The Device Discovery in Bluetooth Scatternet Formation Algorithms

by

Ahmed Jedda

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
for
Master of Science
in
Computer Science

Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering
University Of Ottawa

The undersigned hereby recommend to
The faculty of Graduate Studies and Research
acceptance of this thesis

The Device Discovery in Bluetooth Scatternet Formation Algorithms

Submitted by
Ahmed Jedda

In partial fulfillment of
the requirements for the degree of
Master of Science
in
Computer Science

_____

Nejib Zaguia, Ph.D.
(Thesis Supervisor)

_____

Guy-Vincent Jourdan, Ph.D.
(Thesis Co-supervisor)

Contents

List of Figures

List of Tables

Abstract

The Bluetooth Scatternet Formation (BSF) problem can be defined as the problem of forming wireless networks of Bluetooth devices in an efficient manner. A number of restrictions imposed by the Bluetooth specifications make the BSF problem challenging and unique. Many interesting solution algorithms have been proposed in the literature to solve this problem. In this thesis, we investigate the BSF problem. We concentrate on problems introduced by the procedures of device discovery of the Bluetooth specifications and on the different solutions used by BSF algorithms to deal with these problems. We study also in this thesis problems introduced by the specifications of link establishment in Bluetooth due to their close interaction with the device discovery specifications.

We survey and categorize the different device discovery techniques used by BSF algorithms. This categorization is then used as a basis to identify the different theoretical computational models used to study BSF algorithms. We argue, in this thesis, that the currently available models for Bluetooth wireless networks do not model adequately, in most cases, the complexities of the Bluetooth specifications and we show that these models were oversimplified in many cases. A general computational model will be useful as a starting point to design BSF algorithms and to compare the different and numerous BSF algorithms – especially in term of the execution time efficiency. In this thesis, we provide a set of suggestions that will help in the creation of such model.

We survey a number of studies that examined in more depth the specifications of device discovery in Bluetooth. We survey also other studies that attempted to simplify the Bluetooth network model, either by suggesting modifications on the Bluetooth specifications or by the use of communication technologies other than Bluetooth. Finally, we present some experiments accompanied with analyses to show the complexities of the Bluetooth specifications and their sensitivity to minor changes (whether in the specifications or in their implementation).

Acknowledgment

I would like to thank my supervisor, Dr. Nejib Zaguia for all his support and the motivation he gave me during the time I have been working on the thesis. His experience and his thought-provoking questions have helped me immensely and have aided me refine my understanding of the problem. I would like also to thank my co-supervisor, Dr. Guy-Vincent Jourdan for all his help and motivation. I greatly appreciate the effort he made to get into the very minor details to understand my ideas. The discussion sessions with both Dr. Zaguia and Dr. Jourdan were more than helpful. Their constant drive for perfection has helped me greatly throughout this work. I would like also to thank all my other professors in University of Ottawa and Carleton University.

I would like to thank Dr. Qihe Wang and Dr. Dharma P. Agrawal for making their UCBT (University of Cincinnati BlueTooth) simulator freely available. Special thanks to Dr. Wang for answering some of our questions.

I would like to thank my family; my father Hassine, my mother Neziha, my sisters Jihen, Kawther and Sara. I could always count on their support during the difficult times in life. Finally, I would like to thank all my friends, wherever they are.

Chapter 1

Introduction

1.1. Background

The technological advance in radio communication introduced to us what is called the wireless ad-hoc networks. These are self-organizing networks that are formed from wireless devices communicating with each other without the need of any infrastructure. In these networks, each device (or node) works as a terminal and a router at the same time. The nodes are allowed to join or leave the network at any time. Nodes may also move from one geographical position to another at any time. However, ad-hoc networks that support mobility are called mobile ad-hoc networks (MANET).

An ad-hoc network, whether mobile or fixed, is suitable for many scenarios and applications. For instance, such networks can be used for surveillance purposes, disaster relief, conferences or battlefield environments [1] [2]. These networks can also allow mobile robots to communicate with each other in order to perform complex tasks.

Different wireless radio technologies were proposed to enable wireless ad hoc networks; one of them is Bluetooth [3] [4]. The technology is promising in many ways. First, Bluetooth devices are efficient with respect to power consumption. Second, Bluetooth devices use a communication mechanism that is noise-resistant and robust against interference with neighboring Bluetooth devices. Third, the cost of Bluetooth radio devices is relatively low. Last but not least, and more importantly, Bluetooth offers ad-hoc networking capabilities. These main advantages of Bluetooth, as well as its deployment in a wide range of today's laptops, mobile phones and other devices, make this technology well suited for wireless ad-hoc networks.

Bluetooth was introduced by a Special Interest Group formed by a number of IT companies. Among these companies, we find Ericsson, Nokia and IBM, not mentioning others. Originally, Bluetooth was a combination of different solutions for some issues found in wireless networks. These solutions were later assembled together and standardized under the name "Bluetooth", which is borrowed from the name of the 10^{th} century, second king of Denmark, King Harald Bluetooth, who was famous for uniting dissonant Scandinavian tribes into a single kingdom [5].

1.2. Problem Statement

But while the Bluetooth technology addressed and solved a number of problems in wireless ad-hoc networks, it also raised many others, some of which are still not solved. Among these problems, we find those related to the Bluetooth *piconets* and *scatternets*. According to the specifications of Bluetooth [3] [4], if two nodes want to communicate with each other, they must belong to the same piconet or scatternet. A piconet is a star network with one master and up to seven slaves. Piconets are allowed to have up to 255 slaves, however, at most seven of them can be active at one time, all the others would be parked (i.e. not active). Any packet sent between two nodes that belong to the same piconet must pass through the master node. Therefore, the master node controls the packet flow in the piconet. To efficiently do this task, the master must deploy an *intra-piconet scheduling algorithm*.

A scatternet is a combination of multiple interconnected piconets. To interconnect two piconets, we use *bridge* nodes, which are nodes that have multiple roles in different piconets. A bridge node must divide its time between the piconets it belongs to using an *inter-piconet scheduling algorithm*. According to the specifications, a node cannot be master of more than one piconet, while it can be slave in an unlimited number of piconets.

The problem of assigning a role to each node in the scatternet (e.g. master or slave, bridge or non-bridge) – with considering some performance criteria - is called the *Bluetooth Scatternet Formation (BSF).* This problem, and the scheduling-related

problems as well (i.e. *inter-piconet* and *intra-piconet scheduling algorithms*), are defined in the Bluetooth specifications. Nevertheless, no explicit solution to these problems is given in the specifications. Instead, these problems were left open to the research community. Many algorithm solutions for these problems were proposed in the literature. A comprehensive survey on Bluetooth Scatternet Formation algorithms can be found in [6]. Other surveys on scheduling related problems can be found in [7] and [8]. In this thesis, we concentrate more on the BSF problem.

Many criteria need to be taken care of when forming efficient scatternets. For instance it is necessary to limit the size of the scatternet's piconets to eight to avoid *parked* nodes (i.e. not active) in a piconet. It is desirable also that the formed scatternet assigns the roles on its nodes in a way that divides the communication tasks fairly. In some cases also, it is desirable that a scatternet allow the nodes to join or leave on-the-fly, and at a reasonable cost. All these criteria and others make the BSF problem challenging and unique.

However, we believe that there are two features in the Bluetooth specifications that distinguish BSF algorithms and distributed algorithms on Bluetooth wireless networks in general. These two features are related to the communication mechanisms used by Bluetooth. These features make Bluetooth different than traditional single-channel radio networks (also known as broadcast radio networks) such as the IEEE 802.11 networks, and different than the traditional wired networks that are widely used in the field of theoretical distributed computing. The first of these features is related to the Frequency Division Multiplexing (FDM) communication method that is used by Bluetooth. FDM divides the communication frequency medium into a number of non-overlapping frequency's ranges, allowing therefore multiple communicators to share the medium. The FDM technique used in Bluetooth is the Frequency Hopping Spread Spectrum (FHSS). Using this technique, a transmitter and a receiver utilize a pseudo-random sequence of frequency hops (or channels), that is known to both, in order to exchange messages through these hops, using one different hop at each time slot. With the FHSS technique, two pairs of neighboring communicating devices can communicate using separate channels

without significantly interfere the communication of each others. This is contrary to traditional wireless radio networks, in which it is assumed that all neighboring devices share the same radio medium.

The main problem introduced by FHSS lies in the necessity of setting up an agreement among the communicators on the utilized pseudo-random sequence of frequency hops. This feature of Bluetooth complicates the problem of the device discovery. Solutions are needed to allow neighboring devices discover each other on the same frequency and time (since Bluetooth devices always alternate in a pseudo-random sequence of frequency hops), and to let the pair of communicating devices agrees on a pseudo-random sequence.

The second feature that, we believe, distinguishes distributed algorithms of Bluetooth wireless networks from other networks algorithms lies in the introduction of the Bluetooth states. According to the specifications, if a device wants to discover a neighbor, it goes to the INQUIRY state and keeps alternating rapidly within a range of frequency hops announcing its existence by broadcasting small-sized special packets called ID packets. These packets, although called ID, do not contain the identity of their sender. On the other side, the device that wants to be discovered goes into the INQUIRY SCAN state and alternates at a slower rate in the same range of frequencies scanning for neighboring inquirers. When an inquirer and a scanner meet on the same frequency, then both exchange some packets that are sufficient for them to establish a link. After that, if the communicating devices want to establish a link, then one of them goes into the PAGE state and the other must go to the PAGE SCAN state. The devices, then, exchange some messages until a link is established between them.

These two features of Bluetooth (i.e. the FHSS and the Bluetooth states) lead to new complexities and restrictions imposed on the distributed algorithms of Bluetooth wireless networks. These restrictions and complexities, if not addressed properly, may cause a BSF algorithm be executed in a relatively significant long time. For instance, we compare the results of simulation experiments of two independent

studies. The first study examined a BSF algorithm [9], while the second study [10] examined the performance of two reactive routing algorithms for wireless ad-hoc networks; DSR [11] and AODV [12]. In these algorithms, the nodes communicate with their neighbors, mainly, by means of messages broadcasting. The networks of [11] were composed of Bluetooth devices solely, while those of [10] were of IEEE 802.11 devices. Comparing the results of the two papers, it is found that broadcasting in IEEE 802.11 networks can be executed in about 28 times faster than in the Bluetooth networks, and this ratio may be even larger than that. However, in term of other performance metrics, it was shown that Bluetooth perform better [9]. We argue that the reason behind this weakness in the execution time of the BSF algorithm lies in the improper addressing of the problems introduced by the FHSS techniques and the Bluetooth states (i.e. INQUIRY, INQUIRY SCAN, PAGE and PAGE SCAN states). This comparison shows the criticality of the restrictions imposed by the Bluetooth specifications on the Bluetooth wireless networks distributed algorithms. This example, and similar ones, is given in greater details in Chapter 3.

1.3. Objectives

Because of their criticality, we concentrate in this thesis on the problems that are introduced by the FHSS communication technique and those introduced by the different states of Bluetooth (i.e. INQUIRY, INQUIRY SCAN, PAGE and PAGE SCAN states). In other words, we investigate the problems that are introduced by the device discovery and link establishment procedures as they are defined in the Bluetooth specifications. We also investigate the different solutions to deal with these problems, with much focus on the solutions that the BSF algorithms used. We believe that these problems, and the proposed solutions for them, are necessary for the abstraction of the Bluetooth specifications to a suitable theoretical computational model that serves as a basis to study BSF algorithms, and other distributed algorithms on Bluetooth wireless networks.

We note herein that the scatternet formation problem is the most studied network problem (if not the sole) that is solved with distributed algorithms on

Bluetooth wireless networks. Other problems, such as leader election and broadcasting for example, can also be addressed. However, due to the restrictions imposed by the Bluetooth specifications, the solutions of these classical problems as they currently exist in the literature of theoretical distributed computing would have to be adapted to Bluetooth wireless networks.

1.4. Contributions

In this thesis, in chapter 3, we survey the different device discovery techniques, which are the mechanisms that solve the problems of the device discovery and link establishment procedures. We categorize these techniques into static and dynamic discovery techniques. We divide BSF algorithms, as well, into static and dynamic algorithms, depending on the device discovery technique they use. The nodes in static BSF algorithms execute in an external (or static) phase a device discovery technique that allows them to know the identities of a subset of its neighbors. After the device discovery phase, the nodes execute the formation algorithm. Examples of static algorithms are BlueStars [12] and BlueMIS [14]. Dynamic algorithms, on the other hand, make the networks nodes implicitly discover their neighbors while executing the formation algorithm; making it possible for the nodes to join the scatternet on-the-fly. An example of dynamic algorithms is the algorithm of Law *et al.* [15] . To survey the different discovery techniques, we had to survey the current BSF algorithms and to check the solutions they used to solve the device discovery and link establishment procedures problems since it is rare to find in the literature a research paper that solely presents a device discovery technique.

During studying the different discovery techniques, we observed that there is, to some extent, a "lack of shared knowledge" among the researchers of the field concerning the problems that are introduced by the device discovery and link establishment procedures of Bluetooth. For instance, we found that some proposed device discovery techniques required modifications on the Bluetooth specifications, but yet they have their equivalent in the literature. These equivalent techniques perform the same task as the proposed techniques, do not require any modifications

on the specifications of Bluetooth, and perform even better than the specification non-complaint solutions sometimes. We believe that such a "lack of shared knowledge" had a negative impact on the advance of the research in BSF algorithms. In our opinion, one of the sources of this problem lies in the unconventional nature and the large number of the problems (or restrictions) introduced by the specifications of Bluetooth.

The schema we used for the categorization of BSF algorithms (i.e. static and dynamic) is used as well to identify the different theoretical communication models used to study the BSF problem. We describe and illustrate these models and we make some remarks on them. We argue that the currently available models for Bluetooth wireless networks do not model adequately the complexities of the Bluetooth specifications in most cases. We show also that these models were often oversimplified.

In chapter 4, we look at different approaches to study the device discovery procedures of Bluetooth (other than the approach followed by BSF algorithms). We survey a number of studies that examined in more-depth the device discovery procedures of Bluetooth. These studies, generally, started with an analysis of the specifications in order to find the features that cause problems. Some of these studies proposed also some solutions to overcome these problems. These solutions can be categorized into those that suggested modifications on the specifications of Bluetooth and those that suggested the use of a wireless technology other than Bluetooth to bypass the device discovery problems. Also, we cite and briefly describe some studies that formally analyzed the device discovery specifications and others that used hardware devices in their experiments. Given that these studies examined in details the device discovery in Bluetooth, and given that they provided solutions that may overcome problems of the specifications, these studies may help understanding in more depth the specifications of Bluetooth and the Bluetooth wireless networks model.

In chapter 5, we give two case studies that show the complexities of the Bluetooth specifications and their many features that have a large impact on the performances of BSF algorithms, and therefore cannot be considered always as "details." We show how the Bluetooth technology is sensitive to minor changes (whether in the specifications or in their implementations), and show how such changes can have a substantial impact on a large number of BSF algorithms. In the first case study, we study the impact of the modifications introduced in version 1.2 of the Bluetooth specifications on BSF algorithms. Up to now, the effect of these modifications on BSF algorithms had not been analyzed. We show that these modifications have a substantial positive impact on static BSF algorithms. We conducted some experiments on a common device discovery technique for static BSF algorithms and compared its performance using ver1.2 and ver1.1 (note that the latter is used in most BSF studies). We found that this discovery technique, using ver1.2, generates a connected topology about 3.5 times faster than it does using ver1.1. It also discovers about 20% more edges. On the other hand and more interestingly, dynamic algorithms are not affected by these modifications. This will be proven using some experiments and some analytical arguments. These results give more applicability to static BSF algorithms and call for more research on dynamic BSF algorithms.

In the second case study of Chapter 5, we show how a minor change in the implementation of the Bluetooth specifications can have a substantial negative impact on static BSF algorithms. This was related to the implementation of *frequency trains* that are used in the INQUIRY state during the device discovery procedures. More details about the two cases are given in Chapter 5. Some of the results of Chapter 5 were already published in [16] and [17].

Before we conclude this chapter, we want to note that simulation experiments have been the only way to compare the different BSF algorithms. However, comparing networks algorithms using simulation experiments is a question of debate in the research community of wireless ad hoc networks. For instance, Pawlikowski *et al.* stated in their study [18] that there is a "a deep crisis of credibility" in the simulations of telecommunication networks, and hence they stand with the opinion

claiming that "one cannot rely on the majority of the published results on performance evaluation studies of telecommunication networks based on stochastic simulation." In the opinion authors of [18], this is caused by the many pitfalls that researchers fall in when conducting simulation experiments. Other studies having similar opinion are [19], [20] and [21]. On the other hand, Stojmenovic in [22] advocated the "use of simple models, matching assumptions and metrics in the problem statement and simulation to provide a basic "proof of concept," and comparison with truly competing solutions." This was based on the fact that "theoretical proofs of performance are difficult (often probably impossible) to derive" [22]. However, simple algorithms over different simulators may have different performances. It was shown in [23] that even straightforward distributed algorithms, such as the flooding algorithm, may have "significant divergences" in its performance when it is executed under different simulators, even if these simulators were "popular". It is worth-mentioning that there have been some attempts in the literature to avoid simulations as much as possible by providing concise enough theoretical models of computations, even if they were "harsh" models [24] (see also [25] and [26]).

In summary, we observed four important points while working on this thesis. These points are:

1. The lack of shared knowledge among the researchers of the BSF algorithms domain on the problems introduced by the specifications of the device discovery and link establishment procedures of Bluetooth.
2. The theoretical models used to study BSF algorithms do not adequately model Bluetooth wireless networks in most cases, and were oversimplified in many cases.
3. The existence of many features in the Bluetooth specifications that have a large impact on BSF algorithms, and therefore cannot be considered always as "details".

4. Simulation experiments are the only way used so far to compare different BSF algorithms. However, simulation is considered by many researchers as an inefficient way to compare algorithms.

Therefore, we believe that the creation of a "general computational model" would be essential to solve the weaknesses found in the current BSF algorithms, and to advance the research in BSF algorithms. What is meant with a "general computational model" is the following:

1. It defines clearly the different restrictions imposed by the Bluetooth specifications on the distributed algorithms.
2. It makes such restrictions as a "shared knowledge" among the researchers of the field, and whence this model can be used as a starting point to design distributed algorithms in Bluetooth wireless networks – including BSF algorithms.
3. This model is used also as a basis to compare and analyze the performance of the different (and numerous) BSF algorithms.

Although BSF algorithms seem to be very hard to be analyzed mathematically, such a general computational model can be used to compare these algorithms – at least informally – as a step to keep simulation as the last step to compare them.

We believe that publicizing problems introduced by the Bluetooth specifications would be one of the important steps to create this model. We believe that the problems that are introduced by the specifications of the device discovery and link establishment procedures are the most important ones that should be considered, since these problems will characterize the means of communication between Bluetooth devices. In fact, this is one of goals of the critics and suggestions that will be found later in this thesis and it is the reason for our concentration on the device discovery and link establishment procedures of Bluetooth in typical.

1.5. Thesis organization

The thesis is organized as follows. Chapter 2 gives background knowledge on the Bluetooth technology, BSF algorithms and the device discovery and link establishment procedures. Chapter 3 surveys the different device discovery techniques used by BSF algorithms. It also surveys the different communication models used to study BSF algorithms. Chapter 4 surveys a number of studies that addressed the problem of device discovery in Bluetooth but followed approaches other than those followed by BSF algorithms. Chapter 5 contains two study cases, the first shows how a minor change in the specifications of Bluetooth can substantially improve the performance of static BSF algorithms, but not dynamic BSF algorithms. The second study case shows how a minor change in the implementation of Bluetooth (although legal) can degrade substantially the performance of the static BSF algorithms.

Chapter 2

The Device Discovery and BSF Algorithms: Basics

We start the chapter by defining basic notions that are found repeatedly in the literature of Bluetooth. We talk about the Bluetooth Scatternet Formation (BSF) problem and some of the suggested algorithm solutions for it. After that, we talk about the device discovery and link establishment procedures in Bluetooth.

2.1. Background knowledge

Bluetooth is a technology that enables ad-hoc short-range wireless networking between devices such as laptops, mobile phones, headsets, printers and others. Its specifications have been developed and licensed by a large group of IT companies; among them giants in their domains such as Nokia, Ericsson, Toshiba and others. The specifications have also been standardized by the IEEE under the IEEE 802 wireless personal area network (WPAN) category. The technology provides low-cost, low-power and noise resistant radio systems that have been quite successful since 1998, when it was first introduced. The best indicator of the technology's success is its deployment in a wide range of today's electronic devices. According to statistics in [27], the marketplace has more than two billion Bluetooth devices in it. The number is rising at an approximated rate of 18 million devices weekly.

Bluetooth operates in the Industrial, Scientific, Medical (ISM) 2.4 GHz frequency bandwidth, which are frequency bands that require no license to use them. Bluetooth uses the Frequency Division Multiplexing (FDM) method for communication. FDM is a method that divides the communication frequency's medium into a number of non-overlapping frequency ranges creating

multiple communication channels from the medium instead of only one, allowing, thus, multiple communicators to share the medium. The FDM technique used in Bluetooth is the Frequency Hopping Spread Spectrum (FHSS), which is a technique well-known for its robustness against interference. FHSS is a technique that makes a sender and a receiver utilize a pseudo-random sequence of frequency channels (or hops), that is known to both, in order to transmit messages through these hops, using one different hop at each time slot. This makes a pair of devices communicate with each other using one channel without disturbing their neighbors. The pseudo-random sequence also makes it difficult for an intruder to intercept their signals. However, the main problem of this technique lies in the way of establishing an agreement among the communicators on the utilized pseudo-random sequence of frequency hops. This is, in fact, the main source of the problem of this thesis.

2.1.1. Piconets and Scatternets

For two Bluetooth devices to communicate, they must either belong to the same *piconet* or the same *scatternet*. A piconet is a star topology that consists of a *master* and up to seven *slaves*. A master can have more than seven slaves (up to 255), but this imposes a penalty on the performance of the piconet[1]. Note that a piconet in its most basic form is simply a bidirectional link between two Bluetooth devices. The master device controls the packet flow within the piconet. Any packet sent from one node to another in the piconet *must* pass through the master. This leads the master to utilize an *intra-piconet scheduling* to control this flow. It is worth mentioning here that one node cannot be a master in two different piconets but can be slave in unlimited number of piconets. Therefore, piconets will be identified by their masters.

A scatternet is an interconnection of multiple piconets. Two piconets may be interconnected either by a *Master/Slave (M/S) bridge*, which is a node that has the role of master in one piconet and slave in another, or by *Slave/Slave (S/S) bridge*,

1 The number 7 came from the fact that a master assigns to each of its neighbors an address called the LT_ADDR that is of 3-bit length. The master has the address of 000 in the piconet. The number 255 appears because of the same concept. There is no convincing reason, though, for having it to be 7 and not 16 or 4 instead for instance.

which is a node that has the role of slave in two different piconets. According to the specifications of Bluetooth, it is allowed to have a bridge being part of many piconets. It is also allowed to have two piconets interconnected by more than one bridge. A bridge node should follow an *inter-piconet scheduling* to schedule its time between the piconets it belongs to. Note, however, that an M/S bridge will have to put its piconet (i.e. the one that it is master for) on hold when it is acting as a slave to other piconets. This implies that all the slaves of this bridge node will be put on hold, and no communication between these slaves, directly through their master, will be allowed. Hence, *M/S bridges* pose a higher penalty on the performance of the scatternet than *S/S bridges* do.

When dealing with Bluetooth scatternets, two major problems have to be addressed. The solutions of these problems were not explicitly mentioned in the specifications of Bluetooth, and instead were left as open problems to the researchers. These problems are:

1) *The scatternet formation*: which is the problem of generating a scatternet and assigning to each node in the scatternet a role (i.e. master, slave or bridge), and,
2) *The scatternet management (or scheduling)*: this is the problem of intra-piconet and inter-piconet scheduling.

We refer the reader to [7] and [8] for comprehensive surveys on the latter problem. In this thesis, we focus more on the Bluetooth Scatternet Formation problem. We define and discuss this problem next.

2.2. The basics of Bluetooth device discovery in Scatternets

For any scatternet to exist, its nodes must first discover each other. Due to the nature of Bluetooth, with Bluetooth nodes using a frequency hopping scheme for communication, it is required first that the nodes find each other in both the frequency and time space. This problem is called the Device Discovery problem.

The specifications of Bluetooth provide only the basic functions and some recommendations for the device discovery problem to be solved. According to the

specifications [2][4], if a device wants to discover a neighbor, it gets into the INQUIRY state in which it keeps alternating rapidly in a range of frequency hops announcing its existence by broadcasting small special packets that do not contain the identity of its sender. On the other side, the device that wants to be discovered gets into the INQUIRY SCAN state and alternates in the same range of frequencies but with a slower rate than the inquirers. If an inquirer and a scanner meet on the same frequency hop, and if the scanner listens to one of the inquirer's special packets, then the scanner answers back with another special packet that contain its identity and some other information. At that moment, the inquirer is made aware of the existence of the scanner and of its identity (while the scanner does not know the identity of the inquirer). This would be sufficient to establish a link between the two devices. More details on this process will be given later in this chapter.

The device discovery procedures mentioned in the specifications are suitable only for simple scenarios, as they assume that the devices are manually assigned the roles of being in the INQUIRY or the INQUIRY SCAN states. Other than that, the previous solution gives both the inquirer and the scanner the minimum amount of knowledge required to establish a link. For instance, the method and the time at which a link will be established are not mentioned in the specifications, and must be considered by the designers. Also, note that the inquirer knows the identity of the scanner while the opposite is not true. Obviously then, more complex scenarios are required in practice. Different device discovery techniques are introduced in the literature to address such scenarios. We give a survey on almost all the currently available techniques, but first, we describe in more depth the problem of Bluetooth Scatternet Formation.

2.3. The Bluetooth Scatternet formation problem

There are a large number of Bluetooth Scatternet (BSF) algorithms in the literature, and there will likely be even more in the near future. Until now, there is no "sufficiently good" algorithm that solves the problem, so there is no standardized yet. "Sufficiently good", however, may have different meanings as there is no standard

method for measuring the "goodness" of a BSF algorithm. Many metrics were suggested for evaluating scatternets. For us, the most important ones are (ordered from the most important to the least):

1. *Time:* the time needed to form the scatternet. This is usually measured by simulation experiments.
2. *Connectivity:* the algorithm must guarantee the connectivity of the scatternet– as long as its nodes originally form a connected network. Before executing a BSF algorithm, a link between two nodes is set if they are in the radio range of each other, but in the context of scatternets, a link is set between two nodes if there is a master-slave relation between them. The term "guarantee" is a little ambiguous since there is always a tiny probability that two neighboring nodes never discover each other. However, these cases are usually ignored by BSF algorithms and it is assumed that two neighboring nodes looking for each other will meet always in a finite time.
3. *Out-degree constraint:* each piconet in the scatternet should not have more than seven slaves. According to the specifications of Bluetooth, if a master has more than seven slaves, it has to *park* them (i.e. make them in the PARK state). A node in the PARK state cannot participate in the current piconet until it is out of it. Obviously, this imposes a penalty on the scatternet performance.

As a consequence of the fact that the bridges nodes participate in their piconets on a time division basis, the following three metrics should be considered by a BSF algorithm:

4. *Number of Piconets:* The number of piconets (masters) should be reduced. Indeed, when it increases, the number of bridges increases as well, and this affects negatively the performance of the scatternet.
5. *Number of M/S bridges:* This number should be reduced. As mentioned before, M/S bridges have a higher penalty on the performance of the scatternet than S/S bridges do.
6. *Number of bridges and their degrees:* the number of bridges should be

minimized as much as possible. However, if we want to preserve the degree-constraint requirement, there will be a limit to such minimization. A BSF algorithm, hence, should try not to make the degree of the scatternet bridges high.

Other important metrics also exist. Examples are the *average shortest path* of the scatternet – compared to the average shortest path in the original network, and the *maximum traffic flow* [28]. Some other metrics related to the capacity of the scatternet were also proposed [29].

Stojmenovic and Zaguia [6] published a survey in 2004 that covers most of the available BSF algorithms at that time. They provided many criteria to divide the BSF algorithms. Among these criteria are the followings:

1. *Single-hop algorithms vs. Multi-hop algorithms:*

   Single-hop algorithms such as [30] [31] [32] [33] assume that each node in the network can communicate directly with another node (i.e. a complete graph). The multi-hop algorithms relax this assumption (see [12] [14] [29] [34]). Most of the multi-hop BSF algorithms assume the Unit Disk Graph (UDG) model for communication. Note that in a UDG model, if two nodes were in each other radio vicinity, then there must be a link between them. This means that it should be assumed that there are no faults or broken links in the network in order to assume the UDG model. We found that all of the BSF algorithms that deterministically generate out-degree constrained scatternets either assume the complete graph model or the UDG model. Bluepleidas [34] is an exception. However, the reader must note that it assumed the uniform distribution of the nodes in the plane. More discussion on BluePleidas can be found later in section 3.1.1 and section 5.2.1.

2. *Centralized algorithms vs. Distributed algorithms:*

   Obviously, in environments where we deal with mobile nodes, a distributed algorithm is preferred. However, there are a number of algorithms, such as [35] for instance, that assumed the existence of a central entity that executes

the formation algorithm or some other optimization procedures. Other centralized algorithms execute a distributed election algorithm to elect a leader. The leader decides then the role of each node in the scatternet [36] [37].

3. *Self-healing algorithms vs. Non-self-healing algorithms:*
   Self healing algorithms provide procedures that allow the on-the-fly joining (addition) or leaving (deletion) of nodes to or from the scatternet. Few of such algorithms exist nevertheless. Examples of single-hop self-healing algorithms are [15] and [32]. Examples of multi-hop self-healing algorithms are [29] and [38]. However, most of the algorithms in literature are non-self-healing algorithms.

Device discovery techniques have a major effect on the performance of the BSF algorithms that use them, especially in term of their execution time. We propose a new criterion to evaluate BSF algorithms based on how a node in the scatternet discovers its neighbors. This criterion will be the one most used in this thesis:

4. *Dynamic algorithms vs. Static algorithms*
   Static algorithms assume that each node knows the identity of a sufficient number of neighbors (or links) in order to guarantee the connectivity of the network before starting the execution of the BSF algorithm. They do this by executing an external *device discovery technique*. Examples of static algorithms are BlueStar [12], BlueMesh [39], Bluenet [28] and BlueMIS [14]. Dynamic algorithms, on the other hand, include an implicit device discovery phase and each time a node discovers another, part of the scatternet is constructed. Dynamic algorithms accept on-the-fly joining of nodes to the scatternet (by definition) and most of them accept on-the-fly leaving. Examples of dynamic algorithms are those of [15], [29], and [40].

An example of a static algorithm is the well-known BlueStars [12]. It is executed in three phases. The first phase is the device discovery. Each node attempts

to discover enough neighbors to guarantee the connectivity of the underlying network. The second phase is where the piconets are constructed. Each node has a unique ID. The nodes that are largest (i.e. has the largest ID) among their neighbors start capturing all their smaller neighbors as slaves, with the condition that a slave belongs to only one piconet in this phase. A node that finds that all its larger neighbors became slaves assigns itself the role of master and captures the smaller neighbors as slaves (also, with the condition that they are not already captured by another master). This creates many isolated piconets, which will be interconnected in the third phase of the algorithm. Neighboring piconets are defined as those that their masters can be interconnected through 1 slave (2-hop) or 2 neighboring slaves (3-hop). Interconnecting each pair of neighboring piconets with one bridge is sufficient to guarantee the connectivity of the scatternet. Whence, BlueStar is a multi-hop algorithm, guarantees connectivity and does *not generate an* out-degree constrained scatternet, since a node may have more than seven smaller neighbors and captures them all.

TSF [32] is an example of a dynamic algorithm. It works in single-hop scenarios. It constructs a tree-like scatternet. The algorithm keeps constructing tree-like *components* and merges them until the full scatternet is constructed. A node, in TSF, divides its time between 1) discovering new neighbors to continue the formation of the scatternet (called as the FORM state), and 2) communicating with the nodes of the already-built scatternet (the COMM state). Three type of nodes exist in TSF; free nodes (i.e. those that do not belong to any tree), root nodes (i.e. the root of their trees) and non-root nodes (i.e. any other node in the tree that is not root). The algorithm of TSF is simply as follows; whenever a free node discovers another free node, it constructs a piconet with it and one of them becomes the root of the constructed tree. If a free node discovers a non-root node, it becomes a slave of a newly discovered non-root node. If a free node discovers a root node, then it destroys the connection with it without making any scatternet-related action. The last rule of the algorithm is that root nodes are only allowed to merge with other roots. This gives an algorithm

that works only in single-hop scenarios, that generates a tree-like scatternet and that is dynamic.

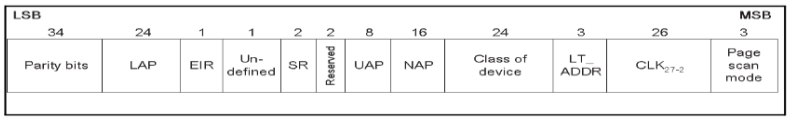

Figure 1: The ID and FHS packets (Source [4]). The ID packet is used for inquiry and paging purposes. It is a small packet of 68-bit length – which makes it robust against interference. It may contain either an Inquiry Access Code (IAC) or a Device Access Code (DAC). The former is common to all devices, and it is used in when a device is inquiring, while the former (the DAC) is used by a device in the PAGE state to contact another specific device in the PAGE SCAN. To page another device, we need to gain some information about it from the INQUIRY process, which will be included in the FHS packet. FHS packets are also special packets. The figure above shows its fields. It contains the address of its sender and its clock. Note that it has 2 reserved bits, and 1 undefined bit for future uses. Some BSF algorithms, such as [41] have utilized these bits. Note also that the FHS packet shown below is of the Bluetooth ver2.1.

2.4. The device discovery and link establishment procedures

We get into the details of the procedures of device discovery and link establishment as they are defined in the specifications. These will be necessary information repeated throughout the thesis.

For two nodes to establish a link, they must perform two operations; the first is the *inquiry*; in which one node gets the address of (or *discovers*) another node. The second is the *paging*, in which a link is constructed. The specifications of Bluetooth state that for one node to discover others, it must switch to the INQUIRY state, and then, keeps broadcasting small special packets (of 68 bits size), called the ID packets. The contents of these packets are common to all Bluetooth devices, and although called ID packets, they do not contain the identity of their senders.

The inquirer's ID packets must be broadcast in different frequency hops in hope of finding other devices listening to the same frequency hop at the same time. The inquirer keeps alternating in a range of frequencies - 32 out of the 79 available hops known to all the devices[2].

2 Bluetooth operates in the range of frequencies 2.4000 GHz to 2.4835 GHz. It divides the medium to 79 channels each of 1 MHz. The rest of the available frequencies are considered as lower and upper band edges, in order to comply with the radio regulations of different countries.

On the opposite side of the connection, a node that wants to be discovered (i.e. a scanner) has to switch to the INQUIRY SCAN state and keeps listening to ID packets generated from the inquirers. The scanner switches between the frequencies hops at a slower rate than the inquirer's in order to increase the chance of finding each other in the same hop. When the scanner receives one of those ID packets, it replies to it with a special packet called the FHS packet. An FHS packet contains 1) its address and 2) its clock value (for more information about the special packets, see Figure 1). The clock value is used for synchronization purposes to make the link establishment procedure faster. These procedures are shown in Figure 2 below.

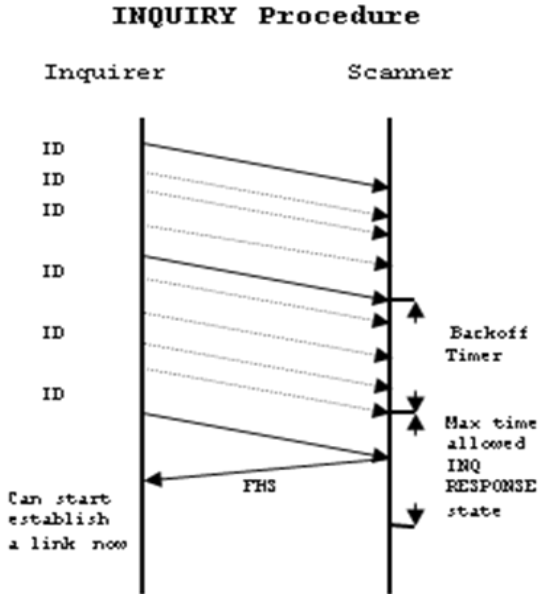

Figure 2: The INQUIRY procedure. The inquirer starts at first by broadcasting ID packets. If one of these packets was received by the scanner, it back-off for a random period of time, and then goes to the INQUIRY RESPONSE state. If it receives there another ID packet, it replies to it with an FHS packet. A link then can be established. Note that this is the inquiry procedures as specified in Bluetooth ver1.1

In Bluetooth ver1.1 and the versions that appeared before it, the scanner back-off (i.e. does not listen to any packet) for a random time before it sends back the FHS packet. In the versions 1.2 and above, the slave sends the FHS packet and, then, back-off for a random time.

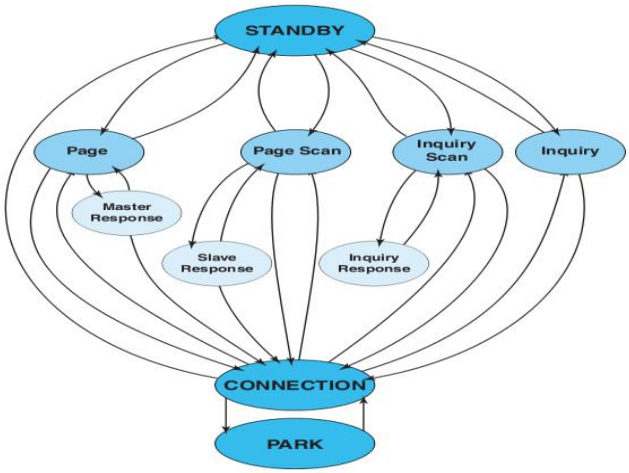

Figure 3: States in Bluetooth (Source [4]). This figure shows the states diagram of Bluetooth. Initially the device rests in the STANDBY state. If a device wants to discover new neighbors, it goes to the INQUIRY state, and if it wants to be discovered, then it goes to the INQUIRY SCAN. If a scanner receives an ID packet from an inquirer it goes to the INQUIRY RESPONSE state and replies to it. The inquirer itself, after receiving a reply from a scanner, may goes to the PAGE state to establish a link. The devices in the PAGE and PAGE SCAN states pass through the MASTER RESPONSE and SLAVE RESPONSE states, respectively, exchanging some messages until they reach the CONNECTION state, in which the link is established and all the connections can take place. A master keeps at most 7 slaves in its piconets in the CONNECTION, while the rest are kept in the PARK state.

The paging procedures (or the link establishment procedures) are shown in Figure 4. Initially, the master node switches to the PAGE state and the slave must switch to the PAGE SCAN state (for more information about the states in Bluetooth, see Figure 3). Using the information obtained from the FHS packet, the master adds an offset value to its clock to have the same clock value of the slave so they are both become synchronized to each other. After that, the master attempts to contact the slave by broadcasting again a number of ID packets. If the slave listens to one of these ID packets, it replies with another ID packet and go to the SLAVE RESPONSE state. When this ID packet is received by the master, it changes its state to the MASTER RESPONSE. Then, the master sends an FHS packet. Note that ID packets do not contain the identity of their senders, while FHS packets do. Whence, the slave did not know the identity of the master before this step! After this, few other messages are exchanged to complete the establishment of the link.

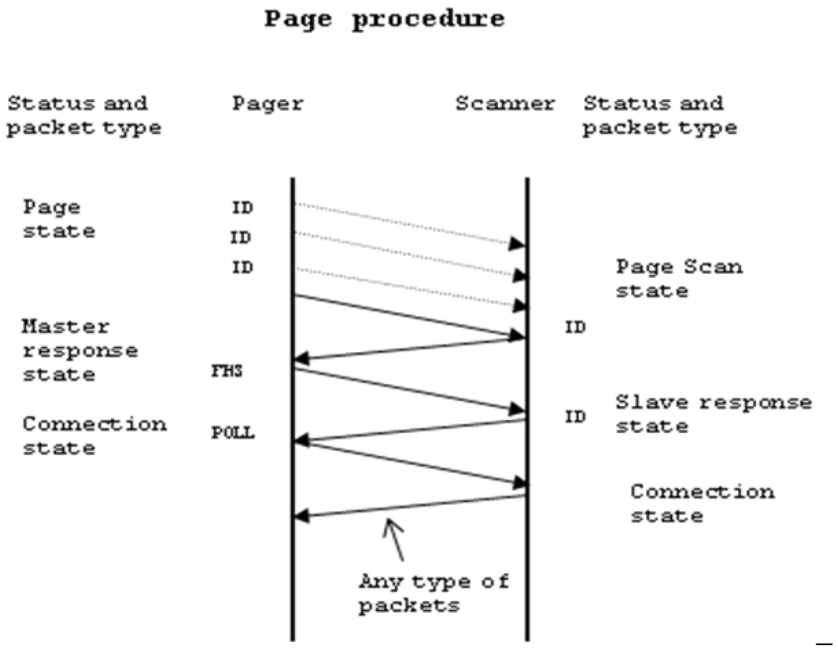

Figure 4: The PAGE procedure. The page procedure is illustrated in this figure. The flow of the packets between the pager and the scanner is shown accompanied with the statuses of both the pager and the scanner.

2.4.1. The frequency trains

In the INQUIRY state, a device alternates between the 32 frequency hop channels using a formula derived from its clock. The formula is shown below in Eq.1:

$$\mathbf{F= [CLK_{16-12} + k + (CLK_{4-2,0} - CLK_{16-12}) \bmod 16] \bmod 32} \qquad \mathbf{(1)}$$

F is the frequency that is used for the inquiry. $\mathbf{CLK_{i-j}}$ are the i^{th} to the j^{th} bits of the device clock **CLK.** The value of CLK_0 is switched each 312.5 µsec. However, a *time slot* in the Bluetooth specification is equal to 625 µsec, which is the time for CLK_1 to change its value. It can be observed that the formula generates only 32 frequency hops, although Bluetooth uses a set of 79 frequency hops. These 32 frequencies are divided into two sets, each of size equals to 16. These two sets of frequencies are called *trains*; Train A and Train B. The variable **k** can have only two different values, and it is used to select the currently used train of frequencies. The trains are switched each 2.56 seconds by default, and can be switched more frequently depending on the

user requirements. According to the specifications, the inquiring device spends 10ms to pass through all the frequencies of a train; thus, each train is repeated 256 times. Note that after 1.28 seconds (i.e. every time the 12^{th} bit of CLK changes); a frequency is swapped between Train A and Train B.

A scanner device, on the other hand, follows a different characteristic of alternation between the 32 frequencies. Firstly, the scanner switches between the frequencies hops at a slower rate than the inquirer's in order to increase the chance of finding each other in the same hop. Secondly, it pseudo-randomly alternates between the 32 frequencies, independently from the value of the train. The scanner changes its scanning frequency each 1.28 seconds. Many implementations set the time to change the scanning frequency to 11.25 msec and let a device spends the rest of 1.28 seconds performing other tasks. The time of 11.25 msec is sufficient for scanning a whole train, which is 10 msec (as it is mentioned above).

2.4.2. Modifications in Bluetooth ver1.2

All the previous features of the device discovery and link establishment procedures apply for Bluetooth ver1.1, which is the mostly used Bluetooth version studied in the literature. However, a new version of the specifications (ver1.2) appeared in 2003, and currently, Bluetooth is in its ver2.1. An even newer version of Bluetooth, ver3.0, appeared in April 2009. Most of the specifications we mentioned above still apply for the new Bluetooth versions. However, Bluetooth ver1.2 presented some modifications on the INQUIRY procedures. These modifications exist in all the versions that appeared after Bluetooth ver1.2.

According to ver1.1 of the Bluetooth specifications, if a scanner device (i.e. a device residing in the INQUIRY SCAN state) receives an ID packet from another inquirer device (i.e. a device residing in the INQUIRY state); the scanner has to back-off for a random amount of time, called the *back-off time* (see Figure 2). While it is backing-off, the scanner is not able to contact or be contacted by any other device. At the end of the back-off time, the scanner goes to the INQUIRY RESPONSE state. In this state, the scanner listens to other ID packets. If it receives any ID packet, whether

it was from the inquirer device that made it backs off or not, the scanner replies with an FHS packet. The scanner, after that, may go directly to the PAGE SCAN state. On the other side, the inquirer, if received the FHS packet, may go to the PAGE state and starts establishing the link. It is also possible for both the scanner and the inquirer to stay at their current states.

The *back-off* timer, in the version 1.1 of Bluetooth, is used to prevent cases of collision at which more than one scanner device receives an ID packet at the same time and from the same sender. In such cases, all the receivers send back their FHS packets at the same time resulting in the collision of the packets. Nevertheless, its disadvantage is that it causes a link to be established in a relatively long time. Also, the backing-off decreases the probability of a successful establishment of a link, because each scanner must receive two ID packets to start establishing a link or at least to announce its existence.

A modification on this procedure was introduced in Bluetooth ver1.2, and it is as follows; if a scanner device receives an ID packet from an inquirer, it goes to the INQUIRY RESPONSE state and replies to the ID packet with an FHS packet directly. At that point, the scanner has two options; either it *backs off* for a random amount of time *or* it goes directly to the PAGE SCAN state and start establishing the link. The difference between the link establishment procedures of ver1.1 and ver1.2 is shown in Figure 5. Note that the back-off timer still exists in the Bluetooth ver1.2. It used also to decrease the probability of collisions but in a different way.

There is another simple modification (or option) that Bluetooth ver1.2 introduced. The specifications call it the *interlaced scanning option*. This option increases the probability of a scanner to be discovered by more inquirers. It addressed a problem related to the frequency trains that are used during the inquiry procedures. According to the specifications, a device uses only 32 frequency hops out of the 79 available if it is in the INQUIRY state. It divides this set of frequency hops into two sets, or trains; called as Train A and Train B.

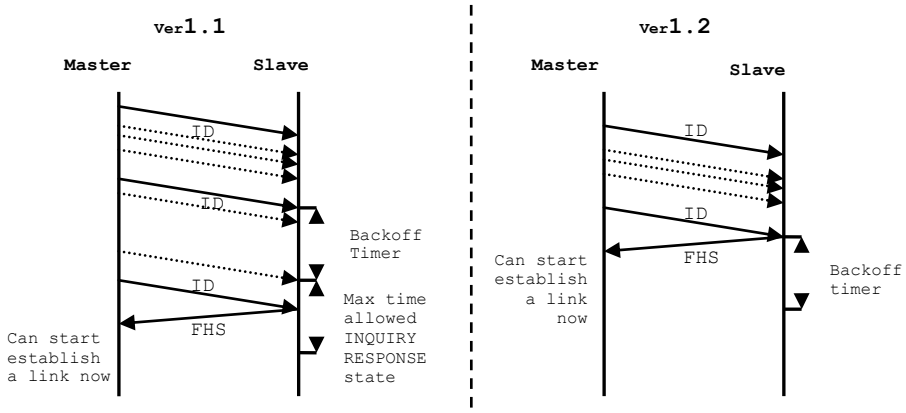

Figure 5: Differences in the link establishment procedures between the Bluetooth versions ver1.1 and ver1.2. The time required starting establishing a link in ver1.2 is faster than this of ver1.1. Note that, in ver1.1, a node may stay in the INQUIRY RESPONSE state for only a finite time.

Recall that in Bluetooth ver1.1 a scanner device selects one frequency from one of the trains (let us say Train A), and keeps listening to it for a period of time called $T_{w\_inquiry\_scan}$. After this period of time is elapsed, it selects a new frequency from the same train and repeats the same procedure. After 2.56 seconds, the scanner selects frequencies from the other train, which happens to be Train B in this case. In Bluetooth ver1.2 and above, and if the interlaced scanning option is activated, the scanner selects two frequencies; one from Train A and another from Train B. The scanner listens for a period of $T_{w\_inquiry\_scan}$ in the first frequency hop, and then, it listens in the other frequency for the same period of time. After two periods of $T_{w\_inquiry\_scan}$, the scanner selects two other frequencies from the two frequency trains and repeats the same procedures.

Chapter 3

Critics on the State of the Art

We start this chapter with a comprehensive survey that covers most of the device discovery techniques that are used by the currently available BSF algorithms. A device discovery technique is defined as the mechanism used to solve the problems of the device discovery. We analyzed many BSF algorithms to know the discovery techniques they use, since it is rare to find in the literature a research paper that solely present a device discovery technique. This survey serves as a basis to understand the different communication models used for Bluetooth wireless networks. We describe these models. We observed that these models do not adequately model the complexities of the Bluetooth wireless networks, in most cases, and were oversimplified in many cases. We give, in this chapter, examples that back our argument.

3.1. Device discovery techniques in BSF algorithms

Device discovery techniques are important because they have a major impact on the execution time of a BSF algorithm. We called the simplest device discovery technique as the *naïve technique*. It works as follows; all nodes are initially in the INQUIRY SCAN state. A node that wants to establish a link with another node switches to the INQUIRY state to discover some of its neighbors. After a period of time, it gets replies from a number of neighboring scanning nodes. It can happen also that no node replies even if there are some scanning nodes in range. The inquirer selects then one of the repliers and attempts to contact it. It does this by changing its state to the PAGE state and starts the paging procedure. The node in the other side must be in the PAGE SCAN. Whence, there should be some mechanism that switches the repliers state (i.e. the scanners that replied to the packets of an inquirer) to the PAGE SCAN state at some time.

As it can be seen, this simple scenario is not efficient for scatternets since it requires the nodes to be manually switched to the INQUIRY state and it does not guarantee connectivity. In other words, if two neighboring nodes perform the exact same task at the same time, then none of them will discover the other, since each of them is in the INQUIRY state. A more automated solution is required therefore.

The available device discovery techniques avoid the manual switching to the INQUIRY state by randomization. Two approaches exist. One approach is to assume that a node selects its role, as a master (INQUIRY) or slave (INQUIRY SCAN), depending on a random variable of its own [42] [43]. The other approach is to make all the nodes alternate between the INQUIRY and INQUIRY SCAN states and when two devices discover each other they perform some actions that help the BSF algorithm. The second approach is more frequent in the Bluetooth Scatternet Formation algorithms (BSF) literature.

We divide the device discovery techniques into static techniques and dynamic techniques; depending on the BSF algorithm that uses them. Next, we give a survey of the available techniques, starting with the static device discovery techniques.

3.1.1. Static device discovery techniques

One of the first papers on BSF algorithms was presented by Salonidis *et al.* [36]. They introduced a BSF algorithm that is centralized, single-hop, non-self-healing and static. They introduced as well the first basic static device discovery technique. All the other static discovery techniques are based on the idea of this technique. The idea is to keep all the nodes execute, for a specific amount of time, a procedure in which each node alternates between the INQUIRY and INQUIRY SCAN states, staying in each state for a random amount of time. This makes the inquiry procedure symmetric (i.e. there is no need to manually assign each node a fixed role as master or slave). Figure 6 depicts this scenario. At the end of discovery procedure, each node knows most of (and sometimes all of) its neighbors and then starts the execution of the BSF algorithm. The results of this paper are very useful,

however, many important implementation details of the proposed device discovery were not considered in it.

Figure 6: Illustration of the device discovery technique of Salonidis *et al.* [36]. This figure illustrates a scenario in which two nodes are executing this discovery technique. The two nodes cannot discovery with each other except if they were in opposite states. These time slots are represented with the rectangles.

More details to this device discovery were considered by Basagni *et al.* in [44] and by Sato and Mase in [45] at the same time. Both papers' results were similar. The two papers added to the work of [36] that as soon as two devices discover each other, a temporary piconet is constructed to exchange some messages and to guarantee mutual (or symmetric) knowledge between the nodes (i.e. if A knows B then B knows A also). For simplicity, we call this technique ALTERNATE. The term ALTERNATE will be used frequently in this thesis. The experiments conducted in [44] were made using the ns-2 environment [46] and a modified version of the BlueHoc C++ library [47], developed by IBM. The experiments dealt with networks having an average nodal degree between 6.9 and 9.2. The authors found that 6 seconds were sufficient to produce, with a high probability, a connected network. In this context, a link in these networks is set between a pair of nodes if and only if they discovered each other. On average, after 20 seconds of simulation time, the experiments did not produce networks where each node knows all of its neighbors. Later, the same team of authors published, in another paper [48], the results of the same technique and concluded that it is "extremely time-consuming to require that each node becomes aware of all its neighbors." We made some experiments on ALTERNATE and we found similar results. Figure 7 below shows the pseudo-code of ALTERNATE.

Having all nodes discovering all their neighbors is of special importance in the Bluetooth Scatternet Formations (BSF) algorithms domain, since it is a condition to have a Unit Disk Graph (UDG). Most static BSF algorithms that guarantee out-degree

```
ALTERNATE (Node A)
while T_Disc > 0
   p ← random(0,1)
   if (p < 0.5)
      Inquiry for time = random (T_min, T_max)
      InquiryScan for time = random(T_min, T_max)
   else
      InquiryScan for time = random(T_min, T_max)
      Inquiry for time = random(T_min, T_max)
end
```

Figure 7: The pseudo-code of the ALTERNATE device discovery technique. Recommended values of T_{Disc}, T_{min} and T_{max} are 20 seconds, 0.01 seconds, and 0.5 seconds respectivley. Function random(x,y) returns a a value between x and y drawn uniformley.

constrained piconets (i.e. with seven or less slaves) assume that the networks can be modeled by a UDG, or that there exist a static device discovery technique that can generate such networks (see for example [50][30][14][39][49]). These algorithms base their solutions on the fact that a node, in a UDG graph, has at most 5 neighbors that are not also neighbors with each other.

Since ALTERNATE does not generate a UDG in a reasonable time, and since many BSF algorithms assume the UDG model, the authors of [48] proposed a solution to bypass this problem. Their solution is to make the nodes enter to a replenish phase after the device discovery phase. In the replenish phase, each node exchanges with its discovered neighbors its neighbor list. Then, a node keeps alternating between the PAGE and PAGE SCAN states for a specific amount of time and tries to contact all the neighbors of its discovered neighbors in hope that they also be in its one-hop neighborhood and not yet discovered. The solution gives only statistical guarantees of generating a UDG. Recall that to establish a direct link between two nodes, one of them must be in the PAGE state and the other in the PAGE SCAN state, and this explains why the nodes need to alternate between the two states.

A modified version of ALTERNATE was presented in [34]. For simplicity, we call this technique LIM-ALTERNATE. The idea of LIM-ALTERNATE is to let a node discover or be discovered by at most *c* of its neighbors; where *c* is a constant set to 5, 6 or 7. The authors proved analytically that, with high probability, this technique generates a connected network. They tested also the technique with ns-2 [46] simulations. They ran over the networks that LIM-ALTERNATE generated the BlueStars algorithm [12]. Running BlueStars over networks generated by LIM-ALTERNATE will guarantee both the connectivity and the degree-constraint requirement in the formed scatternets, since LIM-ALTERNATE generates degree-bounded networks. The new version of the algorithm is called BluePleidas. Nevertheless, the main problem in LIM-ALTERNATE is that that the authors based their argument over the assumption that the nodes are uniformly distributed on a 2-D plane.

3.1.2. Dynamic device discovery techniques

Dynamic BSF algorithms construct small components of nodes, and they continuously merge them together until there are no more nodes to add to the scatternet. A component can be a single node, a piconet or a scatternet. The nodes of a component schedule their time between communicating with the other nodes of the component and discovering other nodes of other components to merge with them. The way of scheduling the discovery procedure differs from one algorithm to another. Dynamic BSF algorithms introduced four ways for making a component discovers its neighboring components, listed next, (and after that we get into greater details):

- **Case 1)** Only specific node(s) in a component perform the discovery procedures; as in the case of [15] and [51],
- **Case 2)** All the nodes of a component successively perform the discovery procedures; as in [40] and [43]
- **Case 3)** All the nodes of a component independently perform the discovery procedures, as in SHAPER [29] and TSF [32] , and

- **Case 4)** All the nodes in a component perform the same discovery procedure at the same time [52].

Case 1) only specific node(s) perform the discovery procedures:

One of the first dynamic BSF algorithms to appear was designed by Law, Mehta and Siu [15]. As the other dynamic BSF algorithms, it builds components and merges them. Each component has one leader. The leader selects one node in the component (let us call it the discoverer) and asks it to alternate randomly between the INQUIRY and the INQUIRY SCAN states. The discoverer has either to be a slave to the leader or the leader itself, if it had no slaves. The other nodes do not do any action related to the device discovery. If a discoverer finds another discoverer, then both perform some actions to merge their components together. A new component is built then and a new leader is elected. The new leader selects another discoverer and the whole procedure is repeated then. If a discoverer does not find any other node after a specific period of time, it assumes that its component contains already all the nodes in range.

One issue of this technique is that it guarantees connectivity only for single-hop scenarios – assuming that it is always possible for an inquirer to find a scanner in a finite time. The only solution to this problem is to let each node in the component performs the discovery procedures at the same time. Note that this technique was used also in [51].

Case 2) all the nodes successively perform the discovery procedures:

Ghosh *et al.* in [40] introduced a dynamic BSF algorithm, called BTSpin, with a new discovery technique, which they called the spin technique. The idea of the technique is to make a master of a piconet orders one of its slaves to perform a spin for a specific amount of time, where a spin means performing an inquiry followed by an inquiry scan[3]. As soon as a slave completes its spin, it informs its master and the

[3] It should be noted that a spin is defined in the original paper as an inquiry followed by an inquiry scan. This is different than the random alternation between the two states that is followed by other dynamic BSF algorithms.

master, then, orders another slave to perform a spin. The master repeats this process until it passes through all its slaves, then after that, the master itself makes a spin. When two nodes discover each other, some action is taken to merge the two neighboring piconets.

Note that it is stated in the algorithm of [40] that the masters order their slaves to perform a spinning. Thus, it is not the case that each node in the component successively performs the discovery procedure (as the title of this section indicates), because there may be many piconets (masters) in one component, and thus, many nodes performing a spin. A pure solution that let each node in a component *successively* perform the discovery procedure does not yet exist. Such a pure solution guarantees connectivity (as it is the case for the device discovery technique of BTSpin [40]), but the execution time of the BSF algorithm that uses it will presumably be longer. We note that a very similar technique to the one we just described was introduced in [43].

Case 3) All the nodes independently perform the discovery procedures:

SHAPER [29] and TSF [32] use the same discovery technique. In a component, each node schedules its time, independently (i.e. without any organization with the other nodes of the network), between communicating with the nodes of the same component and discovering new nodes. A node that is aiming to discover new nodes switches randomly between the INQUIRY and INQUIRY SCAN states. When two nodes discover one another, a merging action is performed.

Such a technique guarantees connectivity, however it has one major limitation. In the Bluetooth specifications, if a node was in the INQUIRY state, then it cannot do anything other than the inquiry. Same thing applies for the paging process. Thus, assume that we have three nodes A, B and C on a straight line (i.e. A and C are not neighbors, but B is neighbor of both). Assume B is in its INQUIRY state searching for new undiscovered neighbors. Assume that A and C are trying to send messages to each other. Because B is in the INQUIRY state, the link between A and C is blocked until B finishes inquiring. The number of such blocked links increases as

the network grows in size, and this imposes problems. The problem exists in other dynamic BSF algorithms, but it is much clearer in SHAPER [29] and TSF [32]. More analysis of the device discovery techniques of SHAPER and TSF can be found in Section 5.2.2.

Case 4) All the nodes in a component perform the same discovery procedure at the same time

Another discovery assumption would be to let all the nodes in a component perform the same discovery procedure at the same time. Joung and Hang in [52] introduced this new technique, which is called BlueTag. In BlueTag, each node initially follows a random sequence of alternation between the INQUIRY and INQUIRY SCAN states. A node uses its ID as a seed to generate this random sequence of alternation. A node may also use a variable other than its ID. Let us call this variable root. As two nodes discover each other, a link is constructed, and both nodes use a newer value of root to generate a new random sequence of alternations. The new value of root can be, for example, the largest ID value in a component. This change in the root value makes all the nodes of a component follow the same sequence of alternation between the INQUIRY and INQUIRY SCAN states. Whence, two nodes belonging to the same component never discover each other.

The technique is promising since it let each node in the scatternets perform the discovery procedures, and it guarantees the connectivity. It also prevents cases where two nodes of the same component discover each other, which is considered in many cases as a penalty. However, we do not have any knowledge about how fast it generates a connected network since the authors of the paper do not present any experiments or any BSF algorithm. To our knowledge, there is no BSF algorithm that uses this discovery technique.

3.1.3. A note on the current device discovery techniques

One goal of surveying the different device discovery techniques is to enhance the understanding of the details of the problems of Bluetooth Scatternet Formation,

since every BSF algorithm must deal with this problem. We observed while working in this survey that there is, to some extent, a lack of shared knowledge among the researchers of this research area of the problems introduced by the Bluetooth specifications and the restrictions they impose on BSF algorithms. This survey helped us to find some of the strengths of the available device discovery techniques. For instance, we found that two BSF algorithms introduced two new device discovery techniques that require modifications on the specifications of Bluetooth, but yet, existing discovery techniques are sufficient for those algorithms to work efficiently and without any negative side-effects. One of those algorithms is the work of Chiasserini and Marsan in [41], who introduced an interesting dynamic BSF algorithm, but did not focus much on its device discovery phase.

The authors of [41] introduced a device discovery technique that is based on the one used by the TSF algorithm [32], which was described in section 3.1.2. Recall that in dynamic BSF algorithms a node has two tasks; the intra-scatternet communication and the discovery of new neighboring nodes. In the algorithm of [41], a node that has the task of discovery goes to the INQUIRY state, broadcasts its ID packets, collects information about all the scanners that replied to it, and then attempts to contact the most suitable node(s) among them. Recall at this point that for a node to establish a new link it must be in the PAGE state and the other node must be in the PAGE SCAN state at the same time. However, the authors of [41] do not present any method for synchronizing the switching of the communicator nodes to the PAGE and PAGE SCAN states. For the inquirers, we believe that the simplest solution to this problem (and perhaps the most efficient) is to make them go directly to the PAGE state after receiving a reply from a scanner. For the scanners, the case is different. However, only two possible methods can be followed by the scanners. The first method is to make the scanners go to the PAGE SCAN state for a fixed period of time directly after replying to ID packets. The second method is to make the scanners alternate randomly between the PAGE SCAN and INQUIRY SCAN states. The second method is inefficient, since the probability of discovering new neighbors is decreased because the scanners spends less time in the INQUIRY SCAN state. On the

other hand, the first method makes this discovery technique becomes exactly as the technique of TSF [32].

Therefore, we claim that if the technique of TSF was followed with no modifications at all, then the algorithm of [41] will work efficiently and with no negative side effects. Note also that the authors proposed using some empty slots (i.e. bits) in the FHS packet that were kept for future purposes (see Figure 1, page 28). In this case as well, we see that the use of the empty slots is not necessary since the device discovery technique of TSF forces us to construct a piconet, and as a consequence, the data that will be stored in those slots can be exchanged after the construction of the piconet in a negligible amount of time.

Furthermore, the authors of [53] introduced a static BSF algorithm. They introduced with it a device discovery technique. However, their discovery technique required some modifications on the Bluetooth specifications. Same as ALTERNATE, their technique idea is to make all the nodes alternate between the INQUIRY and INQUIRY SCAN states. However, instead of constructing a temporary piconet, the technique bypasses this phase by increasing the size of ID packets so that they include more information. The modified ID packets include, among other changes, the identity of the sender, the number of its neighbors, and some other values necessary for the algorithm. However, we claim that, if ALTERNATE was employed then the algorithm works efficiently, with no negative side-effects and may be better. This is for two reasons, the first is that the time of constructing the temporary piconets and exchange the information is negligible. This was found during our experimentations with ALTERNATE. It is also because the increase in the size of ID packets, as it is the case in this algorithm, makes them less robust to interference. Thus, the percentage of collisions increases, and therefore, the execution time of the algorithm increases as well. Interestingly, the same discovery technique was later used by [54] and [55].

From our point of view, we see that the modifications that are intended to only a specific algorithm or more should be avoided as much as possible, and if we were obliged to make such modifications then it would be better to generalize the

suggested modification (i.e. to make it useful for all other BSF algorithms). The specifications of the device discovery and link establishment processes in Bluetooth were the most modified by the BSF research community, mainly because of their criticality and high impact on BSF algorithms. We believe that the details of these processes should be considered more carefully when dealing with BSF algorithms. We believe that the neglecting of such details (intentionally sometimes and unintentionally some others) are one of the main reasons behind the unavailability of a standardized BSF algorithms until now.

3.2. Modeling Bluetooth wireless networks

Different "computational models" are used to study BSF algorithms. Interestingly, the schema that we used to categorize the BSF algorithms into static and dynamic algorithms serves as well in the categorization of these computational models. In this section, we examine these models. Some of our observations about them are given in this section as well. We show that these models do not model adequately the complexities of the Bluetooth specifications most of the times. We show also that these models were often oversimplified. We believe that the cause of this issue lies in the problems introduced by the device discovery and link establishment procedures. Therefore, we think that these problems need to be investigated further.

3.2.1. Communication models of static BSF algorithms

Static BSF algorithms assume initially that the Bluetooth nodes are distributed on a space. The nodes form a vicinity graph, in which the vertices are the radio nodes, and an edge is set between two nodes if they are in the radio vicinity of each other. All available BSF algorithms assumed that all nodes have the same radio range. The nodes, then, execute a static device discovery technique. The discovery technique generates a new graph, in which an edge is set between two nodes if they discovered each other. Depending on the assumption taken by an algorithm, and depending on the discovery technique used, the generated graph may be a Unit Disk Graph (UDG),

a sub-graph of a UDG (such as the case of the networks generated by ALTERNATE), or a degree-bounded sub-graph of a UDG (such as the case of the networks generated by LIM-ALTERNATE).

Note that most of the static BSF algorithms that form out-degree constraint scatternets assume that the networks can be modeled by a UDG. However, the literature does not present any technique with the capability of generating deterministically a Unit Disk Graph – even in error free environments. It seems that generating a UDG would be possible only with the use with of a technology other than Bluetooth. Note that generating a UDG would be simpler in the case of single-channel radio systems – if an error-free environment was assumed. This is because for the Bluetooth devices to communicate with each other, they must be in the same frequency at the same time, contrary to single-channel radio systems where it is sufficient for the communicating devices to be in each other radio range at the time of the communication. At this point, we notify the reader that simplifying assumptions, such as UDG or error-free environments does not always have negative effect, but may help in a better understanding of the problem in hands. Some interesting ideas and algorithms appeared in the literature because of such simplifying assumptions.

After running the device discovery technique, static BSF algorithms model the network as a traditional wired network (in most cases). This is because each two communicating nodes use one single channel without a significant disturbing of their neighbors. Therefore, the network can be modeled as if there is no shared communication medium, such as in IEEE 802.11 networks and Ethernet-like networks, and each two communicating nodes have a separate link.

The exchange of messages between communicating nodes is done through PAGE messages. However, paging imposes a problem that is not found in traditional wired networks. The cost of PAGE message varies depending on the mechanism used to solve this problem. This problem is caused by the condition that for two nodes to establish a link one of them must be in the PAGE state and the other in the PAGE SCAN state at the same time. Not much attention is given to this problem in the literature of BSF algorithms, despite its importance. Few researchers gave solutions to

this problem (perhaps unintentionally sometimes). We surveyed the different solutions of this problem.

We called the first solution that we found for this problem as the *total neighborhood ordering mechanism*; used by BlueStars [12] and BlueMesh [39]. The largest node among its neighbors (i.e. those that their unique identifier is larger than all their neighbors) exchange messages with all its smaller neighbors, while the nodes that have larger neighbors wait until all of them send a message. A node that received messages from all its larger neighbors exchanges messages with all its smaller neighbors. If each node received a message from all its larger neighbors, and then exchanged messages with all its smaller neighbors, the exchange of messages can go in the opposite direction (i.e. the smallest node of a neighborhood exchanges messages with all its larger neighbors, while the nodes that have smaller neighbors wait for their messages). This process is called a *message exchange round*. Interestingly, the performance of BSF algorithms that utilize this mechanism can be compared *analytically* by the number of *message exchanges rounds* they need to complete the algorithm. However, not all static algorithms use this mechanism.

A second similar solution is to assume a *unique node* in the network and build an ordering of the nodes starting from it (see Bluetree [56] and ODBT [57] for example). The unique node serves as a root of a tree and is ranked *first*; its one-hop neighbors are ranked *second* and etc. The condition of having a unique node requires the execution of an election algorithm. The performance of an election algorithm over Bluetooth wireless networks was never studied.

The third solution, and perhaps the most common, is the *randomization*. A node that is expecting the receipt of a message goes to the PAGE SCAN state, while if a node wants to send a message then it alternates randomly between the PAGE and PAGE SCAN states. This solution can affect *negatively* the execution of BSF algorithms. For instance, Bluenet [28] is one of the algorithms that use this randomized solution. In [48], Bluenet was compared to the algorithms of [12], [56] and [49] which all do not use this randomized solution. One of the results of this

comparative study was that Bluenet has a relatively long execution time mainly because of this mechanism.

It should be noted that many BSF algorithms did not consider these problems that are introduced by the PAGE messages. Note also that it is possible sometimes to use the *total neighborhood ordering* mechanism in an algorithm instead of the randomization mechanism. In [58], we proposed three different implementations to the BlueMIS algorithm [14]. Two of them followed the *total neighborhood ordering* approach to deal with the problem switching between PAGE and PAGE SCAN states. The other implementation followed the randomized approach. It was found that the algorithm performs poorly using the randomized approach, compared to the other implementations. In fact, the algorithm sometimes did not terminate, even after a long period of time. However, making a BSF algorithm use the total neighborhood ordering mechanism may be impossible in some algorithms without major changes on their cores.

3.2.2. Communication models of dynamic BSF algorithms

Dynamic BSF algorithms assume initially that the nodes are distributed in a space, usually a 2-D plane. Over the vicinity graph, the nodes run a dynamic device discovery technique accompanied with a formation algorithm. Contrary to the static BSF algorithms, presenting the UDG in dynamic BSF algorithms is possible (under the assumption of an error-free environment). This is because dynamic BSF algorithms utilize a dynamic device discovery technique and because it is usually assumed in these algorithms that an inquirer will discover a scanner in a finite period of time. Recall that we argued that in the previous section that it is not possible, in a reasonable time, for a static BSF algorithm to assume a UDG given the currently available static device discovery techniques. Also note that dynamic BSF algorithms assume that an inquirer will discover a scanner in a finite time without specifying whether this time was reasonable or not.

Most dynamic BSF algorithms assume that the sole difference between this model and the traditional wired networks model is the need to employ a dynamic

device discovery technique. We believe, however, that there are some problems that most (if not all) dynamic BSF algorithms did not focus on. These problems will distinguish the model of dynamic BSF algorithms from other traditional networks models. A dynamic BSF algorithm deals with three types of messages. Each type has a different cost. These types are:

1. *Inquiry messages*: these are messages between two nodes that never discovered each other before. In fact, they are the most expensive and their cost depends mostly on the dynamic device discovery employed.
2. *Page messages:* these are messages between two nodes that know already the identity and clock value of each other. This is the same type of messages we find in the static BSF algorithms model.
3. *Intra-scatternet messages:* since dynamic algorithms build components and keep merging them and since components may be piconets or scatternets, then there is a need in some cases for two nodes in the same scatternet to exchange messages. The cost of these messages depends on the *inter-piconet scheduling* algorithms used and on the device discovery technique used. For instance, assume that two nodes (call them *x* and *y*) need to communicate with each other. There is only one path between these two nodes. If one of the nodes in this path was inquiring for or paging some of its neighbors, then the communication between node *x* and *y* is blocked until all the nodes of the path are free to foreword the messages of *x* and *y*. To solve this issue and similar ones, the nodes of the network need to use an efficient scheduling algorithm. It should be noted herein that the scheduling algorithms and formation algorithms were in most cases studied separately [8]. This is perhaps because of the difficulties of addressing the two problems at the same time.

We note that in both the dynamic and static BSF algorithms communication models, there are many types of messages each with a different cost. The cost of such messages does not depend on the link used (such as in [59], [60] and [104]), not on time (such as in [61]), but on the mechanisms that are used by the BSF algorithm.

Considering this fact, BSF algorithms designers need to avoid situations (or mechanisms) that may cause costly messages. Even more, this can help in the analysis and comparisons of the different BSF algorithms, as it can be possible sometimes to compute the number of messages of each type. Although the existence of different types of messages (i.e. with different costs) can complicate more the analysis of BSF algorithms, but considering such a fact can help, at least, in informal comparisons of BSF algorithms and not making simulation experiments as the only way for such a task. Note that we are focusing on the execution time of the algorithms which are the most critical weakness in BSF algorithms [62].

We presented in this section some observations that show that not all the problems and complexities introduced by the Bluetooth specifications were considered in the theoretical communication models used to study BSF algorithms. These observations and remarks can facilitate the establishing of a general model for studying Bluetooth wireless networks and can help in designing more efficient protocols.

3.2.3. A note on on-demand BSF algorithms

BSF algorithms are sometimes divided into two categories; reactive and proactive [54]. All the BSF algorithms that we talked about in this thesis so far are proactive algorithms. Reactive algorithms are also called on-demand BSF algorithms. These algorithms form a scatternet only when needed and then destroy it when there is no need for it. This leads as a consequence to a longer network life-time as the network's nodes save more energy by spending less effort in the maintenance of the scatternet links. This schema of categorization was used for routing protocols in wireless ad-hoc networks, and in fact, the idea of the reactive BSF algorithms is originally taken from the reactive routing protocols, e.g. AODV [12] and DSR [11], which find the route of a message at the time of transmitting it.

We analyze on-demand BSF algorithms and provide an example that shows the difference between the communications models used in Bluetooth wireless networks and traditional wireless networks. We also claim in this section that the

device discovery techniques used in the reactive BSF algorithms can be categorized under the static and dynamic discovery techniques, despite the different principle of work of reactive and proactive BSF algorithms.

Most of the on-demand BSF algorithms follow the same principle of work. If a node *x* wants to contact another node y, it generates a message announcing in it that it is looking for node *y*. This message is then flooded in the network until it is received by node *y* or until this message traverses "many" hops. If node *y* receives the message of *x* through node *z* (where node *z* can also be node *x* itself), then node *y* sends back an answer to *z* and this response message is then recursively transmitted until it reaches back node *x*. The formed scatternet will hence be formed from the nodes along the path from node *y* to node *x*. Different algorithms use different mechanisms to assign roles (i.e. master or slave, bridge or non-bridge) to the nodes of the scatternet, but these mechanisms are usually not complex.

However, many problems need to be addressed in order to implement this idea using Bluetooth devices. These problems are mainly raised by the specifications of Bluetooth. These problems are usually related to the device discovery and link establishment procedures in Bluetooth. One main problem that an on-demand BSF algorithm needs to address is the broadcasting problem. As it is already known, Bluetooth uses a connection-oriented communication mechanism. This means that for two nodes to communicate with each other, they must be either part of the same piconet or scatternet.

In general, the broadcasting in on-demand BSF algorithms is implemented using the following approach. A transmitter node starts inquiring for all its neighbors by residing in the INQUIRY state. After spending a specific time in this state, the node starts contacting separately each neighbor it discovered. However, a problem is introduced because of this residence in the INQUIRY state. This problem can be seen clearly in this following scenario. Assume a scenario in which we have four nodes, x_1, x_2, y_2 and y_1, all aligned on a line (see Figure 8 below). Assume that at the same moment, both x_1 and y_1 wants to contact each other. Node y_1 starts the execution of the algorithm and attempts to discover its neighbor y_2. Node x_1 does the same with x_2.

Now, both x_2 and y_2 go to the INQUIRY state to discover each other. Herein, unless nodes x_2 and y_2 alternate between the INQUIRY and INQUIRY SCAN states, the two nodes will not discover each other.

Figure 8: Problems in on-demand BSF algorithms I. This scenario shows one of the problems that on-demand BSF algorithms need to address. The circles represent the nodes; x_1, x_2, y_2 and y_1, while the lines represent the links between the nodes. If both y_1 and x_1 wanted to contact each other at the same time, then y_2 and x_2 may need to alternate between the INQUIRY and INQUIRY SCAN states for them to discover each other.

We note that Liu, Lee and Saadawi suggested in [9] different implementations for broadcasting in Bluetooth. It was assumed in their solution that no two transmitter nodes start establishing a route at the same time. Therefore, the scenario of Figure 8 that we just described cannot occur in their experiments. Also, we note that Zhang and Riley in [63] proposed an on-demand BSF algorithm that follows the same approach to implement the broadcasting. Their solution as well does not suffer from this problem. This is because it was assumed in their algorithm that there is one special node that it is the destination of all the other nodes and it cannot be a source at any time also.

The broadcasting mechanism that on-demand BSF algorithms use, which we discussed above, causes another problem as well. This problem can be seen clearly in the following scenario. Assume that we have a network such as the one shown in Figure 9. Assume that node *x* is the source while node *z* is the destination. Node *y* receives a message from *x* and forwards it to node *z*. Node y needs to listen to the response from *z*. Thus, it resides in the PAGE SCAN state. Node y needs also to listen to the inquiries of node *w*. Whence, it needs to alternate between the INQUIRY SCAN and PAGE SCAN states. Zhang and Riley in [63] were the first to consider this problem and to give a solution to it.

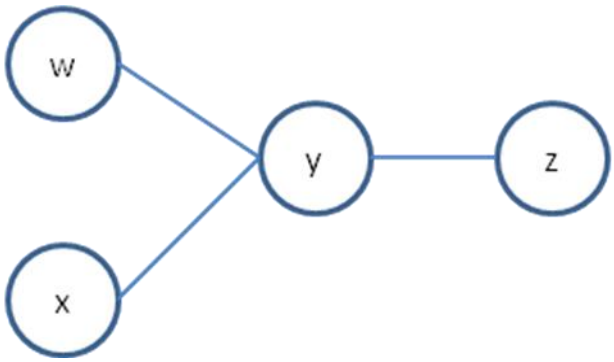

Figure 9: Problems in on-demand BSF algorithms II. This is another scenario that shows some problems the on-demand algorithms need to address. The circles represent nodes and the lines represent the links. If only x wanted to contact node z, then node y may need to alternate between INQUIRY SCAN and PAGE SCAN states.

We presented two scenarios that show the complexities of dealing with on-demand BSF algorithms. Some BSF algorithms proposed simplifying assumptions to solve them (such as the case of [9] and [63]), while some other algorithms did not concentrate on the execution time and thus did not address these problems in their algorithms (see [64] and [65] for example). We think that in order to solve these issues, the nodes should run an efficient scheduling algorithm that schedules the time of a node between the following states:

1. INQUIRY SCAN: to listen to undiscovered neighbors requesting communication through this node
2. PAGE SCAN: to listen to responses forwarded by this node
3. INQUIRY: to search for new neighbors
4. PAGE: contact neighbors that are already discovered. It can happen that a node does not go to the PAGE state but this will force it to inquire for its neighbors each time. Recall that the INQUIRY procedures are much more expensive than the PAGE procedures.

Describing these problems that we have just discussed, we wanted to show that the states of INQUIRY, INQUIRY SCAN, PAGE, PAGE SCAN states imposed new problems on on-demand BSF algorithms running over a Bluetooth wireless network that are not found in the other wireless networks. These states can degrade

the performance of an on-demand algorithm significantly, especially in term of the execution time of these algorithms. For instance, we find a significant difference between the execution times the distributed algorithms running over Bluetooth networks and those running over IEEE 802.11 networks – which are the most used to study wireless ad-hoc networking. We compare the results of two independent research papers. The first studied the execution time of two on-demand routing algorithms running over IEEE 802.11 networks. The second paper studied the performance of an on-demand BSF algorithm. All these algorithms use the same principle of work and depend heavily on broadcasting operations.

Perkins *et al.* compared in [10] the performance of two on-demand routing algorithms, AODV [12] and DSR [11], using IEEE 802.11 devices. They measured the *Average end-to-end delay of data packets* which is the time required for a route to be discovered. In one set of experiments, 50 nodes were distributed in a rectangular plane of 1500 m x 300 m area. The radio range was set to 250. The worst case of average delay was found to be 1.6 seconds approximately, setting 40 transmitter nodes at one time. In the case of 10 transmitter nodes, the average delay was less than 0.1 seconds.

We compare these results with those of [9] – which we already discussed in this section. The on-demand BSF algorithm of [9] has a similar idea to AODV and DSR. The authors executed their experiments on a network modeled as the one shown in Figure 10, and assumed in all experiments only one single transmitter (node 0 in Figure 10). The authors considered many implementations of their algorithm. The most efficient implementation in term of execution time, and does not require modifications on the specifications, was found to have an average end-to-end delay of 45 seconds (if the route length was set to 4 hops). If the route length was set to 7, then the average delay may reach 80 seconds and more. The authors also considered the scenario in which a node knows previously the identities and clocks of its neighbors. Note that in such a scenario, there is no need for the INQUIRY procedures. They found that the average delay was 10 seconds if the route length was set to 4 hops and may reach 15 seconds if the route was set to 7 hops. In the case of fixed nodes (i.e. no

mobility), such a scenario may be practical as the INQUIRY procedures are needed only once.

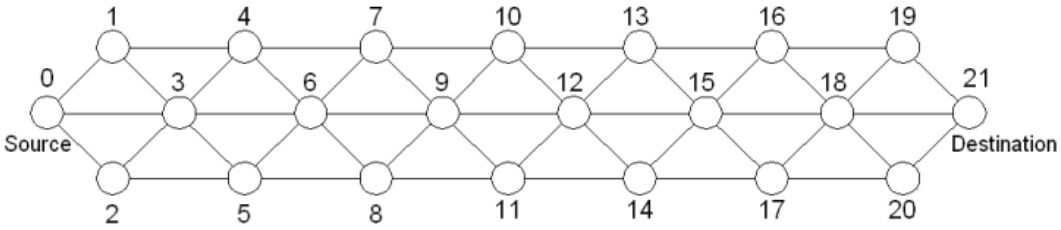

Figure 10: Network used for the experiments of [9] (Source [9]).

Although the two experiments were not conducted under the same environment, and therefore this comparison is not very accurate, but the difference is significant and shows that the execution time of on-demand BSF algorithms is relatively high. It should be noted, however, that Bluetooth networks perform better than IEEE 802.11 in term of other performance metrics, such as the network utilization (see [9] and references therein). This shows the benefits of Bluetooth that other wireless technologies do not provide. This is one example that shows the significant differences between Bluetooth wireless networks and other wireless networks. It shows also the significance of the problems that are introduced by the INQUIRY, INQUIRY SCAN, PAGE and PAGE SCAN states of Bluetooth[5].

At this point, the reader should note that on-demand BSF algorithms, sometimes, attempted to solve the problems that are introduced by the Bluetooth states by posing simplifying assumptions, but never proposed general-purpose solutions that work in all situations. However, one interesting general-purpose solution to these problems was introduced by Kawamoto, Wong and Leung in [53].Their idea is to build a control scatternet, and an on-demand routing protocol is then executed over this scatternet. The formation of the control scatternet starts by running a device discovery technique that is very similar to ALTERNATE. This means that the discovery of neighboring nodes, which is the most time consuming process, is executed only once during the execution of the algorithm. To solve issues

[5] The reader can find another comparative study of different on-demand routing algorithms and that addressed the execution time of such algorithms in reference [105].

similar to the ones we showed above (See Figure 8 and Figure 9), the authors introduced a new scheduling mechanism.

Anyway, note how similar the algorithm of [53] is to the proactive algorithms. However, it is usually considered among the reactive BSF algorithms. What is more interesting, though, is that this algorithm used a static device discovery technique to solve some of its issues, and it is very similar to ALTERNATE. Also, note that device discovery techniques that are employed in [9], [63] and others existing so far are similar to the naïve device discovery technique (i.e. one inquirer and all others are scanners).

Chapter 4

The Device Discovery Problem with Different Approaches

We observed from our work in the previous chapter that the communication models that are used currently to study BSF algorithms do not, in most of the cases, adequately model the Bluetooth wireless networks.

In this chapter, we survey studies that examined the device discovery procedures using an approach other than the one followed by BSF algorithms. The results of these studies help in understanding in more depth the Bluetooth wireless networks model.

These studies attempted in some cases to simplify the Bluetooth wireless networks model by either suggesting modifications on the Bluetooth specifications or by bypassing the Bluetooth device discovery procedures using a wireless technology other than Bluetooth. This survey includes also the results of some studies that formally analyzed the device discovery procedures, and some other studies that used hardware devices in their experiments. We start with the studies that suggested modifications on the Bluetooth specifications.

4.1. The Bluetooth specifications and suggested modifications

We start this section with a dynamic device discovery technique that we did not include in the previous chapter for the reason that it attempted, in contrary to the previously mentioned discovery techniques, to bypass a problem introduced by the device discovery procedures by suggesting modifications on the Bluetooth specifications.

This problem lies in the fact that one node must be in the PAGE state and another in the PAGE SCAN state at the same time in order to establish a link. All the previous techniques force a device in the INQUIRY SCAN to go directly to the PAGE SCAN as soon as it replies to an ID packets from an inquirer, and forces the inquirer, as well, to go to the PAGE state as soon as it receives a reply from a scanner[6]. Wang in [42] introduced a new way to synchronize the switching to the PAGE and PAGE SCAN states. However, this solution required modifications on the specifications of Bluetooth.

In this proposed solution, each node decides randomly and independently whether it should be an inquirer (i.e. in the INQUIRY state), or whether it should be a scanner (i.e. in the INQUIRY SCAN state). An inquirer broadcasts a new type of special packets that contain its address, its clock and the time at which it (i.e. the inquirer) will go to the PAGE SCAN state. When a scanner receives such information, it switches to the PAGE state at the time specified in the special packet, and attempts to contact the inquirer (which will be in the PAGE SCAN state at that time). When a connection between the two devices is established, the role of the devices is switched, so that the inquirer becomes the slave and the scanner becomes the master.

4.1.1. Suggestions for modifications on the Bluetooth specifications

The author of [42] was not the only one to suggest modifications on the Bluetooth specifications. In fact, suggestions to make modifications on the Bluetooth specifications are common in the literature. We list a number of similar studies focusing more on the ones that treated the specifications of Bluetooth device discovery.

4.1.1.1. Problems in the INQUIRY procedures

One of the attempts to modify the Bluetooth specifications in order to improve the device discovery procedures is the work of Zhang and Riley [63]. The authors

[6] For more information about the INQUIRY procedures, see Section 2.4, page 30 and Figure 2.

proposed a modification to the inquiry process in order to guarantee mutual (or symmetric) knowledge in a faster way. They presented a new type of special packets they called EID (Extended ID) packets. These packets contain the identity of the master and its clock. They are sent by the inquirers as soon as they receive an FHS packet from a scanner (i.e. a device in the INQUIRY SCAN). This step reduces the need to construct a temporary piconet to guarantee mutual knowledge between the inquirers and the scanners (as in the case of ALTERNATE and LIM-ALTERNATE for example). The authors of the paper tested the performance of LIM-ALTERNATE with the new modification they proposed. They showed that LIM-ALTERNATE gives better connectivity of the resulted networks. However, the average time needed for generating a connected network, which is more important in the context of this thesis, was not considered in their work. The same modification appeared also in another paper of the same authors [66].

Zhen and Kim in [67] made also some modifications on the ID packets. We have to revisit some details in the specifications in order to understand their modifications. According to the specifications, the inquirers follow a formula derived from their clock values to select the frequency hop that will be used to transmit the ID packets. The Bluetooth clock ticks each 312.5 μsec. Thus, following this formula, an inquirer uses a different frequency each 312.5 μsec. The inquirer sends two ID packets in two different frequencies in each time slot, which is set in the specifications to 625 μsec. After sending two packets in one time slot, the inquirer listens for the scanners responses for another time slot. However, in practice, a radio device requires a period of time to be able to switch from one frequency to another. In Bluetooth devices, this period of time is equal to 244.5 μsec, which is the difference between the time of a half slot (i.e. 625/2 = 312.5 μsec) and the time of the transmission of one ID packet, which is 68 μsec since 1 bit is sent in 1 μsec in Bluetooth. The modifications of the authors came from their observation that nowadays most of the radio devices are able to switch from one frequency to another in about 100. Therefore, they proposed to add 144 more bits to an ID packet to allow it to have more information. An interesting thing in this study, that we do not find in

many others, is that it checked the physical details of the radio devices to find whether the solution it proposed is feasible or not.

Liu, Lee and Saadawi used the proposed solution of [67] in an on-demand scatternet formation algorithm they introduced in [9]. Their algorithm requires a lot of broadcasting operations between neighboring devices. However, since Bluetooth is a connection-oriented communication technology (i.e. a link must be established between two devices for them to send messages to each other), the broadcast operations are a time-consuming. The authors argued that the inquiry procedure of Bluetooth can be useful for broadcasting relatively short messages if the extended ID packets of [67] were used, which we have just described. However, they found that they need more bits than the 212 bits allowed in [67] for their algorithm to work properly. Therefore, they suggested using two ID packets or more to separate the load of an inquiry packet. Let us assume that the length of the message that will be frequently broadcasted is 250 bits, which means that it will be separated in sequence of two of the ID packets introduced in [67]. The scanners replies to an inquirer only if they received all the messages of the sequence. Obviously, this modification makes the inquiry process a little longer in time. However, it lets the broadcast operation be much more efficient compared to the case where no modifications on the specifications were presented, as it is shown by the experiments that were conducted in the study.

Basagni, Bruno, Petrioli and Mambrini in [48] proposed to add to the standard ID packet the identity of its sender. According to the Bluetooth specifications, the address of a device is of 48 bit length. If we follow the solution of [9], which we just talked about above, then the addition of 48 bits will cause no problems. However, the authors of [48] preferred a solution in which the inquirer sends one ID packet in each time slot instead of two packets. The authors tested the impact of their suggested modification on the performance of ALTERNATE. They checked the percentage of generating a connected network against time. In the case of networks with low average nodal degree (less than 8), they found that their modifications made ALTERNATE perform worst compared to the specifications complaint

implementation. This is contradictory to the case of networks with high average nodal degree (about 28) where the modifications made ALTERNATE generate about 20% more connected networks after 10 seconds of simulation.

4.1.1.2. The back-off timer problem

The authors of [48] made also some modifications on the specifications. However this time, they attempted to give a solution to the back-off timer problem. To explain the back-off timer problem, we have to revisit the details of the link establishment procedures in Bluetooth. According to ver1.1 of the Bluetooth specifications, if a scanner device receives an ID packet from another inquirer device; the scanner has to backoff for a random amount of time called the back-off time. While backing off, a scanner is not able to perform any operation related to the inquiry procedures. After the back-off time terminates, the scanner goes to the INQUIRY RESPONSE state listens again to ID packets. As soon as it receives any ID packet, whether it was from the inquirer device that made it back off or not, the scanners answers back the ID packet with an FHS packet. As it has been mentioned previously, FHS packets are special packets that include the identity of their senders and the value of their senders' clock. After sending back an FHS packet, the scanner may go directly to the PAGE SCAN state. On the other side, a node that received the FHS packet may go to the PAGE state and starts establishing the link. Both the scanner and the inquirer may also stay at their current states; i.e. INQUIRY and INQUIRY SCAN states respectively.

The back-off timer, in the version 1.1 of Bluetooth, is used to prevent cases of collision at which more than one device receives an ID packet at the same time and from the same sender. In such cases, all the receivers send back their FHS packets at the same time leading to the collision of these packets. Nevertheless, the disadvantage of this timer is that it causes a link to be established in a relatively longer time. Also, the backing-off decreases the probability of a successful establishment of a link, because each scanner must receive two ID packets to start establishing a link or at least to announce its existence (by sending back the FHS

packet which holds its identity as well). For instance, it may happen that a scanner device goes to the INQUIRY SCAN state after being in the INQUIRY RESPONSE state, and there it may not receive any other ID packet, and thus, a link will not be established. Note that this scanner device may receive an ID packet from a node other than the one that made it go to the INQUIRY RESPONSE state, and may establish a link with it after that.

Basagni *et al.* in [48] suggested eliminating the back-off timer completely. They tested the performance of the ALTERNATE with the new modification. They found that the performance of ALTERNATE was significantly improved. Using the new modifications, they found that ALTERNATE generates about 35% (15%) more connected networks in 10 seconds (20 seconds) compared to specifications complaint implementation.

Welsh, Murphy and Frantz in [68] and [69] performed some experiments to investigate the device discovery phase procedures in Bluetooth. They suggested either reducing or even eliminating the back-off timer. They tested their modification under a simple scenario in which only one of the nodes is assigned the role of master (i.e. inquirer) and all the others are slaves (i.e. scanners). The authors of the paper were interested in the time required to receive at least one reply from any scanner. They found that eliminating the back-off timer would improve the performance of the experiments by about 70%, and it would improve more if only one single train was used (recall the trains, Train A and Train B, that are used during the INQUIRY procedures). However, it was found that the collision increases as the back-off timer value decreases. This is because the reduction of the back-off timer, instead of its complete elimination, keeps its benefits as packets collisions preventer and, at the same time, makes the link establishment procedure faster.

Zaruba and Chlamatac in [70] suggested changing the value of the back-off timer adaptively to the number of neighboring inquirers. Deciding the number of neighboring inquirers can be done with the use of some extra sensing hardware.

Similar suggestion was presented by Chakrabotary *et al.* in [71]. However, for these authors, it was preferred to reduce the back-off timer instead of completely

eliminate it. Their results were obtained by mathematically analyzing the problem with some accompanied simulation experiments. The authors also suggested using a single frequency hops train. Similarly, the suggestion of using a single frequency train can be found in [69] and [72]. A single train can have the meaning of either using the 32 frequencies altogether such as in [71], or simply using one of the two trains (i.e. Train A or Train B) always such as in [69].

In the work of Kardos and Vidacs [73], the authors first started with an analysis of the factors that they believe introduce problems in the device discovery and link establishment procedures. They listed four main factors: 1) the need to have one node in the INQUIRY and another in the INQUIRY SCAN state. 2) The inefficiency of energy consumption, since an inquirer has to send a large number of ID packets to discover other nodes. 3) The asymmetry of the inquiry process (i.e. the inquirer knows the identity of the scanner, while the scanner does not because of the ID and FHS packet), and 4) the separation of the INQUIRY and PAGE processes. The authors then presented a modification on the Bluetooth specifications to avoid the limitations caused from these factors. Their idea was to make the whole process of link establishment be done on one state only, called the RECON state. This made them to replace the ID and FHS packets with other type of packets that contain more bits. They used the device discovery technique of Salonidis *et al.* [36] as a benchmark. They showed a superiority of their solution – 300% faster with 20% less energy consumption. In our opinion, such results are very promising. However, the solution requires many changes on the modifications, thus more detailed study on the side-effects of the new introduced modifications would be necessary.

4.2. Bypassing the Bluetooth device discovery with other technologies

Some papers, such as [74][75][76][77] and [78], were interested in improving the performance of the device discovery in Bluetooth using other technologies or methods. Most of these papers focus on the problem of service discovery, and thus do not provide a lot of insight into the improvements they can bring to BSF

algorithms. These papers usually assume some specific scenarios that may be found in real-life and then designs a system that use Bluetooth and another wireless technology to solve some of the issues that the Bluetooth technology impose, such as the long time of the device discovery for example.

Salminen *et al.* in [77] considered the following scenario: there is a mobile node equipped with Bluetooth for radio communication and an RFID (Radio Frequency IDentification) tag. There are some other devices – such as printers, or any other device that offer a service – all equipped with Bluetooth for radio communication and an RFID transceiver; let us call them service devices. The RFID tag of the service devices are active (i.e. send some sort of information once a passive RFID tag pass in proximity of them), while the RFID tags of the mobile nodes are passive. If a mobile phone wants to use a service in proximity of it, it will not have to perform any type of standard Bluetooth device discovery, but instead, it will have to get close to the service device, and the service device's RFID tag then sends to the mobile node the necessary information to establish a link. This step shortens the device discovery procedure to milliseconds – since the paging usually is complete in milliseconds if the clock value of the slave (i.e. the node in PAGE SCAN) is known[7]. Woodings *et al.* [75] worked on a similar scenario but used the IrDA instead of RFID.

Note that IrDA requires that a line-of-sight between the two communicators be not interrupted, and thus, the discovery of unnecessary service devices will be avoided. The same case applies to the RFID case in Saliminen *et al.* [77] work just mentioned.

Hall *et al.* [78] proposed a different use of RFID to improve the device discovery procedure. They proposed to equip each device with both a Bluetooth radio and an RFID transceiver. An RFID transceiver can stimulate an RFID tag or transceiver to send back some information. The RFID transceivers are used to wake up other devices from their sleep mode, which is a step that saves energy. Also, the

7 This became a known fact as it is published in many papers and we have found it in our experiments too. The authors of [77], however, found that the paging process takes about 1.8 seconds. They *suspected* that this happened because of not knowing the clock value of the slave, but they did not fix the problem.

RFID transceivers are used to exchange the information necessary to bypass the inquiry procedure (i.e. the address and clock of the slaves). Note that RFID transceivers can have radio range of about 30 meters, and can exchange about 128 bits of information – which are sufficient to exchange the address of the slave and its clock. Unfortunately, there are no experiments for this study.

The most promising technology to solve the issues of Bluetooth device discovery is the IEEE 802.11b which is a single-channel radio standards (i.e. does not use a frequency hopping technique, but use only one frequency hop) and it is becoming as widespread as Bluetooth. However, IEEE 802.11b and Bluetooth both work in the same frequencies range (2.4 GHz – 2.4835 GHz), and this causes a problem of coexistence. Lansford *et al.* [79] showed, by experiments, that the coexistence of IEEE 802.11b and Bluetooth may cause significant degradation in the performance of both. Afterward, they mentioned some solutions to the problem. For instance, they suggested to make some modifications in the radio bands assignment regulations, and/or to make some modifications in the specifications of IEEE 802.11b, Bluetooth, or both. They also suggested the use of a radio communication technology, other than IEEE 802.11b, that uses a band different than the 2.4 GHz; such as the IEEE 802.11a or HiperLan2 for example. Yet, the authors mentioned some limitations in using such radio systems.

Details of the problem of coexistence will be omitted from this thesis since it requires good knowledge in the Physical layers of Bluetooth and IEEE 802.11. However, the Bluetooth Special Interest Group (SIG), the group responsible of developing the Bluetooth specifications, is working on using the powers of IEEE 802.11 for higher speed communication (since IEEE 802.11 has a faster data rate then Bluetooth). There is no intention, though, to use IEEE 802.11 for the device discovery [80].

Liberatore *et al.* in [62] proposed to use two Bluetooth devices instead of one. Such an approach changes the discovery from a half-duplex to full-duplex. Recall that an inquirer device cannot scan or perform any other communication task. Therefore, having two devices, one of them always inquiring and the other idle or scanning,

would make the discovery procedures full-duplex. The authors claim that this approach leads to a higher probability of new neighbor discovery and shorter time. They prove their claims with experiments and a formal analysis.

4.3. Studies with formal analysis

A number of papers deal with the Bluetooth device discovery from a probabilistic point of view. Such studies are useful since they lead to the knowledge of the real factors that affect the performance of the device discovery. Most of these studies tried to solve the following question: assume that we have two (or more) nodes, one (or more) in the INQUIRY state and another (or others) in the INQUIRY SCAN state but none of the nodes is alternating between the two states, as in the case of ALTERNATE for example. Then, how long should an inquirer be in the INQUIRY state to discover the other nodes?

The specifications recommend 10.24 seconds in error-free environments and an additional 10 seconds, if needed, in error prone environments. Peterson *et al.* [81] made a thorough mathematical analysis and some simulation experiments to prove that it is only 5.12 seconds that are required for an inquirer to discover 99% of all the scanning devices if using Bluetooth ver1.1. The number becomes 3.84 seconds if ver1.2 is used and 1.28 seconds if ver1.2 using the interlaced option of ver1.2 is used. However, they mentioned that if multiple inquirers were involved then a mathematical analysis would be very hard. This is due to the many random variables someone needs to deal with when considering the more complex device discovery techniques. There have been some attempts to understand more complex scenarios, e.g. [72] and [36], but they usually use simplifying assumptions.

Dufflot *et al.* [82] made a detailed formal analysis of the Bluetooth device discovery phase – assuming only one inquirer and one scanner. They used, for their study, an automated technique for the formal checking of systems that exhibits stochastic behavior; called probabilistic model checking. It is a method for calculating the likelihood of the occurrence of certain events during the execution of a system. The author of the paper argued that this method gives more accurate results than

discrete-event simulators. They used PRISM [82] as a tool to perform their study. However, it can be noted from what they have mentioned that modeling more complex systems with such exhaustive analytical models is hard, and requires a lot of computer resources. Such complexities and others are the reason why studying formally the performance of device discovery techniques (when taking in mind a big number of nodes) is considered a very hard task.

4.4. Studies with hardware implementations

Not many studies in the BSF domain considered experiments with hardware devices. We found only few of such studies ([84], [85] and [68]). The lack of such studies is mainly due to their high implementation cost; for instance, some BSF algorithms simulation studies use experiments with 100 nodes or more [39] [48] and hence it would be hard to perform and understand experiments with this large number of real devices. Even more, it would be hard, if hardware devices were used, to track the experiments (i.e. to know what actually happened during the experiments).

The study of [85] included some experiments on the performance of ALTERNATE using hardware devices. The study's experiments considered a scenario with only two nodes. This does not give much significance to the results of these experiments. Welsh *et al.* experiments in [68] included 5 devices; one of them was assigned the role of the inquirer and all the others were scanners. The study of [84] attempted to test the performance of the LIM-ALTERNATE device discovery technique using real hardware devices. This time, two types of experiments were conducted; each included 10 devices from a different manufacturer. It was found that the some optional features of the specifications (which are at the same time necessary to LIM-ALTERNATE) were not implemented by both manufacturers. This has led the authors of the paper [84] to use a different device discovery technique (although they considered it as a modification of LIM-ALTERNATE). The new technique idea is to let all the devices at first alternate between the INQUIRY and INQUIRY SCAN, and, there, each node collects the information of all (or most of) its direct neighbors. Then, after a timeout, each node alternates between the PAGE and PAGE SCAN and

attempts to contact the neighbors it discovered in the inquiry phase – not contacting more than 7 neighbors. Obviously, this is a different technique than LIM-ALTERNATE, and the results of these experiments do not, hence, reflect exactly the performance of LIM-ALTERNATE. Anyway, the main result the authors were focusing on in this paper is the call of Bluetooth chips manufacturers to include in their chips all the necessary features needed by Bluetooth Scatternet Formation (BSF) algorithms.

4.5. Comments and critics

During our revision to the studies that suggested modifications on the Bluetooth specifications (section 4.1.1), it took our attention that all of them considered Bluetooth ver1.1. In reality, four new Bluetooth versions appeared after ver1.1 at the time of writing this thesis. In these newer versions, the Bluetooth Special Interest Group (SIG), which is the group that supervises the development of the specifications, introduced some modifications on all parts of the specifications in order to keep the technology up-to-date. Among those, it introduced new ones on the procedures of device discovery and link establishment. The most interesting modifications on the procedures of device discovery were introduced in Bluetooth ver1.2. The modifications were a small change in the method that a scanner node (i.e. a node residing in the INQUIRY SCAN state) replies to an ID packet received by an inquirer node (i.e. residing in the INQUIRY state). This small change was found to have a substantial positive impact on static BSF algorithms, but not on dynamic BSF algorithms. These results were found from some experiments we conducted and present their details in the next chapter. These results, in fact, show that that the strengths and weaknesses of the Bluetooth specifications are still not all covered. In our opinion, we believe that the strengths of the Bluetooth specifications cannot be discovered without a deep study of the details of this technology (as all the studies surveyed in this chapter did).

In fact, most of the studies that we included in this chapter attempted to analyze in depth the specifications, using different approaches. These studies usually

started with an analysis of the factors that cause problems in the specifications of device discovery or link establishments, and then suggested solutions to overcome these problems. Therefore, these studies can help as well in understanding Bluetooth wireless networks.

However, it should be mentioned again that these studies were included in this chapter because they addressed the problems of device discovery not from the BSF algorithms point-of-view. Most of these studies used simple scenarios in their experiments. For example, Salminen *et al.* [77] assumed that the Bluetooth will be used as a communication mean of some devices that offer a service. These devices can be printers, projectors, displays, DVD, music players and others. A user, in their scenario, navigates in this service space and asks to use the desired service. Obviously, such scenario is not suitable for general-purpose BSF algorithms. We should note that some of the studies included in this chapter considered the ALTERNATE and LIM-ALTERNATE device discovery techniques, but none of such studies consider in details the effect of their solutions on BSF algorithms. One other observation we had during our work on this chapter is that there are no metrics that are repeatedly used to measure the performance of the solutions of these studies.

Chapter 5

Case Studies

In this chapter, we show examples of some features in the Bluetooth specifications that can have a large impact on BSF algorithms, and therefore, cannot be considered details in this context. We present the results of two case studies that are consisted of experiments and analyses to show some examples of the complexities of the Bluetooth specifications. We show how can simple changes in the Bluetooth specifications (or in their implementation) have a substantial impact on a large number of BSF algorithms. We first show a case where a small change in the specifications of Bluetooth can substantially improve the performance of the static BSF algorithms, but not dynamic BSF algorithms. The second case, however, shows how a small change in the implementation of the specifications substantially degrades the performance of the static BSF algorithms.

The analysis in the first case study can be used to understand more the communication models of static and dynamic BSF algorithms.

5.1. Case study 1: a small change, a substantial improvement

In this section, we study the impact of the modifications introduced in version 1.2 of the Bluetooth specifications on BSF algorithms. Up to now, the effect of these modifications on BSF algorithms has not been analyzed. We show that these modifications have a substantial positive impact on static BSF algorithms. We conducted some experiments on the ALTERNATE device discovery, which is the most commonly used in static BSF algorithms, and compared its performance using ver1.2 and ver1.1 (note that the latter version is used in most BSF studies). We found that this discovery technique, using ver1.2, generates a connected network about 3.5

times faster than it does using ver1.1. It also discovers about 20% more edges. Dynamic algorithms, on the other hand, are not affected by these modifications. This will be proven using some experiments and some analytical arguments.

Two new modifications on the device discovery procedures were presented in Bluetooth ver1.2; the first addressed the back-off timer and the other was the interlaced option scanning (see section 2.4). There are some studies that studied these modifications such as [81], [82] and [87], but nevertheless, their experiments addressed only simple scenarios that do not reflect the complexities of BSF algorithms and those of the device discovery techniques that they use. However, it should be noted that the effect of these modifications on BSF algorithms was not yet considered in the literature.

We analyze the effect of these modifications on the performance of common device discovery techniques and on the performance of the BSF algorithms; dynamic and static. Our experiments were conducted on three device discovery techniques; ALTERNATE, LIM-ALTERNATE and the discovery technique used in Law's algorithm [15] . In our opinion, those three techniques are sufficient to have an idea on all the other available discovery techniques used in BSF algorithms.

5.1.1. Why Bluetooth ver1.2 and not Bluetooth ver2.1 instead?

At this point, it is worth mentioning that the majority of the results of the studies related to Bluetooth Scatternets - whether concerned with the device discovery or BSF algorithms - were obtained using simulations of the Bluetooth ver1.1. However, the specifications of Bluetooth are continually updated to meet the requirements of nowadays. The current version is Bluetooth ver2.1. (Bluetooth ver3.0 appeared in April 2009.) More modifications related to the device discovery procedures than those we just mentioned were introduced in the versions of Bluetooth after ver1.1. One of these modifications is the Extended Inquiry Response (EID) option which was introduced in the Bluetooth ver2.1. With this option, a scanner device that receives an ID packet can send another message, called the EID message,

just after the sending of the FHS packet. A scanner can use this EID message to send additional information that may be useful to the inquirer[8].

However, we see that the two modifications of ver1.2 that we presented (i.e. the one related to the back-off timer and the interlaced scanning option) have more impact on Bluetooth Scatternet Formation (BSF) algorithms than all the other standardized modifications. Because of this, whenever we mention ver1.2 in this thesis, we mean with it ver1.2 and all the versions that appeared after it.

5.1.2. Experiments and results

In this section, we talk about our experiments and discuss the results we obtained. We started our experiments by testing the ALTERNATE discovery technique and compare its performance using Bluetooth ver1.1, Bluetooth ver1.2 and Bluetooth ver1.2 with the interlaced scanning option. We considered ALTERNATE for our experiments because it is the most widely used device discovery technique.

We surveyed the literature for Bluetooth networks simulators. Many simulators were found (such as [47], [89], [90], [91], [92], [93] and [48]); some of them do not meet our requirements while others are not freely and publically available. The simulator that can meet most of our requirements is UCBT [88]. In any case or the other, it is neither expected nor advised to use a simulator as it is without deep knowledge of its details and perhaps even some changes to its code [21]. We checked in detail most of the source code of UCBT and we made some modifications on it in some cases.

At first, we generated 250 connected Unit Disk Graphs (UDG). We divided the graphs into five equal groups depending on the number of nodes in them. We had graphs with 30,50,70,90 and 110 nodes; all distributed uniformly in a region of 30m x 30m. This is done to control the average nodal degree of the networks. For the parameters of ALTERNATE, we used the recommended values in [48]. T_{Disc}, the time to run the experiment, is set to 20 seconds. The period of time that a device spends in

[8] For more about the INQUIRY procedures, see section 2.4 (page 30) and Figure 2.

the INQUIRY state or in the INQUIRY SCAN state is uniformly drawn from the interval [T_{min}, T_{max}], where T_{min} was set to 0.01 seconds and T_{max} was set to 0.5 seconds.

We looked at first at the time required for the networks to be connected; which we called the connectivity time. Initially, we consider that there is no link between any two nodes in the network even if they were in range of each other. A link between two nodes is set each time they discover each other. A link, in this context, is bidirectional, used for analysis purposes, and does not mean that there is a physical relation of Master/Slave between the two nodes. In order to measure the connectivity time, we considered the median time for connectivity (i.e. the average of all the connectivity times of our experiments) and the average time at which 98% of the experiments generated a connected. We calculate the latter by removing the 2% experiments with the longest connectivity times, and then we calculate the maximum connectivity time among all the other 98% experiments. The results are shown in Table 1 and Table 2. ALTERNATE, using ver1.1, requires on average approximately 6.3 seconds to get the connectivity of 98% of the networks. On average, only 3.2 seconds are sufficient for ver1.1 to generate a connected network.

The superiority of ver1.2 (with and without the interlaced scanning option) is clear in Table 1. Bluetooth ver1.2 required on average only 3.4 seconds to make 98% of the networks connected, while if the interlaced scanning option is activated, then only 1.8 seconds are required for the same purpose. In other words, using the interlaced scanning option improves the performance of ALTERNATE by about 3.5 times! Note that the results of ALTERNATE running under ver1.1 match those published in [48]. In the following, we use ver1.2 for ver1.2 without interlaced scanning, while we use interlaced scanning to indicate ver1.2 with interlaced scanning.

Table 1: ALTERNATE - Results of the 98% connectivity time of the networks

Number of nodes	Ver1.1	Ver1.2	Interlaced
30	8.9	8.1	1.7
50	5.2	1.7	1.4
70	5.3	2.2	1.5
90	6.6	2.4	1.8
110	5.5	2.5	2.2
Average	6.3	3.4	1.8

Table 2: ALTERNATE - Results of the average connectivity times of the networks

Number of nodes	Ver1.1	Ver1.2	Interlaced
30	4.4	1.8	0.9
50	2.9	1.2	1.1
70	2.7	1.3	1.2
90	3.1	1.5	1.2
110	2.9	1.5	1.4
Average	3.2	1.5	1.2

Note that the connectivity times for the networks with 30 nodes in ver1.1 and ver1.2 are quite similar; 8.9 for ver1.1 and 8.1 for ver1.2. Note also that the difference between the two versions becomes clearer when the number of nodes is increased. This similarity in the case of 30 nodes and dissimilarity in the other cases happens because the number of critical links in the network (i.e. the links that cause network's disconnectivity if deleted) is increased as the average nodal degree of the network decreases.

We studied also the rate at which new links are being added to (i.e. discovered in) the networks. Figure 11 and Figure 12 show this rate for networks with 30 nodes and 110 nodes. It can be seen that ver1.2 is faster than ver1.1, and that that the interlaced scanning option makes the performance slightly better. This is because, in ver1.2, a node in the INQUIRY SCAN state starts establishing a link directly after the receipt of the first ID packet of an inquirer. In ver1.1 instead, when a node in the INQURY SCAN receives an ID packet, it must first wait for duration of time drawn at random, called the back-off timer, then after the back-off, it has to wait for the receipt of another ID packet to start establishing a link. Therefore, in ver1.2, the probability of a successful establishment of a link is increased and the time needed to establish is also decreased. As a consequence, this leads to the faster rate of links discovery. Note that the back-off timer still exists in ver1.2; however, when ALTERNATE is used with this version, then the back-off timer has no impact. In other words, if we conducted our experiments on ALTERNATE using Bluetooth ver1.1 and we set the backoff timer to be a random value between [0, 0] (i.e. always

zero), then we the performance of ALERNATE will be same using the two Bluetooth versions. Some researchers suggested the elimination of the back-off timer (see section 4.1.1.2); however, the reader should note that this is not the solution followed in Bluetooth ver1.2.

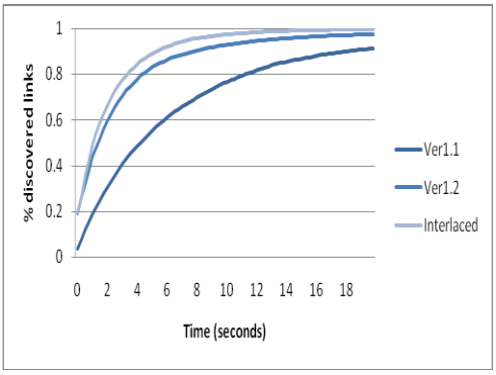

Figure 11: ALTERNATE - The percentage of discovered links in the networks over time (number of nodes = 30)

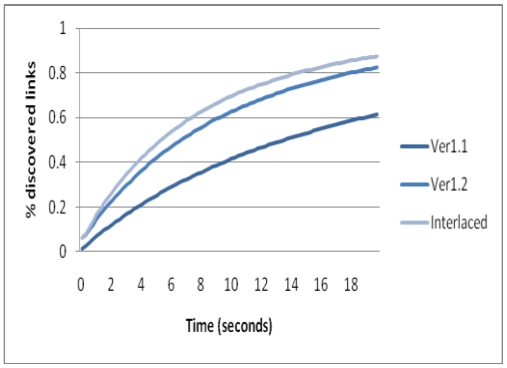

Figure 12: ALTERNATE - The percentage of discovered links in the networks over time (number of nodes = 110)

Using the interlaced scanning option, the rate of discovery is even faster than that of ver1.2. This can be seen clearer in the networks with 110 nodes. This occurs because an inquirer has a higher probability of discovering new neighbors when it uses the interlaced scanning option. With this option, an inquirer alternates between the two frequency's trains (i.e. Train A and Train B) compared to when this option is deactivated. The probability of discovery of new neighbors, if this option is activated, is close to that if the inquirer is inquiring in parallel in the two trains.

According to a number of researchers such as [94], the connectivity of uniformly distributed Unit Disk Graphs (UDG) is a function of the nodal degree of the graph. This is perhaps the explanation behind the short connectivity time of ALTERNATE using the new versions, since the fast rate at which new links are discovered in the networks implies a fast increasing rate for the average nodal degree in the networks. This means that most of the networks we tested using the interlaced scanning option had enough average nodal degree to guarantee connectivity after only 1.75 seconds on average; compared to 6 seconds on average if ver1.1 was used.

Note that after some time, almost no new links will be discovered and thus the percentage of discovered neighbors does not reach 100% (see Figure 11 and Figure 12). Thus, even with the new modifications on the specifications, it cannot be expected that a complete UDG can be discovered by using ALTERNATE only. However, a very high percentage of discovered links (near 100%) can be achieved – in networks with low nodal degree. This means that it may be possible to use some techniques beside ALTERNATE to get a UDG at a reasonable cost. Recall that the authors of [48] introduced a technique that is run after the termination of ALTERNATE to generate UDG's. Their solution, though, guarantees the generation of a UDG with statistical guarantees only.

This high percentage of links' discovery is also interesting because some BSF algorithms may try to work now with networks that are not exactly UDG's but are very close to it. In other words, it may be logical to design BSF algorithms that tolerate the inexistence of a UDG, and still generate efficient scatternets.

5.2. The impact of the new modifications of Bluetooth ver1.2 on BSF algorithms

We now evaluate the impact of the modifications of Bluetooth ver1.2 on BSF algorithms. We argue that static BSF algorithms, but not dynamic BSF algorithm, will be mainly and significantly affected by these modifications.

5.2.1. The case of static BSF algorithms

The actual execution time (T_{ae}) of this type of algorithms is the time spent in the device discovery phase (T_{dd}) plus the execution time (T_e) of the algorithm. For simplification of implementation and analysis, static BSF algorithms studies usually assume that all nodes are synchronized to start the device discovery phase at the same time, and then all the nodes agree on a timeout for the termination of this phase. This timeout is chosen in a way that guarantees, with high probability, the connectivity of the generated networks. Our experiments and those of [48] recommend that this timeout be set to 6 seconds if Bluetooth ver1.1 is used. But if Bluetooth ver1.2 is

used, our experiments show that this timeout may be reduced to 2 seconds or even to 1.77 seconds if the interlaced scanning option is activated. Because the required time for connectivity in ALTERNATE is significantly decreased using Bluetooth ver1.2, the timeout value will be lower, and hence, the device discovery time (T_{dd}) and the actual execution time (T_{ae}) of these algorithms will also be lower. Note that T_{ae} ranges between 1 second and 18 seconds on average, while T_{dd} is between 6 to 10 seconds using ver1.1 and, if ver1.2 is used, its values become between 1.8 – 2.85 (i.e. decrease by about 3.5 times as our experiments showed).

At this point, a note should be mentioned about the execution time (T_e) of the static BSF algorithms. Basagni *et al.* showed in [48] that, when they published their work, the fastest available BSF algorithm was BlueStars [12]. This algorithm is one of the simplest BSF algorithms. We briefly described how BlueStars work in Chapter 2. The execution time of BlueStars is dependent on the number of links in the network. Observing this fact, Dubashi *et al.* [34] introduced a new algorithm called BluePleidas. This algorithm is actually the BlueStars algorithm running over a networks generated by a newer version of ALTERNATE. We called this newer version LIM-ALTERNATE. The idea of LIM-ALTERNATE is to let the nodes of the network discover or be discovered by at most *c* neighbours (where *c* is 5, 6 or 7). Whence, it reduces the number of links in the network, and guarantees the generation of out-degree constrained scatternets as well. The authors of [34] claim that BluePleidas is the fastest static BSF algorithm. The fast execution time is achieved with a nice trade-off with other scatternets properties, such as the average shortest path and the probability of disconnectivity of the generated networks.

We conducted some experiments on LIM-ALTERNATE to study it in more depth, since it can be considered the core of the BluePleidas algorithm. More details about the experiments are given in the next section.

5.2.1.1. LIM-ALTERNATE Experiments

We made some experiments on LIM-ALTERNATE. We compared its performance against ALTERNATE. We found that, in term of the time required to

generate a connected network, there is no significant difference between the two techniques. But in term of other performance metrics, ALTERNATE was superior.

Our experiments were executed using the same simulation environment used for the experiments of ALTERNATE. The value *c* was set to 7 (i.e. the maximum degree of any node is not allowed to exceed 7). As with ALTERNATE, we generated 250 UDG networks grouped into five groups with different number of nodes (30, 50, 70, 90 and 110). All networks were connected Unit Disk Graphs.

We were concerned at first with the time required to generate a connected topology. We ran our experiments for 20 seconds. Then, we searched, in the results, for the time at which the networks became connected. As in the experiments of ALTERNATE, we considered the median time for connectivity (i.e. the average of all the connectivity times of our experiments) and the average time at which 98% of the experiments generated a connected. We calculate the latter by removing the 2% experiments with the longest connectivity time, and then calculate the maximum connectivity time among all the other 98% experiments. Table 3 and Table 4 show the results.

Table 3: LIM-ALTERNATE - The time at which 98% of the networks become connected

	Version 1.1		**Version 1.2**		**Version 1.2 Interlaced**	
	ALT	**LIM-ALT**	**ALT**	**LIM-ALT**	**ALT**	**LIM-ALT**
30	8.714	13.337	8.0876	5.0866	1.5017	2.3803
50	5.646	7.0688	1.7138	2.487	1.378	1.3878
70	5.198	4.9822	2.2474	1.729	1.4213	1.7264
90	5.606	4.5129	2.2825	2.0606	1.787	1.8415
110	3.760	6.7061	2.4905	2.1071	2.2074	2.1822

Table 4: LIM-ALTERNATE - The average (median) time for connectivity in all networks

	Version 1.1		**Version 1.2**		**Version 1.2 Interlaced**	
	ALT	**LIM-ALT**	**ALT**	**LIM-ALT**	**ALT**	**LIM-ALT**
30	4.511	3.26563	1.0839	1.1481	0.9707	1.015
50	2.8853	2.5664	1.1039	1.164	1.0769	1.1014
70	2.6133	2.5678	1.3061	1.161	1.1248	1.1329
90	2.4524	2.5269	1.4376	1.1681	1.1521	1.1443
110	2.6546	2.7563	1.4382	1.2188	1.28	1.3131

As it can be noticed from Table 3 and Table 4, LIM-ALTRENATE is affected by the new modifications in the same way as the ALTERNATE. In other words, there is no significant statistical difference between the two techniques in terms of the connectivity times. However, that does not mean that LIM-ALTERENATE is better (since it has similar connectivity time with the additional feature of generating degree-bounded networks). Observing the number of disconnected graphs generated by both techniques and all Bluetooth versions (see Table 5), we find that LIM-ALTERNATE generates less connected networks (although not significantly). These experiments, however, do not reflect the real percentage of generating disconnected networks by both techniques. It would be better to conduct more experiments, and test the performance of both techniques under nodal distributions other than the uniform distribution.

Table 5: Difference between ALTERNATE and LIM-ALTERNATE in the number of the generated networks that were disconnected (out of 50 experiments)

	Version 1.1		**Version 1.2**		**Version 1.2 Interlaced**	
	ALT	**LIM-ALT**	**ALT**	**LIM-ALT**	**ALT**	**LIM-ALT**
30	2/50	4/50	1/50	3/50	0/50	1/50
50	0	1	0	1	0	0
70	0	0	1	0	0	0
90	0	0	0	0	0	0
110	0	1	0	0	0	0

Nonetheless, if we study the *ratio of original/generated networks average shortest path* with both techniques, we see that ALTERNATE is significantly superior over LIM-ALTERNATE. We mean by original networks those formed from the Unit Disk Graph, while we mean by the generated networks those that were generated by the discovery techniques. Networks generated by LIM-ALTERNATE have lower average shortest paths since LIM-ALTERNATE restricts each node to have a constant degree value. In our experiments, the degree is not allowed to be more than 7. The average shortest would have been lower if we restricted the nodal degree to 5 or 6. Figure 13 and Figure 14 depict the difference between the networks generated by both techniques in term of the ratio of the average shortest paths

between the original networks and the generated networks. As it can be seen from Figure 13, the average shortest path, in the case of LIM-ALTERNATE, is decreased as the number of nodes increases. This is because the average nodal degree in the networks of LIM-ALTERNATE is constant while it is not in the case of ALTERNATE. This is considered as a disadvantage of LIM-ALTERNATE, and perhaps it is the only major disadvantage of LIM-ALTERNATE and BluePleidas.

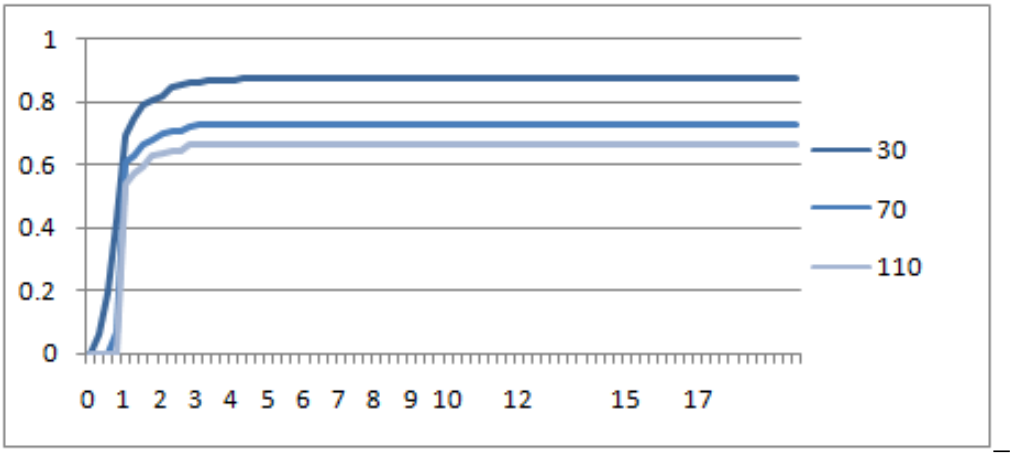

Figure 13: LIM-ALTERNATE - Ratio of Average Shortest Path between the original and the generated networks; using Bluetooth ver1.2 with the Interlaced Scan activated.

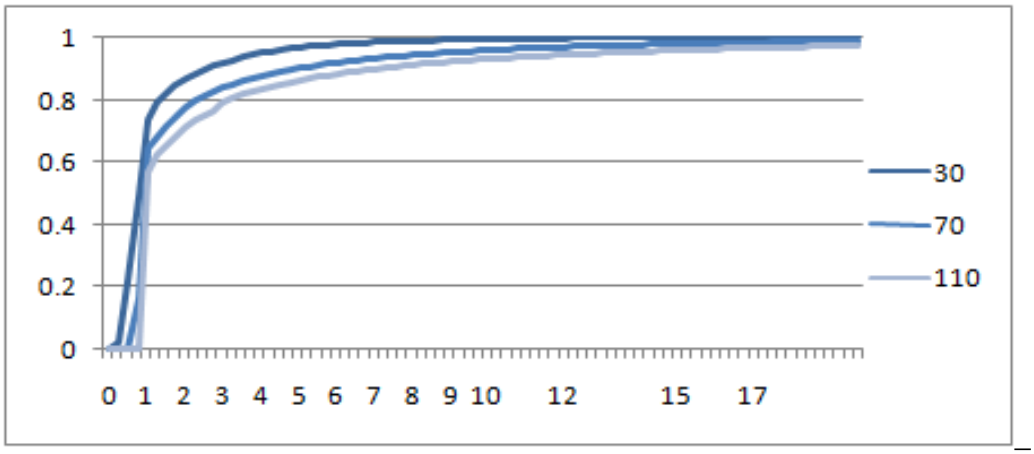

Figure 14: ALTERNATE - Ratio of Average Shortest Path between the original and the generated networks; using Bluetooth ver1.2 with the Interlaced Scan activated.

5.2.2. The case of dynamic BSF algorithms

We have seen in Section 5.1.2 that the faster connectivity time of ALTERNATE in ver1.2, and consequently the improved execution time of static BSF

algorithms, is mainly caused by the faster rate of new links' discovery in the networks. However, we believe that this does not cause improved execution time for the currently available dynamic BSF algorithms. This is either because 1) the probability of discovering new neighbors in the dynamic BSF algorithms will not be affected by the new modifications or 2) because the device discovery part (in some of these algorithms) is not the most dominant part in deciding their execution times. Again, the reader is reminded at this point that the modifications that we mentioned in this thesis are the only ones we found impacting BSF algorithms in all the versions after ver1.1 (see section 5.1.1).

We will show how that analyzing the different device discovery techniques used in BSF algorithms is sufficient to study the impact of the modifications of Bluetooth ver1.2 on dynamic BSF algorithms. The analysis that will be giving can be considered as an application of the categorization schema of static and dynamic algorithms that we used throughout this thesis. We start with a brief revision of dynamic device discovery and BSF algorithms, and then we consider separately each device discovery technique. The reader should note that we are considering only the currently available BSF algorithms, and therefore, we do not consider the BlueTag [52] technique because it is not used in any BSF algorithm.

Dynamic BSF algorithms are based on the idea of the merging of components. A component is defined as a set of connected nodes. A component can be an isolated node, a piconet or a scatternet. The components are dynamically merged together until a large scatternet that contains all the nodes is obtained. Each node divides its time between 1) intra-component communication and 2) discovering new neighboring devices. Dynamic BFS algorithms use different methods to make components discover new neighbors. Three methods exist in the currently available BSF algorithms:

1. Make only specific node(s) in a component perform the discovery procedures; as in the case of [15] and [51],

2. Make all the nodes of a component successively perform the discovery procedures; as in [40] and [43],

3. Make all the nodes of a component independently perform the discovery procedures; as in SHAPER [29] and TSF [32].

Case 1) only specific node(s) in a component perform the discovery procedures

This dynamic discovery technique was used as it is in both Law's *et al.* [15] algorithm and in [51], so this analysis applies to these algorithms and other similar ones. Therefore, we consider only Law's algorithm [15]. We want to show that the probability of discovering new neighbors in Law's algorithm will not be affected by the modifications of Bluetooth. The idea of Law's algorithm is to fill the piconets of the scatternet with as many slaves as possible. Each component has one leader. Depending on random variable of its own, a leader either goes to the INQUIRY state to seek new neighbors, or it assigns one of its slaves to scan for new neighbors. This means that a leader must be a master. In case the leader was a master with no slaves, then it may itself scans for neighboring devices.

Accordingly, each component delegates only one of its member nodes to seek or scan for new neighboring nodes. It is a condition that a delegated node be either a master or a slave, and not an S/S bridge. Each time two nodes of different components discover each other, they perform an action to merge the two components. In Law's algorithm [15], the merging action depends on the size of the piconets of the two delegated nodes.

Because in Law's algorithm [15] only one node in a component runs the discovery procedures at a time, the number of active discoverer nodes, which are the nodes that are in the INQUIRY or in the INQUIRY SCAN states, will be decreased with time, and as a consequence, the probability of discovering new neighbors will be decreased as well with time. Note that a low number of discoverers leads to a low degree of the discoverers nodes which leads also to a lower probability of discovering neighbors. Thus, the decrease of the number of active discoverers in the network as the algorithm is being executed is the dominant factor in deciding the probability of

discovering new neighbors, and as a consequence to that, the algorithm performance will not be much affected by Bluetooth ver1.2 faster rate of discovering new links.

We made some experiment to back up our analysis. In these experiments, we ran Law's algorithm using Bluetooth ver1.1, Bluetooth ver1.2 and Bluetooth ver1.2 with interlaced scanning. We focused on the execution time of the algorithm under the three versions. Because Law's algorithm is a dynamic BSF algorithm, the term of execution time is a little ambiguous. For us, the algorithm terminates at the time of the last successful merge action. Or in other words, the algorithm terminates at the time at which the last discovery between two nodes happens.

We found that there is no significant impact of the proposed modifications of Bluetooth ver1.2 on Law's algorithm execution time. The results of the experiments indeed were found to match the results of our analysis. The experiments we made were run using the UCBT simulator. The implementation of the algorithm was done by Wang and was used in the author's PhD thesis [42]. We cancelled the effect of noise in these experiments to focus more on the impact of Bluetooth ver1.2 modifications. We generated 200 UDG networks. The networks were grouped into four equally-sized sets, depending on the number of nodes (30, 40, 50 and 60 nodes). All the nodes were in range of each other, since this algorithm is designed to work in single-hop scenarios only. The results of the experiments are shown in Table 6.

Table 6: The average execution time of Law's algorithm against the number of nodes. The execution time is measured in seconds.

Number of Nodes	Ver1.1	Ver1.2	Interlaced Scanning
30	40.943	40.684	43.619
40	43.242	43.194	46.125
50	41.783	47.615	50.32
60	44.299	52.127	51.013

Case 2) All the nodes of a component successively perform the discovery procedures

BTSpin [40], which is another dynamic BSF algorithm, can be analyzed similarly to Law's algorithm [15]. The two algorithms use different types of discovery techniques, but both techniques can be analyzed similarly. We want to

show, as we did in the Law's algorithm case, that the shorter time of a link establishment process, which led to the faster rate new links discovery, which is also the main reason behind the improvement of ALTERNATE, has no effect on the time execution of BTSpin [40].

In BTSpin, a component is defined as a connected set of nodes that form either an isolated node or a piconet. In contrary to other BSF algorithms, a component in BTSpin cannot be a scatternet. The masters of the components ask each one of their slaves to perform a spin one at a time. A spin is defined as an inquiry followed by a scan (i.e. inquiry scan). Thus, the number of the discoverer nodes in the network at a time is equal to the number of piconets at the same time. If this number is large and close to the number of nodes in the network, then we may expect the algorithm to be affected by the modifications of ver1.2. However, on average, the number of discoverer nodes is low, and as a result, a faster link establishment will not have much impact on the performance of BTSpin. Whence**,** the time to establish a link will not be a dominant part in determining the performance of the discovery technique used in this algorithm**,** and therefore, the faster rate of links' discovery obtained by Bluetooth ver1.2 has no large impact on the algorithm execution time. This argument works also for all the other dynamic BSF algorithms that use this device discovery technique.

Case 3) All the nodes of a component independently perform the discovery procedures

We want to show that the device discovery part is not the most dominant part in deciding the execution times of the dynamic BSF algorithms that use this discovery technique. Before starting our analysis, it is worth mentioning that that the inquiry procedures are usually completed in the order of seconds, while the paging procedures are completed usually in the order of milliseconds However, the inquiry procedures do not always dominate the execution time of a BSF algorithm. This is seen clearer in the case of static BSF algorithms, since the nodes in these algorithms *inquire* for their neighbors only at the device discovery phase. For more clarity, let us consider the following example. Basagni *et al.* [48] studied and compared four major

static BSF algorithms, which are BlueStars [12], Bluenet [28], Bluetrees [56] and LSBS [49]. The authors of the paper found that the BSF algorithm execution times may be in the range of 0.5 seconds to 18 seconds. Note that not a single inquiry is required during the execution of these algorithms; but only pagings. In this comparative study, Bluenet [28] was found to be a relatively slow algorithm simply because of the techniques it uses for the communication between the nodes. This was related mainly to the fact that one node must be in the PAGE state and another in the PAGE SCAN state at the same time to establish a link and how BlueNet solved this issue.

We will see next how two major dynamic BSF algorithms, SHAPER [29] and TSF [32], have their execution times dependant mainly on the techniques they use for the communication between the nodes. Thus, Bluetooth ver1.2 still does not have a major impact on these two algorithms.

TSF works in single-hop scenarios while SHAPER works in single-hop and multi-hop scenarios. Both algorithms are similar, and in fact, SHAPER is a modified version of TSF. Both algorithms dynamically construct a tree-like scatternet. As in all other BSF dynamic algorithms, a node in these two algorithms divides its time between 1) discovering new neighbors to continue the formation of the scatternet (the FORM state) and 2) communicating with the nodes in the already-built scatternet (the COMM state). A node spends a random time T_{form} in the first state and a random time T_{comm} in the second state. In the FORM state, a node alternates between the INQUIRY and the INQUIRY SCAN states searching for new neighbors.

In TSF and SHAPER, all the nodes independently alternate between the FORM and the COMM states. Note that if T_{comm} was set to zero, then these algorithms will be affected by the new modifications of ver1.2 as ALTERNATE does. Things are not that simple however. We show that the hidden complexities of TSF and SHAPER have a great impact on their execution time and whence the device discovery technique that they use is not affected by the modifications of Bluetooth ver1.2.

Three types of nodes exist in TSF and SHAPER; free nodes, which are nodes that do not belong to any tree, root nodes, which are the roots of their trees, and the non-root nodes, which are nodes that are not roots of their trees, The algorithm of TSF works as follows; If a free node discovers another free node, then one of them is elected as leader and it becomes then the root of a new constructed tree. If a free node discovers a non-root node, it becomes its slave. If a free node discovers a root node, then it destroys the connection with it and does not make any merging function. Root nodes are only allowed to merge with other roots. Note that non-root nodes are not allowed to merge except with free nodes, and root nodes are also not allowed to merge except with other root nodes. This means that the probability of having a successful link is decreased rapidly with time, becoming then the dominant part in analyzing the execution time of TSF. In fact, the probability of successful discovery will decrease *exponentially* since the number of discoverer nodes also decreases exponentially.

Note that the merging decision in TSF is taken locally by the two nodes that discovered each other. In other words, they do not need to exchange messages with other nodes. This is, in fact, a useful property in TSF that we do not find in SHAPER. This will be explained next.

SHAPER allows the non-root nodes to make connections with free and non-free nodes. This gives the algorithm more complexities, but it makes the algorithm work in multi-hop scenarios. Note that TSF does not work in single-hop scenarios. Because of this additional modifications to TSF, SHAPER needs, whenever two trees are merged, to inform some of the nodes in the merged components about the merge decision, such as the root of tree for example. This makes the merging decision no longer local. This means that these messages exchanged in the case of merging will need to be postponed sometimes, as in the case when one of the nodes that need to be notified is residing in the INQUIRY or PAGE state since a node residing in these two states cannot be interrupted externally. All the other nodes that are on the path to these nodes can also postpone the message. This postponed time can be considered as a penalty. In fact, this time can be in the order of seconds since a node may reside in

the INQUIRY state for seconds in order to make the inquiry efficient (see Section VII of [29]). The non-local merging decision and the postponed time that results from it and may take up to seconds in some cases are two factors that impose a costly penalty on the execution time of SHAPER. In fact, the non-local merging decision shows that T_{comm} in SHAPER cannot be set to zero, and hence it cannot be treated as ALTERNATE. These factors show again that the techniques that SHAPER utilizes have more impact on its execution time. In other words, SHAPER is not affected by the modifications of Bluetooth ver1.2 as ALTERNATE does.

The four algorithms we have just analyzed, Law's algorithm [15], TSF [32], SHAPER [29] and BTSpin [40] are considered as the major dynamic BSF algorithms in the literature. We showed in Chapter 2 that their device discovery techniques are the only discovery techniques used in the currently available dynamic BSF algorithms, and hence what apply to them apply also to the other dynamic BSF algorithms. The reader is notified herein that there are very few dynamic BSF algorithms available in the literature. The dynamic device discovery technique of BlueTag [52] was not considered in this section because it is not used by any BSF algorithm. Thus, as we claim that these BSF algorithms are all not affected by the modifications of Bluetooth ver1.2, we can conclude that all the available dynamic BSF algorithms are also not affected by the same modifications.

Note that, in most dynamic BSF algorithms, it was not the specifications of Bluetooth that mostly affected the execution times of these algorithms, but instead it was the techniques that these algorithms use. In our opinion, we consider this as a weakness in dynamic BSF algorithms, and whence we believe that more research work is still needed on this type of algorithms.

5.3. Case study 2: a small change, a substantial degradation

We showed in the first case study how a simple change on the Bluetooth specifications can substantially improve a wide range of BSF algorithms, mainly in term of their execution times. Our arguments were backed with a set of experiments and analyses. In this case study, we show how a minor change on the implementation

of the specifications of Bluetooth can substantially degrade the performance of the ALTERNATE device discovery technique, and as a consequence, degrade the performance of the static BSF algorithms. Again, we back our arguments with a set of experiments and analyses. From all these experiments and analyses, we want to show the complexities of the Bluetooth specifications.

This small detail that we will be discussing was found by chance during our work with ALTERNATE. It is related to the frequency hops trains used during the INQUIRY state (see section 2.4.1]). The specifications of Bluetooth (in all its versions) state that Train A must always be the first train used in any new PAGE procedure. The specifications also states that it "does not matter" which train a device start with at the beginning of a new INQUIRY procedure. In other words, we must start the PAGE hopping sequence in Train A, but we can start the INQUIRY in either Train A or Train B (see page 257 and page 258 in [4]).

We found that the UCBT simulator [42], which is the one that we used for our experiments, starts each INQUIRY procedure with Train A. This was some kind of a simplification of the source code to make the INQUIRY and the PAGE procedures implementations as similar as possible. This implementation is legal according the Bluetooth specifications. However, this way of implementing the INQUIRY procedure made the performance of ALTERNATE degrades significantly.

If we restrict ourselves with starting each new INQUIRY procedure with Train A as in the case of PAGE procedures, then we will have a surprising degradation in the performance of ALTERNATE. Following this implementation of the INQUIRY procedure, we found in our experiments that ALTERNATE will not be able to discover more than 60% of the links in a network even in a very long time of simulation (40 seconds)[9] and even in error-free environments!

If such a detail was ignored or missed, or if the actual performance of ALTERNATE was similar to this, then many major static BSF algorithms available in literature would be based over a weak foundation. This is because most static BSF algorithms use ALTERNATE as their device discovery technique.

[9] This result was obtained from experiments that we did not publish in this thesis.

Let us assume that ALTERNATE performance is actually similar to that, there will be two consequences of such a bad performance. Firstly, discovering at most 60% of the links leads to a higher probability of generating disconnected networks, especially in networks with low average nodal degree (between 6 and 8 degrees/node). Both ALTERNATE and LIM-ALTERNATE will be affected in the same way. Secondly, most static BSF algorithms, with the exception of BluePleidas [34], assume the Unit Disk Graph (UDG) model when they deal with the problem of the formation of degree-constrained scatternets. Obviously, discovering at most 60 % of the links makes it not practical to assume that ALTERNATE may generate networks that are, at least, similar to Unit Disk Graphs (UDG), even with the use of other techniques such as the replenish technique that was introduced in [48] (see section 3.1.1).

Fortunately, ALTERNATE does not have this bad performance. However, we had to make some changes on the implementation of the UCBT simulator that caused this problem. In our implementation, we kept the counter that controls the switching of the trains always running. Recall that the selection of a train is controlled by a variable that has two values only. This variable is controlled by a counter that toggles its value each 2.56 seconds (See the k variable of equation (1) – that was described in section 2.4.1). Following our implementation, if the inquiry procedures were executed continuously, as in the case of ALTERNATE, then the inquiry starts in most cases with the train that was last used in the previous inquiry, as long as the counter that controls the selection of the trains was not changed by the timeout. In the cases where the inquiry procedures happen in a relatively long time after one another, the selected train will be kind of randomly chosen. Note that neither our implementation nor the implementation of UCBT, which caused the bad performance, is illegal according the Bluetooth specifications.

5.3.1. Experiments and results

In our experiments, we generated 100 connected UDG networks. We grouped the networks into two equally sized sets depending on the number of nodes which

were 30 and 110. We used Bluetooth ver1.2 for our experiments. Figure 15 shows a comparison between the rates of discovering new links using the two implementations; in the first implementation, we restricted ourselves with starting each new INQUIRY with Train A. In the following, we call this implementation as the *restricted case*. The second implementation relaxes the restriction that is imposed in the first implementation. We kept the counter that controls the switching of the **k** variable (i.e. the value of the train) always running. We call the second implementation as the *non-restricted case*. Note that all the experiments that we conducted on ALTERNATE in section 5.1.2 (i.e. the first case study) were obtained with the *non-restricted case*.

Note that the percentage of discovered links in the experiments with 30 nodes in the *restricted case* did not exceed 60%. Note also that the same percentage does not increase much after approximately 12 seconds. On the other hand, in the experiments of the *non-restricted case*, we see that rate of discovering new links in the networks with 30 nodes reached approximately 100%. Thus, such a small change in the implementation of ALTERNATE could degrade its performance, in term of the discovery of new links, by about 40%!

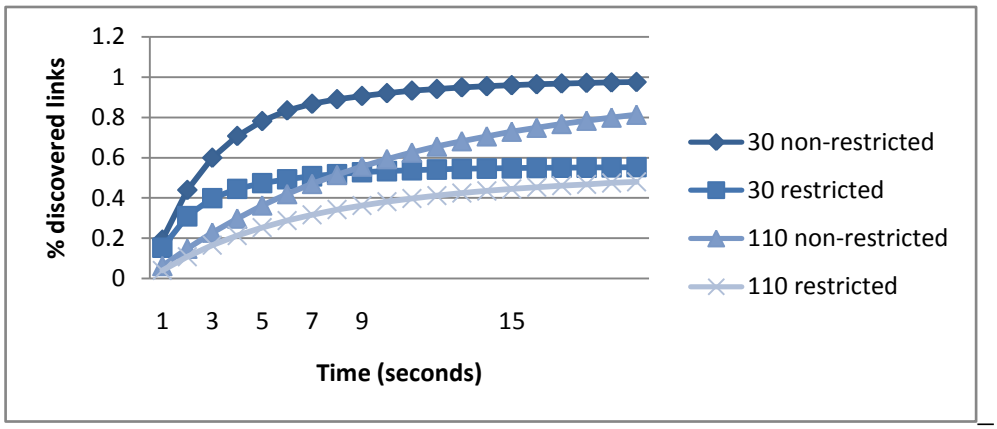

Figure 15: Impact of using Train A as the initial train in each new INQUIRY. The case of networks with 30 and 110 nodes (*restricted* means that Train A is always used as the first train). These results were obtained using Bluetooth ver1.2.

We obtained other results that show us the sensitivity of dealing with Bluetooth. We studied the percentage of generating connected networks. We show

such percentages using the *restricted case* under Bluetooth ver1.2 and Bluetooth ver1.1 respectively. As in the *non-restricted* case, the time required to generate a connected network is shorter in the case of Bluetooth ver1.2. However, the number of the connected networks was much lower. It is much clearer in the experiments with 30 nodes. In both the *restricted* and *non-restricted* cases, the percentage did not exceed 54%, compared to approximately 98% in the non-restricted case.

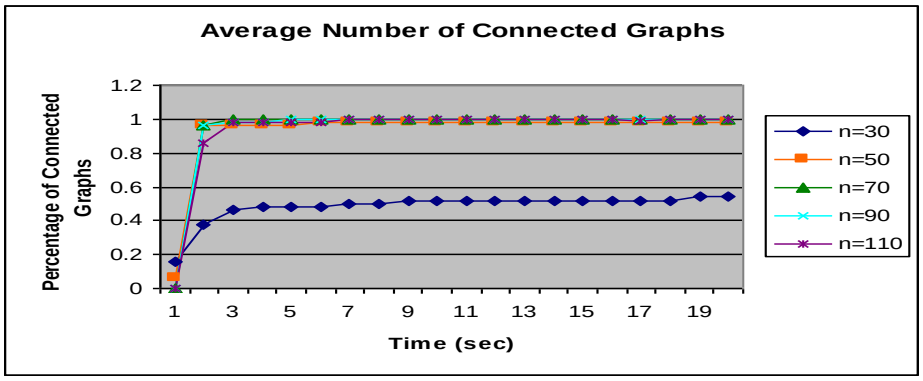

Figure 16: Bluetooth ver1.2 percentage of connected networks – Restricted Implementation

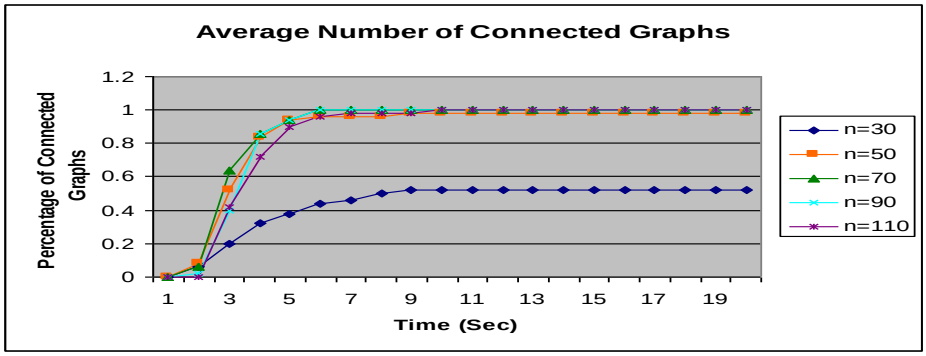

Figure 17: Bluetooth ver1.1 percentage of connected networks – Restricted Implementation

Therefore, we found two interesting phenomena during our work with these experiments; the first phenomenon is that at most 60% of the nodes were discovered when we use the *restricted case implementation*. The second is that the percentage of generating connected networks would decrease if the same implementation was used. This percentage of disconnected networks appears in a clearer way in the case of networks with low average nodal degree (see Figure 16 and Figure 17, experiments with 30 nodes). Obviously, the second phenomenon is caused as a consequence of the first.

We give an informal explanation of the first phenomenon. In order to do that, we need to briefly revisit the INQUIRY procedures. Recall that an inquirer follows a frequency hopping sequence that is derived from its clock and it is as follows:

$$\mathbf{F = [CLK_{16-12} + k + (CLK_{4-2,0} - CLK_{16-12}) \bmod 16] \bmod 32} \qquad \mathbf{(2)}$$

The formula was described already in section 2.4.1**. F** is the frequency used for inquiry. $\mathbf{CLK_{i-j}}$ are i^{th} to j^{th} the bits of the clock and **k** is the variable that controls the selection of the trains. In the specifications, the inquiry frequencies are the 0-5 and 53-78. For simplicity, we will assume that these hops are spread from 1-32. As a result, **k** can be either 1 or 17. The clock ticks each 312.5 μsec. Variable **k** toggles its value each 2.56 seconds.

Figure 18 depicts the sequence of the frequency hops generated by Eq. 2. Each line of those shown in Figure 18 is a sequence of frequencies that the inquirer follows. The inquirer spends 10 msec to visit all of the frequencies of one line. At the toggle of the 12^{th} bit of the clock (i.e. after 1.28 seconds), a new line is generated. Therefore, each line of Figure 18 is repeated for 1.28 seconds (or 1.28 seconds/ 10 msec = 128 iterations). Note that the between an even-numbered line and its following line, there is only one different frequency hop. This was caused by the toggle of the 12^{th} bit of CLK. This, in fact, is a swap of two frequencies; each belongs to a different train. Note also that between an odd-numbered line and its following line, there are 15 different frequency hops. This is caused by the toggle of the **k** variable (i.e. after each 2.56 seconds). It is very important to note at this point that the scanner device's frequencies are derived from its clock and are not dependant on the train value (i.e. the **k** value). We can assume that these frequencies are drawn randomly from the 32 available frequencies.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	2	3	20	21	22	23	24	25	26	27	28	29	30	31	32
17	18	19	20	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	23	24	25	26	27	28	29	30	31	32
1	2	3	4	5	6	7	24	25	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	27	28	29	30	31	32
1	2	3	4	5	6	7	8	9	10	11	28	29	30	31	32
17	18	19	20	21	22	23	24	25	26	27	28	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	13	14	31	32
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	32
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
17	18	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	21	22	23	24	25	26	27	28	29	30	31	32
1	2	3	4	5	22	23	24	25	26	27	28	29	30	31	32
17	18	19	20	21	22	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	25	26	27	28	29	30	31	32
1	2	3	4	5	6	7	8	9	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24	25	26	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	29	30	31	32
1	2	3	4	5	6	7	8	9	10	11	12	13	30	31	32
17	18	19	20	21	22	23	24	25	26	27	28	29	30	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	16

Figure 18: The inquirer's hopping sequence. Trains toggle each 2.56 seconds.

In the restricted-case, if we assume that an inquirer spends a long time (i.e. 10 seconds and more) in the INQUIRY state, then we have roughly the same sequences of Figure 18. However, the reader should note that, in our implementation of ALTERNATE (whether in the restricted or non-restricted case), a node spend less than 1.28 seconds in the INQUIRY or INQUIRY SCAN state. Therefore, and since we are obliged to use Train A each time a new INQUIRY stars, we find that a node hardly use Train B. However, not using Train B does not mean that half of the frequencies will not be used. Figure 19 shows the frequencies sequence visited by an inquirer if Train B was never used (i.e. the value of **k** never toggles). As it can be seen from Figure 19, two frequencies from different trains are swapped each 1.28 seconds. Whence, after 16x1.28 = 20.48 seconds, all the frequencies of Train A are swapped with those of Train B.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32

Figure 19: The inquirer's hopping sequence if the frequency trains never toggle.

Let us consider the difference between the Figure 18 and Figure 19. While an inquirer in the non-restricted case needed only 6.40 seconds to pass through all the 32 frequency hops, the inquirer needed 20.48 seconds for the same task in the restricted case. In the restricted case, 19 different frequencies were visited by an inquirer in 5.12 seconds, while in the non-restricted case and in the same time, 31/32 of the frequencies were visited. Given this slow rate of visiting the inquiry frequency hops, an inquirer node will have a lower probability of discovering its neighbors. We suspect that if it was not allowed to swap frequencies between the trains each 1.28 seconds, then the probability would be a little higher. It is worth mentioning that this explanation is informal and it would be more useful to provide a mathematical analysis to this phenomenon. We still believe, however, that the source of the problem is caused by the slower rate of changing inquiry hopping frequencies which is found in the non-restricted case.

The second phenomenon (i.e. the high percentage of generating disconnected networks) can be seen as a consequence of the first observation (i.e. at most 60% of the links were discovered). Since the percentage of discovered links does not exceed 60%, and since the distribution of nodes in the plane is uniform, and thus, the links are also uniformly distributed, the nodal degree of the generated networks would also not exceed the 60% of the value of the nodal degree of the original Unit Disk graphs. This matches the results that we obtained from the experiments. We link this fact to the results of [94] which related the probability of the connectivity of a UDG to the number of neighbors a node is connecting to (i.e. the average nodal degree), and found that the nodal degree should be logarithmic to the number of nodes in the network to get the connectivity. Hence, in the case of networks with 30 nodes, 0.6 x 30 is on the range of the values of the average nodal degree that are sufficient to make a uniform UDG network connected. This is, perhaps, why we find approximately 50% of the generated networks and the other 50% were disconnected.

In summary, our objective in these case studies was to show how some simple changes in the implementation of Bluetooth can have major impact, whether positive

or negative, on the performance of a device discovery technique or a BSF algorithm. We believe that all the experiments presented in this chapter, accompanied with the analyses, are sufficient to prove our argument. The two case studies give examples of some features in the Bluetooth specifications that have a large impact on BSF algorithms and cannot be considered always as “details”.

Chapter 6

Summary and Conclusion

6.1. Summary

In this thesis, we investigated the different solutions used by BSF algorithms to address the problems that are introduced by the device discovery and link establishment procedures of the Bluetooth specifications. We started with a survey of the different device discovery techniques used by BSF algorithms. We categorized them into static and dynamic techniques. We also categorized BSF algorithms into static and dynamic algorithms, depending on the device discovery technique they use.

We observed during our survey that there is, to some extent, a "lack of shared knowledge" of the restrictions introduced by the specifications of Bluetooth that are imposed on BSF algorithms. We believe this has a negative impact on the advance of the research in this field.

We studied then the different theoretical models used to study BSF algorithms. We analyzed these models and criticized some of them. We argued that these theoretical models did not model adequately, in most cases, Bluetooth wireless networks and these models were oversimplified in many cases. This was done by showing that many problems, mainly those that are related to the device discovery and link establishment procedures, were not addressed properly (or even not considered sometimes) when studying BSF algorithms. We gave different examples to such problems in static, dynamic and on-demand BSF algorithms.

We also surveyed a number of works that concentrated on the device discovery procedures problems using different approaches to address them (other than the approach followed by BSF algorithms). We categorized these approaches into four categories: 1) studies that proposed modifications on the specifications, 2) studi-

es that proposed the use of a wireless technology other than Bluetooth to bypass the problems of the device discovery procedures, 3) studies that formally analyzed the device discovery procedures and 4) studies that used hardware devices in their experiments to study the device discovery procedures. These studies usually started with an examination of the factors that negatively affect the performance of the device discovery procedures. We believe that these studies help in understanding in more depth the device discovery procedures of Bluetooth. These studies also help in understanding the Bluetooth wireless networks model.

In chapter 5, we conducted some experiments to study the performance of different device discovery techniques. We studied the performance of ALTERNATE, LIM-ALTERNATE and the device discovery technique of Law's algorithm [15] using Bluetooth ver1.1 and Bluetooth ver1.2. The change made on ver1.1 was small. However, the impact of this small change was substantial. We found that ALTERNATE, using ver1.2, generates a connected a connected network generates about 3.5 times faster than it does using ver1.1. It also discovers about 20% more edges. We found that LIM-ALTERNATE is affected in the same way ALTERNATE is affected by ver1.2. Interestingly, we found that the device discovery technique of Law's algorithm [15] is not affected by Bluetooth ver1.2. In reality, we argued that all the currently available dynamic BSF algorithms will not be affected by ver1.2.

We conducted some other experiments on ALTERNATE; using different implementations of the specifications. These implementations dealt differently with the frequency trains used during the inquiry procedures. The first implementation, which we called the restricted case, starts each inquiry procedure in Train A, while the second implementation, which we called the non-restricted case, keeps the counter that controls the switching of the trains (i.e. between Train A and Train B) always running. Both implementations are legal according to the specifications of Bluetooth. Interestingly, we found from the experiments we conducted that ALTERNATE using the restricted case does not discover more than 60% of the networks' edges. If the performance of ALTERNATE was similar to that, then all static BSF algorithms will be negatively affected.

6.2. Conclusion

Four main points were observed during our work in this thesis. These points are:

1. The lack of shared knowledge among the researchers of the BSF algorithms domain on the problems introduced by the specifications of the device discovery and link establishment procedures of Bluetooth.
2. The theoretical models used to study BSF algorithms do not adequately model Bluetooth wireless networks in most cases, and were oversimplified in many cases.
3. The existence of many features in the Bluetooth specifications that have a large impact on BSF algorithms, and therefore cannot be considered always as details.
4. Simulation experiments are the only way used so far to compare different BSF algorithms. However, simulation is considered by many researchers as an inefficient way to compare algorithms.

These four points can be considered as weaknesses in the domain of BSF algorithms. We believe that in order to overcome these weaknesses, a "general computational model" needs to be created and used as a basis to study BSF algorithms. What is meant with a "general computational model" is the following:

1. It defines clearly the different restrictions imposed by the Bluetooth specifications on the distributed algorithms.
2. It makes such restrictions as a "shared knowledge" among the researchers of the field, and whence this model can be used as a starting point to design distributed algorithms in Bluetooth wireless networks – including BSF algorithms.
3. This model is used also as a basis to compare the performance of the different (and numerous) BSF algorithms. Although BSF algorithms seem to be very hard to be analyzed mathematically, such a general

computational model can be used to compare these algorithms – at least informally – so to keep simulation as the last step to compare them.

In other words, this general model should define clearly what can be done (and at what cost) and what cannot be done. This model should also identify the restrictions imposed by the Bluetooth specifications on BSF algorithms in a manner that make them well known among the researchers of the domain.

All our work considered BSF algorithms, since the BSF problem is the most studied network problem (if not the sole) in the literature. However, we believe that other classical problems of theoretical distributed computing, such as election, broadcasting or tree formation, should also be studied with considering the restrictions that are imposed by the Bluetooth specifications. Note that some of these classical problems were studied under different distributed computational models. For instance, the election problem was studied under the asynchronous and synchronous wired networks [95] [96] [97][98], the broadcast radio networks model [99][100], the self-stabilizing model [101] and the mobile agents' model [102][103], but it was never studied with the same intensity under the Bluetooth networks model. Note that some BSF algorithms use some ideas of such classical algorithms. For instance, some algorithms such as [56] run a phase for electing a unique node. However, the election problem was never studied under the Bluetooth networks model. Also, the algorithms of TSF [32] and SHAPER [29] both imported ideas from the classical algorithms of tree formation, such as [95] for example. Moreover, we find some BSF algorithms that use the same principle of reactive routing algorithms such as [9] and [63] for example. Therefore, studying these classical problems (and other different problems) under the Bluetooth networks model would be useful for the theory of distributed computing and for BSF algorithms. We see that the Bluetooth network model presents some interesting and challenging problems to the domain of theoretical distributed computing. Also and because the many interesting features of Bluetooth, such as the low cost and low power consumption of Bluetooth devices, solving these classical problems under the Bluetooth networks model would help the field of wireless ad-hoc networking.

In this thesis, we investigated some essential problems all BSF algorithms need to deal with, which are those that are introduced by the device discovery and link establishment procedures. We surveyed many solutions to such problems. We criticized some of these solutions as well. We also conducted some experiments on some device discovery techniques to study them in more depth. One of the goals of this thesis was to identify some of the problems that are introduced by the device discovery and link establishment procedures of Bluetooth, and to make such problems commonly known among the researchers of the field. We believe that publicizing such problems will help in understanding in more depth the Bluetooth Scatternet Formation problem, and will help the study of the distributed algorithms of Bluetooth wireless networks as well.

References

[1] IETF Manet charter, *http://www.ietf.org/html.charters/manet-charter.html*. [online]

[2] I. D. Chakeres and J. P. Macker. Mobile ad hoc networking and the IETF. *SIGMOBILE Mobile Computing Communication Review* 10 (2006), no. 1, 58-60.

[3] Bluetooth Core Specifications v1.1.

[4] Bluetooth Core Specifications v2.1 + EDR, www.bluetooth.com [last visited in 29/3/2009]

[5] J. Kardach. How Bluetooth got its name. *EE Times Scandinavia*. Available at: http://www.eetimes.eu/scandinavia/206902019?cid=RSSfeed_eetimesEU_scandinavia [last visited in 24/2/2009]

[6] Stojmenovic. I, and Zaguia, N. "Bluetooth scatternet formation in ad hoc wireless networks," Chapter 9 in: J.Misic and V. Misic. *Performance Modeling and Analysis of Bluetooth Networks: Network Formation, Polling, Scheduling, and Traffic Control*. Auerbach Publications (2006), 147-171.

[7] J. Misic and V. Misic. *Performance Modeling and Analysis of Bluetooth Networks: Polling, Scheduling, and Traffic Control*. Auerbach Publications. (2006).

[8] R.M. Whitaker, L. E. Hodge and I. Chlamtac. Bluetooth scatternet formation: A survey. Ad Hoc Networks 4 (2005), no. 3, 403-450.

[9] Y. Liu, M.J. Lee and T.N. Saadawi. A Bluetooth scatternet route structure for multi-hop ad hoc networks. IEEE Journal of Selected Areas in Communications. 21 (2003), no. 2, 229-239.

[10] C.E. Perkins, E.M. Royer, S.R. Das and M.K. Marina. Performance comparison of two on-demand routing protocols for ad hoc networks. IEEE Personal Communications 8 (2001), no. 1, 16-28.

[11] D. Johnson and D. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks*. Chapter 5 in: T. Imielinski and H. Korth, Eds. *Mobile Computing*, Kluwer (1996).

[12] C. E. Perkins and E. M. Royer. Ad Hoc On-demand Distance Vector Routing. In: *Proceeding of the 2nd IEEE workshop of Mobile Computing Systems and Applications* (1999), 90–100.

[13] C. Petrioli, S. Basagni and M. Chlamtac. Configuring BlueStars: multihop scatternet formation for Bluetooth networks. IEEE Transactions on Computers 52 (2003), no. 6, 779-790.

[14] N. Zaguia, Y. Daadaa and I. Stojmenovic. Simplified Bluetooth scatternet formation using maximal independent sets. Integrated Computer-Aided Engineering 15 (2008), no. 3, 229-239.

[15] C. Law, A.K. Mehta and K. Siu. A new Bluetooth scatternet formation protocol. Mobile Networks and Applications 8 (2003), no. 5, 485-498.

[16] A. Jedda, N. Zaguia and G-V. Jourdan. Analyzing the discovery phase in Bluetooth Scatternet Formation Algorithms. In: *proceeding of the International Conference of Distributed Computing and Systems (ICDCS) workshops, The 2nd IEEE Specialized Ad Hoc Networks and Systems workshop SAHNS 2009.* Montreal, Canada (2009).

[17] A. Jedda, N. Zaguia and G-V. Jourdan. An Analysis of the BluePleidas Algorithm's Device Discovery Phase. In: *proceeding of the International on Information and Communication Technologies and Accessibility (ICTA 2009).* Hammamat, Tunisia (2009).

[18] K. Pawlikowski, H.-D.J. Jeong and J.-S.R. Lee. On credibility of simulation studies of telecommunication networks. IEEE Communications Magazine 40 (2002), no.1, 132-139.

[19] S. Kurkowski, T. Camp and M. Colagrosso. MANET simulation studies: the incredibles. SIGMOBILE Mobile Computing Communication Review 9 (2005), no. 4, 50-61.

[20] T.R. Andel and A. Yasinsac. On the credibility of MANET simulations. Computer 39 (2006), no. 7, 48-54.

[21] J. Heidmann, N. Bulusu, J. Elson, C. Intangonwiwat, K. Lan, Y. Xu, W. Ye, D. Estrin and R. Govindan. Effects of Details in Wireless Networks Simulations. In: *Proceedings of the SCS Multiconference on Distributed Simulation*, Phoenix, Arizona, USA, USC/Information Sciences Institute, Society for Computer Simulation (2001), 3-11.

[22] I. Stojmenovic. Simulations in wireless sensor and ad hoc networks: Matching and advancing models, metrics and solutions. IEEE Communications Magazine 46 (2008), no. 12, 102-107.

[23] D. Cavin, Y. Sasson and A. Schiper. On the accuracy of MANET simulators. In: *Proceedings of the Second ACM international Workshop on Principles of Mobile Computing* (Toulouse, France, October 30 - 31, 2002). POMC '02. ACM, New York, NY, 38-43

[24] T. Moscibroda, R. Wattenhofer "Maximal Independent Sets in Radio Networks." In: *Proceedings of the 24th ACM Symposium on the Principles of Distributed Computing (PODC'05),* Las Vegas, Nevada, USA (2005), 148-157.

[25] F. Kuhn, T. Moscibroda and R. Wattenhofer. "Radio Network Clustering from Scratch." In: *Proceedings of the 12th European Symposium on Algorithms (ESA) (2004),* 460-471.

[26] I. Cidon and O. Mokryn. Propagation and Leader Election in a Multihop Broadcast Environment. In: *Distributed Computing, 12th International Symposium, DISC '98,* Andros, Greece (1998), 104-118.

[27] The official Bluetooth Technology Info Site. www.bluetooth.com [last visited in 12/5/2009]

[28] Z. Wang, R.J. Thomas and J. Haas. Performance comparison of Bluetooth scatternet formation protocols for multi-hop networks. Wireless Networks 15 (2009), no. 2, 209-226.

[29] F. Cuomo, T. Melodia and I.F. Akyildiz. Distributed self-healing and variable topology optimization algorithms for QoS provisioning in scatternets. IEEE Selected Areas in Communication 22 (2004). no.7, 1220-1236.

[30] L. Barriere, P Fraigniaud, L. Narajanan and J. Opatrny. Dynamic construction of Bluetooth scatternets of fixed degree and low diameter. In: *Proceedings of the ACM Symposium on Discrete Algorithms SODA* (2003).

[31] Y. Wang, I. Stojmenovic and X-Y. Li. Bluetooth scatternet formation for single-hop ad hoc networks based on virtual positions. In: *Proceedings of the IEEE Symposium on Computers and Communications*, Alexandria, Egypt (2004).

[32] G. Tan, A. Miu, J. Guttag, H. Balakrishnan. An Efficient Scatternet Formation Algorithm for Dynamic Environments. In: *Proceedings of IASTED CCN*, Cambridge, MA (2002).

[33] T.Y. Lin, Y.C. Tseng and K.M. Chang. A new BlueRing scatternet topology for Bluetooth with its formation, routing, and maintenance protocols. Wireless Communications and Mobile Computing 3 (2003), 17-537.

[34] D. Dubhashi, O. Häggström, G. Mambrini, A. Panconesi and C. Petrioli. Blue Pleiades, a new solution for device discovery and scatternet formation in multi-hop Bluetooth networks. Wireless Networks 13 (2007), no. 1, 107-125.

[35] V. Mehta and M. El Zarki. A Fixed Sensor Networks for Civil Infrastructure Monitoring. Wireless Networks 10 (2004), no. 4, 401-412.

[36] T. Salonidis, P. Bhagwat, L. Tassiulas and R. LaMaire. Distributed topology construction of Bluetooth personal area networks. In: *Proeedings of the IEEE INFOCOM* (2001).

[37] L. Ramachandran, M. Kapoor, A. Sarkar and A. Aggarwal. Clustering algorithms for ad hoc wireless networks. In: proceedings of the *ACM DIALM Workshop* (2000), 54-63.

[38] C. Pamuk and E. Karasan. SF-DeviL: an algorithm for energy-efficient Bluetooth scatternet formation and maintenance. Computer Communications – *special issue on Performance issues of Wireless LANs, PANs and ad hoc networks* 28 (2005), no. 10, 1276-1291

[39] C. Petrioli, S. Basagni and I. Chlamtac. BlueMesh: degree-constrained multi-hop scatternet formation for Bluetooth networks. Mobile Networks Applications 9 (2004), no. 1, 33-47.

[40] J. Ghosh, V. Kumar, X. Wang and C. Qiao. BTSpin - Single Phase Distributed Bluetooth Scatternet Formation. CSE Dept. TR # 2004-06, State University of New York at Buffalo. (2004)

[41] C-F. Chiasserini and M. A. Marsan. A distributed self-healing approach to Bluetooth scatternet formation. IEEE Transactions on Wireless Communications 4 (2005), no. 6, 2649-2654.

[42] Q. Wang, "Scheduling and Simulation of Large Scale Wireless Personal Area Networks," Ph.D. thesis, University of Cincinnati, Cincinnati, USA (2006).

[43] K. Persson and D. Manivannan. A Fault-Tolerant Distributed Formation Protocol for Bluetooth Scatternets. International Journal of Pervasive Computing and Communications 2 (2006), no. 2, 165-176.

[44] S. Basagni, R. Bruno and C. Petrioli. Device Discovery in Bluetooth Networks: A Scatternet Perspective. In: *Proceedings of the Second international Ifip-Tc6 Networking 2002*.

[45] T. Sato and K. Mase. A scatternet operation protocol for Bluetooth ad hoc networks. In: *The 5th International Symposium of Wireless Personal Multimedia Communication* (2002).

[46] The VINT Project. The ns Manual. Available at: http://www.isi.edu/nsnam/ns/ [last visited 18/11/2008]

[47] BlueHoc: Bluetooth Performance Evaluation Tool, http://bluehoc.sourceforge.net/ [last visited 4/3/2009]

[48] S. Basagni, R. Bruno, G. Mambrini and C. Petrioli. Comparative performance evaluation of scatternet formation protocols for networks of Bluetooth devices. Wireless Networks 10 (2004), no. 2, 197-213.

[49] X.-Y. Li, I. Stojmenovic and Y. Wang. Partial Delaunay triangulation and degree limited localized Bluetooth scatternet formation. IEEE Transactions on Parallel and Distributed Systems 15 (2004), no.4, 350-361.

[50] W. Song, X. Li, Y. Wang and W. Wang. dBBlue: low diameter and self-routing Bluetooth scatternet. Journal of Parallel and Distributed Computing 65 (2005), no. 2, 178-190.

[51] M. Medidi and A. Daptardar. A Distributed Algorithm for Mesh Scatternet Formation in Bluetooth Networks. In: *International Conference on Wireless Networks 2004*, 295-301.

[52] Y-J Joung and G-D Hwang. A Simple and Fast Algorithm for Bluetooth Network Formation. In: *NETWORKING 2005*, 1413-1417

[53] Y. Kawamoto, V.W.S Wong and V.C.M Leung. A two-phase scatternet formation protocol for Bluetooth wireless personal area networks. In: *proceedings of the IEEE WCNC 2003 and* Wireless Communications and Networking 3 (2003), 1453-1458.

[54] C. Zhang; V.W.S. Wong and V.C.M. Leung. TPSF+: a new two-phase scatternet formation algorithm for Bluetooth ad hoc networks. In: *Global Telecommunications Conference, 2004. IEEE GLOBECOM '04 vol.6* (2004), 3599-3603.

[55] C. Yang, C. Lin and I. Huang. TPSF+C: A Two-Phase On-Demand Scatternet Formation Algorithm Considering Route Stability. In: *Proceedings of the 22nd international Conference on Advanced information Networking and Applications - Workshops (March 25 - 28, 2008). AINAW*. IEEE Computer Society, Washington, DC, 922-927.

[56] G.V. Zaruba, S. Basagni and I. Chlamtac. Bluetrees - scatternet formation to

enable Bluetooth based ad hoc networks. In: *Proceedings of the IEEE International Conference on Communication* ICC 2001, Helsinki, Finland, (2001), 273-277.

[57] E. Pagano, G-P. Rossi and S. Tebaldi. An On-Demand Bluetooth Scatternet Formation Algorithm. In: *WONS 2004*, LNCS 2928 pp.130-143, 2004, *IFIP International Federation for Information Processing 2004*.

[58] A. Jedda, Different implementations of the BlueMIS algorithm. Unpublished manuscript 2008.

[59] B. Awerbuch, A. Baratz and D. Peleg. Cost-sensitive analysis of communication protocols. In: *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing* (Quebec City, Quebec, Canada, August 22 - 24, 1990). PODC '90. ACM, New York, NY, 177-187.

[60] L. Higham and T. Przytycka. Asymptotically Optimal Election on Weighted Rings. SIAM Journal of Computing 28 (1999), no. 2, 720-732.

[61] S. Jain, K. Fall and R. Patra. Routing in a delay tolerant network. In: *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (Portland, Oregon, USA, August 30 - September 03, 2004). *SIGCOMM '04*. ACM, New York, NY, 145-158.

[62] M. Liberatore, B. N. Levine and C. Barakat. Maximizing transfer opportunities in Bluetooth DTNs. In: *Proceedings of the 2006 ACM CoNEXT Conference* (Lisboa, Portugal, December 04 - 07, 2006).

[63] X. Zhang and G.F. Riley. Energy-aware on-demand scatternet formation and routing for Bluetooth-based wireless sensor networks. *IEEE Communications Magazine* 43 (2005), no.7, 126-133.

[64] B. Zhen, J. Park and Y. Kim. Scatternet Formation of Bluetooth Ad Hoc Networks. In: *Proceedings of the 36th Hawaii International Conference on System Sciences* (2003), 312-319.

[65] C.S. Choi and H.W. Choi. DSR based Bluetooth scatternet. In: *ITC-CSCC 2002*, July 2002, Thailand.

[66] X. Zhang and G.F. Riley. Evaluation and accelerating Bluetooth device discovery. In: *Proceedings of the IEEE Symposium of Radio and Wireless Networks* (2006).

[67] B. Zhen and Y. Kim. Location management support of IP over Bluetooth. Presented at: The *3Gwireless'2002,* San Francisco, CA, May 2002.

[68] P. Murphy, E. Welsh and P. Frantz. Using Bluetooth for Short-Term Ad-Hoc Connections between Moving Vehicles: A Feasibility Study. In: *Proceedings of the IEEE Vehicular Technology Conference (VTC), vol. 1, issue. 55* (2002), 414-418.

[69] E. Welsh, P. Murphy and P. Frantz. Improving Connection Times for Bluetooth Devices in Mobile Environments. In: *International Conference on Fundamentals of Electronics Communications and Computer Sciences (ICFS)*, (March 2002).

[70] G.V. Záruba and I. Chlamtac. Accelerating Bluetooth Inquiry for Personal Area Networks. In: *Proceedings of the 2nd IEEE Globecom 2003*, San Francisco, CA, December (2003), 702 – 706.

[71] G. Chakraborty, K. Naik, D. Chakraborty, N. Shiratori and D. Wei. Analysis of the bluetooth device discovery protocol. Wireless Networks. [Online]. Available at: http://dx.doi.org/10.1007/s11276-008-0142-1

[72] G.V. Zaruba and V. Gupta. Simplified Bluetooth device discovery - analysis and simulation. In: *Proceedings of the 37th Annual Hawaii International Conference on System Sciences (*2004), 5-8.

[73] S.Z. Kardos and A. Vidacs. Performance of a new device discovery and link establishment protocol for Bluetooth. In: *Proceedings of the 6th IEEE GLOBECOM '05* (2005).

[74] J.R. Engelsma and J.C. Ferrans. Bypassing Bluetooth Device Discovery Using a Multimodal User Interface. In: Proceedings of the *Fourth Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services, 2007. MobiQuitous 2007,* 6-10 Aug. 2007, 1-9.

[75] R. Woodings, D. Joos, T. Clifton, and C. Knutson. Rapid heterogeneous connection establishment: accelerating Bluetooth inquiry using IrDA. In: *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)* (March 2002), 342-349.

[76] D. Scott, R. Sharp, A. Madhavapeddy and E. Upton. Using visual tags to bypass Bluetooth device discovery. ACM Mobile Computing and Communication Review 9 (2005), no. 1, 41-53.

[77] T. Salminen, S. Hosio and J. Riekki. Enhancing Bluetooth connectivity with RFID. In: *Proceeding of the 4th IEEE Internationl Conference on Pervasive Computing and Communications (PERCOM'06)*, (March 2006), 36-41.

[78] E. Hall, D. Vawdrey and C. Knutson. RF Rendez-Blue: reducing power and

inquiry costs in Bluetooth-enabled mobile systems. In: *Proceedings of the 11th International Conference on Computer Communications and Networks (CCN 2002)*, October 2002, 640-645.

[79] J. Lansford, A. Stephens and R. Nevo. Wi-Fi (802.11b) and Bluetooth: enabling coexistence. IEEE *Network* 15 (2001), no.5, 20-27.

[80] The official Bluetooth Technology Info Site. Bluetooth Technology to Harness the speed of 802.11. Available at: http://www.bluetooth.com/Bluetooth/Press/SIG/BLUETOOTH_TECHNOLOGY_TO_HARNESS_THE_SPEED_OF_80211.htm. [last visited 4/3/2009]

[81] B. Peterson, J. P. Kharoufeh and R. O. Baldwin. Bluetooth Inquiry Time Characterization and Selection. IEEE Transactions on Mobile Computing 5 (2006), no. 9, 1173-1187.

[82] M. Duflot, M. Kwiatkowska, G. Norman and D. Parker. A Formal Analysis of Bluetooth Device Discovery. International Journal on Software Tools for Technology Transfer (STTT) 8(2006), no.2, 621 – 632.

[83] PRISM - Probabilistic Symbolic Model Checker. Available at: http://www.prismmodelchecker.org [last visited 4/3/2009]

[84] C. Petrioli, C. Pierascenzi and A. Vitaletti. Bluetooth Scatternet Formation Performance: Simulations vs. Testbeds. In: *Proceedings of the IEEE Military Communications Conference, 2006. MILCOM 2006.* (October 2006), 1-9.

[85] D. Bohman, M. Frank, P. Martini and C. Scholz. Performance of symmetric neighbor discovery in Bluetooth ad hoc networks. In: *Proceedings of the German Workshop on Mobile Ad-hoc Networking, WMAN'04*, Ulm, Germany, Dec. 2004.

[86] M.A. Ronai and E. Kail. A simple neighbor discovery procedure for Bluetooth ad hoc networks. In: *Proceedings of the 2nd IEEE Global Telecommunications Conference, 2003. IEEE GLOBECOM '03*. (1-5 Dec. 2003), 1028-1032.

[87] J.-R. Jiang, B.-R. Lin and Y.-C. Tseng. Analysis of Bluetooth device discovery and some speedup mechanisms. Journal of the Institute of Electrical Engineering 11, no. 4, 301–310.

[88] Q. Wang, D. Agrawal. UCBT- Bluetooth extension for NS2 at the University of Cincinnati. Available at: http://www.ececs.uc.edu/~cdmc/ucbt/ [last visited 18/9/2008]

[89] The Bluescat simulator version 0.6. Available at: http://www-124.ibm.com/developerworks/oss/cvs/bluehoc/bluescat0.6.

[90] A. Seth, A. Kashyap and S. Dheeraj. A Simulation Tool for Bluetooth Scatternets: Extensions to Bluescat. Computer Science and Engineering Department, Indian Institute of Technology, Kanpur. Unpublished manuscript.

[91] G. Tan. Blueware: Bluetooth Simulator for ns. MIT Technical Report, MIT-LCS-TR-866, Cambridge, MA, October, 2002.

[92] X. Zhang and G. F. Riley. Bluetooth Simulations for Wireless Sensor Networks Using GTNetS. In: *Proceedings of the 12th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04)*, 2004.

[93] W. Priess, J. Ferreira de Rezende and L. Pirmez. Adaptive inter-piconet scheduling for multipurpose scatternet scenarios. Ad Hoc Networks 3 (2005), no. 2, 141-155.

[94] F. Xue and P.R. Kumar. The number of neighbors needed for connectivity of wireless networks. Wireless Networks 10 (2004), no. 2, 169-181.

[95] R. G. Gallager, P.A. Humblet and P.M. Spira. A Distributed Algorithm for Minimum-Weight Spanning Trees. ACM Transactions on Programming Languages and Systems 5 (1983), no. 1, 66-77.

[96] Y. Afek and E. Gafni. Time and message bounds for election in synchronous and asynchronous complete networks. In: *Proceedings of the Fourth Annual ACM Symposium on Principles of Distributed Computing* (Minaki, Ontario, Canada). PODC '85. ACM, New York, NY, 186-195

[97] G.N. Frederickson and N. Lynch. Electing a leader in a synchronous ring. Journal of ACM 34 (1987), no. 1, 98-115.

[98] B. Awerbuch. Optimal distributed algorithms for minimum weight spanning tree, counting, leader election, and related problems. In: *Proceedings of the Nineteenth Annual ACM Symposium on theory of Computing*

[99] S.P. Levitan and C.C. Foster. Finding an extremum in a network. In: *Proceedings of the International Symposium on Computer Architecture* (1982), 321–325.

[100] S.-H. Shiau and C.-B. Yang. A fast maximum finding algorithm on broadcast communication. Information Processing Letters 60 (1996), no. 2, 81-89.

[101] Y. Afek, S. Kutten and M. Yung. Memory-efficient self stabilizing protocols for general networks. In: *Proceedings of the 4th International Workshop on Distributed Algorithms* (Bari, 1999), J. van Leeuwen and N. Santoro (eds.), vol. 486 of Lecture Notes in Computer Science, Springer-Verlag, pp. 15-28.

[102] D. Deugo. Mobile agents for electing a leader. In: *Proceedings of the 4th International Symposium on Autonomous Decentralized System 1999*, 324-324.

[103] L. Barriere, P. Flocchini, P. Fraigniaud and N. Santoro. Can we elect if we cannot compare? In: *Proceedings of the 15th ACM Symposium on Parallel Algorithms and Architectures (SPAA '03)* (2003), 324-332.

[104] T. Przytycka and L. Higham. Optimal cost-sensitive distributed minimum spanning tree algorithm. *SWAT'96*, 246-258.

[105] A. Boukerche. Performance evaluation of routing protocols for ad hoc wireless networks. Mobile Networks and Applications 9 (2004), no. 4, 333-342.