The International Journal of Parallel, Emergent and Distributed Systems
Vol. 00, No. 00, Month 200x, 1–23

RESEARCH ARTICLE

Toward Better Understanding of the Behavior of Bluetooth Networks Distributed Algorithms

Ahmed Jedda, Guy-Vincent Jourdan, Nejib Zaguia[a*]
[a*] *School of Electrical Engineering and Computer Science*

(*v1.0 released November 2011*)

The use of Frequency Hopping Spread Spectrum (FHSS) in Bluetooth significantly differentiates its networks from classical radio networks. In order to observe such differences, we studied basic algorithms, in particular, neighbor discovery and message exchange algorithms. Some of the major differences are found in the procedures of device discovery and link establishment, which are studied in this paper. We focus on their impact on Bluetooth networks' distributed algorithms. We show through detailed simulation experiments that minor modifications to the Bluetooth specifications or their implementation may significantly affect the performance of well-known neighbor discovery algorithms. We then study the impact of the procedures of link establishment with the purpose of finding time-efficient implementations of communication rounds for Bluetooth networks. We study OrderedExchange and RandomExchange as both algorithms implement communication rounds in Bluetooth, but use the PAGE and PAGE SCAN states differently. Theoretical analysis shows that RandomExchange has a better time complexity, while simulation experiments show that OrderedExchange significantly outperforms RandomExchange in networks with a practical size (110 nodes and less). We use the previous results to improve the time efficiency of Bluetooth Scatternet Formation (BSF) algorithms through the introduction of the time-efficient algorithm OrderedExchangeCMIS. We believe that the study of some other basic algorithms (such as broadcasting, spanning tree, and election) will lead to a better understanding of Bluetooth networks, and as a consequence, to more efficient algorithms that fully leverage the strength of this type of network.

Keywords: Bluetooth, Frequency Hopping, Bluetooth Scatternet Formation, Distributed algorithms

1. Introduction

Our interest in Bluetooth stems from five of its features: 1) its wide availability in the market (about 75% of all phones in the market are equipped with Bluetooth [20]); 2) its efficiency in energy consumption; 3) its robustness against interference and noise; 4) its low cost; and 5) the ad hoc network capabilities that it provides. With these features, Bluetooth has contributed heavily to providing solutions for many wireless ad hoc networks problems (such as, low cost and low energy consumption). A major advantage of Bluetooth comes from using Frequency Hopping Spread Spectrum (FHSS) for communication. FHSS significantly differentiated Bluetooth networks from classical radio networks. Some of the main differences are apparent in the device discovery and link establishment procedures. These procedures are briefly described in the follow.

According to Bluetooth specifications, a Bluetooth node that needs to discover neighboring nodes must switch to a state called INQUIRY. While in the INQUIRY

*Corresponding author. Email: ajedd077@uottawa.ca

ISSN: 1023-6198 print/ISSN 1563-5120 online

DOI: 10.1080/1023619YYxxxxxxxx
http://www.informaworld.com

state, the inquirer broadcasts small packets to announce its existence. A node that wants to be discovered switches to a state called INQUIRY SCAN and listens to inquiry packets generated by its neighbors. If two neighbors discover each other (i.e. met in opposite states), they exchange information that facilitates the link establishment procedures. For the link establishment, a node u that wants to establish a link with a previously discovered neighbor v switches to a state called PAGE and sends small packets specifically designated to v. Node v, on the other hand, must switch to the PAGE SCAN state to listen to the requests of u. Further packets are exchanged to set up the connection.

The Bluetooth specifications list only the guidelines that a Bluetooth network algorithm must follow to implement the device discovery and link establishment procedures. Any method to implement these procedures may be used as long as it complies with the specifications guidelines. These procedures have a major impact on many distributed algorithms especially in terms of their execution time. We found that few research efforts were conducted to develop efficient implementations for the device discovery and link establishment procedures. In this paper we study the INQUIRY, INQUIRY SCAN, PAGE and PAGE SCAN states and their impact on Bluetooth networks. We focus on the execution time of the distributed algorithms of these networks.

In the first part we study the impact of the inquiry states on some neighbors discovery algorithms. The neighbor discovery problem is the problem of letting each node in the network discover all (or a sufficient number of) its neighbors. Bluetooth ver1.2 introduced a minor modification to the device discovery procedures of Bluetooth ver1.1. This modification reduced the time required to establish a link for nodes that discovered each other for the first time. This modification exists in all Bluetooth versions appearing after Bluetooth ver1.2; however, its impact on neighbor discovery algorithms has not been studied. We studied two well-known neighbor discovery algorithms under the two Bluetooth versions. We called these algorithms ALTERNATE (used in [16], [17], [28] and others), and LIM-ALTERNATE [5]. We conducted two sets of experiments on these algorithms, noting a substantial improvement in their performances caused by the modifications of Bluetooth ver1.2. We observed an increase of about 20% in the number of discovered edges.

We perform another set of experiments in which we show how a minor change in the implementation of the Bluetooth specifications can lead to a substantial degradation in the performance of ALTERNATE . The specifications state that, during the device discovery and link establishment procedures, the nodes use 32 frequency channels for the search. These frequencies are divided into two trains, Train A and Train B. A more specific definition of frequency trains is available in 4.2. The specifications state that it does not matter which train a device starts with at the beginning of a new INQUIRY procedure. We showed in this set of experiments that if we forced each new inquiry procedure to start with a specific train and not the other, then the performance of ALTERNATE is significantly degraded (approximately 40% decrease in the number of discovered edges). These experiments are elaborated in Section 4.

The second part of this paper studies the impact of the page states on the communication mechanisms used in Bluetooth networks. We show that major improvements in the execution time of some distributed algorithms may be obtained by the use of a communication mechanism more adapted to Bluetooth networks. Our results are based on the study of *message exchange algorithms*. We define a message exchange algorithm as an implementation of one or more *communication rounds*, where in a communication round every node sends and receives a message to and from all its neighbors. In the literature on Bluetooth network algorithms

are message exchange algorithms. Some examples can be found in [16] [17] [28]. We note as well that many other distributed algorithms, not necessarily in the Bluetooth literature, can be seen as message exchange algorithms. Examples can be found in some solutions of maximal independent set [14][10], coloring algorithms [11], minimum spanning tree [7], self-stabilizing algorithms [23] and others.

In this paper we study two algorithms that implement communication rounds for Bluetooth networks, *RandomExchange* and *OrderedExchange*. The difference between the two algorithms is in the use of the PAGE and PAGE SCAN states. Theoretical analysis show that the time complexities of *RandomExchange* and *OrderedExchange* are $O(\Delta)$ and $O(n)$ respectivley, where Δ is the maximum nodal degree in the network and n is the number of nodes in the network [1]. Simulation experiments on the other hand show the superiority of *OrderedExchange* in relatively small networks of 110 nodes and less. As Bluetooth is mainly used for personalized and indoor networks, we suggest that it is more practical to use *OrderedExchange* to implement communication rounds for Bluetooth networks in comparison to *RandomExchange*.

In Section 6, we introduce applications of our previous results in order to design time-efficient Bluetooth Scatternet Formation algorithms. The problem can be defined as forming Bluetooth network topologies that are efficient with respect to some performance metrics, and this problem has been the focus of most of the research in Bluetooth networks distributed algorithms. Any two Bluetooth nodes communicating with each other must be either in the same piconet or scatternet. A piconet is a star topology of one master node and one or more slaves. As a result, all packets exchanged inside a piconet must pass through its master. It is preferable to limit the number of slaves in a piconet to 7 slaves. Piconets with larger numbers of slaves are allowed, but with a penalty on the performance of the nodes of the piconet. A scatternet is an interconnection of a number of piconets. Piconets can be interconnected to each other by bridge nodes (that is, nodes that have roles in different piconets). Ideally, a scatternet should be connected and outdegree limited (that is, the size of each piconet in the scatternet is limited to 7 slaves). We introduce in Section 6 *OrderedExCMIS*, which is a time-efficient algorithm that forms connected and outdegree limited scatternets. Our algorithm outputs the same results of BlueMIS [28] but with an improved execution time. This improvement is obtained from the use of *OrderedExchange* for the implementation of communication rounds.

The paper is organized as follows. Section 2 includes background knowledge on the Bluetooth specifications. Section 3 surveys some related works. Section 4 is a study of the inquiry states. It includes two case studies that show the impact of some minor changes in the Bluetooth specifications on two well known neighbors discovery algorithms for Bluetooth networks. Section 5 studies the impact of the page states on the communication mechanisms used in Bluetooth networks. Section 6 provides some applications of *OrderedExchange* and Section 7 concludes the paper.

2. Background Knowledge

Bluetooth uses the Frequency Hopping Spread Spectrum (FHSS) technique for communication. Using FHSS, devices communicate using a pseudo-random order of frequency channels, which is known to both communicating devices. FHSS influ-

[1] A detailed analysis can be found in Section 5.3.

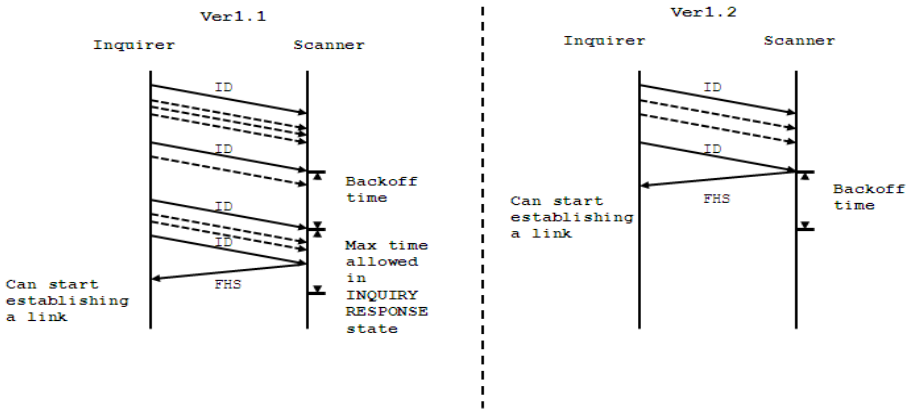

Figure 1.: Differences in the inquiry procedures in Bluetooth ver1.1 and Bluetooth ver1.2. Master is the node that establish the link, while Slave is the node scanning for masters to establish a link

ences the device discovery and link establishment procedures in Bluetooth. These procedures are described in the next two subsections.

2.1 Device Discovery Procedures

The device discovery procedures are controlled by the INQUIRY and INQUIRY SCAN states. A node that needs to discover another (inquirer) switches to the INQUIRY state. To announce its existence, it broadcasts small packets (called ID[1]) in a sequence of different frequency channels, drawn randomly among a preset set. On the other hand, a node that needs to be discovered (scanner) switches to the INQUIRY SCAN state and listens to different frequency channels for neighboring inquirers. When a scanner receives an inquirer's message, its action depends on the Bluetooth version used:

- **For Bluetooth ver1.1 or older:** The scanner *backs off* (that is, does not scan for ID packets anymore), then after a duration drawn randomly it switches back to the INQUIRY SCAN state and listens again for ID packets. If it receives any ID packet within a time limit (not necessarily from the inquirer that made it back off), the scanner sends back to the inquirer a packet called FHS. The FHS packet includes the identifier of the scanner and its clock [2]. The scanner and inquirer have the option to either establish a link or not. Note, that at this step the inquirer know the identity of the scanner, and the opposite is not true (see Figure 1).
- **For Bluetooth ver1.2 or newer:** The scanner sends back the FHS packet. The scanner has the option of either backing off or starting the establishment of the link (which is a procedure initiated by the inquirer) (see Figure 1).

2.1.1 Frequency trains and the interlaced scanning

We show in Section 4 that the previously mentioned modifications to the answer from a scanner to ID packets introduced in Bluetooth ver1.2 have a major positive impact on some neighbors discovery algorithms. We show as well that there is another modification introduced in ver1.2, called *interlaced scanning*, that also has a positive impact on the same algorithms. In the inquiry procedures, both the

[1]Despite the name, ID packets do not contain the identifier of their senders.
[2]Having the clock of the sender facilitates the link establishment procedure

inquirer and scanner alternate within a preset set of 32 frequency channels. These channels are equally divided into two *frequency trains* called Train A and Train B each having 16 frequency channels. Without interlaced scanning, a scanner scans each train alternately for 2.56 seconds[3] each, and each frequency channel is scanned for $T_{w\_inquiry\_scan}$ msec (set to 11.25 msec by default). With interlaced scanning, the scanner scans a frequency channel from a different train each $T_{w\_inquiry\_scan}$ period of time.

2.2 Link Establishment Procedures

Link establishment procedures are controlled by the PAGE and PAGE SCAN states. After two neighbors u and v discovered each other, if node u wants to contact v then u switches to the PAGE state while node v must be in the PAGE SCAN state. Few control packets are exchanged between the two nodes until a link is established.

All algorithms in this paper use the following technique for point-to-point communication. If two nodes u and v want to communicate with each other, they establish a link for a period during which the messages are exchanged, and then the link is destroyed. This technique is used in most Bluetooth network algorithms that assume no nodes mobility.

3. Related Works

Most of the related works available in the literature focus only on the inquiry states. In [6] [9], an analysis of the inquiry procedures is done to obtain faster neighbor discovery algorithms. Other authors proposed to improve the inquiry procedures by modifying the specifications of device discovery [26][2]. None of these studies considered the effects of PAGE and PAGE SCAN states.

Another related study can be found in [13]. In this work, the authors suggested the use of two connected Bluetooth chips instead of one. This would make the procedures of neighbor discovery full-duplex instead of half-duplex. The suggestions can be used to improve the communication mechanisms in Bluetooth networks as well, although the main goal of the authors was to reduce the neighbors discovery time. The work of Zurbes [29] is also close to ours. The author studied the different types of packets in Bluetooth to understand their impact on Bluetooth scatternets and piconets, which is a goal close to our attempt to better understand the behavior of Bluetooth networks.

3.1 Bluetooth Scatternet Formation algorithms

The Bluetooth Scatternet Formation (BSF) problem attracts a large portion of research in Bluetooth networks algorithms. A Bluetooth *scatternet* is defined as a combination of multiple interconnected piconets; where a *piconet* is a star network of one *master* and up to 255 *slaves*. Piconets interconnect with each through *M/S bridges* (a master for one piconet and slave to one or more others), or *S/S bridges* (a slave to more than one piconet). For any two Bluetooth nodes to communicate with each other they must belong to the same scatternet or piconet. BSF algorithms are compared by a number of metrics, a discussion of which can be found in [22]. We focus in this paper on the execution time metric. Some of the well known BSF

[3]This value can be controlled by the developer

algorithms are BlueStars [16], BluePleidas [5], BlueMesh [17] and BlueMIS [28]. We introduce in section 6.1 a modified version of BlueMIS that is more efficient in term of execution time.

3.2 Neighbor Discovery and Message Exchange Algorithms

We focus in this paper on two types of algorithms to obtain our results: neighbors discovery and message exchange algorithms. The neighbor discovery algorithms that we study are referred as ALTERNATE [3] and LIM-ALTERNATE [5]. In ALTERNATE, all the nodes alternates between the INQUIRY and INQUIRY SCAN states. When two nodes discover each other: 1) they establish a temporary piconet; 2) they exchange their identifiers and some other information that may be useful later; and 3) they then destroy the piconet after that. The algorithm terminates after a period of time is passed. ALTERNATE is used in many BSF algorithms such as [28], [16] and [17].

LIM-ALTERNATE [5] is similar to ALTERNATE except that it allows a node to discover (or be discovered by) at most c of its neighbors, where c is a constant integer between 5 and 7. Limiting the size of piconets of a scatternet to 7 slaves is significant to its efficiency. This is because, according to the specifications, a master can control only 7 slaves at a time, all others slaves are kept in a *parking* state (that is, they do not communicate with the master for a specific period of time). This parking introduces a degradation in the throughput of the scatternet. It is shown in [5], theoretically and experimentally, that LIM-ALTERNATE generates, with a high probability, connected degree limited networks given that the nodes are uniformly distributed in the network.

Unfortunately, we did not find any research work that studied message exchange algorithms in Bluetooth networks. When analyzing these types of algorithms, we refer to results obtained from previous research, some of which come from the Bluetooth research community.

4. The Impact of the Inquiry States on Neighbors Discovery Algorithms

In this section we present two case studies examining the impact on the inquiry states on two neighbor discovery algorithms, ALTERNATE and LIM-ALTERNATE. In the first, we show that a minor change in the specifications of Bluetooth introduced an increase of 10% to 30% in the number of discovered edges in both ALTERNATE and LIM-ALTERNATE. We then show how a minor change in the implementation of the specifications may lead to a decrease of about 35% in the number of discovered edges when ALTERNATE is used for neighbors discovery. All our experiments were done using the UCBT simulator [25] which is an NS-2 [24] library specifically designed for Bluetooth.

4.1 First Case Study: A minor change, substantial improvement

We conducted a number of experiments on ALTERNATE under Bluetooth ver1.1, Bluetooth ver1.2 and Bluetooth ver1.2 with the interlaced scanning option. We first generated 250 unit disk graphs and uniformly distributed them in a squared area of 30m $\times$ 30m. Each node had a radio range of 10m. We considered five groups of connected networks with 30, 50, 70, 90 and 110 nodes. For each type of network, we conducted 50 different experiments. For the parameters of ALTERNATE, we used the recommended values of [2].

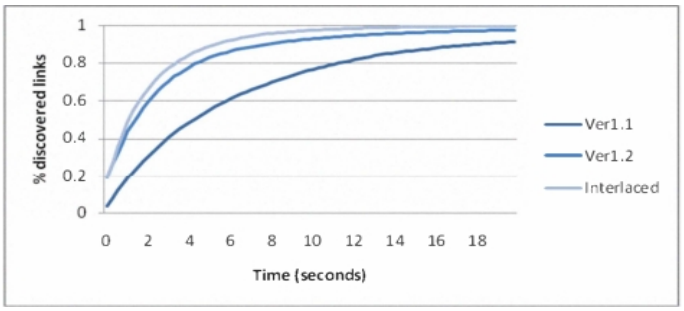

Figure 2.: The percentage of discovered edges against time (number of nodes = 30)

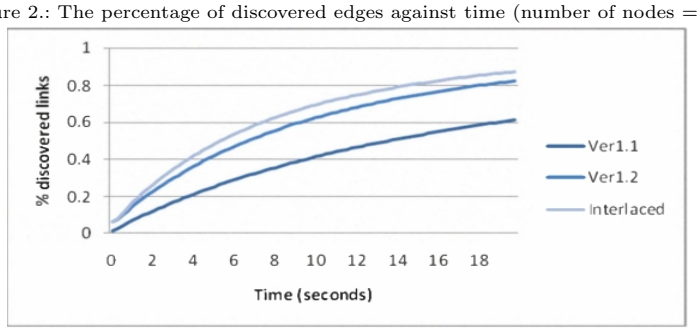

Figure 3.: The percentage of discovered edges against time (number of nodes = 110)

We first studied the rate at which new edges were being added to (i.e., discovered in) the networks. Figure 2 and Figure 3 show this rate for the networks of 30 nodes and 110 nodes. It can be seen that ver 1.2 has a faster rate than ver 1.1, and that that the interlaced scanning option improves the performance slightly. 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 ver 1.1, when a node in the INQUIRY SCAN receives an ID packet, it must first wait for a period of time (the *backoff time*). After the backoff time, it has to wait for the receipt of another ID packet to start establishing a link, which it may not receive at all (see Figure 1). Therefore, in ver1.2, the probability of a successful establishment of a link is increased and the time needed to establish a link is decreased. As a consequence, this leads to a faster rate of links discovery. The improvement obtained from the interlaced option is obvious. A device using this option receives messages from more inquirers as it scans both frequency trains at a higher rate. Also, a device using the interlaced option must necessarily use the modifications introduced in Bluetooth ver1.2.

We now look at the time required for the networks to be connected (the connectivity time). In order to measure the connectivity time, we consider the time at which at least 98% of the network becomes connected. Our experiments are run for 20 simulation seconds each. The results are shown in Table 1. ALTERNATE using Bluetooth ver1.1 requires on average about 6 seconds to reach the connectivity threshold. The impact of the modifications of Bluetooth ver1.2 is clear in Table 1. Bluetooth ver1.2 needs an average of 3.5 seconds to guarantee the connectivity of 98% of the networks. With the interlaced scanning option in Bluetooth ver1.2, only 1.77 seconds on average are needed, which is about 3.5 times better than the connectivity time obtained using ver1.1. Note that the results of ALTERNATE running under ver1.1 match those published in [2].

After a period of time, almost no new edges are discovered and thus the percentage of discovered edges never reaches 100% (see Figure 2 and Figure 3). Thus, even with the new modifications to the specifications, it cannot be expected that

Table 1.: Results of the 98% connectivity (in seconds)

Number of nodes	ver1.1	ver1.2	ver1.2 interlaced scanning
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

a complete unit disk graph can be discovered using ALTERNATE only. The use of ver1.2 and the interlaced scanning option may lead to a high percentage of discovered edges (about 100%) when the nodal degree is low.

We then studied the performance of LIM-ALTERNATE under the two versions. It should be noted that the difference between LIM-ALTERNATE and ALTERNATE lies in that a node in LIM-ALTERNATE is not allowed to discover or be discovered by more than c of its neighbors, where c is 5,6 or 7. As a result, we observed that the connectivity time of LIM-ALTERNATE and ALTERNATE are affected in approximately the same way. Moreover, we found in [8] that there is no significant statistical difference between the connectivity times of ALTERNATE and LIM-ALTERNATE in all versions. The experiments details are omitted from this paper and can be found in [8].

4.2 Second Study Case 2: Another minor change, substantial degradation

As mentioned in Section 2.1.1, the nodes in the INQUIRY and INQUIRY SCAN states alternate between 32 frequency channels. These channels are equally divided into two trains; Train A and Train B. The alternation is based on the following equation:

$$F = [Clk_{16-12} + k + (Clk_{4-2,0} - Clk_{16-12})\%16]\%32 \quad (1)$$

F is the frequency that is used for the inquiry, and % is the modulo operation. Clk_{i-j} are the i^{th} to the j^{th} bits of the device clock Clk. The value of Clk_0 is switched every 312.5 usec. The formula generates only 32 frequency hops. These 32 frequencies are divided into two equal sets, which are Train A and Train B. The variable k can have only two values. It is used to select the current train. The trains are switched every 2.56 seconds (by default), and can be switched more frequently depending on the user requirements. We show in the following that the variable k has a substantial impact on the performance of the ALTERNATE algorithm.

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 state that it does not matter which train a device starts 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 [21]).

To evaluate the impact of starting the inquiry with one train or the other, we consider two implementations of ALTERNATE that differ only on the way they deal with the k variable. In the first implementation, we force a device that enters the INQUIRY state to always start its inquiry in Train A. We call this the *restricted case*. The second implementation keeps the counter that controls the toggling of the k variable constantly running, meaning that a device can start the inquiry

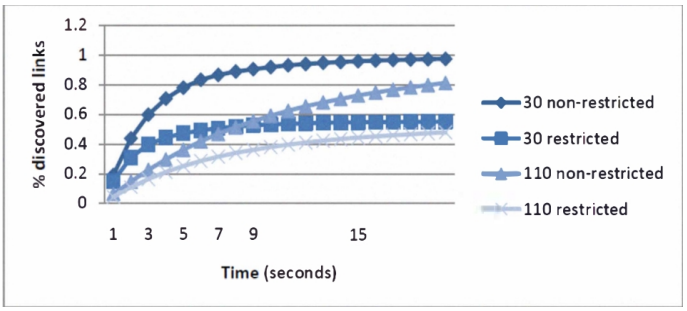

Figure 4.: A comparison of the percentage of discovered edges against time between the restricted and non-restricted case. Experiment simulation time is 20 seconds

procedure in any train. We call this the *non-restricted case*. Note that these two implementations comply with the specifications of Bluetooth.

In this experiment, we generated 250 unit disk graphs. We arranged the graphs into five groups based on the number of nodes (30, 50, 70, 90 and 110). Note that in previous experiments the implementation of ALTERNATE was the non-restricted case. We only report here the 30-nodes and 110-nodes networks. Figure 4 shows a comparison of the percentage of discovered edges against time between the restricted and non-restricted case.

We found a surprising degradation in the performance of ALTERNATE when the restricted case is used. Figure 4 shows that the experiments of the restricted case of ALTERNATE could not discover more than 60% of the edges, while the non-restricted case discovered about 98% in the 30-nodes networks and about 83% in the 110-nodes networks. A minor alteration to the specifications can make a 40% difference in the percentage of discovered edges. Of course, a low discovery percentage leads to a lower percentage of generating connected networks. For instance, we found that about 50% of the networks with 30 nodes were connected after the simulation. For networks with larger number of nodes, the percentage was close to 95%-100%.

We observed two phenomenon in these experiments. The first is the low percentage of edge discovery. The second is the higher percentage of generating disconnected networks. The second is a consequence of the first. We provide in the following a brief explanation of the first phenomena. Recall Eq 1 shown above which is used to generate the pseudo-random sequence of inquiry frequency hops. Figure 5 depicts a sequence generated by Eq 1. Each line of those shown in Figure 5 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 5 appears for 1.28 seconds (or 1.28 seconds/ 10 msec = 128 iterations for each line). Note that between an even-numbered line and its following line, there is only one different frequency hop. This is caused by the toggle of the 12^{th} bit of Clk. This is a swap of two frequencies with each belonging 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 every 2.56 seconds).

It is important to note herein that the frequency sequence the scanner follows is derived from its clock. This sequence is not dependent on the train value (i.e., the k value). We can assume that these frequencies are drawn randomly from all the 32 available frequencies.

Note that, in our implementation of ALTERNATE (whether in the restricted or non-restricted case), a node spends less than 1.28 seconds in the INQUIRY or INQUIRY SCAN states (that is, a node switches from one state to another in less

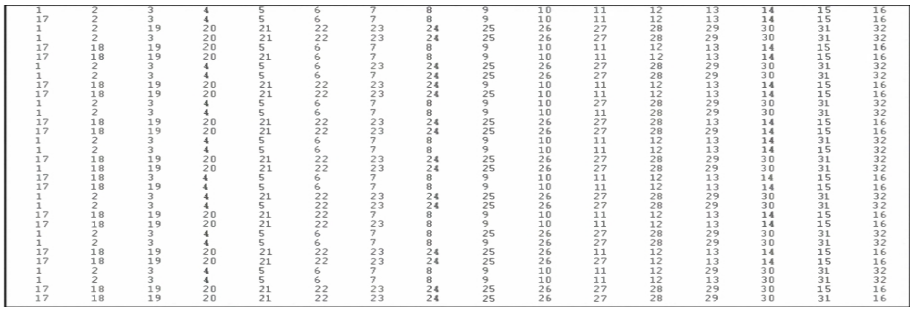

Figure 5.: The inquirer's hopping sequence. The k variable toggles each 2.56 seconds

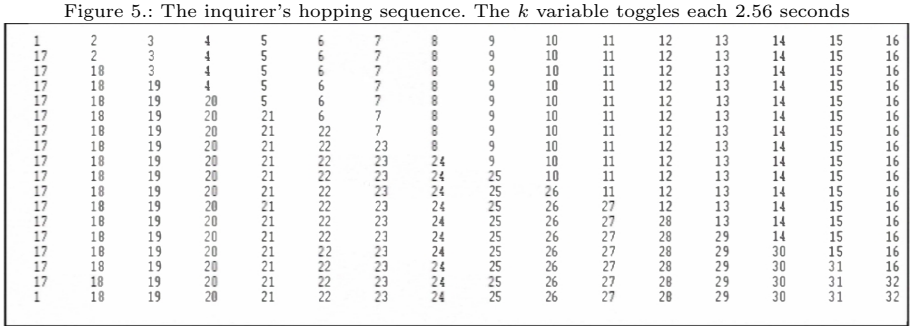

Figure 6.: The inquirer's hopping sequence. The k variable never toggles

than 1.28 seconds). Let us consider the restricted case. Given that the train is switched every 2.56 seconds (See Section 2.1.1), and that we are obliged to use Train A each time a new INQUIRY starts in the restricted case, we find that an inquirer never uses Train B. This means that the value of k points to Train A during the whole period of the device discovery. Figure 6 shows the frequency sequence visited by an inquirer if Train B was never used.

Let us consider the difference between the Figure 5 and Figure 6. In both cases (restricted and non-restricted), a frequency swap (between a frequency from Train A and another from Train B) occurs every 1.28 seconds. In the non-restricted case a new train is used every 2.56 seconds, whereas this does not happen in the restricted case. As a result, the rate of visiting all the 32 hops of inquiry is slower in the restricted case. For instance, 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 non-restricted case 31 frequency hops (out of 32) were visited by an inquirer in a period of 5.12 seconds, while an inquirer in the restricted case visits only 19 frequency hops in the same period of time. Given the slower rate of visiting the inquiry frequencies, and given that the scan frequencies (that is, those visited in the INQUIRY SCAN) are selected randomly among the 32 available frequency hops, we can deduce that an inquirer node has a lower probability of discovering its scanning neighbors in the restricted case. As a consequence, inquirers in the non-restricted case discovers more neighbors in a shorter period of time.

5. The Impact of the Page States on Message Exchange Algorithms

In this section we study the PAGE and PAGE SCAN states. These states control the procedures of link establishment. We show that a careful study of these states may lead to more efficient algorithms in term of execution time.

We observe from the literature that many BSF algorithms are built over *communication rounds* (see [28] [16][17]), where in a communication round, each node

sends/receives a message to/from all its neighbors. We call an algorithm that consists of communication rounds a *message exchange algorithm*. Obviously, algorithms based on communication rounds are not restricted only to BSF algorithms. Examples of similar algorithms can be found in some maximal independent sets algorithms [14] [10], coloring algorithms [11], minimum spanning tree algorithms [7], self-stabilizing algorithms [23] and others.

The purpose of the following experiments is to obtain a better understanding of the behavior of message exchange algorithms in Bluetooth networks. We study two algorithms that implements communication rounds, called *RandomExchange* and *OrderedExchange*. The two algorithms differ only in how they use the PAGE and PAGE SCAN states. We show that message exchange algorithms that use *OrderedExchange* are faster than those that use *RandomExchange* in networks with a relatively small number of nodes. This is mainly due to the fact that the time to transmit a message using *OrderedExchange* is lower than if *RandomExchange* is used.

Three important points of the experiments conducted with the message exchange algorithms used in this paper should be noted:

(1) Neither *RandomExchange* nor *OrderedExchange* require modifications on the Bluetooth specifications to work correctly.
(2) The simulation experiments take into consideration the exact specifications of the baseband and link layers of Bluetooth. The execution time of Bluetooth networks is mostly affected by these two layers.
(3) We implement communication between two neighboring devices using a technique that is used in major BSF algorithms [2]. The technique can be described as follows. For any pair of neighbor nodes to communicate with each other, a temporary piconet of one master and one slave is constructed. The nodes exchange messages. If no more messages have to be exchanged, the piconet is destroyed. The time to establish such piconets is relatively short, given that both nodes already know each other identifiers and the frequency sequence they are following. Such information can be obtained using any neighbors discovery algorithm.

5.1 RandomExchange and OrderedExchange

A Bluetooth node is not able to broadcast a message simultaneously to all its neighbors. Instead, a node needs to establish a link separately with each of its neighbors and send a copy of the same message [1]. This is the main challenge to implementing communication rounds for Bluetooth networks. The procedures of link establishment impose another challenge that needs to solved. To establish a link between two nodes, one of them must be in the PAGE state and the other in the PAGE SCAN state. This may lead to a *deadlock* situation, where two nodes attempt to contact each other indefinitely but never succeed since they are not in opposite states. An algorithm that implements a communication round must avoid such deadlock situations. We describe in the following paragraphs *RandomExchange* and *OrderedExchange*.

RandomExchange is described in Algorithm 1. *RandomExchange* may have a deadlock situation that occurs when two neighbor nodes use the exact same alternation sequence indefinitely. Such situations occur with a low probability. Simulation experiments back this arguments. Interested readers may refer to [18] for a

[1]It should be noted that this is true only in the absence of a scatternet or a piconet that contain all the nodes, which is our case.

comprehensive mathematical analysis that considers a similar case.

Algorithm 1 RandomExchange

1: Each node u alternates randomly and independently between states $PAGE$ and $PAGESCAN$ for periods of time t_{p_i} followed by t_{ps_i} (that is, $\{(t_{p_1}, t_{ps_1}), ..., (t_{p_k}, t_{ps_k})\}$. The values of t_{p_i} and t_{ps_i} are drawn independently and randomly from a distribution T with a mean μ_p.
2: A node u does not switch to the $PAGESCAN$ state if it already received a message from all its neighbors.
3: A node u does not switch to the $PAGE$ state if it already sent a message to all its neighbors.
4: The algorithm terminates once each node send/receive a message to/from all its neighbors.

OrderedExchange uses the unique identifiers of the Bluetooth devices. For simplicity, we assume that the identifier $id(u)$ of a node u is u, where u is an integer unique to each node in the network. For any two neighbors u and v, we say u is a larger neighbor of v if $u > v$. Respectively, we say v is a smaller neighbor of u. *OrderedExchange* is described in Algorithm 2. In *OrderedExchange*, nodes having no larger neighbors start sending messages to all their smaller neighbors. For all other nodes, if a node u receives a message from all its larger neighbors, then u starts sending messages to all its smaller neighbors. We say that *OrderedExchange* is *descending* if the communication starts from the largest nodes to the smaller ones. *OrderedExchange* is said to be *ascending* if the communication starts from the smallest nodes to the larger ones. If we assume that the edge (u, v) and (v, u) are not equivalent, then a round of descending *OrderedExchange* followed by a round of ascending *OrderedExchange* are required to contact all edges of the network. Note that a descending round of *OrderedExchange* can be seen as a transformation of the graph G into a directed acyclic graph G_D such that any edge (u, v) is directed toward v if $u > v$.

Algorithm 2 OrderedExchange at node u (descending)

1: $N_l \leftarrow \{v : v$ is a larger neighbor of $u\}$
2: $N_s \leftarrow \{v : v$ is a smaller neighbor of $u\}$
3: **while** $N_l \neq \emptyset$ **do**
4: Upon reception of a message from neighbor $v \in N_l$, $N_l \leftarrow \{N_l - v\}$
5: **end while**
6: **for** each $v \in N_s$ **do**
7: Send a message to v
8: **end for**

5.2 Algorithmic Models for Bluetooth Networks

A suitable algorithmic model for Bluetooth networks is required in order to mathematically analyze *RandomExchange* and *OrderedExchange*. When modeling Bluetooth networks, we should note two important features. First, packet collisions and interference have a minor impact and thus can be neglected. Second, message broadcasting is not allowed in Bluetooth (that is, for a node to broadcast a message to all its k neighbors, it must perform k sequential send operations). Schmid and Wattenhofer surveyed in [19] the algorithmic models used for wireless sensor

networks. Bluetooth networks cannot be modeled by any of these models because of the inability to broadcast. Interestingly, the Bluetooth networks' model is closer to that of classical wired networks.

We model the network as a geometric graph $G(V, E)$. The vertices represent the nodes. An edge is assigned between two nodes (u, v) if $d(u, v) \leq R$ (that is, the distance between the nodes u and v is less than a threshold R). We assume that each node u has a unique identifier $id(u)$. For simplicity, we let $id(u) = u$. We denote $N(u)$ as the set of neighbors of u. We assume that G is an edge-independent random graph (that is, each edge in the graph is drawn with a uniform probability $p \in O(1)$). These types of graphs have been used widely in the mathematical analysis of wireless networks (see [4]). The synchronous model $\mathcal{SYNC}$ is used to analyze the time complexity (see [11]). The model $\mathcal{SYNC}$ divides the time into equal slots. In a single time slot, a node can receive messages from all its neighbors, perform local computation, and send messages to all its neighbors. This model was used to analyze the time complexities of Bluetooth algorithms in the studies that analyzed execution time mathematically such as [12] [28] [17]. The results of the execution time obtained from our simulation experiments cannot be explained by $\mathcal{SYNC}$. Therefore, we present a slightly modified model we call $\mathcal{SYNC}'$. In the new model, a node can receive messages from all its neighbors, perform local computation, and send messages to *only one* neighbor in one time slot. The theoretical results obtained from $\mathcal{SYNC}'$ match those obtained from simulation experiments.

5.3 Mathematical Analysis of RandomExchange and OrderedExchange

Clearly the time complexity of *RandomExchange* $T_R(n)$ under $\mathcal{SYNC}$ is $O(1)$, since only one communication round is required. For the analysis of the time complexity $T_O(n)$ of *OrderedExchange* we define the graph G_D. The graph G_D is constructed by orienting each edge (u, v) from u to v if $u > v$, where (u, v) is an edge in the graph $G(V, E)$. The graph G_D is thus the directed acyclic graph of $G(V, E)$. We denote the directed path that has the maximum length in G_D as $\pi = \{x_1, .., x_k\}$, where $x_i > x_{i+1}$ for $1 \leq i < k$. The length of π is denoted as $l(\pi)$. Thus, we note that the time complexity of *OrderedExchange* $T_O(n)$ is equal to $l(\pi)$. It was shown in [1] that $l(\pi) \in \Theta(n)$ in edge-dependent random graph. Therefore, $T_O(n) \in \Theta(n)$.

THEOREM 5.1. *Under $\mathcal{SYNC}$, the time complexity of* RandomExchange $T_R(n)$ *is in $O(1)$, whereas the time complexity of* OrderedExchange $T_O(n)$ *is in $O(n)$.*

The analysis is different under $\mathcal{SYNC}'$. For a node to contact all its k neighbors, k time slots are required. Thus, $T_R(n)$ should be expressed in terms of Δ (the maximum degree of the graph G) or $deg(G)$ (the average degree of the graph G). The time complexity of *RandomExchange* is probabilistic due to the random nature of the algorithm. An analysis to a behavior similar to that of *RandomExchange* can be found in [18]. According to [18], the time to establish link between two nodes alternating between PAGE and PAGE SCAN is bounded by a constant (that is, $O(1)$). This makes the average time complexity of *RandomExchange* bounded by $\Theta(deg(G)) \in O(n)$. The time complexity $T_O(n)$ of *OrderedExchange* under $\mathcal{SYNC}'$ varies depending on the strategy (or order) a node uses to contact its neighbors. We consider in this section three strategies:

Strategy 1: The neighbors are contacted in the same order of *OrderedExchange*. Neighbors are contacted from larger to smaller neighbors if *OrderedExchange* is descending, or from smaller to larger neighbors if *OrderedExchange* is ascending.

Strategy 2: The neighbors are contacted in the opposite order of *OrderedExchange*.

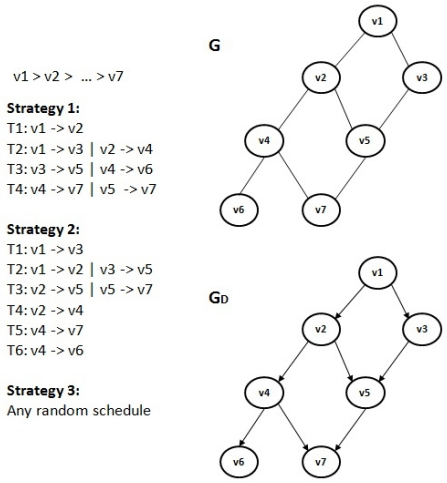

Figure 7.: Example of the three strategies. The order of the nodes is given above. The symbol Ti: $v- > u$ indicates that v sends a message to u at time slot Ti. Note that Strategy 1 required only 4 time slots to terminate, whereas Strategy 2 required 6 time slots

Neighbors are contacted from smaller to larger ones if *OrderedExchange* is descending, or from larger to smaller neighbors if *OrderedExchange* is ascending.

Strategy 3: The neighbors are contacted uniformly randomly (i.e., no specific order). If k is the number of neighbors to be contacted, then each neighbor is selected with a probability $1/k$.

Figure 7 illustrates an example of the execution of *OrderedExchange* using the three strategies. These strategies are important since some algorithms may force the nodes to contact their neighbors using a specific order (see XTC algorithm [27] for example). We analyze in Theorem 5.2 the time complexity of *OrderedExchange* using Strategy 1.

THEOREM 5.2. *Under $\mathcal{SYNC}'$, the worst case time complexity of* OrderedExchange *using Strategy 1 is $O(n)$. The average time complexity is $\Theta(n)$.*

Proof. The time complexity $T_{O_1}(n)$ is at least $\max(\alpha, l(\pi))$. α is the size of the largest set of neighbors to be contacted by any node. Note that $\alpha \leq \delta$. $l(\pi)$ is the length of the longest path π in G_D. We prove that $T_{O_1}(n) \leq n$. Assume that the identifiers of the nodes $V = \{v_1, v_2, ..., v_n\}$ are unique and are assigned from the set $I = \{1, 2, ..., n\}$, where $|V| = n$. Let $\pi = \{x_1, ..., x_k\}$ represents the longest path in G_D where $k = l(G_d)$. Consider any pair of nodes x_i and x_{i+1} in π for $1 \leq i < k$. To maximize $T_{O_1}(n)$, we let each node $x_i \in pi$ contact as many neighbors as possible before contacting x_{i+1}. We denote the set of these neighbors as $m_1(x_i) = \{x_{j_1}, ...x_{j_m}\}$. Any node $x_{j_{i'}}$ for $1 \leq i' < m$ must be less than x_i and greater than x_{i+1} (that is, $x_i > x_{j_{i'}} > x_{i+1}$). The length of such a sequence cannot exceed n. This completes the proof of the first part. Given that $l(\pi) \in \Theta(n)$ on average in edge-dependent random graphs and $p \in O(1)$, we conclude that $T_1(n) \in \Theta(n)$ on average. □

Theorem 5.3 and Theorem 5.4 show that the worst case complexity of *OrderedExchange* using Strategy 2 and Strategy 3 are quadratic with respect to n. Moreover, *OrderedExchange* using Strategy 3 has a lower time complexity than *OrderedExchange* using Strategy 2 in the average case. The proof of both theorems can be found in Appendix 1.

THEOREM 5.3. *Under $\mathcal{SYNC}'$, the worst case time complexity of* OrderedEx-

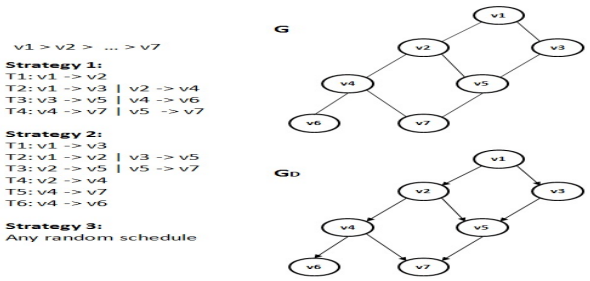

Figure 8.: A comparison of the execution time of OrderedExchange under different strategies

change *using Strategy 2 is* $O(n^2)$.

Proof. See Appendix 1. □

THEOREM 5.4. *Under* $\mathcal{SYNC}'$*, the worst case time complexity of* OrderedExchange *using Strategy 3 is* $O(n^2)$.

Proof. See Appendix 1. □

The average case time complexities of the *OrderedExchange* is more important in our case. Its mathematical derivation is more difficult. We obtain the average execution time of the three strategies using simulation experiments. The experiments show that the average execution time follow the worst case time complexity. The details of these experiments in Section 5.4.

5.4 Simulation results of OrderedExchange and RandomExchange

In this section we discuss the experiments we conducted to analyze *OrderedExchange* and *RandomExchange*. All experiments are implemented using the NS2 network simulator environment [24] and the UCBT library for Bluetooth networks simulation [25]. The networks are modeled as random geometric graphs constructed by randomly placing points on a plane with an area of $30 \times 30m^2$. Each point on the plane represents a node. An edge between two nodes u and v is assigned if and only if the distance between u and v is less or equal to a threshold R. The value of R is set to $10m$ which is the radio range of Bluetooth Class 2 [21]. We categorize our networks according to their size into networks of 30, 50, 70, 90 and 110 nodes.

We compared the behavior of *OrderedExchange* under Strategy 1, Strategy 2, and Strategy 3. The results are shown in Figure 8. The results show that Strategy 1 is the best strategy and follows a linear behavior with respect to the number of nodes. We see that Strategy 2 and Strategy 3 follow a quadratic behavior. Strategy 3 outperforms Strategy 2.

We compared the execution time of *RandomExchange* and *OrderedExchange*. Figure 9 shows the results of the experiments. We can see that *OrderedExchange* outperforms *RandomExchange* in all networks on average. The results are within a confidence interval of 97%.

We conducted experiments under networks with limited average nodal degrees. The average degrees we considered were 5, 10, 15, 20 and 25. To generate a graph with average nodal degree d, the length of the $Nd/2$ edge was set as the threshold R. Every edge with length greater than R is deleted from the network [15]. The results are shown in Table 2. The value P_{ij} of the cell that lies in the intersection of the i^th row and j^{th} column represents the ratio of performance of the algorithms (that is, $P_{ij} = \frac{T_{R_{ij}} - T_{O_{ij}}}{T_{R_{ij}}}$, where T_O and T_R are the average execution time of

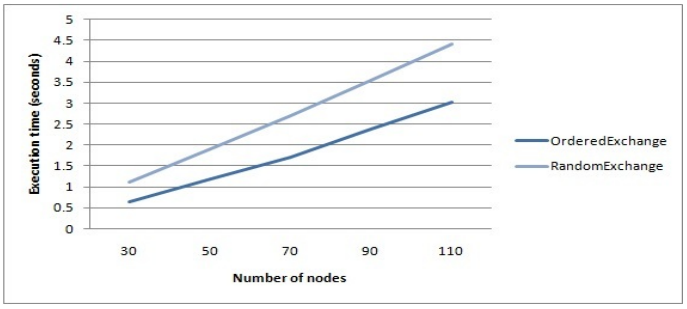

Figure 9.: A comparison of the execution time of RandomExchange and OrderedExchange

Table 2.: Ratio of the average execution time of *RandomExchange* to *OrderedExchange*

number of nodes /degree	5	10	15	20	25
30	0.919	0.746	0.697	0.723	0.772
50	0.890	0.637	0.605	0.627	0.652
70	0.894	0.607	0.629	0.563	0.573
90	0.882	0.630	0.551	0.484	0.506
110	0.846	0.669	0.550	0.521	0.450

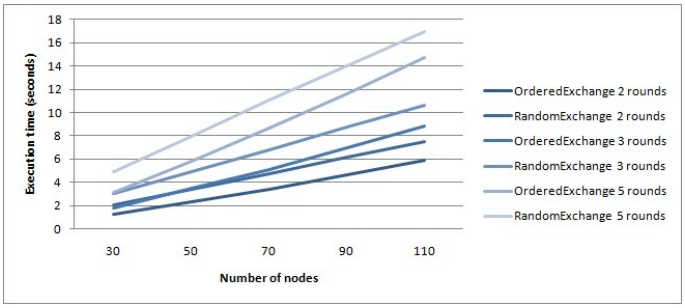

Figure 10.: A comparison of the execution time of RandomExchange and OrderedExchange with multiple consecutive phases

RandomExchange and *OrderedExchange* respectively).

In all experiments, we note that the ratio is always greater than zero, which means that *OrderedExchange* outperforms *RandomExchange* in all experiments. Notice that when the average degree is limited and the number of nodes increases, the performance of *RandomExchange* moves toward being equal to that of *OrderedExchange*.

We studied the performance of *OrderedExchange* and *RandomExchange* when they run z consecutive communication rounds. A node u completes a communication round if it sends and receives a message to and from all its neighbors. A message exchange algorithm runs z consecutive rounds if each node completes z communication rounds. We studied the execution time of both *OrderedExchange* and *RandomExchange* with $z = 1, 2, 3, 4, 5$. Figure 10 shows the results of $z = 2, 3, 5$. We assume that an edge (u, v) is equivalent to (v, u). *OrderedExchange* still outperforms *RandomExchange*. These results can be confirmed with a confidence interval greater than 99.5%. We should note that the average degree of the networks was not limited during these experiments.

We studied the impact of the messages sizes on the algorithms. Since the Bluetooth packets sizes are constant, we modeled the long messages by a series of consecutive messages sent between the sender and the receiver. The results of these experiments still show the superiority of *OrderedExchange*.

We should note that we assumed that in all previous experiments nodes of *Ran-*

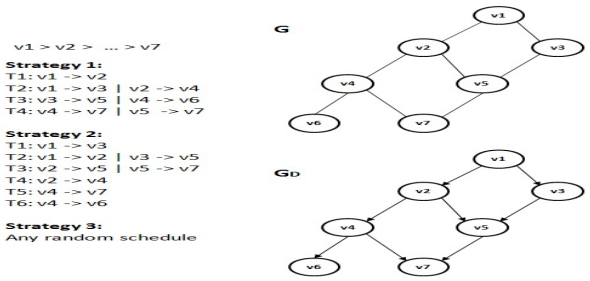

Figure 11.: A comparison of OrderedExchange using Strategy 1 and Strategy 2

domExchange had no order to contact their neighbors, whereas *OrderedExchange* was run under Strategy 1. This was to guarantee that we obtained the performance out of the two algorithms.

We atudied the impact of Strategy 1, Strategy 2 and Strategy 3 on the average execution time of *OrderedExchange*. We studied the worst case time complexity of both algorithms in Section 5.3. Results of the experiments are shown in Figure 11. Interestingly, the average execution time follows a similar behavior to that of the worst case execution time. Strategy 1 gives the best performance with a linear execution time with respect to the number of nodes. Both Strategy 2 and Strategy 3 give a quadratic execution time. We note that Strategy 3 outperforms Strategy 2 similar to what was indicated in the worst case time analysis.

These experiments were all conducted using networks of 110 nodes or less, which is a practical size for Bluetooth networks. If we extrapolate the results of Table 2 we find that *RandomExchange* may outperform *OrderedExchange* in networks with larger sizes and a limited average degree. This shows the correctness of our mathematical analysis, since time complexity analysis is only relevant for a large size of input. The results show that the constant in the time complexity of *OrderedExchange* is larger than the time complexity of *RandomExchange*. This constant represents the cost of transmitting a message. Thus, the results show that *the cost of transmitting a message in a Bluetooth network depends on the communication mechanism used to transmit it*. We conclude that *OrderedExchange* is more efficient than *RandomExchange* in term of execution time in networks of a practical size.

6. Applications

We now discuss applications of the results we found in Section 4 and Section 5. We show first how these results lead to the design of more efficient Bluetooth Scatternet Formation algorithms in terms of execution time.

6.1 Improving Bluetooth Scatternet Formation Algorithms

In Section 4.1, we showed how to improve the performance of ALTERNATE and LIM-ALTERNATE. These neighbors discovery algorithms are the most used in Bluetooth scatternet formation algorithms. Thus, it is obvious that the substantial improvements in ALTERNATE and LIM-ALTERNATE lead to more efficiency in the BSF algorithms that use them.

There are less obvious applications for our results. We focus on applications of *OrderedExchange*. We introduce improvements on the BlueMIS algorithm [28], which is a well known algorithm for Bluetooth Scatternet Formation algorithms. BlueMIS consists of three stages. A simple scatternet is formed in the first stage.

This scatternet is guaranteed to be connected and outdegree limited given that the network is connected and can be modeled as a unit disk graph. A scatternet is said to be connected if there is a path between any two nodes in the scatternet. A scatternet is said to be outdegree limited if the size of its piconets does not exceed 7 slaves. The scatternets formed in the first stage of BlueMIS may contain a large number of masters, which is not preferable in scatternets. The second and third stages are two procedures that heuristically reduce the number of masters with simple local rules. We focus solely on the first stage in this paper. We show that it can be seen as an algorithm based on message exchange rounds. We also show that using *OrderedExchange*, instead of *RandomExchange*, as an underlying message exchange leads to a faster algorithm. It should be noted that, although not formalized, the original design of BlueMIS uses *RandomExchange*.

The first stage of BlueMIS forms *directed local maximal independent sets* (*DL-MIS*). A *DL-MIS* is a directed subgraph $H(V, E')$ of $G(V, E)$, where $E' \subset E$. The set V represents the network nodes. Each node $v \in V$ constructs a local maximal independent set $N'(v)$ of its neighbors $N(v)$ (that is, $N'(v)$ is a maximal subset of $N(v)$ such that no two nodes in $N'(v)$ are neighbors to each other). Symmetries in $N'(v)$ are then broken (that is, for two neighboring nodes u and v such that $u > v$, if $u \in N'(v)$ and $v \in N'(u)$, then u is deleted from $N'(v)$). The set E' is defined as the union of all sets $N'(v)$ for all $v \in V$ (that is, $E' = \bigcup_{v \in V} N'(v)$). The subgraph $H(V, E')$ is connected if $G(V, E)$ is connected. The maximum size of $N'(v)$ for any v is at most five if $G(V, E)$ is a unit disk graph.

Two different implementations of the first stage of BlueMIS are proposed in the following. The first is based on *RandomExchange* (shown in algorithm 3) and the second is based on *OrderedExchange* (shown in Algorithm 4). It is easy to see that algorithm 3 is a message exchange algorithm of one round based on *RandomExchange*. Therefore, its time complexity is $\Theta(deg(G))$; where $deg(G)$ the average degree of G. Note also that the for loop (line 12-14) in Algorithm 3 is necessary for the correctness of the algorithm and therefore algorithm 3 is a message exchange algorithm. Algorithm 4 is a sequence of two consecutive message exchange rounds of *OrderedExchange*.

Algorithm 3 BlueMIS first stage based on RandomExchange - at node u

1: $U \leftarrow N(u)$, $B \leftarrow \emptyset$, $R \leftarrow N(u)$
2: Alternate between states PAGE and PAGE SCAN to send and receive messages.
3:
4: **@ state PAGE:**
5: **while** $U \neq \emptyset$ **do**
6: $v \leftarrow min(U)$
7: u sends a slave message to v
8: $N'(u) \leftarrow \{N'(u) \cup v\}$
9: $B \leftarrow \{B \cup N(v)\}$
10: $U \leftarrow \{U - \{v \cup N(v)\}$
11: **end while**
12: **for** each $b \in \{B \cup N(u)\}$ **do**
13: u sends a dummy message to b.
14: **end for**
15:
16: **@ state PAGE SCAN:**
17: Upon receipt of a message (slave or dummy) from a neighbor v; $R \leftarrow \{R - v\}$
18: Terminate when $U = \emptyset$ and $R = \emptyset$

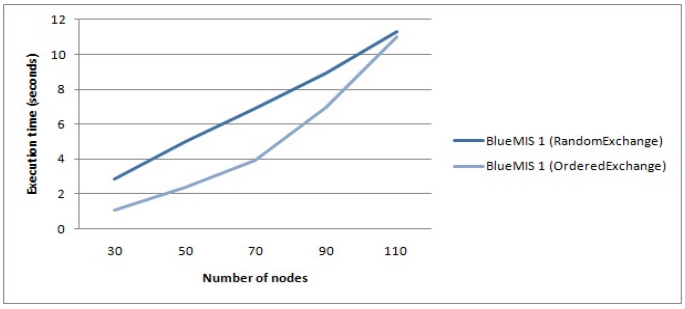

Figure 12.: Execution time of first stage of BlueMiS using RandomExchange and OrderedExchange

Algorithm 4 BlueMIS first stage based on OrderedExchange - at node u

1: u exchanges the list $N(u)$ with all its neighbors, using *OrderedExchange*.
2: u computes $N'(u)$ locally.
3: u exchange the locally constructed $N'(u)$ with all its neighbors, using *OrderedExchange*.
4: u breaks the symmetries found in $N'(u)$, that is if $u \in N'(v)$ and $v \in N'(u)$ and $v > u$ then $N'(u) \leftarrow N'(u) - v$.

Experimental results of Algorithm 3 and Algorithm 4 were obtained by means of simulation experiments. Figure 12 shows the execution time of the two algorithms. The experiments were categorized into different sized networks of number of nodes 30, 50, 70, 90 and 110. Each node was presented as a point in a unit disk graph uniformly distributed in an area of $30 \times 30m^2$. The radio of each node was set to $10m$. *OrderedExchangeCMIS* outperformed *RandomExchangeCMIS* by 172.5%, 113.56%, 74.59%, 27.5% and 2.83% in the networks of size 30, 50, 70, 90 and 110 respectively.

Note in Figure 12 that the execution time of *OrderedExchange* in the simulation experiments tends to be quadratic as the number of nodes increases. This is due to the increase of the size of the neighbor lists exchanged in the first round (see line 1, algorithm 4). This was caused by the way we implemented an *OrderedExchange* communication round in this experiments. We implemented Strategy 1 which optimizes the execution time of *OrderedExchange*. In order to do this, a node u contacts its smaller neighbors $N_s(u)$ from smallest to largest. Let $N_s(u) = \{v_1, v_2, ..., v_k\}$ be the smaller neighbors of u such that v_1 is the smallest neighbor, v_2 is the second smallest, and so on. We let a node u wait for a node v_i for a period of time t_x. In case the time period t_x is passed without u being able to contact v_i, u starts contacting $v_{i+1 mod k}$. A node v_i is not available to u only if it is busy exchanging messages with another node w. Also, the larger the size of the neighbor list exchanged between two nodes, the longer the period of time they are unavailable. This in consequence increases the proportion of neighbors of u which are unavailable. In this case, u does not contact its smaller neighbors in $N_s(u)$ by order from smaller to larger, but instead in a random manner. This leads to the use of Strategy 3 instead of Strategy 1. Recall from section 5.3 that the execution time of Strategy 3 is $\Theta(n^2)$. This explains the quadratic behavior seen in Figure 12. The experiments still show the superiority of *OrderedExchangeCMIS* in scenarios with relatively small number of nodes of 110 and less.

7. Conclusion

In this paper, we attempted to understand the impact of the INQUIRY, INQUIRY SCAN, PAGE and PAGE SCAN states on the behavior of Bluetooth networks in order to design more efficient distributed algorithms. We studied two types of algorithms: neighbor discovery and message exchange algorithms. We showed how minor changes in the Bluetooth specifications or their implementations may lead to a substantial impact on two well known neighbors discovery algorithms; namely ALTERNATE and LIM-ALTERNATE.

Then, we studied two message exchange algorithms suitable for Bluetooth networks, namely *RandomExchange* and *OrderedExchange*. We showed the superiority of the later in practical scenarios with relatively small number of nodes. We used these results to improve the execution time of a well known BSF algorithm BlueMIS.

In future work, we will focus on two questions. First, we will look at Bluetooth networks communication mechanisms using a dynamic network model, in which nodes have the ability to move. The difficulty will arise from the combination of neighbor discovery and messages exchange algorithms. Second we will evaluate other classical distributed algorithms using the model we used in this paper. For example, we will look at algorithms such as spanning trees, broadcasting, and election. In these cases, we hope to be able to optimize known solutions to fully leverage the strengths of Bluetooth networks, and end up with solutions that are significantly more adapted to this technology than the generic solutions that are found today.

References

[1] M. Albert and A. Frieze, "Random graph order," *Order*, vol. 6, no. 1, March 1989.

[2] S. Basagni, R. Bruno, G. Mambrini, and C. Petrioli, "Comparative performance evaluation of scatternet formation protocols for networks of bluetooth devices," *Wirel. Netw.*, vol. 10, no. 2, pp. 197–213, 2004.

[3] S. Basagni, R. Bruno, and C. Petrioli, "Device discovery in bluetooth networks: A scatternet perspective," in *Proceedings of the Second International IFIP-TC6 Networking Conference on Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; and Mobile and Wireless Communications*, ser. NETWORKING '02. London, UK: Springer-Verlag, 2002, pp. 1087–1092.

[4] S. Basagni, A. Faragó, M. A. Nanni, and D. T. Tran, "Increased connectivity at lower cost: The case for multi-radio nodes in multi-hop wireless networks," in *Proceedings of the 28th IEEE conference on Global telecommunications*, ser. GLOBECOM'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 3940–3945.

[5] 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," *Wirel. Netw.*, vol. 13, pp. 107–125, January 2007.

[6] 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)*, vol. 8, pp. 621–632, 2006, 10.1007/s10009-006-0014-x.

[7] R. G. Gallager, P. A. Humblet, and P. M. Spira, "A distributed algorithm for

minimum-weight spanning trees," *ACM Trans. Program. Lang. Syst.*, vol. 5, pp. 66–77, January 1983.

[8] A. Jedda, "The device discovery in bluetooth scatternet formation algorithms," Master's thesis, University of Ottawa, 2009.

[9] A. Jedda, N. Zaguia, and G.-V. Jourdan, "Analyzing the device discovery phase of bluetooth scatternet formation algorithms," jun. 2009, pp. 468–471.

[10] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer, "Fast deterministic distributed maximal independent set computation on growth-bounded graphs," in *Distributed Computing*, ser. Lecture Notes in Computer Science, P. Fraigniaud, Ed. Springer Berlin / Heidelberg, 2005, vol. 3724, pp. 273–287, 10.1007/11561927_21.

[11] F. Kuhn and R. Wattenhofer, "On the complexity of distributed graph coloring," in *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, ser. PODC '06. New York, NY, USA: ACM, 2006, pp. 7–15.

[12] C. Law, A. K. Mehta, and K.-Y. Siu, "A new bluetooth scatternet formation protocol," *Mob. Netw. Appl.*, vol. 8, pp. 485–498, October 2003.

[13] M. Liberatore, B. N. Levine, and C. Barakat, "Maximizing transfer opportunities in bluetooth dtns," in *CoNEXT '06: Proceedings of the 2006 ACM CoNEXT conference*. New York, NY, USA: ACM, 2006, pp. 1–11.

[14] M. Luby, "A simple parallel algorithm for the maximal independent set problem," in *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, ser. STOC '85. New York, NY, USA: ACM, 1985, pp. 1–10.

[15] F. Onat, I. Stojmenovic, and H. Yanikomeroglu, "Generating random graphs for the simulation of wireless ad hoc, actuator, sensor, and internet networks," *Pervasive and Mobile Computing*, pp. 597–615, 2008.

[16] C. Petrioli, S. Basagni, and I. Chlamtac, "Configuring bluestars: Multihop scatternet formation for bluetooth networks," *IEEE Transactions on Computers*, vol. 52, pp. 779–790, 2003.

[17] ——, "Bluemesh: degree-constrained multi-hop scatternet formation for bluetooth networks," *Mob. Netw. Appl.*, vol. 9, pp. 33–47, February 2004.

[18] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire, "Distributed topology construction of bluetooth personal area network," in *Proc. IEEE INFOCOM 2001*, 2001.

[19] S. Schmid and R. Wattenhofer, "Algorithmic models for sensor networks," in *In 14th International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS), Island of Rhodes*, 2006, pp. 51–54.

[20] B. S. I. G. SIG, "Does the word billion mean anything ?" Tech. Rep., Nov. 2008.

[21] ——, *Bluetooth Specifications ver4.0*, 2010.

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

[23] G. Tel, *Introduction to Distributed Algorithms*, second edition ed., 2000.

[24] Vint, "The network simulator - ns-2 - http://www.isi.edu/nsnam/ns/," Tech. Rep.

[25] Q. Wang, "Ucbt - bluetooth extension for ns2 at the university of cincinnati - http://www.cs.uc.edu/ cdmc/ucbt/," Tech. Rep.

[26] ——, "Scheduling and simulation of large scale wireless personal area networks," Ph.D. dissertation, University of Cincinnati, Cincinnati, USA, 2006.

[27] R. Wattenhofer and A. Zollinger, "Xtc: a practical topology control algorithm

for ad-hoc networks," in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, April 2004, p. 216.

[28] N. Zaguia, Y. Daadaa, and I. Stojmenovic, "Simplified bluetooth scatternet formation using maximal independent sets," *Integr. Comput.-Aided Eng.*, vol. 15, no. 3, pp. 229–239, 2008.

[29] S. Zurbes, "Considerations on link and system throughput of bluetooth networks," vol. 2, 2000, pp. 1315–1319 vol.2.

8. Appendix 1

We prove herein Theorems 5.3 and Theorem 5.4. We denote $\pi = \{x_1, ..., x_k\}$ be a longest path in G_D. Since the length of π (that is, $l(\pi)$) is in $\Theta(n)$, then we can assume that $l(\pi) = k = n/t$. Let the identifiers of the nodes be represented by $V = \{v_1, v_2, ..., v_n\}$. Let $r(v_i)$ be a function that returns the rank of a node v_i in the set V (that is, $r(v_i) = i$). Given a longest path $\pi = \{x_1, ..., x_k\}$, the rank $r(x_i)$ of node x_i is less than or equal to $n - i$ (that is, $r(x_i) \leq n - i$). We consider graphs in which each edge exist with a constant probability $p \in O(1)$ (that is, random graphs).

THEOREM 8.1. *Under $\mathcal{SYNC}'$, the worst case time complexity of* OrderedExchange *using Strategy 2 is $O(n^2)$.*

Proof. Let $T_2(n)$ represents the the time complexity of *OrderedExchange* using Strategy 2. Let x_i and x_{i+1} be any two pair of nodes in π for $1 \leq i < k$. For each x_i there exist a set of nodes $m_2(x_i)$ that x_i contacts before contacting x_{i+1}. Then:

$$T_2(n) = \sum_{x_i \in \pi} |m_2(x_i)| + 1 \tag{2}$$

Note that $m_2(x_i)$ is drawn (with probability p) from the set of all available neighbors of x_i that are smaller than both x_i and x_{i+1}. This leads to $|m_2(x_i)| = (r(x_{i+1}) - 1)p$. We know already that $r(x_{i+1}) \leq r(x_i) + 1$. Also, $r(x_i) \leq (n - i)$. This leads to:

$$T_2(n) = \sum_{x_i \in \pi} |m_2(x_i)| + 1 \tag{3}$$

$$= \sum_{i=1}^{k} (r(x_{i+1}) - 1)p + 1 \tag{4}$$

$$\leq \sum_{i=1}^{k} ((r(x_i) + 1) - 1)p + 1 \tag{5}$$

$$\leq \sum_{i=1}^{k} ((n - i) + 1) - 1)p + 1 \tag{6}$$

$$\in O(pnk) \in O(pn^2) \tag{7}$$

□

Note that the time complexity of *OrderedExchange* cannot exceed $|E|$. This can be found in the equation above, since if $p \in \Theta(1)$, then $|E| \in O(n^2)$.

THEOREM 8.2. *Under $\mathcal{SYNC}'$, the worst time complexity of* OrderedExchange *using Strategy 3 is $O(n^2)$.*

Proof. The proof is similar to the previous one. Let x_i and x_{i+1} be any two pair of nodes in π for $1 \leq i < k$. For each x_i there exist a set of nodes $m_3(x_i)$ that x_i contacts before contacting x_{i+1}. Then:

$$T_3(n) = \sum_{x_i \in \pi} m_3(x_i) + 1 \tag{8}$$

The set $m_3(x_i)$ is the set drawn with equal probability from the set of smaller neighbors of x_i except x_{i+1} (that is, $\{N_s(x_i) - x_{i+1}\}$). This makes the $|m_3(x_i)| = |N_s(x_i)| - 1)/2$. The expected value of $|N_s(x_i)| = (r(x_i) - 1)p$. This leads to:

$$\begin{aligned} T_3(n) &= \sum_{x_i \in \pi} |m_3(x_i)| + 1 && (9)\\ &= \sum_{x_i \in \pi} (|N_s(x_i)| - 1)/2 && (10)\\ &= \sum_{x_i \in \pi} ((r(x_i) - 1)p - 1/2) && (11)\\ &\leq \sum_{i=1}^{k} (n - i)p + 1 && (12)\\ &\in \Theta(pnk) \in \Theta(pn^2) && (13) \end{aligned}$$

□

Note from the equation above that $T_3(n)$ is less than $T_2(n)$.