BSF-UED: A Novel Time-Efficient Bluetooth Scatternet Formation algorithm based on Unnecessary Edges Deletion

Ahmed Jedda, Arnaud Casteigts, Guy-Vincent Jourdan, Hussein Mouftah

BSF-UED: A New Time-Efficient Bluetooth Scatternet Formation Algorithm Based on Unnecessary-Edges Deletion

Ahmed Jedda*, Arnaud Casteigts†, Guy-Vincent Jourdan*, and Hussein T. Mouftah*

* SEECS, University of Ottawa, Canada
{ajedd077, gvj, mouftah}@eecs.uottawa.ca

† LaBRI, University of Bordeaux, France
acasteig@labri.fr

***Abstract*—We introduce a new time-efficient Bluetooth Scatternet Formation (BSF) algorithm, called BSF-UED (*Unnecessary-Edges* Deletion). BSF-UED forms connected scatternets deterministically. Heuristics are added to make these scatternets outdegree limited (that is, with no more than 7 slaves per piconet). The performance of the algorithm is evaluated through a range of simulation experiments. BSF-UED is compared against some of the most common BSF algorithms which are BlueStars, BlueMIS I, BlueMIS II, and BlueMesh. We show that BSF-UED provides a good balance between the usual scatternets performance metrics, while being time efficient (nearly 1/3 of the execution time of BlueMesh). BlueStars remains a faster algorithm, but with the major flaw of generating scatternets whose piconets have a large number of slaves.**

I. Introduction

We study in this paper the problem of forming efficient multi-hop ad-hoc topologies of Bluetooth devices (that is, the *Bluetooth Scatternet Formation (BSF)* problem). Our interest in the Bluetooth technology as an enabler for ad-hoc networking comes from the many advantages that it offers, such as a relatively low cost of devices, low energy consumption and low interference with neighboring wireless devices. Moreover, the wide availability of Bluetooth devices in mobile phones, smart watches, security sensors, car key fobs and others is leading toward enabling personal area networks using Bluetooth as an underlying communication technology. Moreover, the Bluetooth Special Interest Group (SIG), which is the group responsible for overseeing its specifications, is continuously introducing modifications to the specifications of Bluetooth and promoting the technology as an enabler of ad-hoc networking (see *Bluetooth Smart* and *Bluetooth Smart Ready*[1]) or as a cable replacement technology.

The BSF problem comes from the design of the Bluetooth technology. According to the Bluetooth specifications [1], two nodes can communicate with each other if they either belong to the same *piconet* or to the same *scatternet*. A piconet is a star topology of Bluetooth nodes, where one of the piconet nodes is assigned the role of *master* while the other nodes are *slaves*. A scatternet is an interconnection of piconets. Two piconets are interconnected through *bridges*. A bridge connecting two piconets is a node that belongs to both piconets. The bridge can be a master in one piconet and a slave in the other (called *Master/Slave (M/S) bridge*), or it can be a slave in both piconets (called *Slave/Slave bridge*). A node may serve as a bridge to more than one pair of piconets. Similarly, two piconets may be interconnected through more than one bridge. Note also that it is not possible to have Master/Master bridges, since a node can be master to only one piconet. This means that the number of masters is the same as the number of piconets. Thus, the terms *master* and *piconet* are used interchangeably in the rest of the paper.

The Bluetooth specifications do not specify algorithms for Bluetooth scatternets formation. The solutions of the problem are left open to researchers. The quality of a scatternet can be measured by many performance metrics. For instance, it is preferred to minimize the number of masters in a scatternet. This is because a master controls the flow of packets between the slaves of its piconet and whence consumes higher energy. Similarly, it is preferred that piconets have few slaves on average. Also, the number of bridges, especially M/S bridges, shall be minimized since a bridge node schedules its tasks between the piconets it belongs to, and thus, a bridge may consume more energy. More specifically, an M/S bridge may cause a degradation in the throughput of its piconet, as all activities in its piconets are postponed when that bridge serves as a slave. A related performance metric is the average numbers of roles per node, measured by the number of piconets a node belongs to. There are other performance metrics. Some of these metrics are conflicting. That is improving one performance metric may degrade another metric.

The large number of such metrics led to the appearance of a large number of BSF algorithms, each of which concentrates on improving a small set of metrics. However, many of these proposed algorithms do not consider the BSF algorithm execution time as an important factor. For example, the BSF algorithms that are introduced as time-efficient are often tested under simple Bluetooth networks simulators such simjava, bluehoc or in-house simulators. Nevertheless, the complex specifications of Bluetooth, especially those of the Baseband and Link layers, have major impact on the execution time of Bluetooth networks algorithms, and therefore, these complex specifications must be taken into consideration when designing any time-efficient BSF algorithm (see [2] for more details).

We introduce in this paper algorithm BSF-UED (BSF based on Unnecessary-Edges Deletion). The main objective

[1]Bluetooth Smart Ready: http://www.bluetooth.com/pages/Bluetooth-Smart-Devices.aspx. Last visited on Apr. 19, 2013

of BSF-UED is to form scatternets in a short execution time. Algorithm BSF-UED builds over a well-known time-efficient BSF algorithm, called BlueStars [3]. However, BSF-UED aims at fixing a major issue in BlueStars, which is the formation of piconets with very large size (that is, number of slaves). A master with large number of slaves is said to have a large outdegree, since a master-slave relationship between two nodes u and v is often represented as a directed edge (u, v). Limiting the outdegree of a scatternet is a performance metric of significant importance. This is because, according to the Bluetooth specifications, a piconet can have at most 7 active slaves. Every other slave must be inactive in the piconet (i.e. *parked*). A parked slave of a given master cannot send or receive messages to or from its master. It is possible to have a piconet with more than 7 slaves, but such piconets may have low throughput and may also degrade the throughput of the scatternet. Note that limiting the outdegree of a given scatternet may cause the disconnectivity of the scatternet. BSF-UED guarantees the connectivity of the formed scatternets. In general, BSF-UED focuses on three conflicting performance metrics: 1) the execution time, 2) the scatternet connectivity, and 3) the outdegree limitation. At the same time, BSF-UED forms scatternets that are efficient with respect to many performance metrics.

Algorithm BSF-UED performance is studied using detailed simulation experiments. We use the UCBT (University of Cincinnati BlueTooth) simulator, which is a simulation library built over NS-2, to implement our experiments. BSF-UED is compared against algorithms BlueStars, BlueMesh, BlueMIS I and BlueMIS II. A brief idea about these algorithms is given in the following. A detailed literature survey is omitted due to lack of space. We refer the readers to [4] for a comprehensive literature survey. BlueStars [3] forms, in a time-efficient manner, connected scatternets that have a low number of piconets and M/S bridges, but potentially a large number of slaves per piconet (more than 7). BlueMesh [5] offers similar qualities together with connectivity, but at the cost of a long execution time. BlueMesh is considered as one of the best algorithms in the literature (see [4]). To our knowledge, BlueMIS I is the first local distributed BSF algorithms (that is, it has a time complexity of $O(1)$). BlueMIS I [6] generates in a time-efficient manner scatternets that are connected and outdegree-limited but with very high number of piconets. In order to overcome this issue, a set of simple rules were introduced in [6]. These rules, which were called BlueMIS II, are executed over BlueMIS I. The algorithms we study in this paper, including algorithm BSF-UED, have unique features. For instance, these algorithms do not assume any knowledge of the nodes positions; which is a strong assumption that significantly simplifies the procedure of scatternet formation (see [7] for a BSF algorithm that uses this assumption). These algorithms can work in multi-hop networks and not only in single-hop networks, and these algorithms form mesh-like scatternets which increases the reliability of the scatternet in comparison to tree-like scatternets such as [8].

Simulation results show that the execution time of BSF-UED is approximately 1/3 the execution time of BlueMesh, while BSF-UED is shown to form scatternets with similar properties to those of BlueMesh. Most of the scatternets obtained during the simulation experiments were outdegree limited to 7. Only one scatternet out of 5000 formed scatternet is not outdegree limited and has a maximum outdegree of 8. We show that BSF-UED forms scatternets with a low average number of roles per node. The details of simulation experiments are given in Section IV.

The paper is organized as follows; Section II gives a formal definition of the BSF problem. Sections III describes the algorithm. Section IV presents and discusses simulation results, and section V concludes the paper.

II. Notations and Assumptions

We model the input network as an undirected graph $G = (V, E)$, where V is the set of Bluetooth nodes, and an edge $(u, v) \in E$ if nodes u and v are neighbors. Two nodes are neighbors if they are in the radio range of each other, and both discovered each other. A scatternet is modeled as a directed subgraph $\mathcal{S} = (V, E')$, such that if $(u, v) \in E'$ then $(u, v) \in E$ and $(v, u) \notin E'$. A directed edge $(u, v) \in \mathcal{S}$ indicates that u is a master of v. A scatternet is *connected* if the directed graph $\mathcal{S}$ is weakly connected. This is because the communication between a master and its slave can be in both directions.

Each node is assumed to have a unique identifier. We say u is larger than v, denoted $u \succ v$, if the identifier of node u is larger than v's. All the nodes have the same radio range. It is assumed that each node knows nodes within its radio range. That is, the graph $G = (V, E)$ is a unit disk graph.

III. Algorithm Description

BSF-UED runs in two phases. The first phase forms isolated outdegree limited scatternets, which are then interconnected during the second phase. Using local rules, each pair of nodes sharing an edge e gives a distinct color to e. This coloring is used then to categorize the edges into three categories $C1$, $C2$ and $C3$. The edges of $C1$ may be necessary for connectivity, but they do not cause the outdegree of the scatternet to exceed 7. The edges of $C3$ do not affect the connectivity of the scatternet, but may affect its outdegree limitation. The edges of $C2$ may be necessary for connectivity and may also impact the outdegree limitation of the scatternet. Given this categorization, we give priority to the edges of category $C1$. We avoid the use of edges of category $C2$, and we use the edges of $C3$ only when needed to avoid the disconnectivity of the scatternet. We explain in the following the procedures of phase 1 and phase 2.

A. Phase 1: piconet construction

The first phase generates a forest of disjoint outdegree-limited piconets such that every node is either a master or a slave in exactly one piconet. Since each node has a unique identifier, there must be at least one local maxima (that is, a node that has an identifier larger than all its neighbors'). This property is used to initiate a wave-like communication process where any node u initially waits to be contacted by all its

larger neighbors. Once u is contacted by its larger neighbors, it starts contacting its smaller neighbors. Each node u has a set of neighbors called preys ($preys(u)$) which u attempts to capture as slaves if u is not already slaved by another master. Node u cannot slave a prey v if v is already slave to another master. This forms disjoint piconets.

The set $preys(u)$ contains initially all the smaller neighbors of u. Thus, the size of $preys(u)$ may exceed 7, and the procedure described above may forms piconets with more than 7 slaves. We introduce therefore the concept of *nodes delegation* that restricts u from capturing more than 7 slaves in phase 1. Each node u stores a variable called its piconet capacity, denoted $\varphi(u)$, which is initially set to 7. A node u uses $\varphi(u)$ to limit the number of slaves in its piconet, if it becomes a master. Let's assume that the size of $preys(u)$ is larger than 7. If node u is already contacted by all its larger neighbors, or if it has no larger neighbors then u starts the following procedure. First, u selects the node with maximum identifier in $preys(u)$, called v. Node v is deleted from $preys(u)$. Node u finds a subset of common neighbors, denoted $CN(u, v)$, that are neighbors to both u and v and have identifiers smaller than both u and v. Two cases may occur. If $CN(u, v) = \emptyset$, then u attempts to capture v as a slave. Otherwise, u delegates v to capture the common neighbors in $CN(u, v)$ to v and does not attempt to capture v or any neighbor in $CN(u, v)$. In such case, u deletes the delegated neighbors from its preys set $preys(u)$. The above process repeats until u has enough capacity to capture the remaining neighbors (that is, $|preys(u)| \leq \varphi(u)$). Once this condition is true, node u attempts to slave all its neighbors in $preys(u)$.

For ease of analysis, we give a distinct color to each edge $(u, v) \in E$, such that $u \succ v$. The edge (u, v) is colored $black$ if u is the master of v. The edge (u, v) is colored $silver$ if u attempted to slave v but v is already slave to another node w. Whenever u delegates to v some common neighbors $CN(u, v)$, the edge (u, v) is colored $blue$. Meanwhile, the edges $\{(u, w)|w \in CN(u, v)\}$ are all colored red. A node u, that is slaved by a larger neighbor, colors every edge (u, v) with a $green$ color, where v is a smaller neighbor to u. An example illustrating the procedures of of phase 1 is given in Figure 1.

1) The piconet capacity: Note that whenever a node u attempts to slave a node v, the piconet capacity of u (i.e. $\varphi(u)$) is decreased by one. This happens whether the attempt is successful (i.e. (u, v) is $black$) or if it is not successful (i.e. (u, v) is silver). It should be noted also that whenever an edge (u, v) is colored red, node v decreases its piconet capacity $\varphi(u)$ by one. The intuition behind this is that there is a chance that u will be a master of v in phase 2. Note that if (u, v) is colored red, then there must be a node w that is neighbor to both u and v, such that $u \succ w \succ v$, and (u, w) is colored $blue$. That is, v is delegated to w. Note also that w may slave v in phase 1, and one way to interconnect the piconet w and u would be through v being a master of u.

2) How to find $CN(u, v)$?: We describe in the following the procedure of selecting a set of common smaller neighbors $CN(u, v)$ to be delegated to v. Let's define $T = \{preys(u) \cap N_s(v)\}$. This set contains the nodes that can be potentially part of $CN(u, v)$. We let $CN(u, v)$ contains the largest subset $T' \subseteq T$, where the size of T' is at most 7, and the size of the set $T \backslash T'$ is at least $\varphi(u)$. Therefore, $CN(u, v)$ contains $\min(|preys(u)|-\varphi(u), 7, |T|)$ of the set T. We assume that the nodes of $CN(u, v)$ are those that have the largest identifiers in T.

Correctness of Phase 1

Given the input graph $G = (V, E)$, the procedures of phase 1 always terminate correctly. We present the following results without formal proofs due to lack of space. First, let E_{color} the subset of those edges that are colored *color* (e.g., $E_{black} = \{(u, v) \in E : c(u, v) = black, u \succ v\}$).

1) Let $G_p = (V, E_{black})$ be a spanner subgraph of the input graph $G = (V, E)$, then G_p is a forest of disjoint piconets.
2) The set of black and silver edges, if added to the scatternet, do not affect the outdegree limitation of the scatternet given $\varphi(u)$ is initially set to 7. Formally, let's $G_{bs} = (V, \{E_{black} \cup E_{silver}\})$ be the spanner subgraph of the input graph $G = (V, E)$ with the set of edges $\{E_{black} \cup E_{silver}\}$. Then, after phase 1, G_{bs} is outdegree-limited to 7, given that $\varphi(u)$ is initially set to 7.
3) The set of blue edges are not necessary for the connectivity of the input graph $G = (V, E)$. This applies to every blue edge (u, v) and the corresponding edge (v, u). Formally, let $\overrightarrow{E} = \{(u, v) : (u, v) \in E, u \succ v\}$, let $\overrightarrow{E}_{blue} = \{(u, v) : c(u, v) = blue, (u, v) \in \overrightarrow{E}\}$, then the graph $G_1 = (V, \{\overrightarrow{E} - ED_{blue}\})$ be the spanner subgraph of the input graph $G = (V, \overrightarrow{E})$. Then, after the phase 1, G_1 is connected.

The procedures of phase 1 are implemented in a wave-like communication rounds. That is, a node cannot start executing the algorithm except if it receives a message from all its larger neighbors. This wave-like communication and the simple delegation rules of phase 1 are major reasons, but not the only ones, behind the time efficiency of our algorithm.

B. Phase 2: Piconets Interconnection

Abstractly, phase 2 creates a meta-graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in which every piconet formed in phase 1 is a vertex in $\mathcal{V}$. Two general steps are performed in phase 2. We first define a set of edges $\mathcal{E}$ that make $\mathcal{G}$ connected, such that each of these edges corresponds to a path in the original input graph $G = (V, E)$. We then apply a technique inspired from BlueMIS I [6] in order to eliminate some edges of $\mathcal{E}$ while keeping $\mathcal{G}$ connected. Essentially, each node u in $\mathcal{V}$ constructs a maximal independent set of its *larger* neighbors, denoted $MIS_l(u)$. (A set of nodes is said *independent* if it does not contain any pair of neighbor nodes; it is *maximal* if the addition of any node makes it no more independent.) Then, u interconnects only to the nodes in $MIS_l(u)$. This technique helps in heuristically minimizing the average and maximum outdegree of the scatternet and the number of bridges. We show in the following how the set $\mathcal{E}$ is constructed.

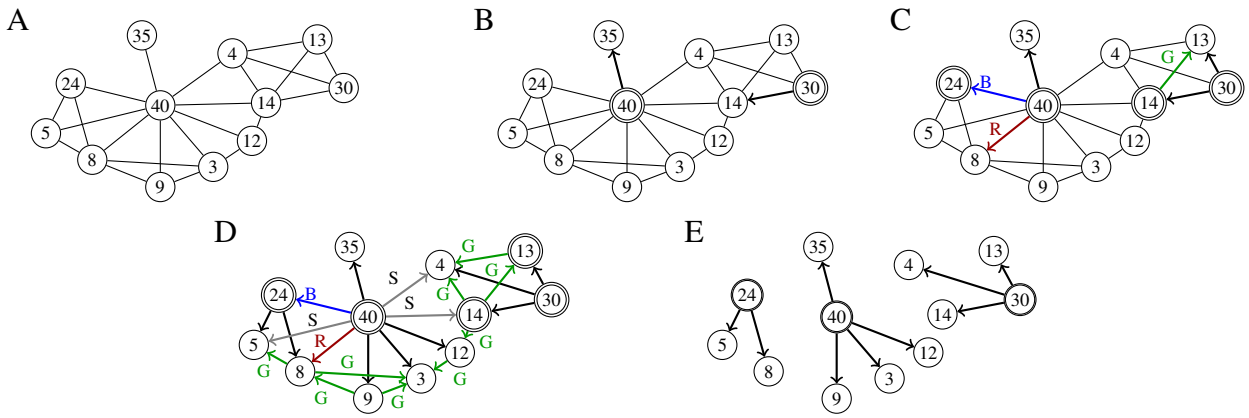


Fig. 1. Example illustrating the procedures of phase 1 executed over the graph shown in (A). Initially, nodes 40 and 30 starts the execution of the algorithm since they are the largest in their neighborhoods. Node 40 and node 30 slave node 35 and 14 respectively, since they are the largest preys (B). Both 40 and 30 decrease the piconets capacities (i.e. $\varphi(40)$ and $\varphi(30)$). Node 40 captures node 30 because $preys(40)$ is greater than $\varphi(40)$ and the set of common neighbors $CN(u, v)$ is an empty set. The next target prey of node 40 is 24 (C). The size of the set $preys(40)$ is greater than $\varphi(40)$ which is equal to 6. Thus, the edge (40,24) is colored red, and the common neighbors between 40 and 24 are 5 and 8. The $CN(40, 24)$ however contains 8 only (see Section III-A2). Node 8 therefore is delegated to 24. Note that a slaved node inform all its smaller neighbors of its status once it is contacted by all its larger neighbors. This is done by coloring the shared edges with green. Node 40 attempts to capture 14 and 4 which are already captured by node 30. It also attempts to capture 5 which is already captured by 24 (D), and hence the edges (40, 14), (40, 4) and (40, 5) are colored silver.

Two *neighbor* piconets u and v share an edge $(u, v) \in \mathcal{E}$ if their masters u and v can be interconnected through one of the path types shown in Table I. Two piconets can be interconnected via 3-hop, 2-hop or 1-hop paths. The 3-hop paths contains two black edges and one green edge. The 2-hop paths must include at least one black edge, whereas 1-hop paths include only one edge colored red connecting two masters. Each path type has a priority such that if more than one path exists between two neighbor piconets, then we consider only the path with the highest priority. The 1-hop paths are given the lowest priority. We avoid such paths since using them for the interconnection process may increase the piconet size of one of the two masters to more than 7 slaves. We give 3-hop paths the highest priority.

The interconnection rules between neighbor piconets is shown in Table I. The rules are shown in priority. That is, interconnection rule I-rule 1 has a higher priority than I-rule 2a, etc .... We show also in Table I the *operation* of the interconnection rule (that is, what master/slave relationship should be added in order to interconnect the two neighbor piconets). Note that in each of the interconnection rules, it is sufficient to add only one master/slave relationship to interconnect the two neighbor piconets.

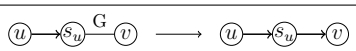
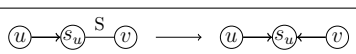
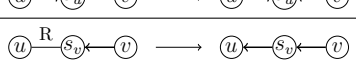
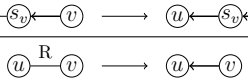
To implement phase 2, each master u collects from each slave s_u. The slave s_u sends information about each neighbor w_i to u. This information is 1) the master of w_i, 2) the piconet capacity value of w_i and 3) the color of the edge (s_u, w_i). Upon the termination of this process, a master u discovers all its neighbor piconets. A master u knows also all the possible interconnection paths connecting it to its neighbor piconets. This procedure, therefore, constructs the meta-graph $\mathcal{G}$.

The next step is to let each node $u \in \mathcal{G}$ (that is, each piconet's master) construct a maximal independent set of its larger neighbor piconets (denoted $MIS_l(u)$). The construction is done in a wave-like communication process, as it is the case of phase 1. Each master u waits to be contacted by all its smaller neighbor piconets' masters. A master u is contacted by a master of a neighbor piconet via one of the paths described in Table I. Once master u is contacted by all its smaller neighbor piconets, it selects the neighbor piconet v with minimum larger identifier - that is not contacted yet - (where the identifier of a piconet is the identifier of its master). Master u then selects the highest priority interconnection rule to interconnect piconet u and v. The interconnection is done via a *gateway* node (as shown in Table I). Let denote such a gateway as s_u. A gateway s_u of piconet u may be a slave to of u or it could be u itself. Master u sends to s_u a message containing the identifier of the node s_v that must be interconnected with, where s_v and s_u are neighbors and s_v is the gateway of the neighbor piconet v. Master u also informs s_u about the interconnection operation (that is, s_u is master of s_v or the opposite).

Let assume that s_u is the gateway of u that is interconnected to piconet v. Upon the completion of the interconnection procedure, s_u requests from v all its neighbor piconets $N_p(v)$. The gateway s_u informs u about any piconet $w \in N_p(u)$ that is also a larger neighbor piconet of v. For each piconet w with such property, piconet u assumes that w is interconnected. Formally, piconet v would be included in $MIS_l(u)$, but every piconet w is not in $MIS_l(u)$. For every neighbor piconet w with such property, master u sends a message to w to inform it that there is no necessary interconnection between both piconets. This message is sent via one of the gateways of u and w.

We give in the following a remark about selecting the best interconnection rule. If a master u finds that there are multiple paths to interconnect with a neighbor piconet v, such that all these paths have the same priority, then u selects the path that include the gateway s_u that has the smallest piconet capacity $\varphi(s_u)$. Moreover, whenever a gateway s_u becomes a master to another gateway, the piconet capacity of s_u ($\varphi(s_u)$) is

TABLE I
THE SET OF INTERCONNECTION RULES IN PHASE 2 (ASSUME THAT $u \succ v$)

Interconnection rule	Illustration	Operation
I-rule 1		s_u captures s_v.
I-rule 2a		s_u captures v
I-rule 2b		v captures s_x or u captures s_x
I-rule 2c		s_v captures u
I-rule 3		v captures u

decreased by one.

C. Correctness of Phase II

We present in the following some results that show the correctness of phase 2 and whence algorithm BSF-UED. We omit the proofs for lack of space.

1) Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be such that $(u, v) \in \mathcal{E}$ if and only if u and v are two piconets that can be interconnected with any of the *interconnection rules* shown in Table I. Then, $\mathcal{G}$ is connected.
2) If a graph $G = (V, E)$ is connected, then the graph $G' = (V, \{(u, v) \in E | v \in MIS_l(u)\})$, where $MIS_l(u)$ is the maximal independent set of larger neighbors of u, is also connected.
3) The graph $\mathcal{G}$ remains connected as long as each vertex $u \in \mathcal{V}$ keeps an edge (u, v) for each $v \in MIS_l(u)$, where $MIS_l(u)$ is the maximal independent set of larger neighbors of u. Note that this is a consequence of the previous result.
4) The graph $\mathcal{S}$ that includes all the master/slave relationships built in phase 1 and phase 2 is a connected scatternet that include all the nodes in V.

One property of the previous interconnection procedure is that if no red edges used for interconnection, then the algorithm forms outdegree limited scatternets. Note that black and silver edges do not cause excess in the outdegree of the scatternet, while blue edges are not used in the scatternet construction and do not cause disconnectivity of the scatternet (see Section III-A). Therefore, it remains to show that green edges do not cause excess in the outdegree limitation of the scatternet. Consider a scenario in which a node u is master to node v. In the interconnection piconet, node v shall use more than 7 green edges to connect with neighboring piconets. Assume that node v must be the master of all the nodes on the other end of its green edges (call them green nodes of v). Let's assume that interconnection rules not involving red edges are not used. Then, there is a maximal independent set of the green nodes of v that is of size at most five. This is because edges between these green nodes are either silver or green. This result, however, assume that red edges are not used in the interconnection process.

We adapt to these results by introducing the following rule that improves the properties of the scatternet formed. This rule eliminates virtually all outdegree unlimited piconets, while not increasing the execution time of the algorithm. The rule is as follows. Assume that piconet u assigned gateway s_u to become master of gateway s_v. Assume s_u is already a master to 7 slaves. Node s_u checks if s_v is a master to less than 7 slaves. If this is the case, then s_v becomes the master of s_u instead. If s_u had less than 7 slaves, this rule is not applied.

IV. SIMULATION EXPERIMENTS

We compare BSF-UED against BlueStars, BlueMesh, BlueMIS I and BlueMIS II. We used an improved version of BlueMIS I introduced in [9]. This improvement gives the same scatternets of BlueMIS I but in a shorter execution time. BlueMIS II execution time is also improved using the same methods of [9]. The performance metrics we consider are: 1) execution time, 2) maximum piconet size (or maximum outdegree), and 3) average number of roles per node. Other performance metrics are included in a longer version of this paper. Our simulation experiments are conducted using the UCBT simulator [10]. The networks are modeled as unit disk graphs. Each graph is constructed by placing points uniformly at random in a $30 \times 30m^2$ plane. An edge connects two points if the euclidean distance between them is less than a threshold t set to 10 m, which is generally taken as the radio range of Bluetooth. A graph is considered for experiment only if it was connected. We form five sets of graphs, each with a different size (30, 50, 70, 90 and 110 nodes). Each set consists of 1000 graphs.

Figure 2 shows a comparison of the execution time of the studied algorithms. The reason behind the outperformance of BlueStars, BlueMIS I and BSF-UED is the use of simple rules to form the scatternet and the fewer number of *communication rounds* in these algorithms, where a communication round is defined as a round in which each node sends and receives a message to and from all its neighbor nodes. We studied different implementations of such rounds in [2]. Algorithms BlueStars, BlueMesh and BSF-UED use a wave-like communication round implementation, whereas BlueMIS I and BlueMIS II were modified to use such a wave-like communication. This wave-like communication round implementation, called OrderedExchange, was shown to be a time-efficient implementation of communication rounds for Bluetooth networks. Based on this analysis of execution time, BlueStars outperforms all other algorithms with respect to

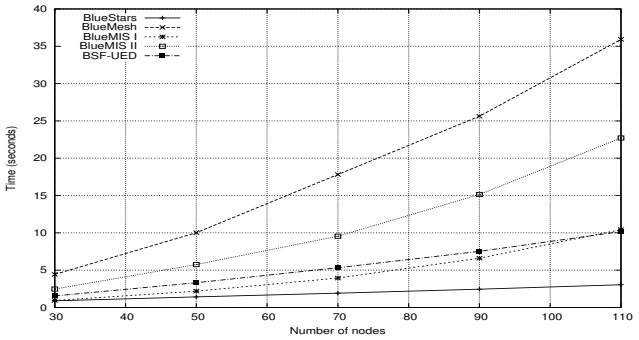


Fig. 2. Comparison of the execution time

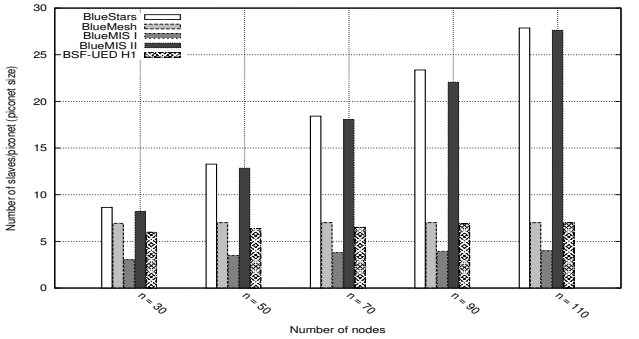


Fig. 3. Comparison of the maximum piconet size

this performance metric, since it simple and it uses very few communication rounds. BlueStars however forms scatternets with very large sized piconets (see Figure 3). BlueMesh fixes the issue of large sized piconets of BlueStars with the cost of a higher execution time, which is about 3 times the execution time of BSF-UED. BlueMIS I has a similar execution time of BSF-UED. Note that this is execution time is obtained from the improvements we introduced in [9]. Note that the original execution time of BlueMIS I is significantly larger than that shown in Figure 2 (see [9] for more details).

Figure 3 shows the average maximum outdegree of the scatternets formed. BlueMIS II and BlueStars are the worst algorithms with respect to this metric. In networks of 110 nodes, these algorithms may form scatternets with maximum outdegree equals about 30 on average. BlueMIS I forms scatternets with the smallest average maximum piconet size. This is because the slaves of a master in BlueMIS I is a maximal independent set of its neighbors, which is of size at most 5 in unit disk graph and it is less on average. BlueMesh on average forms scatternets of at most 7. For BSF-UED, we find that most scatternets formed are outdegree limited. In fact, in our experiments, we find that only one scatternet (out of 5000 experiments) is outdegree unlimited (with maximal degree 8!). BSF-UED is very close to the optimum, which makes us believe that additional heuristics to the algorithms could be further improved to achieve this deterministically.

The number of roles of a node is the number of piconets it belongs to. The average number of roles per node is the sum of number of roles among all nodes divided by the number of nodes. The results are shown in Figure II. BlueStars outperforms all other algorithms in this metric. This is mainly due to the large sized piconets of BlueStars scatternets. BSF-UED, on the other hand, achieves an average of 2.4 roles per node outperforming whence the other algorithms that do not suffer from non-limited outdegree scatternets.

TABLE II
COMPARISON OF THE AVERAGE ROLE PER NODE

	BlueStars	BlueMesh	BlueMIS I	BlueMIS II	BSF-UED
30	1.37	1.61	2.37	1.94	1.46
50	1.30	2.06	2.73	2.24	1.67
70	1.25	2.56	2.98	2.45	1.94
90	1.21	2.96	3.11	2.57	2.21
110	1.18	3.27	3.21	2.67	2.48

V. CONCLUSION

Algorithm BSF-UED is shown to be time-efficient in comparison with major similar BSF algorithms, while at the same time achieve a good balance between the quality metrics of scatternets. Despite not achieving outdegree limitation deterministically, BSF-UED rarely generates outdegree unlimited scatternets. A future work is to solve this issue of BSF-UED. We see that the well studied BSF problem may still be open for new improvements. An important problem in BSF is the generation of outdegree limited scatternets. This problem, mostly studied for complete and unit disk networks, is not thoroughly studied for the case of arbitrary networks, which are obviously more practical.

REFERENCES

[1] B. SIG, *Bluetooth Specifications ver4.0*, 2010.
[2] A. Jedda, G.-V. Jourdan, and N. Zaguia, "Towards better understanding of the behaviour of Bluetooth networks distributed algorithms," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 27, no. 6, pp. 563–586, 2012.
[3] C. Petrioli, S. Basagni, and M. Chlamtac, "Configuring BlueStars: Multihop scatternet formation for Bluetooth networks," *IEEE Transactions on Computers*, vol. 52, no. 6, pp. 779–790, 2003.
[4] I. Stojmenovic and N. Zaguia, *Bluetooth scatternet formation in ad hoc wireless networks*. Auerbach Publications, 2006, ch. 9, pp. 147–171.
[5] C. Petrioli, S. Basagni, and I. Chlamtac, "BlueMesh: degree-constrained multi-hop scatternet formation for Bluetooth networks," *Mobile Networks and Applications*, vol. 9, no. 1, pp. 33–47, 2004.
[6] N. Zaguia, Y. Daadaa, and I. Stojmenovic, "Simplified bluetooth scatternet formation using maximal independent sets," *Integrated Computer-Aided Engineering*, vol. 15, no. 3, pp. 229–239, 2008.
[7] X. Y. Li, I. Stojmenovic, and Y. Wang, "Partial Delaunay triangulation and degree limited localized Bluetooth scatternet formation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 4, pp. 350–361, 2004.
[8] F. Cuomo, T. Melodia, and I. F. Akyildiz, "Distributed self-healing and variable topology optimization algorithms for QoS provisioning in scatternets," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 7, pp. 1220–1236, 2004.
[9] A. Jedda, G.-V. Jourdan, and H. T. Mouftah, "Time-efficient algorithms for the outdegree limited bluetooth scatternet formation problem," in *IEEE Symposium on Computers and Communications (ISCC), 2012*. IEEE, 2012, pp. 132–138.
[10] Q. Wang, "UCBT - Bluetooth extension for NS2 at the University of Cincinnati," Tech. Rep. [Online]. Available: http://www.cs.uc.edu/ cdmc/ucbt