An autoconfiguration method for IEEE 802.11 based MANETs using Bluetooth

An autoconfiguration method for IEEE 802.11 based MANETs using Bluetooth

José Cano Reyes, Eduardo Burgoa, Carlos T. Calafate, Juan-Carlos Cano, Pietro Manzoni

Abstract— **The availability of both Wi-Fi and Bluetooth technologies on currently available devices offers the possibility to combine both in order to make the most out of their capabilities. In this paper we propose using Bluetooth technology to solve the configuration problem of the terminals conforming an IEEE 802.11-based ad-hoc network. The main objective of mobile ad-hoc networks (MANETs) is to extend the connectivity range of nodes through packet forwarding, thereby avoiding the use of a fixed infrastructure. However, since configuration of nodes is a complex issue, we provide a fast and reliable solution to auto-configure MANET terminals. Our solution is adequate for the quick setup and deployment of rescue, military or any other sort of organized team. We present the architecture of the proposed system by means of a simple example, along with the applications developed for both client and server. We conclude with some experiments to assess the performance of the proposed architecture.**

Keywords— **MANET, Wi-Fi, Bluetooth, ad-hoc networks, autoconfiguration, service discovery.**

I. Introduction

MOBILE ad-hoc networks, usually referred to as MANETs, are autonomous systems composed of independent mobile terminals which communicate among themselves using any sort of wireless technology. The terminals are free to move about and organize themselves to conform an IP-based network. Each node operates not only as an end-system, but also cooperates on routing and packet forwarding tasks. If we see it from the IP layer perspective, a MANET is a Layer-3 multi-hop network implemented over a collection of links. Therefore, each node pertaining to the network is, in principle, acting as a Layer-3 router in order to provide connectivity to other nodes. Each node maintains host routes to the rest of nodes within the network, in addition to network routes to destinations outside the MANET, if any.

One of the main advantages of MANETs is that they do not require any fixed infrastructure or a centralized administration. Therefore, they are an attractive networking option for connecting mobile devices quickly and spontaneously.

Concerning their applicability, we believe that MANET technology will be decisive to meet Weiser's paradigm of "anywhere and at any time" connectivity [1]. Ad-hoc networking can be applied anywhere where there is little or no communication infrastructure, or the existing infrastructure is expensive or inconvenient to use. Ad-hoc networking also allows devices to maintain dynamic connections to the network, as well as easily adding and removing devices to and from the network. The set of applications for MANETs is diverse, ranging from large-scale, mobile, highly dynamic networks, to small, static networks that are constrained by power sources. As an example, we can imagine a group of persons with their PDAs in a business meeting where no network support is available; in that case they can easily connect their devices by forming an ad-hoc network. This is just one of the many examples where these networks may be used.

MANETs, due to their flexibility and adaptation capabilities, are also characterized by being complex to deploy and maintain in an automatic manner. One of the main problems when deploying an ad-hoc network is the configuration of nodes. IETF's Ad hoc Network Autoconfiguration (autoconf) Working Group [2] is currently seeking a solution to configure MANET nodes in a totally distributed manner. Examples of solutions that can be found in the literature are PACMAN [3] and the OLSR extensions proposed by Clausen and Bacceli [4]. Their scope, though, only focuses on the assignment of IP addresses to network interfaces. Currently, MANETs rely mostly on the IEEE 802.11 technology [5], also known as Wi-Fi, which requires additional setup of the different MAC layer parameters before any IP address assignment can take place. Moreover, to activate multi-hop relaying, a common routing protocol must be started by all the nodes, thereby achieving successful participation in the MANET.

In this paper we propose a solution that makes the MANET initialization process fully automatic by relying on Bluetooth technology. Bluetooth [6] has been standardized by the IEEE 802.15 working group [7], and is currently one of the most deployed and used wireless technologies. An important property of Bluetooth technology is that it supports the concept of services, device/service discovery, as well as secure connection and authentication. This, alongside with its low battery consumption characteristics and its reduced radio range, makes it ideal for exchanging private and confidential information in a very simple and straightforward manner.

We propose using the aforementioned characteristics of Bluetooth technology to simplify the tasks of users that wish to integrate a certain IEEE 802.11-based MANET by offering a Bluetooth service specifically designed for that purpose. In our work we denote this service as *MANET_Autoconf*.

Department of Computer Engineering. Polytechnic University of Valencia, Spain. E-mail: jocare@doctor.upv.es, edburser@teleco.upv.es,{calafate,jucano,pmanzoni}@disca.upv.es

This work has been partially funded by the *Generalitat Valenciana*, Spain under grant GV05/245 and by FEDER and the *Ministerio de Educacin y Ciencia*, Spain, under Grant TIN2005-07705-C02-01.

The rest of this paper is organized as follows: in section 2 we will summarize the system architecture. Section 3 refers to some design issues and, in section 4, we discuss some initial experimental results. Finally, section 5 presents our conclusions along with references to future work.

II. System architecture

The main objective of our auto-configuration system is to allow non-expert users to start a MANET in a simple and effective manner. Traditionally, the solution to this problem requires all users to adjust the IP configuration of their wireless interface to a same subnet, avoiding addresses that have been picked up by other users; examples of solutions to this problem are being studied by the Ad-Hoc Network Autoconfiguration Working Group. Besides IP settings, all users must start the daemon of a same routing protocol and, if IEEE 802.11 technology is used, they must also adjust their Wi-Fi configuration (e.g., the type of encryption, the channel used, etc.) according to a consensus. Such tasks are tedious and time consuming since they must be repeated at every single terminal participating in the MANET. Also, the fact that this task requires a certain degree of expertise impedes that MANETs experience a generalized acceptation and use, reason why we have looked for a technique to solve the problem more efficiently.

The solution we propose is using the Bluetooth wireless interface, nowadays available on almost every computing device, to automate the process of MANET integration. With this purpose we have one device acting as a server, which will be responsible for registering the *MANET_Autoconf* Bluetooth service, and that makes sure that all devices are configured with different IP addresses but with the same routing protocol/Wi-Fi parameters, thereby enabling the whole process. The rest of Bluetooth devices will function as clients, searching for that service so as to retrieve the MANET configuration parameters, and automatically applying that configuration afterwards. Therefore, we assume that all terminals have both a Bluetooth and an IEEE 802.11 interface. The Bluetooth interface is merely used to obtain the configuration parameters required to join the MANET, while the Wi-Fi interface will allow the station to participate actively in the MANET for different purposes, such as sharing information with other nodes or participating in any sort of peer-to-peer communication.

Figure 1 depicts the proposed system architecture; it is possible to notice, on one hand, three different kinds of elements: Bluetooth clients waiting to be configured with the suitable parameters, the configuration server waiting for incoming Bluetooth connections, and actual MANET nodes. On the other hand, it is possible to observe the coexistence of two networks operating with different wireless technologies: the Bluetooth network (composed by nodes that are linked by discontinuous lines) on the left part of the picture, and the Wi-fi network (composed by nodes that are linked by continuous lines) on the right side of the picture.

Every station that wants to join the MANET must first connect to the configuration server via Bluetooth, possibly competing with other stations also waiting to be configured. To do that it must perform an *inquiry* action to discover nearby Bluetooth devices. Afterwards it must check the different devices found in sequence until it finds the one offering the desired service (*MANET_Autoconf*); this is done by making use of the Service Discovery Protocol (SDP), part of Bluetooth's framework. SDP provides the client with a mechanism for the discovery of services available on a certain device, along with the service attributes. SDP is designed to be decoupled from the service itself, and so it does not provide any mechanism or protocol to use these services. This way, once a device offering the desired service is discovered, the client must proceed to establish a separate L2CAP or RFCOMM connection with the server to carry out the designated task, in our case download of the desired configuration parameters. The process of connection establishment is done using the appropriate L2CAP or RFCOMMM port, as advertised through SDP.

The configuration server must make sure it is visible by other devices, so that any device discovery attempts are successful. Besides, it must register the *MANET_Autoconf* service and advertise it so that clients can proceed to discover the service afterwards. The server will also be listening to the appropriate L2CAP or RFCOMM port for incoming connections. Taking into consideration the limitations of Bluetooth technology on the number of stations in a same piconet, the server can attend up to 7 concurrent clients. If more are waiting to be configured, they must wait until one of the clients completes the configuration process and releases the resources occupied. As we will show in section IV, the configuration time for each node is typically low, and so this should not be a design limitation.

When a client successfully establishes a connection with the server and requests the configuration parameters, the server must generate a customized XML file with all the required information and send it back to the client. This XML file will contain all the necessary information for that station to successfully join the MANET. The configuration parameters include the station's IP address and mask, the routing protocol used (e.g. DSR [8], AODV [9], OLSR [10]) and all the information required to configure the Wi-Fi interface (SSID, channel, etc.). Figure 1 shows an example of the XML configuration file sent by the server to a client.

By allowing the server to determine the IP address of each client we are able to solve the problem of IP address assignment referred earlier in a centralized way. It is important to point out that the server must assign IPs in increasing order and must maintain, at any moment, a registry of the connected clients (for example, in another XML file).

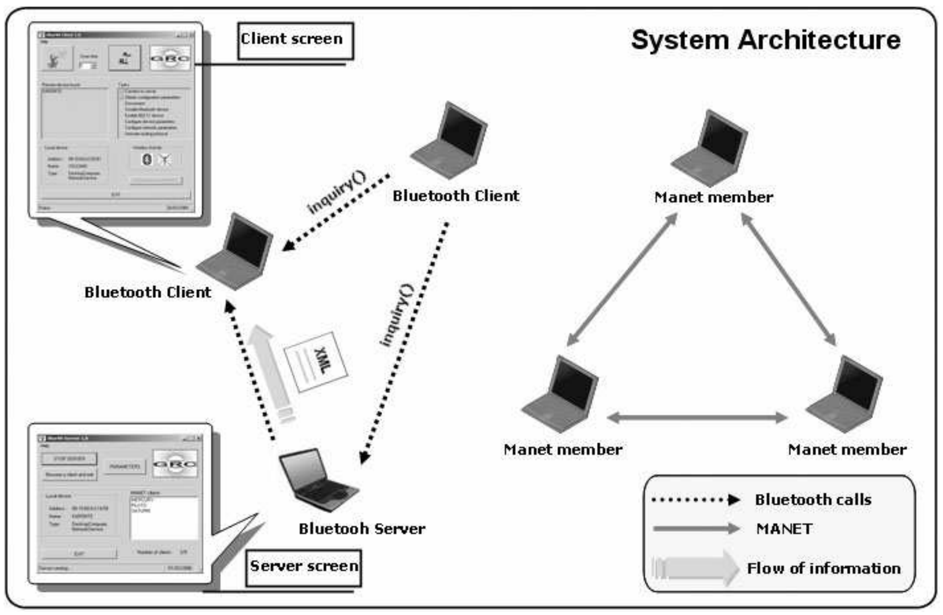
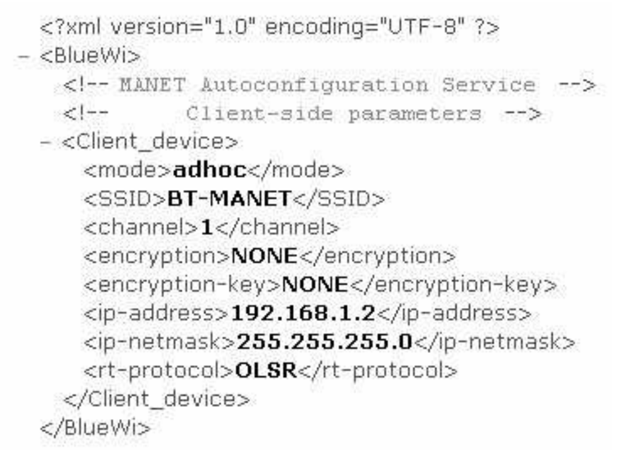


Fig. 1. System architecture.



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <BlueWi>
    <!-- MANET Autoconfiguration Service  -->
    <!--      Client-side parameters  -->
  - <Client_device>
      <mode>adhoc</mode>
      <SSID>BT-MANET</SSID>
      <channel>1</channel>
      <encryption>NONE</encryption>
      <encryption-key>NONE</encryption-key>
      <ip-address>192.168.1.2</ip-address>
      <ip-netmask>255.255.255.0</ip-netmask>
      <rt-protocol>OLSR</rt-protocol>
    </Client_device>
  </BlueWi>
```

Fig. 2. XML configuration file for a client.

After a client station receives its configuration data it must switch to Wi-Fi mode, which means that the Bluetooth interface is disconnected and the Wi-Fi interface is activated. With this technique we reduce to a minimum the interference between Bluetooth and Wi-Fi technologies, a problem that affects both annexes *b* and *g* of the IEEE 802.11 standard [11].

When the Wi-Fi card is enabled the client station can then proceed to apply the new configuration settings. By doing so it will automatically join the MANET, being able to communicate with other mobile stations that have configured themselves previously. Figure 3 describes this process, showing the sequence of actions taken according to all the steps referred previously.

The server station can optionally join the MANET it offers support for. In case it decides to do so, it can join the MANET right from the beginning (possibly suffering from Bluetooth/Wi-Fi interference), or only after all the expected stations are configured. In the latter case it will perform a sequence of actions similar to that performed by clients, disconnecting the Bluetooth interface and enabling Wi-Fi. Evidently, when this occurs, no more users can be automatically configured since the Bluetooth server no longer exists. For this reason the server always maintains data about the number of configured devices, as well as the maximum number of devices expected.

III. Design issues

We have implemented our application for both Windows and Linux operating systems, and in the future we plan to develop a version of it for Pocket PC also.

Concerning the Windows-based application, it uses, among other elements, the Microsoft Bluetooth stack [12]. We have chosen this Bluetooth stack because it is the one included by default on both standard and mobile Windows editions, and also because there are useful libraries that we can use freely. To develop the application we have used the .NET platform combined with the Visual Basic programming language, achieving an object oriented solution that is powerful and yet simple. Actual access to the Bluetooth protocol stack is done using the external Bluetooth library offered by OpenNETCF, which was developed by Peter Foot [13].

Concerning the Linux-based solution, we have designed both a console and a graphical version. That way it can operate both as a service offered by the system and as an interactive application. To control the Bluetooth interface we have used the BlueZ API [14], which offers a high degree of functionality. The console solution was written in C, while the graphical application was designed in C++ using QT libraries [15].

We have designed a client and a server application.

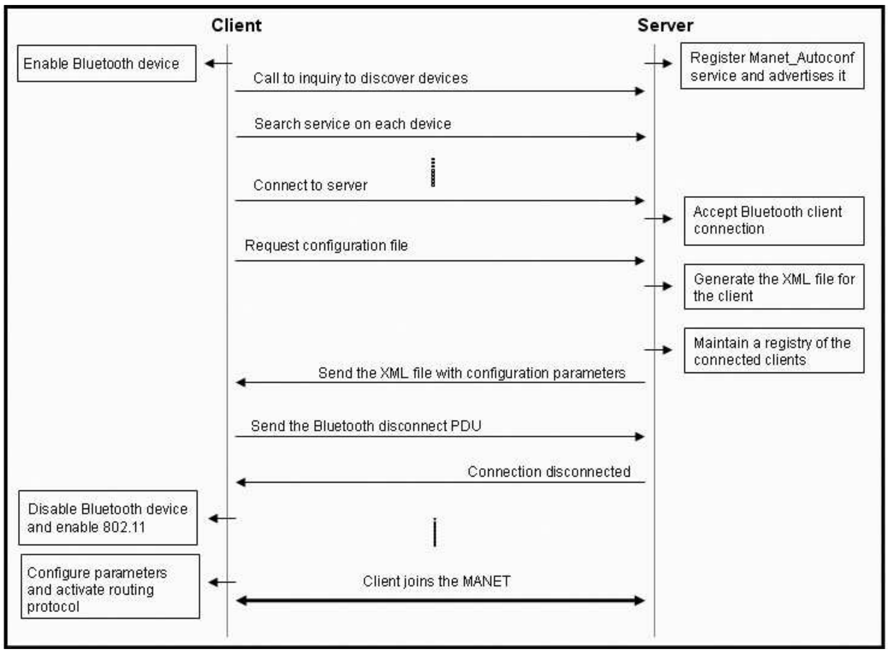
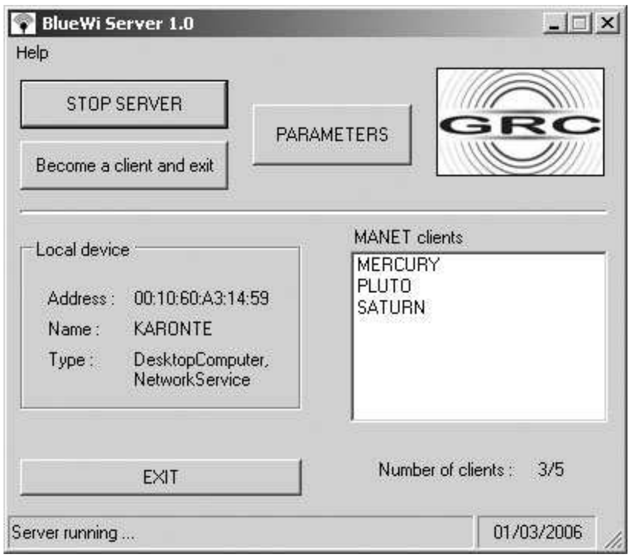

Client
Server
Enable Bluetooth device
Register Manet_Autoconf service and advertises it
Call to inquiry to discover devices
Search service on each device
Connect to server
Accept Bluetooth client connection
Request configuration file
Generate the XML file for the client
Maintain a registry of the connected clients
Send the XML file with configuration parameters
Send the Bluetooth disconnect PDU
Connection disconnected
Disable Bluetooth device and enable 802.11
Configure parameters and activate routing protocol
Client joins the MANET

Fig. 3. Flow diagram between client and server.

Both of them offer information about the local device, such as the MAC address, the public name and the type of device (laptop, PDA, etc.).

Figure 4 shows the graphical server interface, depicting a situation where there are three clients who have connected to the server and joined the MANET successfully. This screenshot represents the environment depicted in figure 1. As it can be seen, we are able to start and stop the server just by clicking the corresponding button. If the server is stopped the Bluetooh interface may be active but the *MANET_Autoconf* service is not registered, avoiding any connection attempts. Moreover, we can set the different configuration parameters, which include the desired IP address and subnet, the maximum number of clients allowed to integrate the MANET, the routing protocol used and the different Wi-Fi parameters. The application also allows a manager to control the number of devices already configured, being these devices identified either by the MAC address of by the device name.

The clients, by using the interface shown in figure 5, can initiate the discovery of Bluetooth devices within range (which, for standard Bluetooth technology, is of about 10 meters); this is done by means of a "find" button, represented by a satellite dish. The discovery time is adjustable by the user as multiples of 1.28 seconds; this value is related to the time a device takes to scan sets of Bluetooth frequencies. The list of devices shown at the interface is restricted to those offering the *MANET_Autoconf* service to simplify the user's tasks. After a successful discovery action the client can proceed to connect to the selected device to retrieve the configuration parameters. A list of all the required tasks is also made visually available by the application; for each successfully completed task, the corresponding element of the list will be checked with a green icon. That way the client can be aware of the current status and possibly identify errors/problems during the process.



Local device
Address : 00:10:60:A3:14:59
Name : KARONTE
Type : DesktopComputer, NetworkService

MANET clients
MERCURY
PLUTO
SATURN

Number of clients : 3/5

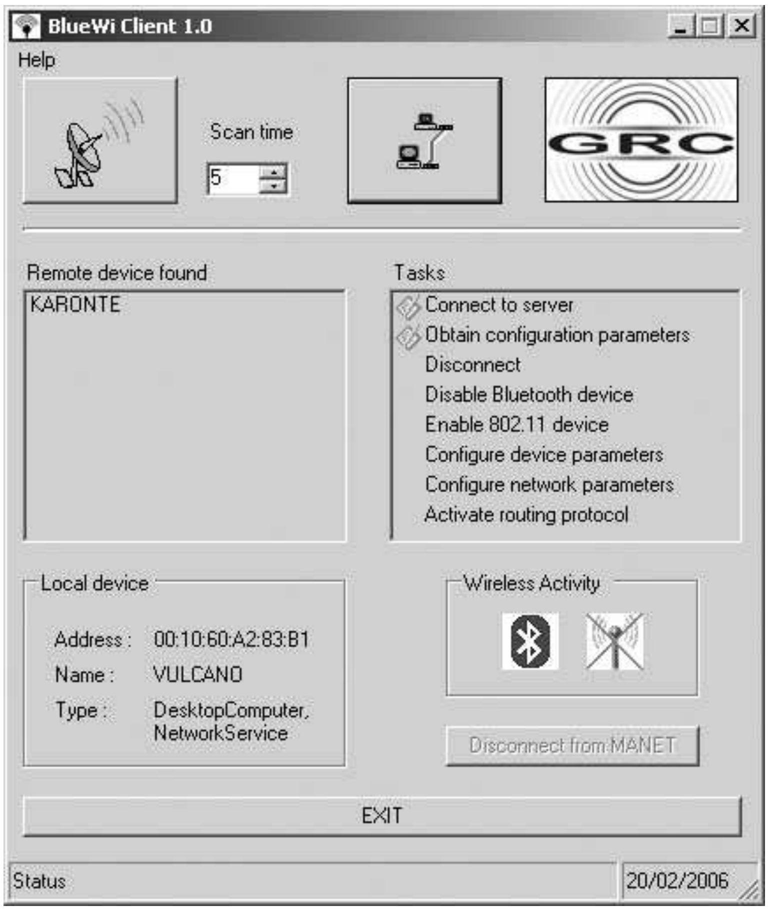
Server running ...
01/03/2006

Fig. 4. Server application capture.

Below the task list we have two icons that indicate, at any moment, which wireless interface is active. Finally, two buttons are also included that allow the client to disconnect from the MANET (restarting the configuration process) and to exit the application.

IV. Experimental results

In order to assess the impact of some of the parameters referred earlier, we have conducted a set of experiments to test the performance of our application under different conditions. The main purpose was to measure the times relative to inquiry, service discovery, and download of configuration data.


BlueWi Client 1.0

Help

Scan time 5

Remote device found

KARONTE

Tasks

Connect to server
Obtain configuration parameters
Disconnect
Disable Bluetooth device
Enable 802.11 device
Configure device parameters
Configure network parameters
Activate routing protocol

Local device

Address :	00:10:60:A2:83:B1
Name :	VULCANO
Type :	DesktopComputer, NetworkService

Wireless Activity

Disconnect from MANET

EXIT

Status 20/02/2006

Fig. 5. Client application capture.

These measurements are very important since they will offer an estimate of the time a node takes to get configured and connected to the MANET.

Concerning the inquiry time, it can be adjusted at user's will as multiples of 1.28 seconds as referred before, allowing to achieve a trade-off between latency and the probability of discovering a device. We measure the duration of that procedure through samples of 100 independent test runs. The results show that the inquiry time takes about 6.42 seconds on average, which means that the operating system introduces an overhead of about 20 ms.

Figure 6 shows the results of an experiment where we measure the time consumed in the inquiry process plus service discovery and, depending on whether the service is found or not, the time for the download of the configuration file (denoted as *With Service* in the figure). As it can be seen, distance has a great impact on performance; for instance, service discovery time increases from about 0.5 seconds to 1 second as distance increases. We can also observe that the values for successful service completion are always higher than those without service, as expected. This value is kept at about 0.3 seconds independently of distance for up to 10 meters (the theoretical limit). Afterwards service times start increasing, achieving 0.6 seconds for a distance of 12 meters.

Based on these results, we now offer an estimate of the time required for successful service discovery

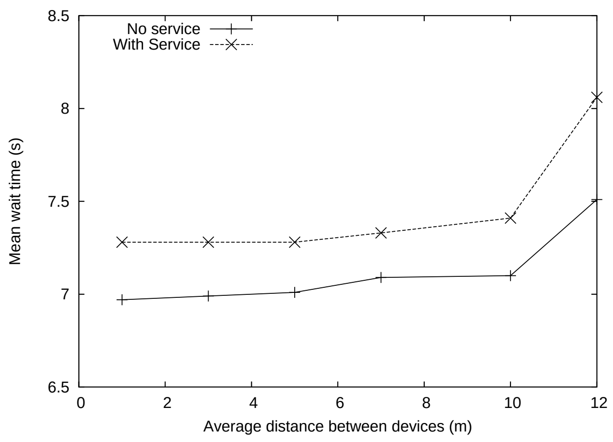


Fig. 6. User wait time for inquiry plus service discovery attempt while varying distance between Bluetooth devices.

and download of configuration data as the number of Bluetooth devices in the surrounding environment increases. These results are presented in figure 7 for an average distance between Bluetooth devices of 1, 5, 10 and 12 meters (inquiry time not included). As it can be seen, if the only nearby Bluetooth device is the one offering the *MANET_Autoconf* service, then the process completes very quickly - typically in less than 2 seconds. However, if other Bluetooth equipped nodes are also detected, then the client

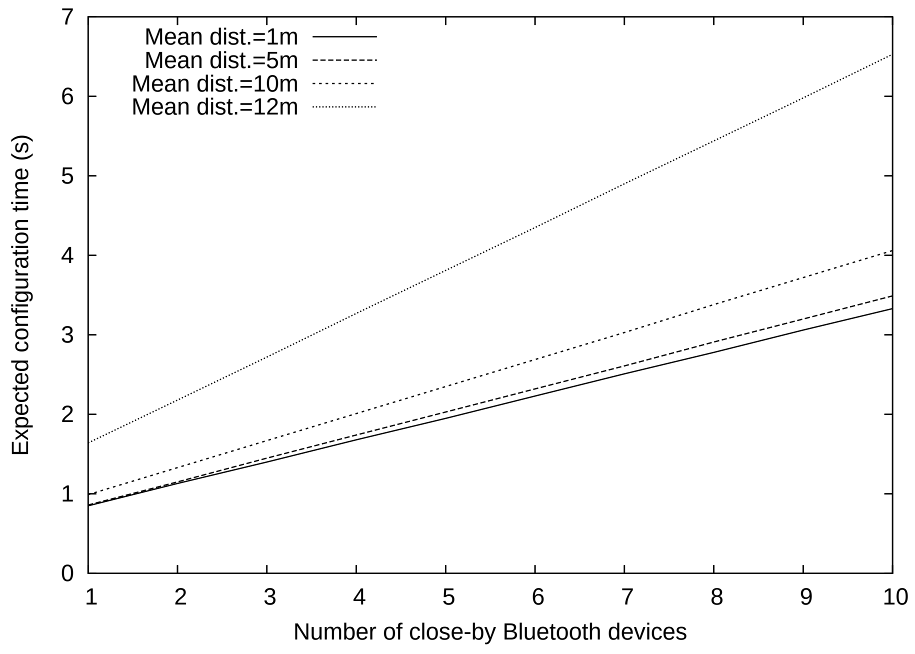


Fig. 7. Estimated configuration time when varying the number of close-by Bluetooth devices for different distance values.

must scan them in sequence until the desired service is found and the download of configuration data takes place. In such case, the expected configuration time will increase linearly with the number of nodes, possibly reaching relatively high values. For this reason, we propose that all devices attempting to configure themselves through this service change their Bluetooth settings so as to make themselves invisible to other nodes in terms of *inquiry*, thereby reducing as much as possible the mean node configuration time.

V. Conclusions

The proliferation of wireless computing devices is due to their low cost and high flexibility, reason why these count with an increasing number of supporters worldwide. However, pervasive computing hasn't yet built upon the fabric of our everyday lives since the use of these new technologies is still rather complex for unexperienced users.

In this paper we propose an effective solution to stimulate the use of MANETs by relying on Bluetooth technology. The proposed architecture makes use of Bluetooth services, device/service discovery, etc., to make the process of node configuration fully automatic, allowing clients to join a MANET in a simple and intuitive manner by using the application we have developed.

We evaluated the proposed architecture in terms of expected configuration time and its dependency with distance, showing that users are expected to complete the entire process of joining a MANET in only a few seconds.

As future work we plan to develop a version of our client application for several other devices and operating systems.

References

[1] Mark Weiser, "Some computer science issues in ubiquitous computing," *Communications of the ACM*, vol. 36, no. 7, pp. 75–84, July 1993.

[2] IETF, "Ad hoc network autoconfiguration (autoconf) charter," http://www.ietf.org/html.charters/autoconf-charter.html.

[3] K. Weniger, "Passive Duplicate Address Detection in Mobile Ad hoc Networks," in *Proceedings of IEEE WCNC*, New Orleans, March 2003.

[4] Thomas Heide Clausen and Emmanuel Baccelli, "A Simple Address Autoconfiguration Mechanism for OLSR," in *IEEE International Symposium on Circuits and Systems" (ISCAS05)*, Madison, Wisconsin USA, June 2005.

[5] IEEE/IEC Std 802.11, *Wireless LAN Medium Access Control(MAC) and Physical Layer (PHY) specifications*, The Institute of Electrical and Electronics Engineers, Inc., August 1999.

[6] Promoter Members of Bluetooth SIG, *Specification of the Bluetooth System - Core. Version 1.1*, Bluetooth SIG, Inc., February 2001.

[7] "IEEE 802.15 Working Group for WPAN," More information at http://www.ieee802.org/15/.

[8] David B. Johnson, David A. Maltz, and Yih-Chun Hu, "The dynamic source routing protocol," Internet Draft, MANET Working Group, draft-ietf-manet-dsr-10.txt, July 2004, Work in progress.

[9] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das, "Ad hoc on-demand distance vector (AODV) routing," Request for Comments 3561, MANET Working Group, http://www.ietf.org/rfc/rfc3561.txt, July 2003, Work in progress.

[10] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," Request for Comments 3626, MANET Working Group, http://www.ietf.org/rfc/rfc3626.txt, October 2003, Work in progress.

[11] Ratish J. Punnoose, Richard S. Tseng, and Daniel D. Stancil, "Experimental Results for Interference between Bluetooth and IEEE 802.11b DSSS Systems," in *Proceedings of IEEE Conf. On Vehicular Technology*, Atlantic City, NJ, USA, October 2001.

[12] "Microsoft Bluetooth Protocol Stack," Microsoft support for Bluetooth on Windows XP and Windows CE: http://msdn.microsoft.com/library/.

[13] P. Foot, *OpenNETCF Bluetooth Library*, Available at http://www.opennetcf.org/, 2002.

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

[15] Troll Tech, "Qt - Cross-Platform C++ Development," http://www.trolltech.com/products/qt.