

# Simplified bluetooth scatternet formation using maximal independent sets

Nejib Zaguia<sup>a</sup>, Yassine Daadaa<sup>a</sup> and Ivan Stojmenovic<sup>b,a</sup>

<sup>a</sup>*SITE, University of Ottawa, Ottawa, Ontario K1N 6N5, Canada*

*E-mail: {zaguia, ydaadaa}@site.uottawa.ca*

<sup>b</sup>*Electronic, Electrical & Computer Engineering, The University of Birmingham, Birmingham, UK*

*E-mail: stojmenovic@storm.ca*

**Abstract.** Bluetooth standard allows the creation of piconets, with one node serving as its master and up to seven nodes serving as slaves. A Bluetooth ad hoc network can also be formed by interconnecting several piconets into a scatternet. Given a set of Bluetooth nodes which are positioned so that their unit disk graph is connected, the Bluetooth scatternet formation (BSF) problem is to select piconets, and assign master and slave roles in each piconet, so that the obtained scatternet is connected, has some desirable properties and good performance with respect to some metrics. In this article we propose BlueMis, a new BSF protocol based on maximal independent sets. BlueMis is a two phase protocol, in which the first phase is discovery and the second phase, is scatternet formation. In the discovery process, two hop neighbors are discovered. The second phase of BlueMis uses two iterations. In the first iteration a piconet containing a maximal independent set is constructed for every device, while the second iteration attempts to simplify the scatternet structure and to delete piconets not essential for the connectivity. This novel protocol is an attempt to simplify the BlueMesh procedure. Simulation results show that the average number of iterations in BlueMesh ranges from 2.7 to 4.5, compared to only 2 for BlueMis. The reduced overhead is a major gain and it represents our primary contribution in this paper. BlueMis is implemented and compared with BlueMesh, in terms of various characteristics. Experiments show that BlueMesh performs slightly better, in terms of the number of piconets and the number of slaves per piconet.

## 1. Introduction

Mobile ad hoc networks (MANETs), or simply ad hoc networks, received significant attention in recent years due to their potential applications in battlefield, emergency disaster relief, and other application scenarios. Ad hoc networks consist of a collection of geographically distributed nodes that communicate with each other over a wireless medium. Mobile ad hoc networks are self organizing, rapidly deployable, and require no fixed infrastructure. They are comprised of wireless nodes, which can be deployed anywhere, and which must cooperate in order to dynamically establish communications using limited network management and administration. Nodes in an ad hoc network may be highly mobile, or stationary, and may vary widely in terms of their capabilities and uses. The objective of this type of network is to accomplish increased flexibil-

ity, mobility and ease of management relative to infrastructure wireless networks. Ad hoc networks usually have no fixed network infrastructure for routing traffic, and transmission range is limited by power constraints, frequency reuse and channel effects. Bluetooth and Wi-Fi are two widely adopted technologies for wireless communication between users/devices. They are competing and complementary at the same time, since Bluetooth is more suitable for short communications and provides direct communication between any two neighboring nodes, while Wi-Fi generally requires an access points and is applied over somewhat longer distances.

Bluetooth technology [2] is emerging as one of the most promising enabling technologies for ad hoc networks. Bluetooth is a short-range radio Frequency (RF) specification for short-range, point-to-point and Point-to-multi-point voice and data transfer intended to re-

place the cable(s) connecting portable and/or fixed electronic device. Key features are ad hoc connectivity, robustness, low complexity, low power, and low cost [2]. It operates in the 2.4 GHz unlicensed Industrial, Scientific and Medical (ISM) band, the regulatory range of this frequency band is 2.400–2.4835 GHz. With its low cost and its wide deployment in a high percentage of today's PDA's, laptops, mobile phones and other devices, Bluetooth seems to be one of the strongest candidates to make a real Personal Area Network (PAN). Since its first days of release, the Bluetooth specification is having a great success and is continuously updated to match the requirements of PAN's. Bluetooth devices collect themselves into a piconet which is the basic Bluetooth networking unit. A piconet is defined as a Bluetooth network with no more than 8 active devices, arranged in a star shape consisting of 1 master device and up to 7 slave devices. The piconet uses a specific frequency hopping pattern that can be determined by observing certain fields in the Bluetooth address and clock of the master. The hopping pattern is of pseudo-random ordering that goes through the 79 frequencies in the ISM band. Bluetooth offers full-duplex communication by employing a Time-Division Duplex (TDD) scheme. Before any two Bluetooth devices can communicate with each other, they must go through the device discovery procedure which consists of two phases, inquiry and paging. During the Inquiry phase defined by Bluetooth standard [2] as an asymmetric process, in order for a device to discover other devices, it uses the inquiry channel and acts as a master, while the device to be discovered will have to use the inquiry scan channel and act as a slave. The limit of only 8 active devices in a Bluetooth piconet can be overcome by the formation of scatternets. The network topology resulting from the connection of two or more piconets is called scatternet. However, the Bluetooth specification does not define the structure of the scatternet and no solutions for scatternet formation are provided. In the past few years, scatternet formation has become an area of intense research activity. Scatternet formation protocols should deal with major issues such as the mobility of the devices and their power efficiency.

When two Bluetooth devices establish communication, one of them assumes the role of a master node while the other is a slave node. Two nodes in a scatternet can communicate by finding a route between them, where each hop is a master-slave pair of nodes from the same piconet. Slaves in one piconet can participate in another piconet as either a master or slave and are named bridges. A node may serve as the master in at

most one piconet, and as a slave in unlimited number of other piconets. However, while it serves as slave in other piconet, its own piconet (if it has master role in any of them) will be idle.

A closely related problem is neighbor/device discovery, how two nodes find each other and establish communication. Almost all proposed solutions assume that Bluetooth technology is used for both neighbor discovery and data communication in the created scatternets. Most BSF protocols use the device discovery scheme described in [9].

Several criteria are used to evaluate BSF protocols (see detailed description in [5,6]) although connectivity and degree limitation of the constructed topologies are main goals of any protocol. Obviously, there is tiny probability that two nodes will never find each other, therefore connectivity cannot be guaranteed even for a network consisting of two nearby nodes. Thus the assumption is natural, and connectivity is judged subject to established neighbors knowledge.

The scatternet formation protocols should have (at least) the following goals in mind: minimizing the number of piconets, and therefore the number of master nodes; minimizing the number of slave roles for each master node; minimizing the number of master-slave bridges. Protocols can be further evaluated according to their message complexity, time complexity and memory requirements.

We assume that all nodes discovered all their neighbors in the initialization phase. We assume the unit disk graph model of wireless communication, where two nodes can communicate with each other if and only if the distance between them is  $\leq R$ , where  $R$  is the transmission radius. We consider multi-hop scenarios, where some nodes are not within transmission range of each other, but are connected via other nodes. Our proposed protocol does not depend on a central entity and each node makes formation decisions solely based on the information from its neighbors (possibly also 2-hop neighbors).

This article is extended and improved version of our conference paper [4]. Our new protocol is a major improvement to the protocol we described in [4]. It adds a second iteration that attempts to further simplify the scatternet structure and to delete piconets which are not essential for the connectivity. We noticed an improvement of about 20% on the number of piconets compared to the protocol in [4].

Section 2 presents a literature review. Section 3 describes our new MIS (maximal independent set) based BSF protocol. Section 4 presents experimental data on some evaluation criteria. Conclusions and references complete this article.

## 2. Related work

A comprehensive survey of BSF protocols is given in [12]. Numerous approaches for scatternet formation have been proposed in the literature since the Bluetooth specification was published. The Bluetooth scatternet formation algorithms proposed in the literature may be classified into two categories: The first category work properly for single hop network, this category assumes that all Bluetooth devices are within radio transmission range of each other (e.g. [1,3,9]). The second category work properly for multi hop network (work for single hop network as well), this category assumes that all devices are not necessarily in a transmission range of each other. The first known solution is proposed in [9], it is based on a leader election process to find a coordinator that collects topology information and then assigns roles to devices and form the topology. Then a centralized algorithm is applied at the leader to assign roles to the devices on the network. Several papers deal with the more general case of multi hop topology (e.g. [8,10,11]). In [10] the first two multi hop scatternet formation protocols are proposed. In both cases the resulting topology is termed a Bluetree. The number of roles taken by each node is limited to 2 or 3, the proposed solutions are not time efficient and the protocols assumes that all nodes start the formation process at the same time. In [11] a three phase dominating set based scatternet formation protocol for Bluetooth network is proposed. In the first phase the unit graph is constructed and a localized sparse graph is extracted. In the second phase, a degree reduction technique Yao sub-graph is applied simultaneously on all nodes with excessive degree (more than 7), to limit the degree of each node to 7, in the third and last phase master-slave roles are assigned based on dominating set membership. However no thorough evaluation of this protocol is provided and this technique requires each node to be equipped with additional hardware to provide to the node its current location which may be expensive and the proposed solution is not feasible in the absence of such hardware.

Petrioli and Basagni [7,8] described a scheme (called *BlueMesh*) that guarantees connectivity and limits the number of slaves in each piconet. Their scheme does not require position information, but instead the local information is extended to two-hops, with a two round device discovery phase for obtaining necessary information. It is a modified clustering process, where selection of slaves is performed in such a way that, if a master has more than seven neighbors, it chooses up

to seven slaves among them so that it can reach all the others via the chosen ones. Such coverage is always possible with up to five slaves [10]. The scatternet formation proceeds in iterations. Nodes that participate in a given iteration perform the modified clustering process. Initially all nodes are undecided. In every iteration, init-nodes (nodes having the largest weight among their immediate undecided neighbors) create piconets, by choosing at most seven neighbors as slaves, and deleting the remaining edges. The iteration stops when all nodes are decided. All created masters, together with the slaves that are not selected for links with slaves from other piconets, withdraw from the next iteration. Simulations show that the created scatternets have a low average number of roles per node, with an average path length increase between 20% and 80%. The number of iterations grows slowly with the number of nodes (about 4.5 for 200 nodes). The method may show weaknesses on some other metrics, especially about the worst-case number of slave roles a node can assume. For instance, in case of dense networks (e.g., complete graph), the second largest node in a neighborhood may end up serving as slave to all the masters in the same neighborhood. Nevertheless, among all methods that do not use position information, the method [7,8] appears to be currently the best available method for multi-hop networks.

## 3. BlueMIS

Our new BSF protocol attempts to simplify the *BlueMesh* procedure. The protocol essentially interprets the slave selection as the *maximal independent set* problem, and in contrast with *BlueMesh* where the number of iterations could be large, our new protocol reduces the process to two iterations, after the initial discovery process. At the end of the discovery phase, performed at each node, all devices discover their one-hop neighbors. During the construction of the *maximal independent set* for each node, there will be exchanges of neighbors' information as needed. Pairs of neighboring devices may exchange information on their one-hop neighbors, thus getting to know all the nodes which are at most two hops away.

The maximal independent set  $MIS(X)$  of a set of nodes  $X$  is a set of nodes  $Y$  from  $X$  such that no two nodes from  $Y$  are connected ('independent set'), and  $Y$  is not a proper subset of another set with the same property ('maximal'). After learning one-hop neighbors, the first iteration of our protocol will serve

to achieve two goals. Each node creates piconet by selecting MIS of its neighbors as its potential slaves. If both neighbours  $u$  and  $v$  select each other as slaves, then we keep only one relationship, based on an arbitrary key so that node with lower  $key2$  remains master. If certain node remains without slaves, it does not need to keep its piconet.

In the second iteration we perform independently two simple procedures to further simplify the scatternet structure. If a master  $u$  has a unique slave  $v$  and  $u$  is not a slave in any piconet, then  $v$  becomes a master (if  $v$  is not already a master) and  $u$  joins  $v$  as slave. Moreover, if all the slaves of a master node  $u$  are themselves master nodes in other piconets then  $u$  loses its piconet and joins the piconets of its slaves. However this is rarely the case, thus even if a master node  $u$  has slaves that are not masters themselves, we try first to transform the scatternet into another one where every slave of  $u$  is a master node in another piconet. If that is possible,  $u$  will join all piconets of its slaves (which are masters themselves) and delete its own piconet. This procedure will keep connectivity and respects the 7 slaves limit.

Our procedure is based on applying certain key(s) for decisions and it will be expressed in terms of a general key. Specific options available for selecting keys are: *node ID*, *-ID*, *(degree, ID)*, *(-degree, ID)*, *(slave degree, ID)*, *(slave degree, degree, ID)*. *Degree* of a node is the number of its 1-hop neighbors. *Slave degree* is the number of slaves selected by MIS procedure explained below. When key is a record composed of an ordered list of subkeys, the comparison is made using lexicographic ordering; i.e. by the first subkey; if equal then compare the second subkey; if also equal then the third subkey is used. Keys can also be based on link to neighbors. For example, the number of common neighbors of  $A$  and  $B$  can be used as the key. This may give priority to select further neighbors, thus creating shorter paths.

Several criteria are used to evaluate BSF protocols although connectivity and degree limitation of the constructed topologies are the main goals of any protocol. Obviously, there is tiny probability that two nodes will never find each other, therefore connectivity cannot be guaranteed even for a network consisting of two nearby nodes. Thus, it is a natural assumption that connectivity is judged subject to established neighbor's knowledge.

The protocol *BlueMis* has two phases: a discovery phase and a second phase with two iterations.

#### Iteration 1 of BlueMis

All nodes run independently the procedure *ComputeMIS* in order to create its own piconet by selecting

a *maximal independent set* of its neighbors as slaves. Each node has two keys,  $key1$  and  $key2$ , known to all its neighboring nodes. To find  $MIS(u)$ , the node  $u$  chooses a node  $v$  from  $Z$ , the set of neighbors of  $u$ , with minimal  $key1(v)$ . Node  $v$  is declared a slave of  $u$  if the following condition is satisfied: node  $u$  will get the value of  $key2(v)$ , checks if  $u$  is already a slave of  $v$  and whether  $key2(u) < key2(v)$ , and then decides whether it should declare  $v$  as its slave. If  $v$  is selected as slave for  $u$  then  $v$  is eliminated from  $Z$ , together with all  $v$ 's neighbors. This is repeated until  $Z$  becomes empty.

#### ComputeMIS ( $u$ )

```

1 MIS( $u$ )  $\leftarrow$   $\emptyset$ ; Master( $u$ )  $\leftarrow$  true
2  $Z \leftarrow N(u)$ ;
3 while  $Z \neq \emptyset$ 
4   do  $v \leftarrow$  Node in  $Z$  with smallest  $key1$ ;
5   Page ( $v$ ,  $u$ ,  $N(v)$ ,  $MIS(v)$ );
6   if  $u$  in  $MIS(v)$ 
7     if  $key2(u) < key2(v)$  then
8        $M(v) \leftarrow M(v) \cup \{u\}$ 
9        $MIS(v) \leftarrow MIS(v) - \{u\}$ 
10       $M(u) \leftarrow M(u) - \{v\}$ 
11       $MIS(u) \leftarrow MIS(u) \cup \{v\}$ 
12   Otherwise
13      $M(v) \leftarrow M(v) \cup \{u\}$ 
14      $MIS(u) \leftarrow MIS(u) \cup \{v\}$ 
15    $Z \leftarrow Z - (N(v) \cup \{v\})$ ;
16 if  $MIS(u) = \emptyset$  then Master( $u$ )  $\leftarrow$  false

```

Notice that in line 16, every node will check independently its set of slaves  $MIS(u)$  and if this set is empty then  $u$  deletes its own piconet.

During this construction, and in line 7, we will also break any symmetric master/master relation between two nodes, using  $key2$ . We assume that  $key2$  contains the unique ID as part of the key (for tiebreaking).

During the execution of *ComputeMIS*( $u$ ), node  $u$  keeps the Information ( $N$ ,  $MIS$ ,  $Master$ ,  $M$ ) where  $N(u)$  is the set of neighbors of  $u$ ,  $MIS(u)$  is the set of slaves of  $u$ ,  $Master$  is a Boolean, i.e. whether  $u$  is a master or not (a slave), and  $M(u)$  the set of nodes that are masters and having  $u$  as a slave.

#### Iteration 2 of BlueMis

In the second iteration we perform independently two simple procedures to further simplify the scatternet structure by deleting piconets that are not essential for the connectivity of the scatternet, therefore lowering the number of piconets.

We define two new procedures: *SlaveProc* and *MasterProc*. Every node will run independently either the

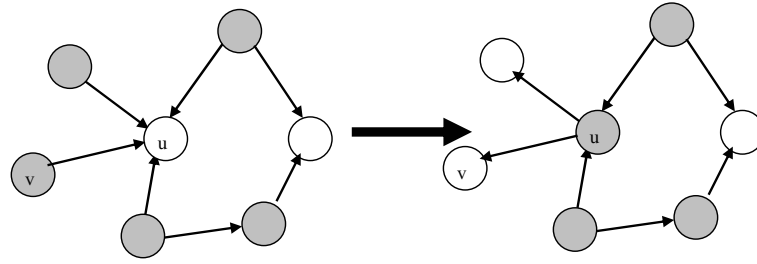


Fig. 1. Slave procedure.

procedure: *MasterProc* if it is a master or the procedure *SlaveProc* if it is a slave. However we want to ensure that all slave procedures are run before the master procedures. Therefore, if  $u$  is a slave, then it runs *SlaveProc*. However, if  $u$  is a master, then it waits for the paging of all its slaves that are not masters before running *MasterProc*. Both procedures will attempt to delete only piconets with a master device  $u$  that is not a slave in any other piconet, that is, when  $M(u)$  is empty.

The master procedure *MasterProc* is run by every master  $u$  and it consists of checking whether  $M(u)$  is empty and all slaves of  $u$  are themselves masters. In that case the piconet of  $u$  will disappear and  $u$  joins the piconets of its slaves, as long as they do not exceed the limit of seven slaves.

The Slave procedure *SlaveProc* procedure is run by every slave  $u$  and it consists of checking all nodes  $v$  in  $M(u)$  to see whether  $M(v)$  is empty and  $MIS(v) = \{u\}$ . That is if  $u$  is the unique slave of a master  $v$  and node  $v$  is not a slave in any other piconet, then the piconet of  $v$  is deleted,  $u$  becomes a master (if  $u$  is not already a master), and  $v$  joins  $u$  as slave (see Fig. 1). This procedure will stop if the number of slaves of the node  $u$  becomes seven.

**SlaveProc (u)**

```

1 If Master(u) = false then
2   u Pages all nodes v in M(u) to get (MIS(v), M(v))
3   If MIS(v) = {u} and M(v) = ∅ and |MIS(u)| < 7 then
4     Master(u) ← true,
5     Master(v) ← false
6     MIS(u) ← MIS(u) ∪ {v}
7     MIS(v) ← ∅
8     M(v) ← {u}
9 Page (done) all nodes in M (u)
    
```

For every master node  $u$ , *MasterProc*( $u$ ) checks the properties of the piconet of  $u$  and decides whether this piconet can be deleted. This simplification is only considered for masters  $u$  that are not slaves for any

other piconet, that is when,  $M(u) = \emptyset$ . In order to be able to delete the piconet of  $u$ , we try first to transform it into another one where every slave of  $u$  is itself a master slave. If that is possible,  $u$  will join all the piconets of its slaves (which are masters themselves) whenever they have room, that is, the number of slaves of the piconet will not exceed seven. In the case that the master  $u$  is able to join all of its slaves' piconets, then it deletes its own piconet (see Fig. 2). This procedure will be finalized assuming that the seven slaves limit is respected.

A master node will check whether all of its slaves  $v$  satisfy one the following two conditions:

1. The node  $v$  is itself a master node;
2. The node  $v$  is not a master node and there is a node  $w$  in  $N(u) \cap M(v)$ .

If condition 2 is satisfied, *Masterproc* will make  $u$  as a slave of  $w$  and  $v$  is deleted from the set of slaves of  $u$ , assuming that there is room in the piconet of  $w$  (see Fig. 3):

$$\begin{aligned}
 MIS(u) &\leftarrow (MIS(u) \cup \{w\}) - \{v\} \\
 M(w) &\leftarrow M(w) \cup \{u\} \\
 M(v) &\leftarrow M(v) - \{u\}
 \end{aligned}$$

If for all slaves of  $u$ , either one of the conditions 1) and 2) holds, then we perform the transformation to try to remove the piconet of  $u$ . All slaves of  $u$  become master nodes and  $u$  joins the piconets of its slaves if they have room for another slave and then deletes that slave from its set of slaves. If this transformation is successful for all slaves of  $u$ , then  $u$  deletes its own piconet.

For a master node  $u$ , we set a Boolean value *Simplify*( $u$ ) to be true if  $M(u) = \emptyset$  and one of the above two conditions holds for every slave  $v$  of the piconet of  $u$ .

**MasterProc (u):**

```

1 WaitPage (done) from all slave neighbors
2 If Master (u) = true then u Pages all nodes v in MIS (u) to get (N (v), MIS (v), Master (v), M (v))
    
```

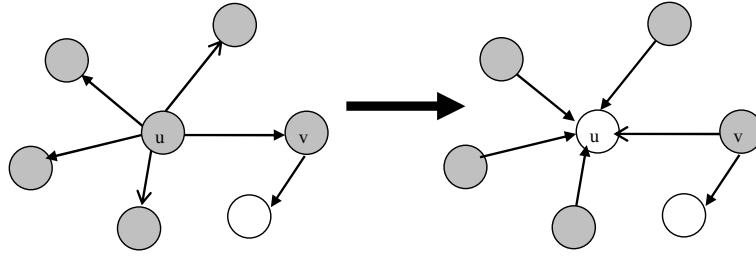


Fig. 2. Master procedure, case 1.

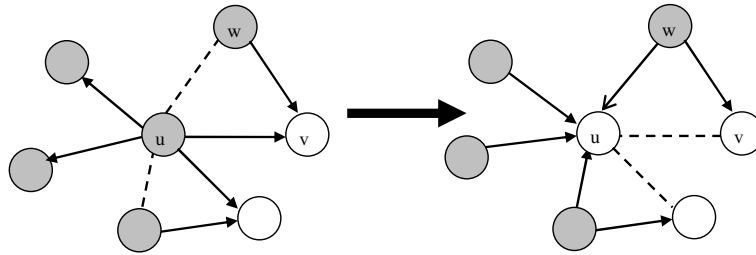


Fig. 3. Master procedure, case 2.

```

3  If Simplify ( $u$ ) = true then for every  $v$  in MIS
(u)
4    If Master ( $v$ ) = true and  $|\text{MIS}(v)| < 7$  then
5       $M(u) \leftarrow M(u) \cup \{v\}$ 
6       $M(v) \leftarrow M(v) - \{u\}$ 
7       $\text{MIS}(v) \leftarrow \text{MIS}(v) \cup \{u\}$ 
8      If there is  $w$  in  $N(u) \cap M(v)$  and  $w \notin \text{MIS}(u)$ 
and  $|\text{MIS}(w)| < 7$  then
9         $\text{MIS}(w) \leftarrow \text{MIS}(w) \cup \{u\}$ 
10        $M(u) \leftarrow M(u) \cup \{w\}$ 
11        $M(v) \leftarrow M(v) - \{u\}$ 
12 If  $\text{MIS}(u) = \emptyset$  then Master ( $u$ )  $\leftarrow$  false

```

It may happen that two neighboring masters try to simplify their piconets at the same time. We thus added the condition  $w \notin \text{MIS}(u)$  in line 8 of *MasterProc* to avoid both masters using each other at the same time for simplification and thus potentially leading to a disconnected topology. Notice too that *MasterProc* may create piconets where the slaves do not form necessarily an independent set.

While the number of slaves of each master is limited, and the scatternet is connected, the number of slave roles for each node is not limited, and in some cases, e.g. complete graph, one node can be selected as a slave to all other nodes. This is the same problem shared with BlueMesh. The main advantage of the new algorithm is reducing the number of iterations to two and therefore obtaining a faster BSF.

Let  $R$  be the nodes' transmission range and assume that two nodes  $u$  and  $v$  are neighbors if and only if their

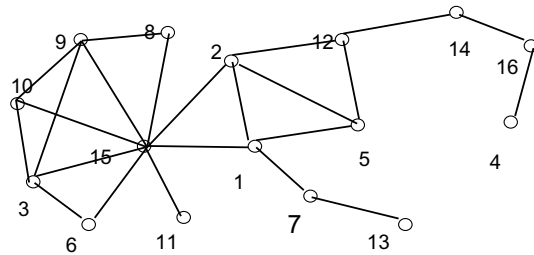


Fig. 4. Initial topology.

Euclidian distance is at most  $R$ . It is a well known fact that in the unit disk graph, any device  $v$  can reach all its neighbors by selecting at most five among them. Since the MIS of any node will have at most 5 nodes, and during the execution of *ComputeMis* no more than  $O(n)$  such exchanges of neighbors will happen, where  $n$  is the number of devices. There will be no need for symmetric knowledge among neighbors, specifically, if a node  $u$  is aware of a node  $v$  at most two hops away, then  $v$  does not need to have the same information about  $u$ .

#### An example illustrating BlueMis

The ComputeMIS protocol is illustrated in Figs 4, 5 and 6. We will use the node ID appearing besides each device, as *key1*. A line between any two devices indicates that they have discovered each other during the discovery phase. Devices with a grey color are the masters and the others are the slaves. An arrow from a

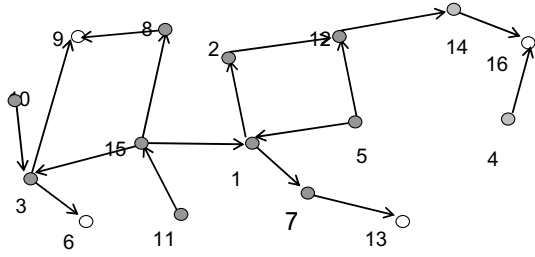


Fig. 5. Scatternet topology produced after iteration 1 of BlueMis.

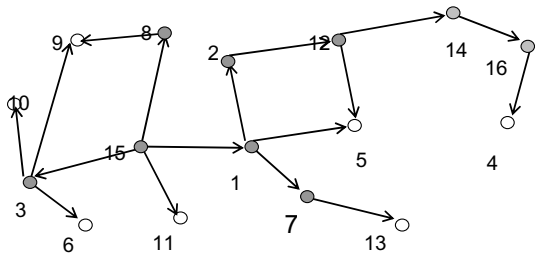


Fig. 6. The scatternet topology produced by BlueMis.

device  $u$  to a device  $v$  indicates that  $v$  is a slave of the master  $u$ .

In the first iteration, each node creates its own piconet by selecting its maximal independent set. At the same time, we avoid the situation where two neighbors  $A$  and  $B$  select each other as slaves. For instance, node 15 selects as a first potential slave a neighbor with a minimal  $keyI$ , which is node 1. Then node 15 pages node 1 to check for any inconsistency with a symmetric Master/Master relation. Node 1 will return the list of its neighbors  $N(1)$  and its list of slaves  $MIS(1)$  chosen so far. Node 1 will select the nodes  $\{2, 7\}$  as its  $MIS$ . Since 15 is not a part of  $MIS(1)$ , the node 1 is selected in  $MIS(15)$ . Device 1 and all of its neighbors, i.e.  $\{2, 5, 7, 15\}$  are taken out of the list of the potential slaves for node 15. The next node to be selected as a potential slave for 15 is node 3. Then node 15 pages node 3 to check for any inconsistency with a symmetric Master/Master relation. Node 3 will return the list of its neighbors  $N(3)$  and its list of slaves  $MIS(3)$  chosen so far. Since node 3 will select nodes  $\{6, 9\}$  as its  $MIS$ , node 15 is not a part of  $MIS(3)$  and thus 3 is selected as a part of  $MIS(15)$ . The same thing will happen with the node 8, next to be selected in  $MIS(15)$ . The next and last node to be selected as a potential slave for 15 is node 11. Then node 15 pages node 11 to check for any inconsistency with a symmetric Master/Master relation. Node 11 will return the list of its neighbors  $N(11)$  and its list of slaves  $MIS(11)$  chosen so far. In case 11 did already choose 15 as one of its slaves

and since 15 is larger than 11, node 15 will select 11 as its slave and 11 will take 15 off its list of slaves. Otherwise, node 15 selects 11 as a slave and no changes are made to the lists of node 11. If later in the process, node 11 tries to select 15 as slave, it will page 15 for its list of slaves, however, since 11 belongs to that list and moreover  $11 < 15$ , nothing will be done. By the end of this first iteration of our protocol, each device will check whether its constructed  $MIS$  is empty. For instance, in this example all the devices 6, 9, 13 and 16 should delete their piconets, since they have no slaves (see Fig. 5).

In the second iteration, each of the slave devices  $\{6, 9, 13, 16\}$  will run independently the procedure *Slaveproc*. For instance node 16 will execute *Slaveproc*. Since its neighbor 4 is a master with unique slave 16, then device 16 becomes a master and 4 becomes its slave. Nothing will be done for the other slaves since they don't have the right property.

When *Slaveproc* finishes for all slave nodes, each of the master devices  $\{1, 2, 3, 5, 7, 8, 10, 11, 12, 14, 16\}$  will run independently the procedure *MasterProc*.

The *MasterProc* will have an effect on nodes 5, 10, 11 since they can be simplified: they are not slaves in another piconet and all of their slaves are masters themselves. All nodes 5, 10 and 11 will join the piconets of their slaves and delete themselves. The obtained scatternet is shown in Fig. 6.

Our next theorem shows that the protocol *BlueMis* creates a connected scatternet topology.

**Theorem 1.** For any choices of  $keyI$  and  $key2$ , the protocol *BlueMis* creates a connected scatternet topology, if the initial graph is connected and arbitrarily defined.

The proof of Theorem 1 is an obvious consequence of the following three lemmas.

**Lemma 1.** For any choices of  $keyI$  and  $key2$ , and after running the procedure  $ComputeMIS()$ , the scatternet remains connected, if the initial graph is connected and arbitrarily defined.

*Proof of Lemma 1.* We will prove that constructed scatternet contains a minimal spanning tree (MST) as its subgraph. Let the 'length' of each edge be 1. To distinguish edges, we make them different and uniquely sortable by defining  $key(B,A) = key(A,B) = (keyI(A), keyI(B))$  where  $keyI(A) < keyI(B)$ . Note that neighbors  $B$  of node  $A$  are considered as slaves in ascending order of  $key(A,B)$ . We then follow Kruskal's MST

construction algorithm, where edges are considered for inclusion in MST in ascending order, and included if and only if their addition does not create a cycle. Let  $(K2, K3)$ ,  $key1(K2) < key1(K3)$  be a pair of nodes that did not select each other as slaves. We show then that this edge is not in MST. Since  $K3$  did not select  $K2$ , there exist common neighbor  $K1$ , selected by  $K3$  as slave, with  $key1(K1) < key1(K2)$  (node can be eliminated only by neighbor with lower key). Then  $key(K1, K3) < key(K2, K3)$ , and  $key(K2, K1) < key(K2, K3)$ . Therefore edges  $(K1, K2)$  and  $(K1, K3)$  have lower keys than edge  $(K2, K3)$  and were already considered previously for MST inclusion. Thus pairs  $(K1, K2)$  and  $(K1, K3)$  are already connected within MST (either by being in MST or by presence of cycles connecting them). But then the pair  $(K2, K3)$  is already connected in MST, and is not included in MST. It follows that all MST edges are selected for master-slave relations, and the scatternet remains then connected. Regardless of  $key2$ , the elimination of master slave pair  $AB$  when pair  $BA$  already exists, and elimination of piconets left without any slave, obviously does not create partitions.  $\square$

**Lemma 2.** For any choices of  $key1$  and  $key2$ , and after running *Slaveproc* of *BlueMis*, the scatternet remains connected, if the initial graph is connected and arbitrarily defined.

*Proof of Lemma 2.* Consider any pair of devices  $(u, v)$  and let  $u = w_1, w_2, \dots, w_n = v$  be an alternating master/slave path that connects  $u$  to  $v$ . in the original scatternet topology. Since *Slaveproc* changes only the roles of some devices, then the same path remains in the scatternet topology. Moreover, this procedure acts only on masters of degree 1 and their unique slave, thus the only possible change is either on  $u = w_1$  that becomes a slave of  $w_2$  or to  $w_n = v$  that becomes a slave of  $w_{n-1}$ , and in both cases we still have an alternating path. Therefore, after running *Slaveproc* of *BlueMis*, the scatternet remains connected.  $\square$

**Lemma 3.** For any choices of  $key1$  and  $key2$ , and after running *MasterProc* of *BlueMis*, the scatternet remains connected, if the initial graph is connected and arbitrarily defined.

*Proof of Lemma 3.* Consider any pair of devices  $(u, v)$  and let  $u = u_1, u_2, \dots, u_n = v$  be an alternating master/slave path that connects  $u$  to  $v$ . in the original scatternet topology. A problem could occur if one of the nodes  $u_i$  in this path lost its status as a master node.

Since the procedure applies only to masters which are not slaves in another piconet, the only possibility is  $u_i = u_1 = u$ . or  $u_i = u_n = v$ . Without loss of generality we may assume that the transformation affected  $u_1 = u$ . There are two possibilities to consider:

The device  $u_2$  is a master: the device  $u$  deleted its own piconet and became a slave in the piconet of  $u_2$ . In this case the original path between  $u$  and  $v$  remains an alternating master/slave path.

The device  $u_2$  is a slave in the piconet of  $u$  and it can be simplified. That is, there is another master node  $w$  in  $N(u) \cap M(u_2)$  and  $|MIS(w)| < 7$ . In this case  $w$  became a slave of  $u$  and  $u_2$  is not anymore a slave of  $u$ . Thus, the original path is replaced by the new alternating master/slave path:  $u = u_1, w, u_2, \dots, u_n = v$ .

Notice that *MasterProc* cannot create new devices with the property  $M(v) = \emptyset$ .  $\square$

## 4. Experiments

We present our experimental results for *BlueMis* and a comparison with *BlueMesh* in terms of various characteristics. We have simulated *BlueMis* to demonstrate its effectiveness in generating limited degree and connected scatternet with certain desirable properties. We implemented the *BlueMesh* protocol within the same environment in order to compare it properly with our proposed scheme. This choice is based on the fact that among all methods that do not use position information, *BlueMesh* has great advantages (No more than seven slaves per piconet, an average of 2.5 roles per node, route lengths not significantly longer than shortest path in the network) and appears to be currently the best available method for multi-hop networks.

We used a simulator of BT-based ad hoc networks with 200 nodes, implemented in Java. In this research the BT stack is not implemented and the study is limited to network-layer details. Our simulation results are obtained by means of a ns2- based simulator that implements the BT stack.

The simulation parameters were chosen to match closely those used in [8] for *BlueMesh* to ensure a fair comparison. We concentrated our evaluation on how the topology was formed from a set of disjoint and discoverable devices.

We generate various scenarios randomly based on network size and the dimension of the geographic area. For our simulations, the number of Bluetooth devices has been varied between 40 and 200 devices; each de-

vice has a maximum transmission range of 10 meters. The devices are randomly and uniformly scattered in a geographic area which is a square of side  $L$ . The resulting geographic network topology graph is a unit disk graph; it is assumed that two nodes are in each other's transmission range if and only if their distance in the plane (Euclidean distance) is  $\leq 10$  m.  $L$  has been set to either 40 meters or 60 meters to permit the testing and comparison of the protocols of moderately ( $L = 60$  m) to heavily ( $L = 40$  m) dense networks. More than 95% of the total generated graphs are disconnected for networks with forty and fifty nodes, thus we omit this class of graphs during the simulation.

Nodes are numbered as they are generated. This numbering serves as node  $ID$  in BSF decisions. In our experiments we choose the  $ID$  as the key for *BlueMesh*, and for the *BlueMIS* the node  $ID$  is used as  $key1$  and ( $slave\ degree, ID$ ) as  $key2$ . To better understand the characteristics of the generated graphs, we provide some measurements taken during the simulation for both models. Table 1 shows some characteristics of the randomly generated networks.

As expected, our protocol does not perform well when the network is dense. In fact, if all  $n$  devices see each other (that is, all devices are within the range of each other) then *BlueMIS* will produce  $(n - 7)$  piconets. In the first iteration, device 1 will choose device 2 as a slave and all other devices will choose device 1. Apart from device 2, which will delete its own piconet after losing the Master/Master tiebreaking with device 1, all other devices will keep their own piconets. In iteration 2, only 6 devices will join the piconet of device 1 and all other piconets will stay unchanged.

We evaluate our proposed protocol based on five primary metrics: average shortest path ratio, average of master/slave roles, average number of roles, number of created piconets, and finally the average number of slaves per piconet.

A major gain of our proposed protocol over *BlueMesh* is the number of iterations. It represents our primary contribution. *BlueMesh* is significantly affected especially by large network sizes, contrary to our approach where the number of iterations is constant, independent of the network size. Simulation results show that *BlueMesh* uses on average 2.7 to 4.5 iterations. The number of piconets produced is equivalent to the number of selected master nodes in the scatternet. It is important to keep this metric as low as possible, because this may increase the network overhead. The number of piconets produced by *BlueMIS* is much lower for moderately dense networks than for dense

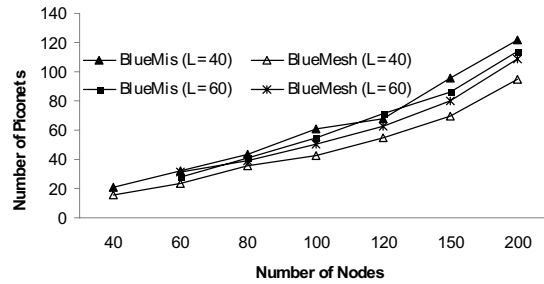


Fig. 7. Number of piconets.

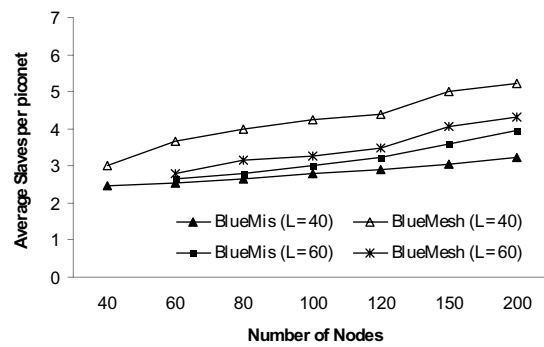


Fig. 8. Average number of slaves per piconet.

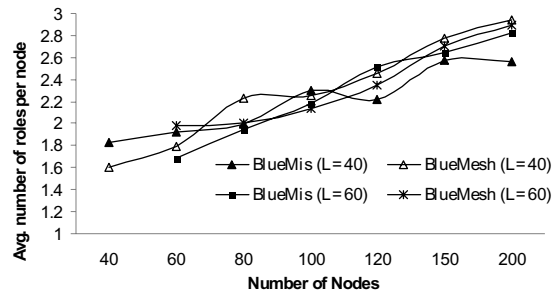


Fig. 9. Average number of roles per node.

networks. We obtain similar numbers as in *BlueMesh* when the network is moderately dense; however there is a loss of about 10 to 15 % for the number of piconets when the network is dense (see Fig. 7).

Similar situation happens with the number of slaves per piconet and the average number of roles (see Figs 8 and 9). For moderately dense network *BlueMesh* and *BlueMIS* produces results in the same range when the network is moderately dense network, however *BlueMesh* is better than *BlueMIS* for dense networks.

*BlueMIS* produced network topologies with an average shortest path length better than in *BlueMesh*. For moderately dense networks, we noticed a gain of 5 to 10% (see Fig. 10).

Table 1  
Generated heavily and moderately dense networks

N		60	70	80	100	120	140	160	180	200
Average degree	L = 40	11.34	11.66	15.7	18.96	22.02	25.46	28.52	31.4	36.4
	L = 60	6.2	6.37	7.13	9.56	11.22	14.52	18.24	23.9	26.4
Maximal degree	L = 40	19.5	21.4	26.9	30.1	34.3	39.4	45.7	51.2	56.8
	L = 60	11.3	13.5	15.7	18.2	19.3	22.3	26.9	28.4	31.2
Minimal degree	L = 40	2.5	3.0	3.2	3.6	4.6	5.8	6.1	6.7	7.1
	L = 60	1	1.3	2.1	3.5	4.7	5.4	5.9	6.3	7.8
Average shortest path	L = 40	3.35	3.06	2.84	2.80	2.71	2.69	2.62	2.57	2.49
	L = 60	4.69	4.64	4.52	4.33	4.24	4.02	3.85	3.82	3.76
% connected graph	L = 40	43	65	71	82	85	90	96	99	99
	L = 60	9	11	15	22	31	43	57	69	77

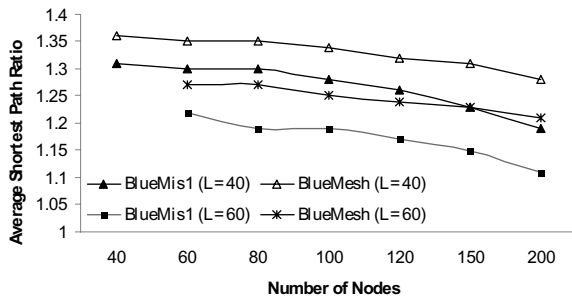


Fig. 10. Average shortest path ratio.

## 5. Conclusions

In this paper, we introduced BlueMis, a new protocol for the establishment of multihop wireless adhoc network of Bluetooth devices. It is based on maximal independent set and guarantees connectivity and degree limitation. It is executed at each node with no prior knowledge of the network topology, thus being fully distributed.

Our major contribution in this paper is the simplicity of BlueMis. Unlike BlueMesh that uses on average 2.7 to 4.5 iterations, the second phase of our new protocol is executed in two iterations.

The designed protocol is implemented and compared, in terms of various characteristics, with BlueMesh, which we consider to be the best competing protocol. In our experiments, we focused on the following parameters: number of piconets, average number of slaves per piconet, average number of roles per device, average shortest-path length. Simulations show that BlueMesh performs slightly better than BlueMis in terms of the number of piconets and the number of slaves per piconet.

An interesting and major open problem in the area is to design Bluetooth scatternet formation algorithms that will guarantee connectivity and degree limitation (for both the master and the slave roles) without using the position information.

## Acknowledgements

This research is supported by Canada NSERC Collaborative Research and Development grant CRDPI 319848-04, and the UK Royal Society Wolfson Research Merit Award.

## References

- [1] A. Aggrawal, M. Kapoor, L. Ramachandran and A. Sarkar, Clustering algorithm for wireless ad hoc networks. Boston, Massachusetts, USA: s.n., August 2000. *The fourth International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 54–63.
- [2] Bluetooth Special Interest Group. Bluetooth Specification Version 1.2. [Online] November 2003. <http://www.bluetooth.com>.
- [3] C. Law, A.K. Mehta and K.Y. Siu, Performance of a new Bluetooth scatternet formation protocol. Long Beach, California, USA: s.n., Oct 2001. *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc2001)*, 183–192.
- [4] Y. Daadaa, I. Stojmenovic and N. Zaguia, Bluetooth scatternet formation of wireless devices based on maximal independent sets, *Proc. ICTA 2007*, Tunisia, 241–247.
- [5] L.E. Hodge and R.M. Whitaker, *What are Characteristics of Optimal Bluetooth Scatternets*, Mobiquitous, Boston, Aug. 2004.
- [6] K.E. Persson, D. Manivannan and M. Singhal, *Bluetooth Scatternets: Criteria, Models and Classification*, Ad Hoc Networks, 2004.
- [7] C. Petrioli and S. Basagni, Degree-constrained multihop scatternet formation for Bluetooth networks, *Proc. IEEE GLOBECOM 2002*, Taipei, Taiwan, November 2002.
- [8] C. Petrioli, S. Basagni and I. Chlamtac, BlueMesh: Degree-constrained multi-hop scatternet formation for Bluetooth networks, *Mobile Networks and Applications* 9 (2004), 33–47.
- [9] T. Salonidis, P. Bhagwat, L. Tassiulas and R. LaMaire, Distributed topology construction of Bluetooth personal area networks, *Proc. IEEE INFOCOM*, 2001.
- [10] G.V. Zaruba, S. Basagni and I. Chlamtac, Bluetrees – scatternet formation to enable Bluetooth based ad hoc networks, *Proc. IEEE ICC*, 2001, 273–277.
- [11] I. Stojmenovic, Dominating set based Bluetooth scatternet formation with localized maintenance. *Fort Lauderdale: CD Proc. IEEE Int. Parallel and Distributed Processing Symposium and Workshops*, April 2002.

- [12] I. Stojmenovic and N. Zaguia, Bluetooth scatternet formation in ad hoc wireless networks, Chapter 9, in: *Performance Modeling and Analysis of Bluetooth Networks: Network Formation, Polling, Scheduling, and Traffic Control*, J. Misic and V. Misic, eds, Auerbach Publications, Taylor & Francis Group, 2006, pp. 147–171.