Blue Pleiades, a new solution for Device Discovery and Scatternet Formation in multi-hop Bluetooth networks*

Devdatt Dubhashi, Olle Häggström
Chalmers University

Gabriele Mambrini, Alessandro Panconesi, Chiara Petrioli
University La Sapienza of Rome

January 11, 2005

Abstract

In this paper we introduce a novel and unified approach to the problems of device discovery and scatternet formation for the BlueTooth standard. We introduce a stochastic model for Bluetooth device discovery and prove that a protocol based on very simple local rules generates a topology that, with high probability, is connected and, crucially, has constant maximum degree. Based on this, we develop a new protocol for device discovery and scatternet formation for multi-hop BlueTooth networks. By means of extensive ns2 simulations we show that our solution is simple to implement, fast and requires low overhead, both for the device discovery and the scatternet formation phases, and leads to better performance when compared to the major approaches so far proposed in the literature.

1 Introduction

The Bluetooth (BT) technology, as described in the Specifications of the Bluetooth System Version 1.1 is one of the most promising enabling technologies for pervasive and ubiquitous computing. In this paper we provide basic results to the fundamental problems of *device discovery* and *scatternet formation*, i.e. on how Bluetooth nodes can become aware of their neighbors (device discovery), can partition themselves into groups, called *piconets*, and finally on how such groups can be joined together to form a connected multi-hop ad hoc network, the so-called *scatternet*.

To describe our contribution we first review the two main unsolved problems in this context. A more thorough discussion of the existing literature is deferred to the next section.

The first problem concerns the device discovery phase. Unlike other standards, Bluetooth requires that network nodes become aware of their neighbors (i.e. nodes within transmission range). Once two neighbors become aware of each other a reliable communication channel between them can be established. The resulting topology is called the *BT topology*.

As shown in [3], requiring each node to discover *all* neighbors is too time consuming. Therefore existing approaches have relied on time-outs. Their goal is to discover *enough neighbors* so that the resulting BT topology, and hence the resulting scatternet, is connected. Fortunately, as shown

*Research supported by EC projects MAGELLAN and DELIS.

in [12], reasonable time-outs– with each node seeking neighbors for about 8 seconds– give good statistical guarantees that the BT topology be connected. However, criteria like these, based on absolute time limits, are not satisfactory because they offer a reliable guide only under normal conditions. They are too dependent on the current state of the technology and on unpredictable external factors such as the actual level of interference.

Thus, one would like to develop more robust criteria for device discovery.

The second problem is also related to the Bluetooth standard. In the scatternet formation phase, the BT topology is divided up into a set of so called *piconets*. The piconets are then connected in a resulting *scatternet*. A piconet consists of a *master* and of some of its neighbors in the BT topology, the so-called *slaves*. The master is in charge of the communication within each piconet. This simple "one-hop" network, the master's piconet, may include several slaves, but no more than 7 of them can be actively communicating with the master at the same time. If a master has more than 7 slaves, some slaves have to be *parked*. To communicate with a parked slave a master has to *unpark* it, while possibly parking another slave. This parking-unparking procedure results in time-consuming operations and in communication overheads so that it is highly desirable to avoid it.

In this paper we show how these two problems can be solved simultaneously by means of simple, fast, and cost-effective protocols.

The solution is based on a simple geometric random graph model that abstracts away some of the essential features of the BT device discovery mechanism. In this model, each node tries to establish a link with c other nodes chosen uniformly at random within transmission range, for a total of ρ tries. A node that already is connected (has links to) c^* other nodes will neither try nor accept any more connections. This model is based on the fact that the Bluetooth device discovery machanism does select neighbours at random within transmission range and, as a first approximation at least, it treats all potential neighbors uniformly. We show that when the nodes are distributed uniformly, the above process generates with high probability a connected network. Crucially, both c and c^* can be constant and $\rho = O(\log n)$, where n is the number of nodes. The bound on ρ is a time bound on the running time of the protocol.

Thus the following protocol for Bluetooth device discovery suggests itself: as soon as a node has discovered c neighbors, it proceeds to the next phases of piconet formation and interconnection; nodes with less than c neighbors will exit device discovery upon timeout expiration. We remark that our theoretical result is likely be useful in several different contexts. With hindsight it also explains the results of [12]. Note however that running the device discovery with time-outs in itself does not ensure a constant maximum degree, unlike our new device discovery procedure.

In spite of the fact that our simplified model only approximates how a device discovery scheme actually works [15], the resulting criterion adopted for the device discovery appears to be robust and effective in pratice. In fact, extensive simulations show that $c = c^* = 6, 7$ are excellent choices that guarantee the connectivity of the BT topology with high probability.

Once the device discovery has been run, to build an ad hoc network the BT nodes have to be partitioned into piconets and such piconets have to be interconnected. In the specifications, this is implemented by allowing each node to assume multiple roles. A node can be a master in one piconet and a slave in one or more other piconets, or a slave in multiple piconets. Devices with multiple roles act as *gateways* to adjacent piconets thus *forming* a scatternet, which is the basic routing infrastructure of a Bluetooth wireless network.

Another contribution of this paper is the design of an effective solution for scatternet formation.

We will show that if we combine our new device discovery with the BlueStars protocol introduced in [13], we obtain a simple, fast and effective scatternet formation protocol that overcomes the limits of the previous solutions. This protocol is dubbed here the *Blue Pleiades* protocol.

A particularly nice feature of Blue Pleiades is that it avoids completely the expensive parking and unparking procedure, for our new device discovery guarantees that no node has more than 7 neighbors in the BT topology. The existence of high degree nodes in the BT topology was essentially the major problem of the solution in [13]. In spite of this drawback, the protocol in [13] was proven superior to all other existing solutions under a host of relevant metrics. We will show that Blue Pleiades, while retaining all the good features of [13], completely avoids the problem. Furthermore it also exploits the advantages of the sparser BT topology for faster operations. Indeed, the thorough ns2 based comparative performance evaluation presented in this paper shows that Blue Pleiades is by far the best performing protocol among the major solutions so far proposed for scatternet formation in multi-hop Bluetooth networks. We remark that Blue Pleiades is a very simple protocol and, unlike others, it does not require any extra-hardware such as GPS positioning devices.

The paper is organized as follows. In the next section we review the major schemes proposed in the literature for device discovery and scatternet formation. Then, in Section 3 we discuss the theoretical underpinnings of our new device discovery approach. In Section 4 the Blue Pleiades protocol is introduced, together with the description of the two scatternet formation protocols selected as benchmarks, BlueStars [13] and Lin Stojmenovic BlueStars (LSBS) [10]. In Section 5, by means of a thorough ns2-based comparative performance evaluation, we will be able to quantify the advantages of the new device discovery approach and of Blue Pleaides over the solutions previously proposed in the literature. We end the paper with some remarks in Section 6.

2 Related Works

The problem of device discovery in Bluetooth networks is one of the first addressed in the literature on Bluetooth. However, all the solutions tackling this problem for multi-hop scenarios are variants of the first protocol proposed in [15].

This scheme was originally designed for the single-hop case. As all devices are within transmission range, only one of the nodes that have discovered each other remains in the discovery phase. This results in a reduced number of neighbors that have to handshake. Among the single-hop scatternet formation protocols using a variant of this scheme (e.g. [15], [8], [5]), [15] uses it for leader election and for gathering at the leader information on the network topology. In [5] and [8] the network is partitioned in components (originally a single device, then a piconet, or a connected scatternet), each of which has a leader. The leader is the only node which survives in the device discovery phase. This is sufficient to allow different components to discover each other, and to be progressively joined to form the final connected scatternet. The critical assumption here is that each node can exit the device discovery phase without affecting the connectivity of the resulting scatternet, provided that someone else among its discovered neighbors (or among the nodes in its component) takes care of device discovery on his behalf.

In the more general multi-hop scenario this is not the case and each node should in general be able to discover *all* its neighbors for the protocol to correctly operate. When applied to multi-hop Bluetooth networks the device discovery scheme presented in [15] still correctly works, statistically allowing each pair of neighbors to be in opposite inquiry modes for enough time to handshake.

However, the time needed for the BT devices to discover *all* their neighbors is far too long to be feasible. For example, after 20s of device discovery only 88% of a node neighbors are discovered in the case of 110 Power Class 3 nodes randomly scattered in an area of square $30m$.

In [3] the reasons for the device discovery being so inefficient have been studied. By means of a thorough ns2 based performance evaluations the authors identified at least two major features of the BT specifications that are responsible for this lack of performance. These are: the overly long backoff interval that adds considerable time to every handshake, and the impossibility of the node in inquiry mode to identify itself in the ID packet, resulting in handshakes between nodes that have already discovered each other. [1]

Changing any of these two features would result in significant improvements. The bottom line is that the need for nodes to be in opposite inquiry modes to communicate, which translates into nodes randomly alternating between inquiry and inquiry scan mode to be able to handshake, requires some time before the different nodes can have the possibility to get in contact.

In [13] it was stressed how discovering the full one-hop neighborhood is not needed for a node to correctly run a scatternet formation protocol in multi-hop BT scenarios. Indeed, for the resulting scatternet to be connected it is sufficient that enough neighbors are discovered so that the global connectivity of the network is maintained in the BT topology. Operating a scatternet formation protocol on a sparser BT topology may also have the advantage of scatternets with a reduced number of slaves per piconet, and of faster protocol operations. The solution proposed in [13] is to set a timeout on the device discovery length, tuned to around 8s, so that the resulting BT topology is connected with high probability. This also provides a tuning knob to limit the size of the generated piconets, which is however unable to provide that strict bound (8 devices) required by the BT technology to operate without the need to park/unpark slaves. A major limit of this solution is that the criterion to exit the device discovery after $8-10$s is based on empirical observations, and may be affected by vagaries of the wireless channel such as the level of interference.

As remarked in the introduction, in this paper we introduce a different approach, based on statistical properties of random geometric graphs. Random geometric graphs have been studied rather extensively both as geometric objects (see for instance the book [11]) and in the context of percolation (see the books [7, 14]). To the best of our knowledge however the model used in this paper has not been studied before. A seemingly related model that has received attention in the context of wireless networking is the *k-nearest model* in which n points are distributed uniformly at random within a unit area and each point is connected to the k nearest points. Although vaguely similar, the two models are rather different. For instance, for the k-nearest neighbour model it is known that for connectivity it is required that each point connects to $\Omega(\log n)$ points [17]. The asymptotic proofs presented in this paper show that, in contrast, a constant number of

[1] Based on the BT specifications, for two nodes to discover each other they have to be in opposite inquiry modes: one has to be in *inquiry mode*, willing to discover its neighbors, and the other one has to be in *inquiry scan* mode, willing to be discovered. The node in inquiry mode (i.e. the *inquirer*) transmits short ID packets, asking the other node to identify itself and provide the clock information needed for a link set up at a later time. ID packets are sent scanning a predefined inquiry frequency hopping sequence at a very fast rate (one hop every half a slot, with the slot time being $625\mu s$). The BT device in inquiry scan mode instead scans the inquiry frequency hopping sequence at a very low rate (once every 1.28s). Thus it does not take long until the node in scan mode receives the ID packet. However, it won't immediately answer it. It will select randomly a time in a 1024 slots *backoff interval*. Only when the backoff timer expires it will listen again to the channel and, upon ID packet reception, will provide its id and clock information. This reduces the probability of collisions among two nodes which receive the ID packet and are scanning on the same frequency. No information is provided by the inquirer to the node in inquiry scan.

connections suffices in our model.

Out of the major solutions so far proposed for scatternet formation in multi-hop Bluetooth networks, [19], [10], [13], [18] and [12] [4], all but [4] rely on the device discovery scheme proposed in [15] to identify a BT topology before the scatternet formation protocol can be run.

In particular, the solution presented in [10] is similar to what is proposed in this paper. The aim of the authors is to build up a connected scatternet in which each piconet has no more than 7 slaves. To this purpose, degree reduction techniques (the Yao construction, originally proposed by Andrew Yao in 1982) are initially applied to the BT topology graph to reduce the number of links at each node to less than 7 without affecting the connectivity of the resulting topology. Each node divides the surrounding plane into k equal angles. In each angle, the node chooses the closest neighbor u, if any. (Ties are broken arbitrarily.) A link between nodes v and u survives the Yao construction phase if and only if v has chosen u and vice versa. All other links are deleted. A cluster-based scatternet formation protocol (which is left unspecified) is then executed on the resulting topology to generate a connected scatternet with no more than 7 slaves per piconet.

However, the degree-reduction technique requires each node to be aware of its own location in the plane (usually provided by an on-board GPS device, or by any suitable inertial positioning system device). Beyond being potentially expensive, this solution is not feasible when such extra hardware is not available. Also, the approach relies on the assumption to operate on the visibility graph, i.e. on the graph where there is a vertex for each BT device, and an edge between each pair of devices that are in each other transmission range, which is modeled as an Unit Disk Graph (UDG), i.e. a graph having a node for each BT device and a link for each pair of BT nodes whose Euclidean distance is no longer than the transmission range. When not all the links of the visibility graph are discovered, as it is the case in practice, extra phases (called *link replenish* phases in the following) must be run before the Yao construction can be correctly operated. Overall this results in complexity, a time and message consuming approach.

The protocol described in [13], termed *BlueStars*, is a fast and very simple clustering scheme to form connected scatternets without using any extra hardware. This solution has the nice feature to select the piconets' masters based on how "fit" a node is to serve as a master (e.g. based on its available energy resources). Once piconets are formed, gateway nodes are selected so that there is an inter-piconet route between all masters that are at most tree hops away (i.e., all adjacent piconets are interconnected). This condition ensures the connectivity of the BlueStars scatternet [6]. However, BlueStars has the major limit not to provide any bound on the piconet size, resulting in the need to park/unpark slaves. When combined with the degree reduction techniques proposed in [10], solution referred to as *Lin-Stojmenovic BlueStars* or LSBS, such limit can be leveraged. The price to pay is the additional complexity, time and overhead needed to run the degree reduction techniques.

Other solutions that successfully limit the number of slaves per piconet are *BlueNet*[18], *BlueMesh*[12], and *BlueTree*[19]. The scatternet formation scheme proposed in [18], *BlueNet*, produces a scatternet whose piconets have a bounded number of slaves: in building a piconet each master can select no more than k slaves. However, no mechanism is introduced to guarantee the global connectivity of the network. Partitioning devices in multiple connected components is a problem from which BlueNet suffers, expecially in sparse to moderately sparse scenarios. Also, the protocol is quite complex, relying on several phases, and time-demanding. The protocol duration is significantly higher than that of other solutions which require extra phases to correctly operate such as the LSBS protocol. A final remark on BlueNet is that it requires nodes to assume several roles (as it divides nodes in several small piconets, that are interconnected only through a high number of extra piconets), which are often Master/Slave roles. This in turns leads to inefficiencies in the scatternet operation, due to the wasted slots associated with each piconet switch, and to the need to freeze all the communications in the piconet of a master which is currently acting as slave in a different piconet.

The scatternet formation protocol described in [19] requires that the protocol is initiated by a designated node (the *blueroot*) and generates a tree-like scatternet in a "wave expansion" fashion. The blueroot starts acquiring as slaves its one hop neighbors. These, in turn, become master themselves and invite their neighbors to join their piconets, and so on. The connectivity of the resulting scatternet is guaranteed (provided that the BT topology on which the protocol operates is connected). Though the basic protocol does not limit the piconet size, a procedure based on geometric properties of networks of devices scattered in the plane is described to re-configure the tree so that each master has no more than seven slaves. However, as in LSBS, BlueTree needs to run extra phases aiming at "bringing back" some links that were not discovered during device discovery, to achieve the consistent view of the neighbors needed for the protocol operation, and in particular for a correct re-configuration of the built scatternet. This adds in time, complexity and overhead. BlueTree has also the disatvantage of not being resilient to disconnections (in case the network is divided in multiple connected components, only the one where the BlueRoot resides can build a scatternet), of requiring all internal nodes to assume Master/Slave roles, and of generating a tree-like scatternet, less robust and efficient than a mesh.

Recently, a new protocol that produces connected mesh-like scatternets with no more than 7 slaves per piconet has been introduced in [12], termed *BlueMesh*. Differently from [10], no location information is needed. In order to form a connected scatternet, the protocol proceeds in successive iterations through which connectivity is achieved progressively. The protocol is fully distributed, resilient to disconnections, guarantees connectivity of the resulting scatternet and, as BlueStars and LSBS, performs well in terms of the number and type of roles assumed by each node. The price to pay over LSBS is that BlueMesh still needs to run the links replenish extra phases needed by LSBS and BlueTree, with the additional complexity of the protocol, which requires some iterations to converge and is therefore time and message consuming itself.

All the mentioned solutions, as well as the one proposed in this paper, refer to the problem of *setting up* scatternets from a set of nodes spread in a geographic area. The problem of *maintaining* scatternets in presence of dynamic addition/removal of nodes, and of nodes mobility has just started to be addressed for multi-hop topologies. (A first solution providing a preliminary account on how to deal with changing network topology has been recently presented in [4].)

In [3], the performance of BlueStars, LSBS, BlueNet and BlueTree have been compared by means of simulations. The results show that BlueStars has by far the best performance in terms of time needed for the scatternet formation, protocol overhead, limited number of generated piconets, efficient distribution of roles to nodes (e.g. reduced use of master/slave roles, limited number of roles per node), short routes length. However, it fails to provide a bound on the number of slaves per piconet, which leads to inefficiencies during the scatternet operation. Among the solutions that provide a bound on the number of slaves per piconet, LSBS is a good compromise leading to good performance. However, the price to pay is extra complexity, the need for all nodes to be aware of their position (e.g. by means of GPS), feature which may not be available, and a significant protocol overhead. Based on these results BlueStars and LSBS have been selected as benchmarks for sake of comparison in this paper.

These results also suggested the potential benefit of designing a device discovery able to directly result in a connected, sparse BT topology, with nodal degree below 7. This would allow to run BlueStars directly, without any extra phase (as instead required in LSBS), maintaining its good features and simplicity. If the method used to sparsify the visibility graph does not rely on additional information, such as the nodes location, then also the need for extra hardware (required by LSBS) would be leveraged. The design of such device discovery is the topic addressed in this paper.

3 Models and Proofs for Device Discovery

In this section we introduce a simple model that abstracts away some of the important features of the Bluetooth device discovery mechanism. This theoretical work lays the foundation for the following sections, where it is shown experimentally that device discovery protocols for Bluetooth based on the algorithms presented here are very effective in practice. During device discovery each node scans its own neighborhood looking for other nodes. When nodes become aware of each other a reliable communication link is established. This device discovery process is random in several respects. For instance, in order to avoid collisions nodes transmit packets by introducing random delays. As a first approximation however the Bluetooth discovery process appears to act uniformly within the transmission radius of a node. This motivates the following definition. The experimental results of the forthcoming sections will vindicate our simplifying assumptions.

Definition 1 *Fix $r > 0$ and positive integers $\rho \geq 1$ and $1 \leq c \leq c^*$. We take $G^n_{r,c,c^*,\rho}$ to denote the geometric random graph defined as follows.*

- *The vertex set consists of n points, picked independently according to the uniform distribution on $[0,1]^2$.*
- *The edges are connections formed by the following asynchronous process: each vertex attempts to connect to c nodes chosen uniformly at random among those within distance r. A vertex which is requested connection declines it if its degree already is c^*. A vertex attempting to connect can make ρ tries for each of its c connections.*

Besides providing a simple model for device discovery, this definition embodies a very simple distributed algorithm to set up a sparse overlay sub-network of the full visibility graph (in which two nodes are connected whenever their distance is at most r). As such it is likely to be useful in other contexts too. In many applications, including Bluetooth, it is crucial to enforce hard degree constraints which is why this feature is explicitly incorporated in the above model. Note that the choices can be performed in a completely asynchronous fashion and take a logarithmic number of steps if $\rho = O(\log n)$ as will be in the following.

For this model we prove the following.

Theorem 2 *Fix $r > 0$ and $c \geq 3$. Set $c^* := \frac{737280}{r^2}c$, and $\rho := \gamma\lceil\log n\rceil$ for a constant $\gamma > 0$. Then, for any $t \geq 1$ and for sufficiently large n,*

$$\mathbf{P}(G^n_{r,c,c^*,\rho} \textit{ is connected}) \geq 1 - O\left(n^{-t}\right). \tag{3.1}$$

Hence,

$$\lim_{n\to\infty} \mathbf{P}(G^n_{r,c,c^*,\rho} \textit{ is connected}) = 1\,. \tag{3.2}$$

Thus, somewhat surprisingly, it is possible to generate connected networks having *constant* maximum degree. Moreover the number of steps taken by each node is only logarithmic in the size of the network. The constant c^* in Theorem 2 is astronomically large. One reason is that in the proofs we use very crude pessimistic estimates and we have not made any great attempt to optimize the constants involved. Rather than looking for the best constants, which could complicate the argument still further, we prefer to establish the fact that the device discovery procedure does generate constant degree connected networks and demonstrate later by means of experiments that the degree is small. Another problem is we take a worst-case approach covering all possible scenarios. The system is asynchronous and can evolve in many different ways. For instance some nodes could be much faster than others. We show high-probability of connectivity *regardless* of the way in which the node operations are scheduled. The model however suggests that *typically* much smaller constants are likey to work. Our implementations will be based on this model, where we optimistically set $c = c^* = 7$ and a (small) time-out is used instead of ρ. As we will see the resulting protocol works very well, generating connected networks very quickly and reliably. In short, the theoretical analysis suggests why in practice a smaller constant would work quite well and we validate this prediction by means of extensive simulations.

We begin by giving an outline of the proof, which proceeds in two steps. First, we show that each vertex is contained in a giant component. The basic tool we use here is the theory of branching processes. Another auxiliary tool that is used repeatedly is stochastic domination. These tools are reviewed briefly in the sequel, with pointers given to the relevant literature. Once the formation of giant compononents is established, the proof of connectivity follows readily. Each vertex is in a giant component, and these components have so many potential links among them that, with high probability, some of them will be established, to connect any two giant components, resulting in a connected network globally.

We begin the proof by reviewing some well-known probability inequalities.

Proposition 3 (Markov's inequality) *Let Ω be a finite set and $f : \Omega \to R^+$ be a non-negative function with average value $\overline{f} := \frac{1}{|\Omega|}\sum_{\omega\in\Omega} f(\omega)$. Then for any $t > 0$,*

$$|\{\omega \in \Omega \mid f(\omega) > t\overline{f}\}| \leq \frac{|\Omega|}{t}.$$

Proposition 4 (Chernoff-Hoeffding Bounds) *Let $X := \sum_{i=1}^{n} X_i$ be the sum of identically distributed, binary random variables X_i such that $\Pr[X_i = 1] = p$ for all i. Then,*

$$\Pr[|X - pn| \geq \delta n] \leq 2\exp(-2\delta^2 n).$$

Let k be an integer such that

$$\frac{\sqrt{5}}{r} < k \leq \frac{4}{r} \tag{3.3}$$

and partition $[0, 1]^2$ into k^2 subsquares of size $\frac{1}{k} \times \frac{1}{k}$ in the obvious way. One point of this choice of k is that it ensures that any two points sitting in adjacent subsquares are within distance r from each other.

The following lemma will be repeatedly used. It says that the points distribute evenly within the area.

Lemma 5 *Let $A \subseteq [0,1]^2$ be an arbitrary (measurable) subset. Let $|A|$ denotes the area of A and let $\#A$ denote the number of points contained in A. Then,*

$$\mathbf{P}\left(\frac{n|A|}{2} \leq \#A \leq 2n|A|\right) \geq 1 - O(e^{-\delta n}) \tag{3.4}$$

In particular, if S denote any subsquare,

$$\mathbf{P}\left(\frac{n}{2k^2} \leq \#S \leq \frac{2n}{k^2}\right) \geq 1 - O(e^{-\delta n}) \tag{3.5}$$

for some $\delta > 0$.

Proof. We have $E[\#A] = n|A|$ and the result follows by applying the Chernoff-Hoeffding bounds (Proposition 4). $\bigtriangledown$

To investigate the connected components of $G^n_{r,c,c^*,\rho}$ we shall employ the following method, which we will call the **sequential discovery procedure** (this is simply a breadth-first exploration). First, select a node v_0 at random (among all n nodes). Then consider the connections that are established by v_0 in the device discovery procedure of Definition 1, and denote the endpoints (other than v_0) of these edges by $v_1, \ldots, v_k$, with $k \leq c$. Then continue with the edges chosen by v_1, and so on, in a breadth first search manner. Each time a new node is encountered, the node reached by it is included in our list of nodes, and the choice is deemed a **success**. Sometimes, the edge leads to a node already seen in this procedure, in which case the choice is said to be a **failure**. At any point of this search procedure, we may stop, and those vertices encountered whose outgoing edges have not been investigated (yet), are called **fresh** nodes.

At various stages of our arguments, we will invoke a comparison between the sequential discovery procedure and a (Galton–Watson) **branching process**. Such a branching process (see, e.g., Harris [9] or Asmussen and Hering [2]) begins with m_0 individuals. Each of these begets, independently of the others, a number of offspring, which has some given distribution f on the non-negative integers. Each of these children then has a number of children for themselves, again independent with distribution f. And so on, again in a BFS manner. One of two things will happen: either the branching process dies out after a finite number of generations, or it survives (forever). Excluding the trivial case where f puts unit mass on 1, it is well known that the branching process has positive probability of surviving if and only if f's first moment is strictly greater than 1.

We shall be particularly concerned with a branching process whose offspring distribution is the binomial distribution $\mathrm{Bin}(2,p)$. This can be compared to the sequential discovery procedure for $c = 2$ in the following way. Suppose that we can show that up until some given stage S of the sequential discovery procedure, each choice of a new node to connect to has probability at least p (conditionally on everything seen so far) of being a success. Then we can make a joint construction (a so-called coupling; see [16]) of the sequential discovery procedure and the $\mathrm{Bin}(2,p)$ branching process in such a way that each individual in the branching process corresponds to a particular node (not shared by any of the other individuals of the branching process) of the sequential procedure, up until the given stage S. We say in this case that the sequential procedure up until stage S **stochastically dominates** the branching process. We will show that the sequential discovery procedure first generates almost surely a set of $\log n$ points and that from then on each point u begets offsprings with distribution $\mathrm{Bin}(2, p_u)$, with $p_u \geq \frac{3}{4}$. It follows from a standard application of stochastic domination that the survival probability of the sequential discovery procedure is at least that of $\log n$ independent branching processes with distribution $\mathrm{Bin}(2, \frac{3}{4})$. In this paper we use stochastic domination repeatedly and refer the interested reader to [16].

We now establish that every vertex is in a giant component.

Definition 6 *Let $s \in (0, 1]$ be some fixed constant. An s-*giant component *of an undirected graph G with n vertices is a connected component of G containing at least sn vertices.*

We first start with the following proposition which asserts that each vertex is in a s-giant component for some $s > 0$.

Proposition 7 *Fix $r > 0$, $t \geq 1$ and $c \geq 2$ and let $c^* := 1152c$ and $\rho := \gamma\lceil \log n \rceil$ for a constant $\gamma > 0$. Then, there is a constant $s > 0$ such that*

$$\mathbf{P}(\text{every node of } G^n_{r,c,c^*,\rho} \text{ is in some } s\text{-giant component}) \geq 1 - O\left(n^{-t}\right)$$

Hence,

$$\lim_{n\to\infty} \mathbf{P}(\text{every node of } G^n_{r,c,c^*,\rho} \text{ is in some } s\text{-giant component}) = 1\,.$$

(In fact we can take $s := \frac{1}{20k^2}$ where k is defined by (3.3).)

In the proof of this proposition and subsequently, we will make repeated use of the following lemma which asserts that at any stage, for a vertex attempting to make connections, there are always plenty of low degree vertices to connect to.

Lemma 8 (Average Degree) *Let $\alpha > 0$ and let P be any set of at least αn points contained in a subsquare S. Then the average degree of a vertex in P at any stage of the device discovery procedure of Definition 1 is at most $\frac{2R^2}{\alpha}c$ where $R := (2 + \frac{1}{\sqrt{5}})r$. Consequently, the number of vertices in P of degree not exceeding $\frac{4R^2}{\alpha}c$ is st least $\frac{\alpha}{2}n$.*

Proof. Consider a square S' of side R centered on S. By Lemma 5, the number of vertices in S' is at most $2R^2n$ with high probability. The total number of edges at the end of the procedure generated by vertices in S' is $2R^2nc$ and these are the only edges which could posibly connect to a vertex in P. Hence the average degree of a vertex in P is at most $\frac{2R^2}{\alpha}c$. The second assertion follows by Markov's inequality (Proposition 3). $\triangledown$

Proof. (Of Proposition 7) We prove the result for $c = 2$ only, which is obviously enough since adding edges is not going to destroy a giant component

Let $a > 0$ be a constant to be chosen below and run the sequential discovery procedure until the outgoing edges of $\ell := a \log n$ nodes are investigated (or until there are no more fresh nodes, in which case we are stuck).

Let us consider a vertex at some stage in the sequential discovery procedure. We will define the vertex to be "successful" if it connects to a node that is "fresh", and, in addititION, if the vertex it connects to has degree at most c^*. Let us estimate the probability of a successful connection. By Lemma 5, we may assume that the event in (3.5) happens, and condition on that event.

First, let us estimate the probability that a vertex succeeds in connecting to a vertex of low degree. Consider a vertex in a subsquare S making connections during the sequential discovery procedure. By lemma 8, if we set $\alpha = 1/2k^2$, the number of vertices in S of degree not exceeding $8k^2R^2 \leq c^*$ is at least $\frac{n}{4k^2}$. Thus, the probability of successfully connecting to a vertex of degree at most c^* is at least $p_1 := \frac{1}{8\pi k^2 r^2}$. Hence there exists a constant $\gamma > 0$ such that after $\gamma \lceil \log n \rceil$ attempts, the probability of failure is at most $\frac{1}{n}$.

Now let us consider the event that when a vertex has made a successful connection to a low degree vertex, it does so to a fresh node. Again, by Lemma 8, the number of vertices in S of degree at most c^* is at least $\frac{n}{4k^2}$. At this stage, there are at most 2ℓ nodes that have already been seen, so each edge has probability at most

$$\frac{8k^2\ell}{n} \tag{3.6}$$

of hitting a node that has already been seen, conditioned on connecting to a low degree vertex.

Taken together, the probability of failing to connect successfully is at most

$$\frac{1}{n} + \frac{8k^2\ell}{n} \leq \frac{9k^2\ell}{n}.$$

Now we estimate the probabilility that at least $t+1$ of the 2ℓ choices was a failure. Recalling the bound

$$\binom{n}{k} \leq \left(\frac{en}{k}\right)^k$$

the probability is at most,

$$\binom{2\ell}{t+1}\left(\frac{8k^2\ell}{n}\right)^{t+1} \leq \left(\frac{16ek^2}{t+1}\right)^{t+1} \frac{\ell^{2(t+1)}}{n^{t+1}} = O(n^{-t}). \tag{3.7}$$

Hence, we have shown that with probability approaching $1 - O(n^{-t})$ we get a connected component with at least $a\log(n) - t + 1$ nodes. But this is not enough to prove Proposition 7, which asserts a component whose size is *linear* in n.

We can, however, continue the sequential discovery procedure from the $a \log n - t + 1$ fresh nodes that we have until the stage S when either a total of $\frac{n}{20k^2}$ nodes have been found (or no fresh nodes remain). Before stage S, the failure probability analogous to (3.8) is estimated by summing the probability of failing to connect to a low degree vertex after ρ tries and the probability of failing to connect to a fresh vertex. By an analogous argument as that used to establish (3.6), the probability of not connecting to a fresh vertex, conditioned on connecting to a low degree vertex, is at most

$$\frac{n/20k^2}{n/4k^2} = \frac{1}{5}. \tag{3.8}$$

Hence the total probability of failure is at most

$$\frac{1}{n} + \frac{1}{5},$$

which, for $n \geq 20$, is at most $\frac{1}{4}$.

It follows that the sequential discovery procedure starting from the $a\log(n) - t + 1$ nodes until stage S stochastically dominates a $\mathrm{Bin}(2, \frac{3}{4})$ branching process with the same initial number of individuals. We therefore get, conditionally on at most t failures associated with the first $a\log(n)$ nodes,

$$\begin{aligned}
&\mathbf{P}(\text{the sequential procedure fails to survive until } \tfrac{n}{20k^2} \text{ nodes are found})\\
&\quad\leq\ \mathbf{P}(\text{a Bin}(2,\tfrac{3}{4}) \text{ branching process starting with } a\log n - t + 1 \text{ individuals dies out})\\
&\quad=\ (\mathbf{P}(\text{a Bin}(2,\tfrac{3}{4}) \text{ branching process starting with 1 individual dies out}))^{a\log(n)-t+1}\\
&\quad=\ (1-\alpha)^{a\log n - t + 1}\\
&\quad\leq\ n^{a\log(1-\alpha)-t+1} \qquad (3.9)
\end{aligned}$$

where $\alpha > 0$ is the survival probability of a $\text{Bin}(2,\frac{3}{4})$ branching process starting from a single individual (an easy calculation shows that $\alpha = \frac{8}{9}$, but we only need the fact that $\alpha > 0$, which follows from the fact that the offspring distribution has expectation $\frac{3}{2} > 1$). The sum of (3.7) and (3.9) is $O(n^{-t})$ as long as we choose a so that $a\log\left(\frac{1}{1-\alpha}\right) > 2t - 1$ whence Proposition 7 follows with $s = \frac{1}{20k^2}$, and we are done $\triangledown$

We now proceed towards the proof of full connectivity.

Definition 9 *A s-giant component C (for some $s > 0$) is said to have a* **core** *in a subsquare S if $|S \cap C| \geq \frac{1}{40k^2}|S|$.*

We start with some easy observations about giant components and cores:

Observation 10 *Let $s = 1/20k^2$ as in the statement of Proposition 7. Then, the following holds:*

1. *There are at most $\frac{1}{s}$ s-giant components.*
2. *Each s-giant component has at least one core.*
3. *Each subsquare has at least one core.*

Proof. The first claim is clear. The second follows from a simple averaging argument: At least one of the k^2 sub-squares contains at least $\frac{n}{20k^4}$ nodes of that connected component, and this square has at most $2n/k^2$ vertices. The third claim too follows from an averaging argument– there are at most $1/s$ components, so in any given subsquare, one of them must have at least $s := \frac{1}{20k^2}$ fraction of the vertices. $\triangledown$

Proof. (of Theorem 2) Expose, in arbitrary order, $c - 1 \geq 2$ edges around each vertex first. By Proposition 7,

$$\mathbf{P}(\text{every node of } G^n_{r,c,c^*,\rho} \text{ is in some } s\text{-giant component}) \geq 1 - O\left(n^{-t}\right)$$

We condition on this event and then expose the remaining edge around each vertex.

Consider the graph G' whose vertices are cores and where two cores are neighbours if there exist two vertices, one in each core, that are connected by an edge. To show $G^n_{r,c,c^*,\rho}$ is connected, it is enough to show that G' is connected

To show that G' is connected, it suffices to show that an arbitrary core is connected to all other cores in its own sub-square and in each neighbouring sub-square. Consider a core P and a different core P' that is either in the same sub-square or an adjacent sub-square. Consider the indicator random variables $X_v, v \in P$ which is 1 if v connects to a vertex in P' and 0 otherwise. We consider $\sum_{v \in P} X_v$ and argue this cannot be too small.

By Lemma 8, the number of points of P' of degree at most $\frac{4R^2}{1/80k^4}c \leq c^*$ is at least $\frac{n}{160k^4}$. Hence, the probability that a vertex in P successfully connects to a vertex in P' is at least the constant

$$p_2 := \frac{n/160k^4}{2\pi r^2 n},$$

regardless of any other connections made previously. The variables $(X_v, v \in P)$ therefore stochastically dominate an independendet set of variables $(X'_v, v \in P)$ with $\mathbf{P}[X'_v = 1] = p_2$. Hence the expected number of edges from a vertex in P to one in P' is at least $\frac{p_2}{80k^4}n$. Now we conclude the proof by applying the Chernoff-Hoeffding bounds (Proposition 4). The probability that there is no such edge is at most $O(e^{-\delta n})$ for some $\delta > 0$. Since there are a constant number of cores, the probability that G' is not connected is also $O(e^{-\delta n})$ (conditioned on the appearance of the giant components previously) $\bigtriangledown$

4 The BlueStars, LSBS and Blue Pleiades protocols

In this section we first review the operations of two of the scatternet formation protocols so far proposed for multi-hop BlueTooth networks: BlueStars [13] and the LSBS protocol [10]. These protocols will be used as benchmarks in the Blue Pleiades protocol comparative performance evaluation and are the basis for the design of the Blue Pleiades protocol itself. Both protocols rely on the device discovery protocol proposed by Salonidis et al. in [15]. We then describe the Blue Pleiades protocol operations.

4.1 BlueStars

The protocol presented in [13], *BlueStars*, proceeds from the device discovery phase into the following two phases of piconet formation and of their interconnection into a connected scatternet. Based on a locally and dynamically computed weight (a number that expresses how suitable that node is for becoming a master, e.g. taking into account the residual energy, or the type of device) and on the knowledge of the weight of its neighbors (obtained during the discovery phase) each node decides whether it is going to be a master or a slave. This decision is taken at a node depending on the decision of the bigger neighbors, and then communicated to the smaller neighbors. The mechanism through which this is implemented is similar to the pecking protocol described below, which is a generic method to establish an ordering among the nodes according to their "weight," i.e., according to a number ≥ 0 locally computed at each node. A node v checks whether it has the bigger weight among its neighbors $N(v)$. If this is the case, that node, called in the rest of the paper an *init* node, executes the following procedure.

```
PeckOrder(v)
1  PageMode
2  for each smaller u in N(v)
3      do Page(u, v)
4  exit
```

An init node goes to page mode and starts paging all its neighbors (which, by definition, are "smaller neighbors," i.e., nodes with a smaller weight) setting up temporary piconets with each

one of them. While the piconet is up, the nodes exchange the information needed for the protocol operation. Then, the piconet is disrupted.
A non-init node u executes the following procedure.

```
SubNode(u)
PageScanMode
for each bigger v in N(u)
    do waitPage(u, v)
PeckOrder(u)
```

Node u goes to page scan mode and waits for a page from all its neighbors with bigger weight ("bigger neighbors"). As soon as node u has received information from the bigger neighbors (i.e., it has been paged by all of them), it becomes the "bottom of the pecking order," i.e., being now the bigger node among those with which it has to exchange the information, it switches to page mode, and starts setting up temporary piconets with all its smaller neighbors (if any). In the following we will refer to the operations described in the peckorder and subnode procedures as the *pecking protocol*.

In particular, a node that decides to be master in BlueStars is either an init node or a node whose bigger neighbors decided all to be slaves. A node that has been told (via paging) by one or more of its bigger neighbors that they are masters, becomes the slave of the first master who paged it. This phase of the protocol leads to the partition of the topology resulting from the discovery phase into piconets with one master and a number of slaves which is not necessarily bounded (if not by the number of neighbors of the master). Notice that no two masters can be neighbors.

After this phase, each master proceeds to the selection of *gateway devices* to connect multiple piconets so that the resulting scatternet is connected. In order to achieve connectivity it is necessary (and sufficient) that each master establishes a path with (i.e., chooses gateways to) all the masters that are two or three hops away [6]. More precisely, masters are two hops away if they can be interconnected by a common slave, called *gateway slave*. This is the preferred way of interconnecting adjacent piconets. Masters are three hops away if they are not two hops away and they can be joined by a pair of *intermediate gateways* (i.e. a pair of neighboring slaves, each belonging to a different piconet among the two that have to be interconnected). In this case the interconnection is achieved by means of an extra piconet, where one of the two slaves is the master and the other a slave. The knowledge about which nodes are the masters two and three hops apart is achieved during the piconet formation phase. Specifically, each node v communicates its role (and possibly the identity and weight of its master) to all its smaller neighbors and to the bigger neighbors that are slaves. If a node is a slave, it waits for the smaller neighbors to communicate the same information. In this way, at the end of the piconet formation phase each node is aware of all its neighbors ID, and of the ID and weight of their masters, which are the information needed in the piconet interconnection phase.

The process of piconets interconnection is based again on a mechanism similar to the pecking protocol, this time executed only among the masters. The details of the rule adopted for consistent gateway selection and for piconets interconnection can be found in [13].

4.2 LSBS protocol

The main aim of the *Li-Stojmenovic/BlueStars protocol* (LSBS), obtained by combining the Yao construction proposed in [10] and BlueStars, is to build up a connected scatternet in which each piconet has no more than seven slaves.

The protocol assumes that each node knows its own identity, a dynamically computed weight that indicates how much that node is suitable for serving as a master (as in BlueStars), and its own location in the plane (usually provided by an on-board GPS device, or by any suitable inertial positioning system device). It is assumed that, as the outcome of the device discovery phase, a node also knows the identity of its neighbors, their weight and their location. By using the pecking protocol, the discovered devices also exchange information about their neighbors (achieving two-hop neighborhood knowledge).

In the description of the algorithm given in [10] it is assumed that nodes are scattered in the plane and that the network graph resulting from the device discovery phase is a connected unit disk graph. Given that the discovery phase does not produce a unit disk graph, for the protocol to correctly operate it is needed to run an extra phase aiming at "bringing back" some links that were not discovered, called in the following the *replenish phase*. This phase can be implemented in the following way.

At the end of a discovery phase all neighboring nodes that have discovered each other exchange the list of the nodes they just discovered, as well as their estimated position. This list exchange can be performed by executing the pecking protocol as described above and leads to the construction at each node v of a set A_v of all nodes discovered by v's neighbors, that appear to lie within v's transmission range and that v did not discover.

Once the list exchange is finished, node v starts contacting the nodes in A_v to verify whether they are nodes within its transmission range (i.e., undiscovered neighbors). This accounts for errors in the position estimates. To this purpose, a node v alternates for a predefined amount of time between page and page scan modes attempting to discover the nodes in its A_v. More specifically, when in page mode v attempts to page one after another (round robin fashion) all the nodes in A_v. Each time two nodes u and v discover each other, they remove each other from their set A_u and A_v and exchange their lists of neighbors. This may lead to new nodes for u and v to be added to their sets A_u and A_v, i.e., to new nodes to page. The length of this phase has to be carefully chosen so to discover all the nodes in A_v that are actually in v's transmission range. Indeed this phase is time consuming and leads to considerable overhead due to the exchange of neighbors lists between neighbors.

The knowledge of the location is then exploited for applying to the "replenished topology" geometric-based techniques to reduce the degree of the network to at most $k \leq 7$. The Yao construction is executed at each node v and proceeds as follows. Node v divides the surrounding plane into k equal angles. In each angle, node v chooses the closest neighbor u, if any. (Ties are broken arbitrarily.)

A link between nodes v and u survives the Yao construction phase if and only if v has chosen u and vice versa. All other links are deleted. To make such decision nodes need to exchange with their neighbors the information on the nodes they selected. This is performed by using again the pecking protocol. Despite some links being deleted, the Yao contruction guarantees the connectivity of the resulting topology [10, 1] (provided that it operates on a connected topology).

Once a connected topology with such a bounded degree has been obtained, the BlueStars algorithm for scatternet formation outlined above uses the nodes' weight for selecting the masters, the slaves and the gateways necessary to form a degree-bounded connected scatternet.

5 The Blue Pleiades scatternet formation protocol

Blue Pleiades was designed to combine the good features of BlueStars and LSBS while eliminating their drawbacks. The idea is quite simple. Based on the theoretical results presented in Section 3 we first modified the device discovery as follows:

> **BluePleaides: device discovery.** *Each device executes the device discovery protocol until it is connected with c neighbors. The constant c can be very small, a number like* 5, 6 *or* 7.

In order to account for nodes in the visibility graph which have less than c neighbors a time limit to the execution of the device discovery protocol is also set. Let G_c denote the subgraph of the visibility graph discovered within such time limit. The new device discovery protocol corresponds to the scheme of Definition 1 where we optimistically set $c = c^*$ and a (small) time-out is used instead of ρ.

In order to validate the theoretical conclusions we performed a first set of experiments based on the simplified model of Section 3.

We now show that this implementation of device discovery works quite fine, generating connected networks very quickly and reliably even for small size networks. In our experiments a number n of points (devices) are distributed uniformly and independently at random in a discretized 1000×1000 point square. The graphs G_r (the visibility graph) and G_c, for different values of c, and r are generated. In our tests, every point in the plots is the average over 1000 different graphs, each with a different initial distribution of points.

Let us now see the outcome of our experiments. Figure 7.1 shows the number of connected components as n increases, for values of the transmission radius r ranging from 40 up to 333 points (pixels). As n increases the graph tends to become globally connected. Note that this is by no means a foregone conclusion. In the k-closest neighbour model for instance, if $k = o(\log n)$, the graph will almost certainly have isolated components [17]. The striking feature of the figure however is that for the visibility graph G_r and G_c, for $c = 4, 5, 6$ or more, the plots are basically completely overlapping. The plot for G_c, $c = 3$ leads to worse performance, requiring a higher number of nodes to result into a connected discovered topology. This outcome is confirmed by all our experiments, for varying values of r and n, ranging from very sparse scenarios to very dense scenarios ($r = 333$, number of vertices in the hundreds).

The conclusion is reinforced by the results shown in Figure 7.2. The figure plots the relative size of the largest connected component of G_c, for $c = 3, 4, 5, 6$, and of G_r for $r = 40, 100, 250, 333$, as the number of vertices grows. The size is relative to the total number of vertices. Once again the striking feature is that the plots for G_c, $c = 5, 6$, and G_r coincide. For $c = 4$ the situation is marginally worse. These empirical facts have important practical implications. In essence they show that, as far as global connectivity is concerned, it does not pay off to set up all possible links. Rather, it suffices to limit the number of links to a very small constant, as the connectivity properties will be maintained. In practice in the simplified model $c = 5, 6, 7$ works quite well.

This is the situation as far as our simplified model is concerned. In the next section we will see that the same situation arises when the graph G_c is generated by simulating the Bluetooth device discovery protocol. Our ns-2 simulations confirm the above observations and show that $c = 6, 7$ might be the best values in practical situations.

It is intuitively obvious that the new scheme for device discovery not only allows to maintain the global connectivity properties of the network, but significantly reduces the number of links in G_c, for small c, over the number of links in G_r. Figure 7.3 confirms this intuition. The figure shows the number of links of the graph when n vertices are distributed uniformly in the square. Each point in the plot is the average taken over 1000 different topologies of G_c, for the same value of c. It is apparent that many links can be saved. The denser the topology (the higher the number of nodes, or the higher the r value), the sparser G_c is with respect to G_r. When $c = 5$ and $r = 250$ ($r = 333$) a 70% (80%) saving in the number of links can be obtained when $n = 100$ nodes are deployed in the area. This results in great savings in terms of time, energy, and traffic congestion.

BluePleaides: piconet formation and interconnection Once a sparse degree-bounded subgraph of the visibility graph is discovered as described above, BluePleiades proceeds by dividing the Bluetooth devices into piconets and interconnecting such piconets into a connected scatternet. This is performed by running the BlueStars algorithm described above over G_c for selecting the masters, the slaves and the gateways necessary to form a degree-bounded connected scatternet.

By adopting the new scheme for device discovery, Blue Pleiades does not require location information, nor does it need to run any extra phase (such as the replenish phase or the Yao construction phase) before the actual scatternet formation can start, resulting into a simple, fast, low overhead solution.

6 Performance Evaluation

This section summarizes the results of ns2 extensive simulations that have been conducted to prove the effectiveness of the Blue Pleiades protocol and to quantify the improvements that can be achieved over previous solutions. Simulations have proceeded in two phases: first we have evaluated the performance of the new scheme for device discovery and compared its behavior with that of the device discovery protocol so far adopted in multi-hop BlueTooth networks [15]. We have then conducted simulations to evaluate the quality of the scatternets generated by the Blue Pleiades protocol, and to compare its performance with that of other solutions proposed in the literature. Based on the comparative performance evaluation of all the major solutions for multi-hop scatternet formation presented in [3], we have selected the LSBS protocol and BlueStars combined with the traditional device discovery as benchmarks to compare our protocol with. Our simulations show that, by exploiting the new scheme for device discovery, Blue Pleiades can achieve performance comparable to what we obtained with BlueStars, while also keeping the number of slaves per piconet to a value below 7, without requiring extra hardware or increasing the complexity of the protocol. The simulation model and the simulation results are detailed in the following two subsections.

6.1 Simulation environment

The performance of the solution proposed for device discovery and scatternet formation has been evaluated by means of a Bluetooth extension to the VINT project network simulator ("ns2") we have developed. We based our extension on BlueHoc, the ns2-based simulator released by IBM. In particular, in order to implement the operations of the solution for device discovery and scatternet formation, we have enriched BlueHoc with mechanisms for *a*) giving to each node the possibility to dynamically assume either the role of master or the role of slave; *b*) handling collisions that might arise during the establishment of a link, and *c*) having a node alternating between inquiry and inquiry scan. (These functions are not available in BlueHoc.) Our simulator strictly follows all the specifications of the BT protocol stack concerning scatternet formation.

In the simulated scenarios, n Power Class 3 BT nodes, with maximum transmission radius of 10 meters, are randomly and uniformly scattered in a geographic area which is a square of side L. We make the assumption that two nodes are in each other transmission range if and only if their Euclidean distance is $\leq$ 10m.

In the simulation results presented here, the number of BT nodes n has been assigned the values 30, 50, 70, 90 and 110, while L has been set to 30m. This allowed us to test our protocol on increasingly dense networks, from (moderately) sparse networks, where only 95% of the visibility graphs are connected ($n = 30$), to highly dense networks. The average degree ranges from 27.9 for $n = 110$ down to 7.4 when $n = 30$ Figure 7.4 (a).

The simulated scenarios refer to the case when all the nodes enter the network within a time T_{acc} (set to $100ms$) and run the device discovery and scatternet formation protocols to self-organize themselves in a Bluetooth scatternet. The value assigned to every node's clock at simulation start is drawn randomly and uniformly in the range 0 and $2^{27} - 1$. All results presented in this section were obtained by running the protocols on 300 connected visibility graphs.

6.2 Simulation results

6.2.1 Device discovery in multi-hop networks

We have run the new scheme for device discovery, with timeout equal to $20s$, as well as the device discovery proposed by Salonidis (referred to as the $c = \text{inf}$ device discovery), the latter for a fixed time equal to $20s$. The effect of the c value on device discovery performance is displayed in Figure 7.4 (a) to Figure 7.7.

The percentage of discovered neighbors over time is reported in Figure 7.5. We notice that setting $c = \text{inf}$ (i.e. all neighbors should be discovered) leads to a significantly higher number of discovered neighbors. However, after $10s$ of device discovery only a fraction of neighbors is discovered (from 96% at $n = 30$ downto 67% at $n = 110$). The higher the density (and the nodes average degree), the longer the time needed to discover all the neighbors. After 20s of device discovery not all the nodes have discovered each other at $n = 110$.

For $c = \text{inf}$, the number of discovered neighbors steadily increases as the device discovery phase proceeds. However, the rate at which new neighbors are discovered decreases with time, as there is a higher chance to handshake again with nodes already discovered. The results obtained after 10s and 20s of device discovery for $c = \text{inf}$ show that it is extremely time-consuming (hardly practical) to require that each node becomes aware of *all* its neighbors.

By adopting the new scheme for device discovery all nodes are able to discover the required number c of neighbors quite fast. This is shown by the curves of the percentage of discovered neighbors, which have a knee and flatten after 4s (8s) at $n = 110$ ($n = 30$). Nodes that have discovered c neighbors leave the device discovery phase, so that interferences get more and more unlikely with time. The fact that BT devices discover the required c nodes in $4-8$s (provided they actually *have* c neighbors) is further confirmed by Figure 7.6. Figure 7.6 shows the distribution of the time spent by nodes in device discovery when varying $c = 5, 7$ and for $n = 30, 70, 110$. After $4 - 8$s the curves get flat, and then increase again to 1 between 18s and 20s. All nodes with more than c neighbors discover c among them within the first $8 - 10$s and exit the discovery phase, while the other nodes exit the device discovery phase only upon timeout (set to $20s$ in the simulations). As expected, the higher the density, the smaller the number of nodes exiting the phase by timeout.

Figure 7.7 displays the percentage of connected topologies after $2, 4, 6, ..., 20$s of device discovery when varying c and for $n = 30, 110$. The curves confirm the effectiveness of the proposed solution: setting $c = 6, 7$ leads to connected BT discovered topologies, and in a relatively short time (within 8s, result that we have used to tune the timeout value in the simulations run to compare the different scatternet formation protocols).

The time needed by the new scheme for device discovery is comparable to what was needed in the $c = \inf$ case, but the new approach has the *extremely important* effect to limit the nodes degree to c. This is shown in Figure 7.4 (a) which plots the average degree of the BT discovered topologies. While no node degree exceeds c, the average degree value lies short below the upper bound. In the next section we will see that this important feature leads both to scatternets with limited number of slaves per piconet and much faster scatternet formation, significantly reducing the overall time needed to build up an ad hoc network out of Bluetooth devices.

6.2.2 Scatternet formation

In the second part of our performance evaluation simulations have been conducted to assess the effectiveness of the Blue Pleiades protocol, to evaluate the "quality" of the scatternets it generates, as well as to compare its performance with that of the BlueStar and of the LSBS protocols. The metrics considered in our performance evaluation include:

1. Measuring the time needed for scatternet formation (once the phase of device discovery has been completed);

2. Counting the average number of piconets;

3. Counting the number of slaves per piconet;

4. Counting the number of roles (either master or slave) assigned to each node,

5. Comparing the average length of the routes between any two BT devices in the scatternet to the average shortest path length between any pairs of nodes in the visibility graph, and

6. Comparing the overhead (number of LMP transmitted packets) associated with running the different scatternet formation protocols.

In all the presented simulations the timeout of the device discovery has been set to 10s.

Protocol duration Figure 7.8 depicts the average time needed by Blue Pleiades (for $c = 6, 7$), BlueStars, and by the LSBS protocol to form a scatternet starting from the BT topologies obtained with device discovery.

Blue Pleiades is by far the fastest protocol. It requires an average of less than 0.5s to build a connected scatternet. BlueStars, that was by far the fastest protocols among those so far proposed, needs up to 2s. The higher the number of nodes (and thus the density), the sparser is the BT topology obtained with the new recipe compared with the BT topology obtained by the device discovery proposed by Salonidis et al. ($c = \text{inf}$). A sparser BT topology means a lower number of nodes that need to be contacted to form the scatternet, and thus improved performance (up to a 75% faster scatternet formation at $n = 110$). LSBS requires an average of 8.65s. The major reason for BlueStars and Blue Pleiades being faster than the LSBS protocol is that the latter needs to run the extra phase with which nodes that discovered each other during the device discovery exchange their neighbors lists (accounting for up to 3.2s at $n = 110$), the replenishing phase, as well as the Yao construction phase (up to $2.3s$). The replenish phase needs to be performed for at least 2s to have the statistical guarantee that all the nodes in the set A_v have been contacted by node v and all needed links are set up.

Piconet number and number of slaves per piconet The number of piconets and their size affect the performance of BT communications. The magic number for piconet size is 7, this being the maximum number of slaves that can actively communicate with the piconet master. Scatternet formation protocols that produce scatternets with more than 7 slaves will incur the overhead of parking and unparking to manage all the slaves (this is the case of BlueStars). Even for protocols that guarantee piconets with less than 8 slaves, there is a tradeoff among small piconets resulting in a higher data rate per slave on one side, incresed complexity (e.g., for scheduling and routing) and interference associated to a higher number of piconets on the other side.

Figure 7.9 shows the average number of piconets in the scatternets formed by Blue Pleiades ($c = 6, 7$), BlueStars and LSBS.

BlueStars generates scatternets with a limited number of piconets, ranging from 22% to 29% of the number of nodes. As the number of nodes increases, they are partitioned into a higher number of piconets. This in turn translates into a higher number of extra piconets needed for sake of piconet interconnection.

This effect is more evident for the LSBS and the Blue Pleiades protocols. Since these two protocols enforce a bound on the number of slaves per piconet (while BlueStars doesn't), they partion the BT devices into a higher number of smaller piconets, thus also requiring more extra piconets for sake of piconet interconnection. Blue Pleiades and LSBS have scatternets with a similar number of piconets (e.g. ranging from 38% to 59% of the network nodes in Blue Pleiades with $c = 6$). Blue Pleiades partitions the BT devices in a smaller number of piconets than LSBS (from 8.1 to 29.7 for $c = 6$ vs. the $11 - 38$ piconets in which the network is partitioned by LSBS) but requires more extra piconets for piconet interconnection. This explains the results shown in the figure in which the curve for the Blue Pleiades protocol crosses that for the LSBS protocol as n (as well as the piconets to be interconnected) increases.

The average number of slaves per piconet is depicted in Figure 7.10. Blue Pleiades and LSBS result in piconets with a bounded number of slaves, and an average number of slaves ranging from 2.3 to 3.5. The size of the "bigger" piconets instead exceeds the threshold of 7 in BlueStars. As mentioned, we use the duration of the device discovery phase as a "tuning knob" for obtaining discovered topologies with a decreased nodal degree. Doing so we could limit the 95th percentile of the number of slaves per piconet to 13. This, however, still requires parking and unparking of slaves, which may lead to time and bandwidth inefficiencies.

Average number of roles per node Critical performance measures for Bluetooth scatternets are the (average) number of roles assigned to each node and the number of nodes with master-slave roles. Nodes with multiple slave roles suffer from the overhead associated with piconet switching (the higher the number of roles, the higher the overhead). Masters-slave roles are critical: all the communications in the piconet of the master are frozen while the master is tuned to a different piconet. For this reason Blue Pleiades, BlueStars and LSBS all try to use slave gateways for joining adjacent piconets. Gateways are the only nodes that can assume master-slave roles, being slaves in a piconet formed for sake of partitioning the BT devices into piconets, and master of an extra piconet built for sake of piconets interconnection. However this can occur only when intermediate gateways were the only solution for interconnecting two piconets.

In Figure 7.11 we show the average number of roles assumed by BT devices in the scatternets produced by Blue Pleiades, BlueStars, and by the LSBS protocol. The average number of roles per node in the scatternets generated by BlueStars, LSBS and Blue Pleiades increases with n (from 1.2 at $n = 30$ to 1.5 at $n = 110$ in BlueStars, from 1.6 at $n = 30$ to 1.9 at $n = 110$ in LSBS, from 1.56 (1.5) at $n = 30$ to 2.48 (2.44) at $n = 110$ when $c = 6$ ($c = 7$) in Blue Pleiades) to take into account the higher number of adjacent piconets that need to be joined (and thus the number of gateways). The increased number of roles per node in LSBS and especially in Blue Pleiades is due to the higher probability for slaves in a LSBS and Blue Pleiades piconet to be selected as gateways to a bigger number of adjacent piconets, and, partially, to the less efficient way of selecting gateways in a sparser topology.

Average shortest path length Independenly of the particular routing protocol to be finally used in the formed scatternet, the lenght of the shortest route is a metric that gives already a measure of how well a routing protocol can perform. Figure 7.4 (b) shows the increase of the average length of shortest paths in the formed scatternets with respect to the same metric in the BT topologies and in the visibility graphs. For $c = \inf$ there is little difference between the shortest path lengths in the visibility graph and the lengths of the same paths in the discovered topologies. For smaller value of c the discovered topology gets sparser and this results in increased average shortest path length. The higher the density, the more the BT topology produced by the new scheme for device discovery is sparse compared to the visibility graph, the higher the increase in the shortest path length. However, the performance degradation introduced by the new recipe for device discovery is much more limited than the one experienced when the visibility graph is sparsified by means of the Yao construction. The increase in the average shortest paths length of the BT topology over the average shortest paths length of the visibility graph ranges from 10% to 44% (16%to 54%) for $c = 7 (c = 6)$ and from 30% to 86.4% when the Yao construction is adopted.

The average increase of route length in the mesh-like scatternets formed by BlueStars with respect to the length of the corresponding routes in the BT topologies is 65%, independently of the increasing number of nodes. This is essentially due to the piconet-based network organization, which may force nodes that are neighbors in the visibility graph to communicate through their common master (if they belong to the same piconet), or through a possibly long inter-piconets route in case they belong to different piconets.

Routes in scatternets formed by the LSBS protocol are from 19% ($n = 30$) to 63% ($n = 110$) longer than routes in BlueStars scatternets, to reflect the higher number of piconets that have to be crossed from source to destination. This increase cannot be accounted to the fact LSBS generates a higher number of smaller piconets (this is also the case of Blue Pleiades, which shows performance close to BlueStars') but appears to be due to the way the Yao construction sparsifies the original topology.

Compared with Blue Pleiades, LSBS partitions the network into a higher number of smaller piconets, but each piconet is interconnected to a lower number of adjacent piconets . This results into a slightly reduced number of roles per node over Blue Pleiades (as less nodes are involved in piconet interconnection) but does not pay off in terms of average shortest path length. This together with the fact the Yao construction set-ups links with the closest neighbors (reducing the distance covered by each link) leads to a significant increase in the number of piconets to be crossed to reach the destination, thus resulting in an average route length much higher than that of all the other protocols.

Packet overhead As a measure of the control message overhead of the scatternet formation protocols, we have counted the number of LMP packets exchanged by the nodes after the device discovery. Figure 7.12 shows the average number of LMP packets per node. As expected, Blue Pleiades shows superior performance due to the sparse topologies over which it operates and the fact nodes need to exchange little information with their one-hop neighbors. BlueStars has good performance due to its low complexity, but the fact it operates on denser topology (thus needing to contact a higher number of nodes) results in a 86.4% increase in the number of LMP packets exchanged over the Blue Pleiades protocol. The remarkable increase in the overhead associated with LSBS over Blue Pleiades (up to 18 times more LMP packets!) is due to the neighbors lists exchanged to achieve two-hop neighborhood knowledge, to the information exchanged during the replenish phase and during the Yao construction. The information communicated is particularly heavy as it includes nodes' ID, weight, location, and estimated clock. All the extra phases needed by the LSBS protocol are ruled out in Blue Pleiades by the adoption of the new scheme for device discovery, resulting in dramatic improvements.

7 Conclusions

In this paper we propose a new scheme for device discovery in multi-hop Bluetooth networks.

Our idea stems from the study of some geometric random graph models generated by means of simple local rules. We formally prove that the networks that they set up are connected with high probability. Based on this theoretical ground, we propose a device discovery protocol in which each node that has discovered c neighbors exits the device discovery phase, $c = 6, 7$. This scheme is robust, maintains the global connectivity of the network while limiting the number of discovered neighbors (thus the BT topology nodal degree) to less than 7.

Based on such results we have designed a new scatternet formation protocol, named Blue Pleiades. Blue Pleiades combines the new device discovery with the BlueStars protocol previously introduced in [13] which was proven performing the best out of the major previous solutions but for the impossibility to provide limits on the piconet size. By exploiting the properties of the device discovery Blue Pleiades generates piconets with bounded size, while maintaining the superior performance of BlueStars as for the other relevant metrics. By means of a thorough ns2 comparative performance evaluation we have quantified the advantage of adopting Blue Pleiades over two protocols selected as benchmarks: BlueStars and Lin/Stojmenovic Bluestars (LSBS). Such protocols were the ones best performing based on the evaluation in [3]. The results of our performance evaluation show that Blue Pleiades is *by far* better performing than LSBS. It requires much less time for piconet formation and interconnection (0.5s vs. 8.65s at $n = 110$), less overhead ($\frac{1}{18}$ of what experienced by LSBS) and has significantly shortest average routes lengths. The only price to pay is a limited increase in the average number of roles per node in high density scenarios (from 1.9 to 2.4 at $n = 110$).

Blue Pleiades also improves over BlueStars, as it not only provides a bound in the piconet size, but also results in faster operations (0.5s vs. 2s at $n = 110$), and reduced overhead, resulting in the best performing solution so far proposed.

References

[1] A.C.C.Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Computing, 11, 1982, 721-736.*

[2] S. Asmussen and H. Hering. *Branching Processes.* Birkhäuser, 1983.

[3] S. Basagni, R. Bruno, and C. Petrioli. A performance comparison of scatternet formation protocols for networks of Bluetooth devices. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications, PerCom 2003*, pages 341–350, Forth Worth, TX, March 23–26 2003.

[4] Elena Baralis Carla-Fabiana Chiasserini, Marco Ajmone Marsan and Paolo Garza. Towards feasible distributed topology formation algorithms for bluetooth-based wpans. In *36th Hawaii International Conference on System Science (HICSS-36), Big Island, Hawaii, January 6, 2003.*

[5] Amar K. Mehta Ching Law and Kai-Yeung Siu. A new bluetooth scatternet formation protocol. In *ACM Mobile Networks and Applications Journal, 8(5), October 2003.*

[6] I. Chlamtac and A. Faragó. A new approach to the design and analysis of peer-to-peer mobile networks. In *Wireless Networks, vol.5, no.3, 149-156, May 199.*

[7] G.Grimmett. *Percolation.* Springer Verlag, 1999.

[8] John Guttag Godfrey Tan, Allen Miu and Hari Balakrishnan. Forming scatternets from bluetooth personal area networks. In *MIT Laboratory for Computer Science, MIT-LCS-TR-826, 2001.*

[9] T.E. Harris. *The Theory of Branching Processes.* Springer Verlag, 1963.

[10] X. Li and I. Stojmenovic. Partial delaunay triangulation and degree limited localized bluetooth scatternet formation. In *Proc. AD-HOC NetwOrks and Wireless (ADHOC-NOW), Fields Institute, Toronto. September 20-21, 2002, 17-32.*

[11] M.Penrose. *Geometric Random Graphs.* Oxford University Press 2003.

[12] C. Petrioli, S. Basagni, and I. Chlamtac. BlueMesh: Degree-constrained multihop scatternet formation for Bluetooth networks. *ACM/Kluwer Journal on Special Topics in Mobile Networking and Applications (MONET), Special Issue on Advances in Research of Wireless Personal Area Networking and Bluetooth Enabled Networks (G. Zaruba and P. Johansson, eds.)*, 2003. In press.

[13] C. Petrioli, S. Basagni, and I. Chlamtac. Configuring BlueStars: Multihop scatternet formation for Bluetooth networks. *IEEE Transactions on Computers, special issue on Wireless Internet (Y.-B. Lin and Y.-C. Tseng, eds.)*, 52(6):779–790, June 2003.

[14] R.Meester and R.Roy. *Continuum Percolation.* Cambridge University Press, 1996.

[15] Leandros Tassiulas Theodoros Salonidis, Pravin Bhagwat and Richard LaMaire. Distributed topology construction of bluetooth personal area networks. In *Proceedings of IEEE INFOCOM'2001.*

[16] T.Lindvall. *Lectures on the Coupling Method.* Wiley, 1992.

[17] Feng Xue and P. R. Kumar. The number of neighbors needed for connectivity of wireless networks. In *To appear in Wireless Networks.*

[18] R.J. Thomas Z. Wang and Z. Haas. Bluenet– a new scatternet formation scheme. In *Proceedings of the 35th Hawaii International Conference on System Science (HICSS 35), Big Island, Hawaii, January 2002.*

[19] G. Záruba, S. Basagni, and I. Chlamtac. BlueTrees—Scatternet formation to enable Bluetooth-based personal area networks. In *Proceedings of the IEEE International Conference on Communications, ICC 2001*, volume 1, pages 273–277, Helsinki, Finland, June 11–14 2001.

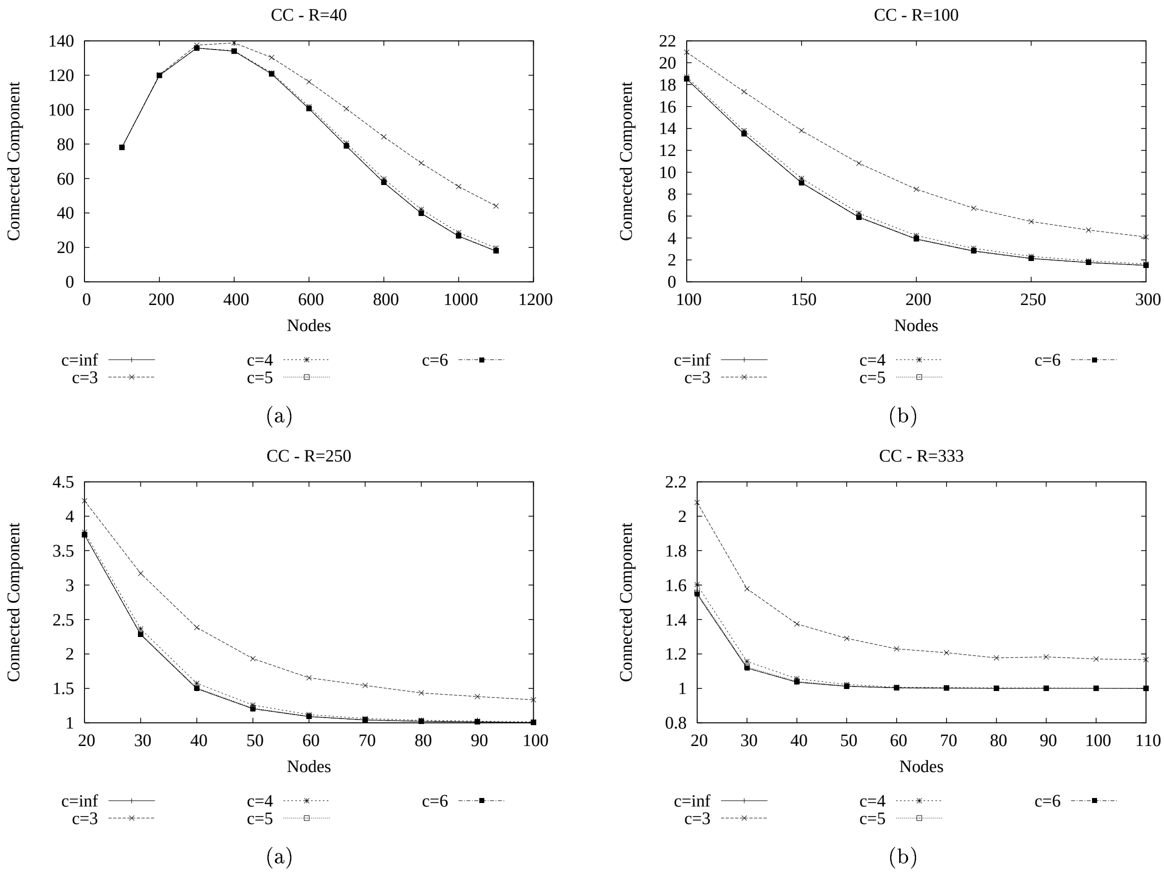
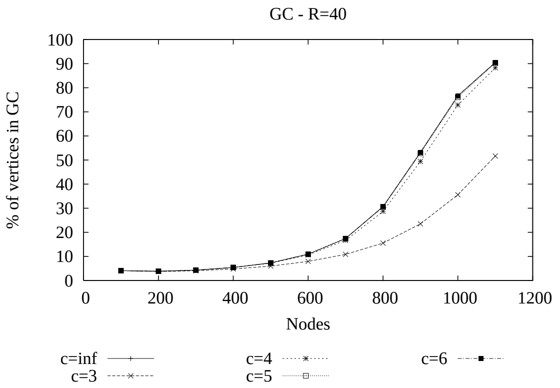
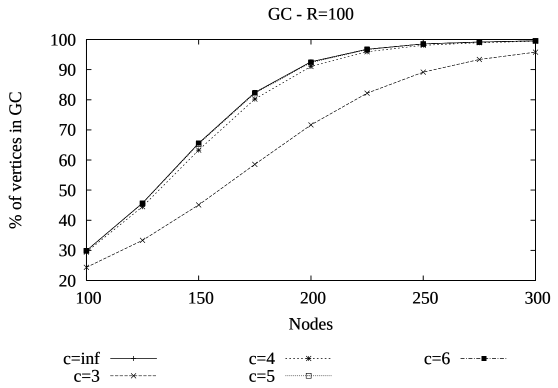


Figure 7.1: Number of connected components of G_c, for $c = 3, 4, 5, 6$ and the visibility graph, denoted as $c = \text{inf}$, for $r = 40, 100, 250, 333$ in a 1000×1000 box.

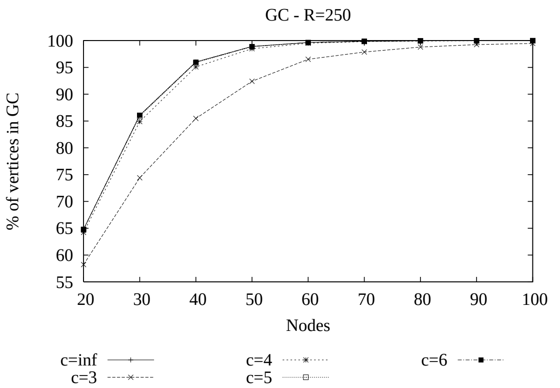
.



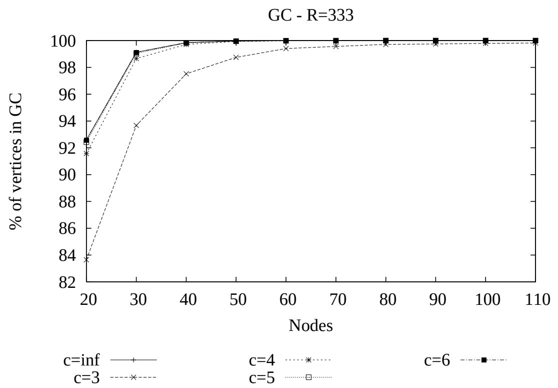
(a)



(b)



(a)



(b)

Figure 7.2: Relative size largest connected component of G_c, for $c = 3, 4, 5$ and the visibility graph G_r (denoted with $c = \infty$) for different values of the transmission radius r

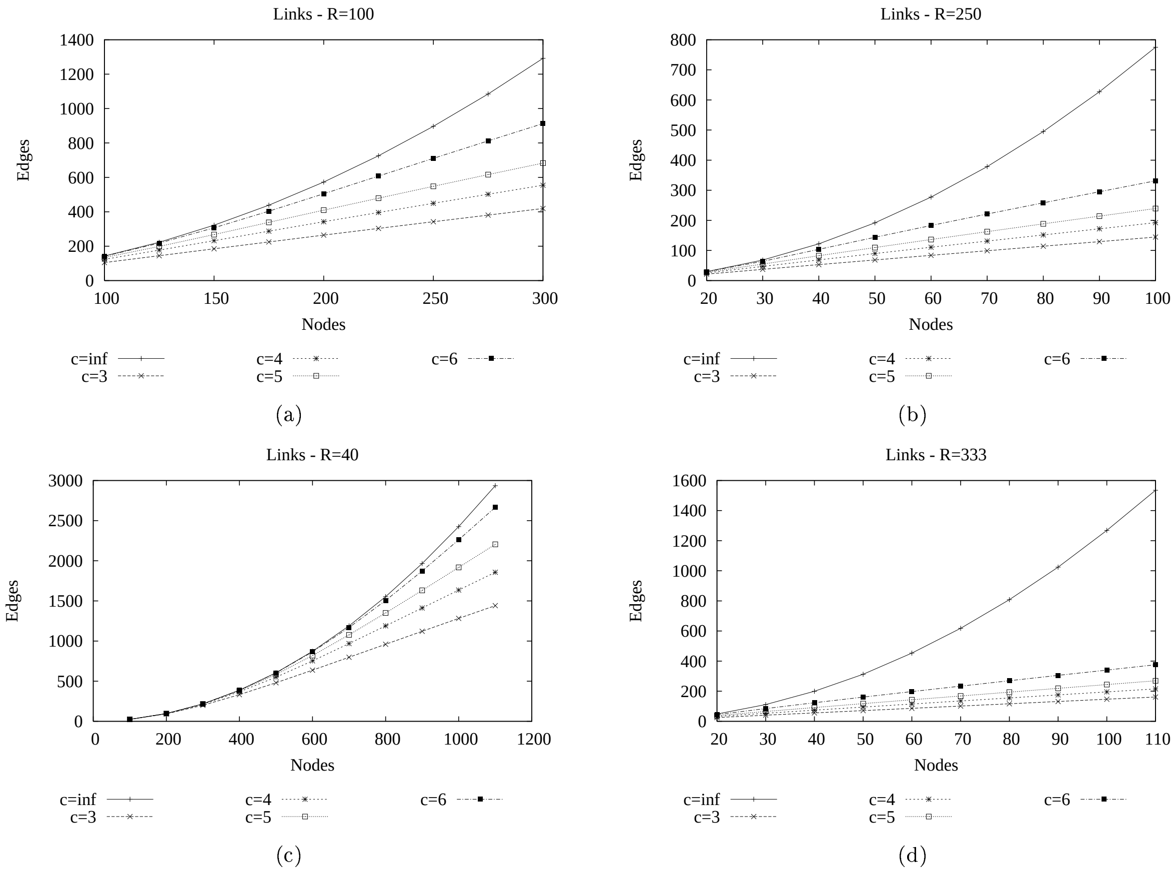
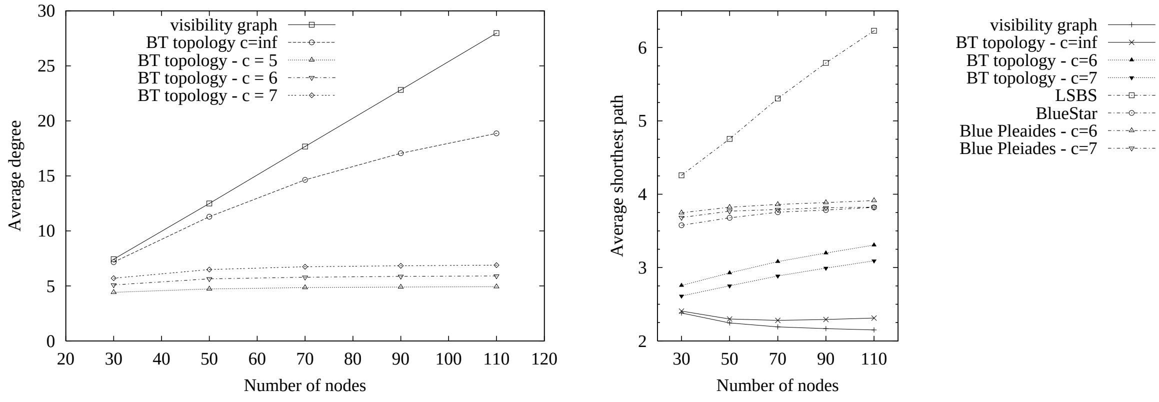


Figure 7.3: Average number of links of G_c and of G_r, for $c = 3, \ldots, 6$ and different values of the transmission radius R in a 1000×1000 box.



(a) (b)

Figure 7.4: (a) Average degree and (b) average shortest path length in the various obtained topologies, device discovery timeout equal to 10s).

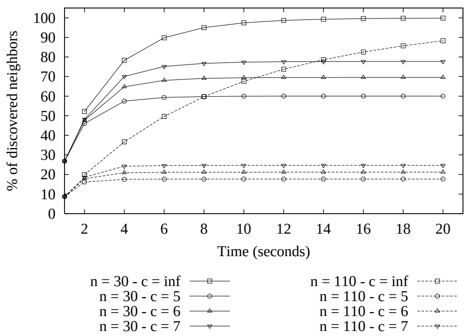


Figure 7.5: Average percentage of discovered neighbors vs. time for $n = 30, 110$ and $c = 5, 6, 7, \mathrm{inf}$.

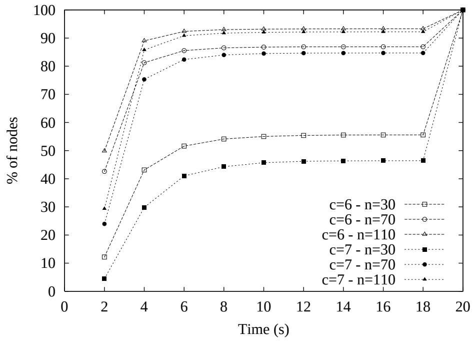


Figure 7.6: Distribution of the time needed for a BT device to complete device discovery: $n = 30, 70, 110$, $c = 6, 7$.

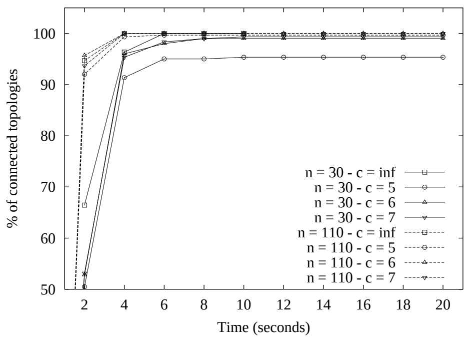


Figure 7.7: Percentage of connected topologies after $2, 4, 6, 8, 10$s of device discovery. Number of BT devices $n = 30, 110$, and $c = 5, 6, 7, \text{inf}$.

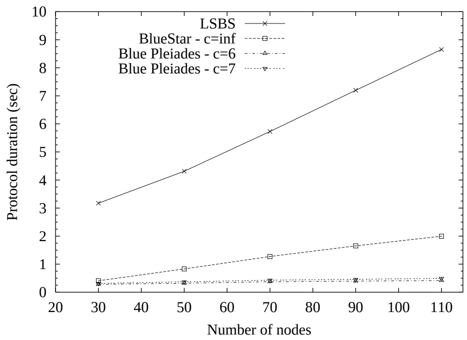


Figure 7.8: Scatternet formation durations: Blue Pleiades protocol ($c = 6, 7$), BlueStars, and LSBS protocol.

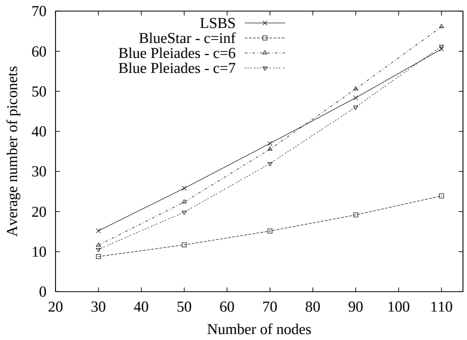


Figure 7.9: Average number of piconets: Blue Pleiades ($c = 6, 7$), BlueStars, LSBS protocols.

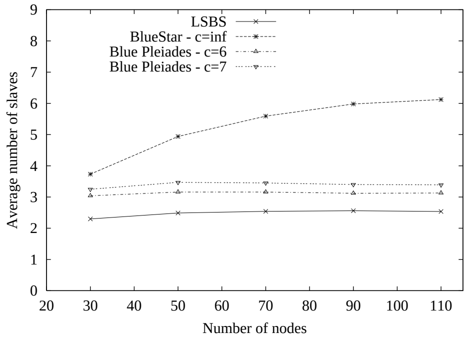


Figure 7.10: Average number of slaves per piconets:. Blue Pleiades ($c = 6, 7$), BlueStars, LSBS.

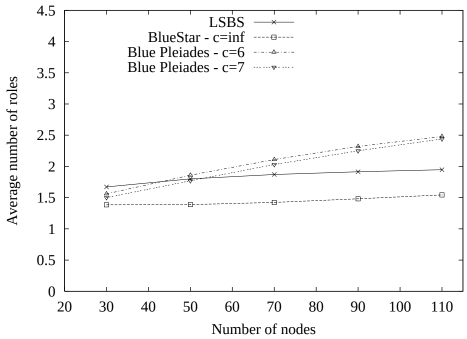


Figure 7.11: Average number of roles per node. Blue Pleiades ($c = 6, 7$), BlueStars, LSBS.

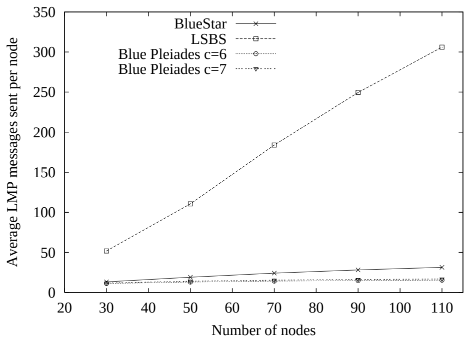


Figure 7.12: Average number of LMP packets per node, 10s long device discovery