page: http://www.cs.rutgers.edu/

[6] "Ubiquitous thin-client computing," Network Computing Laboratory (NCL), Columbia University, NY, USA. Web page: http://www.cs.rutgers.edu/

[7] "The disappearing computer," European Community - Information Society Technologies research program. Web page: http://www.disappearing-computer.net/.

[8] "Ambient agoras: Dynamic information controls in a hybrid world," The Integrated Publication and Information Systems Institute (IPSI). Web page: http://www.ambient-agoras.org/.

[9] "Smart-its: Interconnected embedded technology for smart artefacts with collective awareness," Lancaster University, ETH Zurich, University of Karlsruhe, Web page: http://www.smart-its.org/.

[10] M. Krasnyansky, "BlueZ: Official linux bluetooth protocol stack," http://bluez.sourceforge.net/, 2003.