Mobile Networks and Applications 8, 485–498, 2003


# A New Bluetooth Scatternet Formation Protocol *

CHING LAW, AMAR K. MEHTA and KAI-YEUNG SIU
*Massachusetts Institute of Technology, USA*

**Abstract.** A Bluetooth ad hoc network can be formed by interconnecting piconets into scatternets. The constraints and properties of Bluetooth scatternets present special challenges in forming an ad hoc network efficiently. In this paper, we present and analyze a new randomized distributed protocol for Bluetooth scatternet formation. We prove that our protocol achieves $O(\log n)$ time complexity and $O(n)$ message complexity. The scatternets formed by our protocol have the following properties: (1) any device is a member of at most two piconets, and (2) the number of piconets is close to be optimal. These properties can help prevent overloading of any single device and lead to low interference between piconets. We validate the theoretical results by simulations, which also show that the scatternets formed have $O(\log n)$ diameter. As an essential part of the scatternet formation protocol, we study the problem of device discovery: establishing multiple connections simultaneously with many Bluetooth devices. We investigate the collision rate and time requirement of the inquiry and page processes. Our simulation results indicate that the total number of packets sent is $O(n)$ and that the maximum number of packets sent by any single device is $O(\log n)$.

**Keywords:** Bluetooth, ad hoc networks, resource discovery, topology construction

## 1. Introduction

Bluetooth [4,7,15,22] is an emerging low-cost and low-power short-range radio technology. Many useful applications can be supported by an ad hoc network over Bluetooth. For example, in a conference room, a special announcement can be broadcast to the Bluetooth-enabled mobile phones and handheld computers through an ad hoc network. Bluetooth ad hoc networks can also be used for rapid deployment of EMID (electromagnetic identification) readers [2].

The area of ad hoc networking has gathered significant research interests in recent years. Many studies have concentrated on the routing issues of ad hoc networks [16,23]. These studies usually assume that any two in-range nodes can communicate with each other. Therefore, an ad hoc network can be modeled as a graph such that the in-range nodes are adjacent. For example, simulation-based studies [5,6] of ad hoc routing protocols have been conducted with a link-layer model based on or similar to the IEEE 802.11b standard.

An ad hoc network based on Bluetooth, however, brings new challenges. There are specific Bluetooth constraints not present in other wireless networks. For example, a Bluetooth network is composed of *piconets*. Each piconet contains one master and up to seven slaves. Piconets can be connected into a larger *scatternet* (figure 1) by sharing slaves. As shown by Miklos et al. [14] and Zurbes [26], the configuration of a scatternet has significant impact on the performance of the network. For instance, when a scatternet contains more piconets, the rate of packet collisions increases. Before we can make effective use of Bluetooth ad hoc networking, we must first devise an efficient protocol to form a scatternet from isolated Bluetooth devices.

* This work is supported in part by the MIT Auto-ID Center (autoidcenter.org). Corresponding author: Ching Law, 305 Memorial Drive, Cambridge, MA 02139, USA.

In this paper, we study the problem of scatternet formation in the situation where the devices are in-range of one another. The communication range is at least 10 meters according to the current Bluetooth specification. This means that our formation algorithm should work when the maximum distance between any two devices is at most 10 meters. We will discuss in section 8 how the algorithm should adapt if the assumption is not satisfied.

We adopt a two-layer approach to this problem. First, we investigate how these devices can be organized into scatternets. We design and evaluate the performance of a new scatternet formation protocol. Second, as a subroutine of the formation protocol, we study how the devices can discover each other efficiently.

This paper is organized as follows. In section 2, we discuss the related research on Bluetooth scatternets. To get a better understanding of how our results differ from prior work, readers may skip this section and come back to it after going through the results of this paper. In section 3, we introduce the problem of scatternet formation. Our new scatternet formation protocol is presented in section 4. We present the-

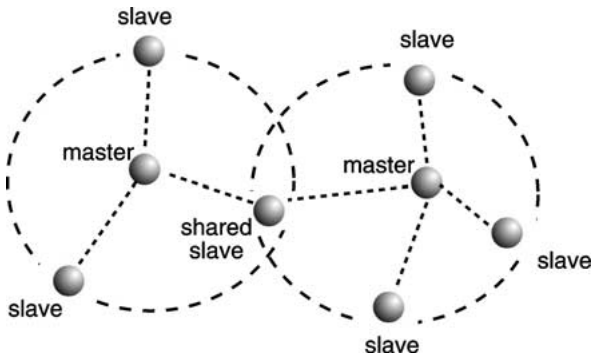


Figure 1. A Bluetooth scatternet.

oretical analyses and simulation results of our protocol in section 5. We discuss device discovery with simulation results in section 6. In section 7 we estimate the overall performance of the protocol. We discuss several variations and extensions to our protocol in section 8 and conclude in section 9. Theoretical results in this paper have appeared in a conference paper [12]. Simulation results in this paper have appeared in a conference paper [10] and a thesis [13].

## 2. Related work

Miklos et al. [14] apply heuristics to generate scatternets with some desirable properties. They evaluate these scatternets of different characteristics through simulations. Johansson et al. [9] perform link-layer simulations of piconets. Raman, Bhagwat, and Seshan [17] argue for cross-layer optimization in Bluetooth Scatternets.

Aggarwal et al. [1] introduce a scatternet formation algorithm. Their algorithm first partitions the network into independent piconets, and then elects a "super-master" that knows about all the nodes. However, the resulting network is not a scatternet, because the piconets are not inter-connected. A separate phase of re-organization is required.

Salonidis et al. [18] discuss the issues of symmetric connection between a pair of Bluetooth devices. In their symmetric protocol, the devices switch states (INQUIRY and INQUIRY SCAN) with a random schedule. In contrast, in our work, the devices switch states periodically, but pick the states randomly.

Salonidis et al. [19] introduce a scatternet formation algorithm – Bluetooth Topology Construction Protocol (BTCP). BTCP has three phases: (1) a coordinator is elected with a complete knowledge of all devices, (2) this coordinator determines and tells other masters how a scatternet should be formed, and (3) the scatternet is formed according to the instructions. A formation scheme is presented in [19] for up to 36 devices. In contrast, our algorithm has only one phase. Since the topology is decided by a single device (the coordinator), BTCP has more flexibility in constructing the scatternet. However, if the coordinator fails, the formation protocol has to be restarted. BTCP's timeout value for the first phase would affect the probability that a scatternet is formed. Our protocol's timeout value for each round only affects the overall performance of the protocol – the scatternet will be formed with certainty. In addition, BTCP is not suitable for dynamic environments where devices can join and leave after the scatternet is formed.

The algorithms in [1,19] depend on a single device to design the scatternet topology and notify other devices. Therefore these algorithms will have time complexity $\Omega(n/k)$, where $n$ is the number of nodes, and $k$ is the maximum number of slaves in a piconet. In comparison, our algorithm consists of a single phase and has $O(\log n)$ time complexity. However, as pointed out in [19], the coordinator election phase dominates the total time requirement. Thus, the advantage of our protocol's $O(\log n)$ time complexity might not be relevant in practice unless the number of devices is very large. Moreover, we note that at least the phase II of BTCP can be modified to run in $O(\log n)$ time, if the topological information is distributed along a tree. However, a tree-based distribution scheme will increase the complexity of the protocol.

Tan [21] gives a distributed Tree Scatternet Formation (TSF) protocol. The extensive simulation results indicate relatively short scatternet formation latency. However, TSF is not designed to minimize the number of piconets. The simulation results suggest that each master usually has fewer than 3 slaves. In comparison, our protocol guarantees that all but one of masters have at least 6 slaves.

Bluetree [25] and Bluenet [24] are scatternet formation protocols for larger-scale Bluetooth networks, in which the devices can be out of range with one another. Simulation results of the routing properties of the scatternets were presented in [24,25]. However, there were no simulation or theoretical analyses on the performance of the scatternet formation process.

## 3. Preliminaries

In this section we introduce some terminologies and performance measures for the scatternet formation problem.

Bluetooth devices share 79 channels of 1 MHz bandwidth in the 2.4 GHz band using frequency hopping. When two Bluetooth devices are connected, one of the devices acts as a *master* and the other device acts as a *slave*. Any Bluetooth device can perform the role of a master or a slave.

A Bluetooth device can discover other devices by the inquiry process. A master in INQUIRY state hops 3,200 times per second according to a 32-channel inquiry hopping sequence. At the same time, a slave in INQUIRY SCAN state changes its listening frequency every 1.28 seconds, along the same sequence.

If the inquiry process succeeds, the master learns the address (which is unique for each Bluetooth device) and the clock of the slave. Then the master and the slave can be connected with the page process. In the page process, the master in PAGE state contacts the slave with a 32-channel page hopping sequence, which is a function of the slave's address and (estimated) clock. Similarly, the PAGE SCAN slave hops with the period of 1.28 seconds along the same sequence. After the master and the slave are connected, they communicate with a hopping sequence over all 79 channels at the rate of 1600 hops per second. This hopping sequence is determined by the master's clock and address.

A *piconet* consists of 1 master and 1–$k$ active slaves.[1] All packets are exchanged between a master and its slaves within a piconet. There is no direct master–master or slave–slave communication. A device can be a slave in several piconets but be a master in only one piconet. The *degree* of a device is the number of piconets to which the device belongs. A device is *unshared* if its degree is 0 or 1. Otherwise, it is *shared*.

[1] $k = 7$ in Bluetooth Specification 1.1 [20].

A *scatternet* is a set of piconets connected through shared devices.

The problem of scatternet formation: How does a collection of isolated devices form a scatternet? The devices are isolated in the beginning; each device is not aware of the other devices. Therefore, the scatternet formation protocol must be distributed. We assume that the devices are in the communication range of each other. Thus, potentially, any two devices can be connected directly.

A scatternet formation protocol has two major performance measures:

- Time complexity – amount of time to form a scatternet. A scatternet should be formed as fast as possible to minimize the delay experienced by the users.
- Message complexity – number of messages sent between the devices. This is important because Bluetooth devices usually operate with limited power. By reducing the number of messages sent, power consumption is conserved.

Futhermore, it is also crucial to have scatternets of good quality. It is not very useful to have scatternets leading to poor communication performance. Thus, we should aim to form a scatternet that facilitates inter-piconet communications. It is not easy to quantify the quality of a scatternet, but we believe the following measures are good indicators:

- Number of piconets – a measurement of a scatternet's efficiency. Since all piconets share the same set of 79 channels, there will be more collisions when there are more piconets. As shown in [26], the burst failure rate increases with the number of piconets.
- Maximum degree of the devices – the maximum number of piconets that any device belongs to. Since the piconets communicate through shared slaves, if a slave belongs to many piconets, then this slave could become the bottleneck of inter-piconet communications. A shared slave has to be time multiplexed between the piconets that it belongs to. Therefore, a shared slave of high degree could become overloaded.
- Network diameter – maximum number of hops between any pair of devices. This will provide us with an estimation of the maximum routing delay of the scatternet.

A good balance among the quality measures is desirable. Consider, for example, a star topology: a single "central" slave is shared by all piconets. In such a scatternet of $n$ devices with every piconet containing $k$ slaves, there are $\lceil (n-1)/k \rceil$ piconets. Although the number of piconets is minimized, minimized (see remark 1), this scatternet probably would not perform very well in practice because the shared slave will be overwhelmed, unless the network is small.

*Remark 1.* Let $k$ be the maximum number of slaves in a piconet. A scatternet of $n$ devices must contain at least $\lceil (n-1)/k \rceil$ piconets.

*Proof.* See appendix A. □

## 4. Scatternet formation

In this section, we first present our scatternet formation protocol and then evaluate its performance and properties by analyses and simulations. The development of this algorithm was inspired by our research on resource discovery algorithms in general networks [11]. The main idea is to merge pairs of connected components until one component is left. Each component has a leader. In each round, a leader either tries to contact another component or waits to be contacted. The decision of each leader is random and independent. Our protocol in [11] forms a complete graph in $O(\log n)$ rounds. In this paper, we apply the same idea to connect Bluetooth devices in $O(\log n)$ rounds.

### *4.1. Algorithm*

Initially, we are given a set of isolated but in-range devices. During the execution of the algorithm, the devices are partitioned into *components*. A component is a set of interconnected devices, and can be a single device, a piconet, or a scatternet. There is one *leader* in each component. For a single-device component, the only member is the leader. For a piconet, the master is the leader. For a scatternet, one of the masters is the leader. When a leader *retires*, it stops being a leader and will be inactive for the rest of the algorithm (unless it becomes a leader again). For any device $v$, let $\mathcal{S}(v)$ be the set of $v$'s slaves. If $v$ is not a master or has no slaves, then $\mathcal{S}(v) = \emptyset$. Let $k \geqslant 2$ be the maximum number of slaves allowed in a piconet. Thus $\mathcal{S}(v) \leqslant k$ for any $v$.

In lemma 2, we will prove the following invariants for the algorithm:

- Each leader either has no slave, or has at least one unshared slave in its piconet.
- Each leader has fewer than $k$ slaves in its piconet, i.e., $|\mathcal{S}(u)| < k$ for any leader $u$.

All leaders execute procedure MAIN in the beginning of each round. We assume a constant $\delta$, such that procedure MAIN and the procedures called by it can be completed in $\delta$ seconds. A good choice of $\delta$ can be found by simulations (see section 6) and by prototyping. We assume that all leaders call procedure MAIN at time $t_0 + i\delta$, for $i = 0, 1, \ldots$, where $t_0$ is the start time.

In the beginning, all devices are leaders. In procedure MAIN, a leader calls SEEK with probability $p \in (1/3, 2/3)$. Otherwise, the leader calls SCAN or asks an unshared slave to call SCAN. During each round, only one device in each component should call SEEK or SCAN.

```
MAIN(leader u)
1   x ← a random number in [0, 1)
2   if x < p (1/3 < p < 2/3)
3     then SEEK(u)
4     else if S(u) = ∅
5             then SCAN(u)
6             else v ← an unshared slave of u
7                  SCAN(v)
```

When a leader executes SEEK, it tries to acquire a new slave (which is running SCAN). However, the leader may not always succeed, because, in any given round, the number of devices running SCAN can be smaller than the number of devices running SEEK. Therefore, if a leader is not able to contact a slave after certain time, it should give up and run MAIN again in the next round. Similarly, SCAN might also fail in any given round. During each round, a matching is found between the SEEK devices and SCAN devices. The number of connections established (size of the matching) is the smaller of the number of SEEK devices and the number of SCAN devices.

```
SEEK(u)
1   u performs INQUIRY
2   if a slave v is found
3     then u connects to slave v by PAGE
4          //S(u) ← S(u) ∪ {v}
5          CONNECTED(u, v)
```

```
SCAN(v)
1   v performs INQUIRY SCAN
2   if v is contacted by a master u
3     then v waits for u in PAGE SCAN
```

We note that SEEK and SCAN devices will go into PAGE and PAGE SCAN modes, respectively, after all inquiries are completed. The amount of time required is investigated in section 6. In general, we make sure that each master is matched to only one slave, and vice versa. When a leader $u$ running SEEK connects to a slave $v$ running SCAN, procedure CONNECTED$(u, v)$ is called.

Procedure CONNECTED$(u, v)$ merges the component of $u$ and the component of $v$. There are several cases:

1. If $v$ is an isolated leader, then $v$ would become a slave of $u$ unless the piconet of $u$ has become full, in which case a new piconet with master $v$ and unshared slave $y$ is created, as shown in figure 2. This is necessary because otherwise $u$'s piconet would be full, violating the second invariant. To satisfy the first invariant, we also need to give the new master $v$ an unshared slave $y$.
2. If $v$ is not an isolated leader, then let $w$ be the master of $v$.
   (a) If the piconet of $u$ and the piconet of $w$ can fit into a single piconet (with at most $k - 1$ slaves because of the second invariant), then $w$ and its slaves join the piconet of $u$, as shown in figure 3.
   (b) Otherwise, we cannot merge the two piconets, and thus we should let $w$ retire.
      (i) If $u$ was an isolated master, $u$ would not have an unshared slave. Thus, we will let $u$ become the slave of retiring master $w$ and let $v$ become an unshared slave of $u$ (figure 4).
      (ii) If the merged piconet is just full, violating the second invariant, we will need to let $v$ become a master and give it an unshared slave $y$ (figure 5).

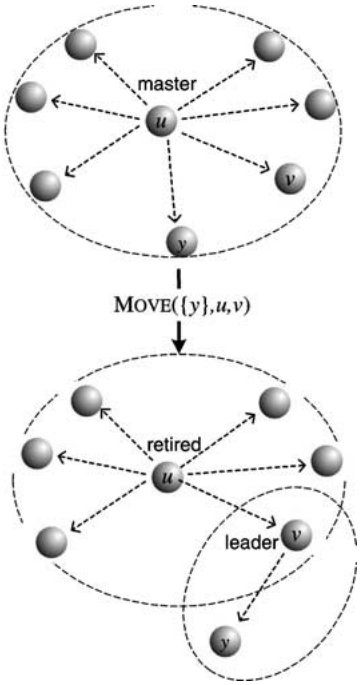


Figure 2. Lines 5–7 in procedure CONNECTED for $k = 7$.

   (iii) Otherwise, we will just try to move as many nodes allowed by the invariants as possible from the piconet of $u$ to the piconet of retiring master $w$, so as to reduce the overall number of piconets (figure 6).

```
CONNECTED(leader u, slave v)
1   if v is a leader
2     then //v was an isolated leader
3          if |S(u)| < k
4            then v retires
5            else u retires // figure 2
6                 y ← an unshared slave of u
7                 MOVE({y}, u, v)
8     else w ← the other master of v
9          w retires
10         switch
11           case |S(u) ∪ S(w)| + 1 < k: // figure 3
12                MERGE(u, v, w); return
13           case |S(u)| = 1: // figure 4
14                MOVE({u}, NIL, w)
15                v disconnects from w; return
16           case |S(u) ∪ S(w)| + 1 = k: // figure 5
17                u retires
18                y ← an unshared slave of u
19                MERGE(u, v, w)
20                MOVE({y}, u, v)
21                v becomes a leader; return
22           case default: // figure 6
23                MIGRATE(u, v, w); return
```

Communications between $u$ and $w$ in CONNECTED, MERGE, MIGRATE, and MOVE are via their common slave $v$.

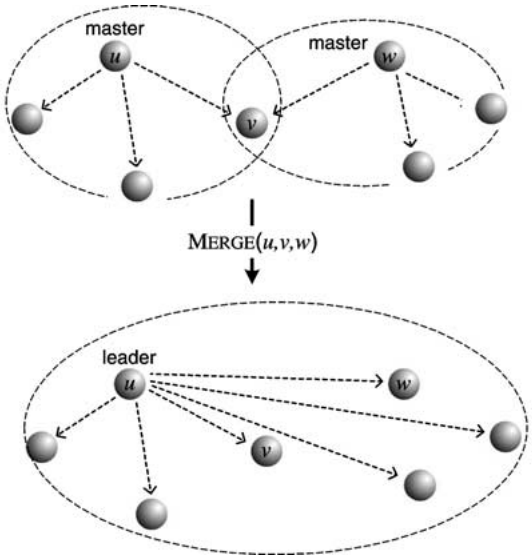


Figure 3. Lines 11–12 ($|\mathcal{S}(u) \cup \mathcal{S}(w)| + 1 < k$) in procedure CONNECTED for $k = 7$.

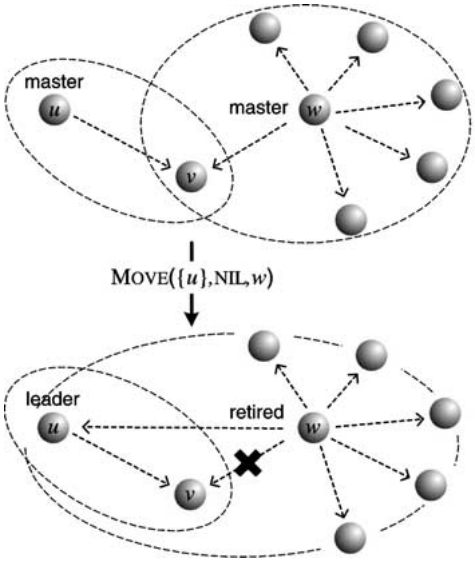


Figure 4. Lines 13–15 ($|\mathcal{S}(u)| = 1$) in procedure CONNECTED for $k = 7$.

Procedure MERGE($u$, $v$, $w$) makes $w$ and all its slaves become $u$'s slaves.

MERGE(master $u$, slave $v$, master $w$
1 $v$ disconnects from $w$
2 MOVE($\mathcal{S}(w) \setminus v$, $w$, $u$)
3 MOVE($\{w\}$, NIL, $u$)

Procedure MIGRATE($u$, $v$, $w$) moves slaves from $\mathcal{S}(u)$ to $\mathcal{S}(w)$ until $\mathcal{S}(w)$ is full or when only two slaves are left in $\mathcal{S}(u)$.

MIGRATE(master $u$, slave $v$, master $w$)
1 $i \leftarrow \min(k - |\mathcal{S}(w)|, |\mathcal{S}(u)| - 2)$
2 *//$i$ is the number of slaves to migrate*
3 **if** $i > 0$
4 **then** $y \leftarrow$ an unshared slave of $u$

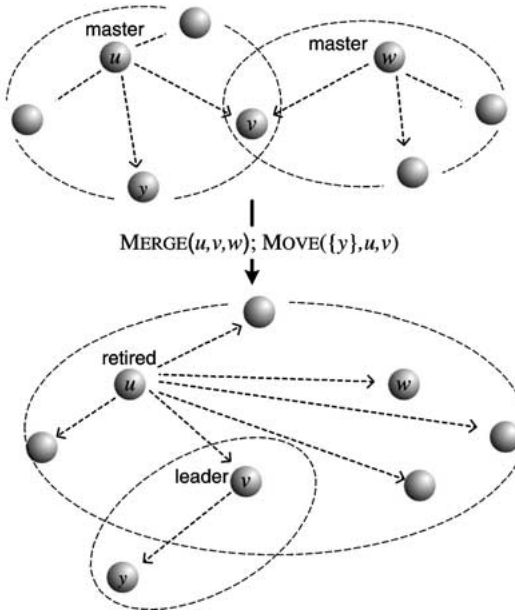


Figure 5. Lines 16–21 ($|\mathcal{S}(u) \cup \mathcal{S}(w)| + 1 = k$) in procedure CONNECTED for $k = 7$.

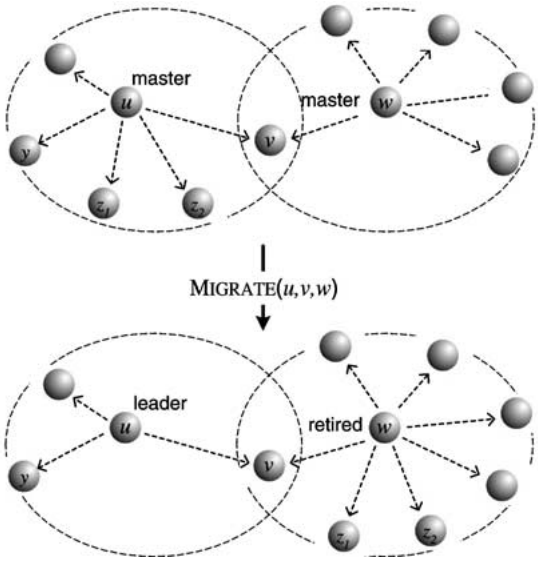


Figure 6. Lines 22–23 (default) in procedure CONNECTED for $k = 7$.

5 $Z \leftarrow \{i$ slaves in $\mathcal{S}(u) \setminus \{y, v\}\}$
6 MOVE($Z$, $u$, $w$)

Procedure MOVE is a subroutine called by CONNECTED, MERGE, and MIGRATE. All devices in set $Z$ disconnect from $u$ and become slaves of $w$.

MOVE(set $Z$, master $u$, master $w$)
1 devices in $Z$ disconnect from $u$
2 devices in $Z$ wait for $w$ in PAGE SCAN
3 $w$ connects to devices in $Z$ by PAGE

Lemma 2 proves the invariants of the algorithm.

**Lemma 2.** During the execution of the algorithm, the following invariants hold:

- Each leader has either no slave, or has at least one unshared slave.
- Each leader has fewer than $k$ slaves.

*Proof.* We will prove the invariants by induction.

In the beginning, all devices are leaders without slaves. So our invariants are satisfied.

Assuming a state satisfying the claim, we only have to make sure that CONNECTED and its calls to MERGE, MIGRATE, and MOVE preserve the invariants, because no piconet is formed or modified in MAIN, SEEK, and SCAN.

Let $\mathcal{S}'(u)$ and $\mathcal{S}'(v)$ be the slaves of $u$ and $v$ after CONNECTED$(u, v)$ is returned.

First, we consider the case that $v$ is a leader (lines 1–7). If $v$ is a leader, it means that $v$ does not have any slave. If $|\mathcal{S}(u)| < k$ (lines 3, 4), $v$ would become an unshared slave of $u$. If $u$ has exactly $k$ slaves (lines 5–7), then one unshared slave $y$ is moved from $\mathcal{S}(u)$ to $\mathcal{S}(v)$. Thus, $\mathcal{S}'(v)$ contains an unshared slave. In this case, $u$ is retired so that it does not need an unshared slave.

Second, we consider the case that $v$ is a slave of a leader $w$. Master $w$ will no longer be a leader, so it does not have to satisfy the invariants. There are four cases:

- $(|\mathcal{S}(u) \cup \mathcal{S}(w)| + 1 < k)$. All devices in $\mathcal{S}(w)$ become slaves of $u$. Device $v$ was an unshared slave in $\mathcal{S}(w)$. After the merge, $u$ is the only master of $v$, so $v$ becomes an unshared slave of $u$. Also, we note that $|\mathcal{S}'(u)| = |\mathcal{S}(u) \cup \mathcal{S}(w)| + 1$ is smaller than $k$ by assumption.
- $(|\mathcal{S}(u)| = 1)$. Leader $u$ will have $v$ as its only slave. Slave $v$ is unshared because it was $w$'s unshared slave.
- $(|\mathcal{S}(u) \cup \mathcal{S}(w)| + 1 = k)$. In this case, $u$ retires so it does not need to satisfy the invariants. When slave $v$ becomes a leader, it obtains an unshared slave $y$ from $u$.
- (Default). Slaves in $\mathcal{S}(u)$ are migrated to $\mathcal{S}'(w)$ until $|\mathcal{S}'(w)|$ is $k$ or $\mathcal{S}'(u)$ contains only two slaves (one of them is $v$). Procedure MIGRATE will always reserve an unshared slave $y$ for $\mathcal{S}'(u)$. By assumption, $w$ had at most $k - 1$ slaves before CONNECTED is called. Therefore, we can move at least one slave from $\mathcal{S}(u)$ to $\mathcal{S}'(w)$. Therefore, $|\mathcal{S}'(u)|$ is at most $k - 1$, because at least one slave is removed after $u$ has obtained slave $v$. □

This algorithm does not minimize the absolute number of messages passed between the devices. This is not crucial in practice, as section 6 will show that most of the packets are sent during the inquiry processes. The current design is a compromise between simplicity of the algorithm and the constant factors of the message complexity of the algorithm.

The last leader will keep calling MAIN even after the scatternet is formed. It is because the leader cannot be certain that all devices are already connected unless it knows the total number of devices. In practice, we can let the leader stop after it has failed to find any device for certain number of rounds. The probability that $n$ leaders fail to make any connections for $l$ rounds is $(p^n + (1 - p)^n)^l$, which is less than $(5/9)^l$ for $n \geqslant 2$ and $1/3 < p < 2/3$.

## 5. Performance and properties

### *5.1. Theoretical results*

#### *5.1.1. Scatternet properties*

We show that the scatternet formed possesses two useful properties: small degrees for shared devices and small number of piconets.

**Lemma 3.** At most one piconet in the scatternet formed by the algorithm contains fewer than $k - 1$ slaves.

*Proof.* When a scatternet is formed, only one component is left. Therefore, except for one piconet, the masters of the other piconets have retired. We will show that a retired master has at least $k - 1$ slaves. Therefore, when the scatternet is formed, all but one of the masters have at least $k - 1$ slaves.

We only need to make sure that if a leader becomes a retired master in a round, it should have at least $k - 1$ slaves, because a retired master will not lose any slave in subsequent rounds. There are four places in CONNECTED that a leader is retired:

- *line 4*: $v$ becomes a slave.
- *line 5*: The test $|\mathcal{S}(u)| < k$ is false. Thus $u$ had at least $k$ slaves before line 7: MOVE$(\{y\}, u, v)$, which reduces the number of $u$'s slaves by 1. Thus $u$ would have $k - 1$ slaves when retired.
- *line 9*: We must show that for all the four cases in lines 11–23, $w$ will have at least $k - 1$ slaves when CONNECTED returns if $w$ remains a master. In the first and third cases, $w$ loses all of its slaves in the procedure MERGE and becomes a slave itself. In the second case (lines 13–15), we have $|\mathcal{S}(u)| = 1$ but $|\mathcal{S}(u) \cup \mathcal{S}(w)| + 1 \geqslant k$ because the condition of the first case is not satisfied. Since $u$ and $w$ share one slave, we have $|\mathcal{S}(u)| + |\mathcal{S}(w)| - 1 + 1 \geqslant k$. Thus, $\mathcal{S}(w) \geqslant k - 1$ before MOVE is called. Master $w$ loses slave $v$, but gains a new slave $u$, so $w$ still has at least $k - 1$ slaves when procedure CONNECTED returns. In the last case (lines 22–23), we have $|\mathcal{S}(u) \cup \mathcal{S}(w)| + 1 \geqslant k + 1$, and MIGRATE will move all devices in the piconet of $u$ to the piconet of $w$ until $w$ has $k$ slaves or $u$ has only 2 slaves left. In the latter case, only one slave in $\mathcal{S}(u) \cup \mathcal{S}(w)$ will not become a slave of $w$. Thus $w$ would have at least $k - 1$ slaves after the MIGRATE operation.
- *line 17*: All slaves of $w$ and $w$ itself become slaves of $u$ in line 19. We note that $u$ and $w$ had $k - 1$ slaves in total (line 16), thus $u$ should have $k$ slaves after MERGE (line 19). MOVE (line 20) would remove one slave from $u$. Therefore, $u$ still has $k - 1$ slaves when CONNECTED returns. □

**Lemma 4.** The algorithm forms a scatternet with $m-1$ devices of degree 2 and $n-m+1$ devices of degree 1, where $n$ is the number of devices and $m$ is the number of piconets.

*Proof.* See appendix B. □

**Theorem 5.** The scatternet formed by the algorithm contains at most $\lfloor (n-2)/(k-1) \rfloor + 1$ piconets.

*Proof.* Consider a scatternet produced by the algorithm. Let $n$ be the number of devices and $m$ be the number of piconets. By lemma 3, at most one piconet has size less than $k$. (A piconet has size less than $k$ if and only if it has fewer than $k-1$ slaves.) Such piconet has size at least 2. By lemma 4, $m-1$ devices have degree 2 and the rest of the devices have degree 1. Therefore, we can conclude that the scatternet contains at least

$$k(m-1)+2-(m-1) = (k-1)(m-1)+2$$

devices. Thus, $n \geqslant (k-1)(m-1)+2$. Since $m$ is an integer, $m \leqslant \lfloor (n-2)/(k-1) \rfloor + 1$. □

Comparing theorem 5's upper bound $\lfloor (n-2)/(k-1) \rfloor + 1$ with remark 1's lower bound $\lceil (n-1)/k \rceil$, we note that our bound is very close to be optimal. For example, when $n = 100$ and $k = 7$, our algorithm forms a scatternet containing at most 17 piconets, while the lower bound requires 15 piconets.

*5.1.2. Asymptotic complexities*

We first derive the algorithm's time complexity and then its message complexity.

**Lemma 6.** During a round with at least 2 leaders, the number of leaders is reduced by a constant fraction with a constant probability.

*Proof.* Let $m \geqslant 2$ be the number of leaders. Let $p$ be the probability that a leader chooses to run SEEK. The algorithm specifies that $1/3 < p < 2/3$. We will assume $p \leqslant 1/2$, because if $p > 1/2$, we can switch the roles of SEEK and SCAN and the proof follows similarly.

During each round, we have a matching between the SEEK devices and the SCAN devices. Let random variable $X$ be the number of SEEK devices in a given round. Since $X$ is distributed binomially with parameter $p$, we have $E[X] = pm$ and $\mathrm{Var}[X] = mp(1-p)$.

Let $\alpha$ be a real number between 0 and 1. If $(1-\alpha)pm \leqslant X \leqslant (1+\alpha)pm$, then at least $(1-\alpha)pm$ connections are made between the SEEK devices and SCAN devices because: (1) there are at least $(1-\alpha)pm$ SEEK devices; and (2) there are at least

$$m-(1+\alpha)pm = (1-p-\alpha p)m \geqslant (1-\alpha)pm$$

SCAN devices since $(1-p) \geqslant p$ if $p \leqslant 1/2$.

Thus, the probability of having at least $(1-\alpha)pm$ connections (size of the matching between SEEK devices and SCAN devices) is

$$\begin{aligned}&\Pr\{\text{at least } (1-\alpha)pm \text{ connections}\}\\ &\quad = \Pr\{(1-\alpha)pm \leqslant X \leqslant (1+\alpha)pm\}\\ &\quad = \Pr\{|X-pm| \leqslant \alpha pm\}\\ &\quad = 1-\Pr\{|X-pm| > \alpha pm\}.\end{aligned}$$

The Chebyshev's inequality states that

$$\Pr\{|X-E[X]| > t\} < t^{-2}\mathrm{Var}[X].$$

By setting $t = \alpha pm$, $E[X] = pm$, and $\mathrm{Var}[X] = mp(1-p)$, we have

$$\Pr\{|X-pm| > \alpha pm\} < \frac{mp(1-p)}{(\alpha pm)^2} = \frac{1-p}{m\alpha^2 p}.$$

Since $m \geqslant 2$ and $p > 1/3$, we have $(1-p)/pm < 1$. Thus we can pick $\alpha$ so that $\alpha^2 > (1-p)/2p \geqslant (1-p)/mp$. Then $c = (1-p)/(2\alpha^2 p)$ is a constant smaller than 1. Therefore,

$$\Pr\{\text{at least } (1-\alpha)pm \text{ connections}\} > 1-c.$$

Each connection reduces the number of leaders by 1. Therefore, with probability at least $1-c$, the number of leaders is reduced by a fraction $(1-\alpha)p$. □

**Theorem 7.** The algorithm forms a scatternet in $O(\log n)$ rounds with probability at least $1-1/n^{\Theta(1)}$,

*Proof.* We note that a scatternet is formed when there is only one leader left. By lemma 6, when there are at least two leaders, the number of leaders is reduced by a fraction with some probability $q$. The probability that the algorithm takes more than $O(\log n)$ rounds to reduce the number of leaders to 1 is at most $q^{\Theta(\log n)} = 1/n^{\Theta(1)}$. □

**Theorem 8.** The expected message complexity of the algorithm is $O(kn)$.

*Proof.* See appendix C. □

**Corollary 9.** If $k$ is a constant, then the message complexity of the algorithm is $O(n)$.

We note that $O(n)$ is the optimal asymptotic message complexity because each device needs to send at least one message.

*5.2. Simulation results*

In this subsection, we investigate the properties and performance of our scatternet formation protocol.

We simulate our scatternet formation algorithm with `simjava` [8], a discrete event simulation package for Java. The probability $p$ that each leader chooses to execute SEEK in each round is 1/2 in our simulations. Following the Bluetooth specification, we set $k = 7$. We start with 2, 4, and 8

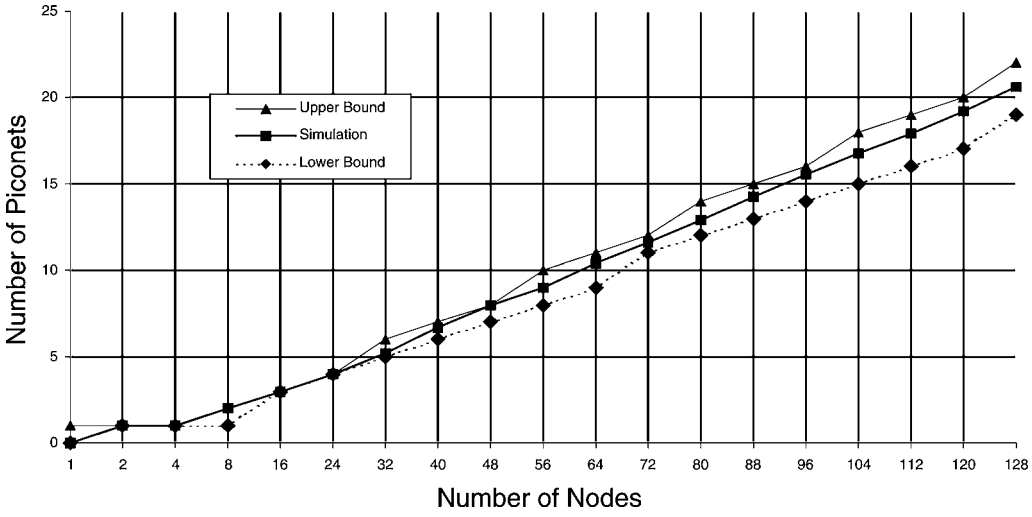


Figure 7. Number of piconets in the scatternet formed, compared to upper bound $\lfloor (n-2)/(k-1) \rfloor + 1$ and lower bound $\lceil (n-1)/k \rceil$, where $k = 7$.

nodes, and then increase by increments of 8 nodes, up to 128 nodes. This allows us to present the results against the number of nodes in linear scale and in logarithmic scale. Each data point in the graphs of this section represents an average of 50 trials.

*5.2.1. Scatternet properties*

First, we found that the maximum degree of the scatternet formed is 1 when there are fewer than 8 nodes and is 2 when there are at least 8 nodes. This means that the maximum degree is optimal except when there are 8 nodes, in which case a maximum degree of 1 is possible.

As we discussed in section 3, it is important to minimize the number of piconets because piconets interfere with each other. Figure 7 shows that the number of piconets formed lies between the protocol's theoretical upper bound $\lfloor (n-2)/(k-1) \rfloor + 1$ and the universal lower bound $\lceil (n-1)/k \rceil$. The largest difference between our simulation result and the lower bound is 2.2 piconets.

The network diameter, which is the maximum shortest path length between any pair of devices, captures the maximum routing delay between any pair of nodes in the scatternet. Although we do not have a theoretical analysis of the network diameter, figure 8 shows that the network diameter grows about logarithmically with the number of devices ($x$ axis is in logarithmic scale).

*5.2.2. Performance*

First, it is crucial that the scatternet is formed as fast as possible, because this translates to the latency experienced by the users. In figure 9, we can see that the number of rounds required to form the scatternet is around $1.2 \log_2 n + 2$. This validates the $O(\log n)$ time complexity theoretical result. In section 6, we will investigate the amount of time required in each round.

Second, as most mobile Bluetooth devices are expected to run on batteries, it is important to minimize the number of messages sent in order to conserve battery power. We can put the messages into three categories:

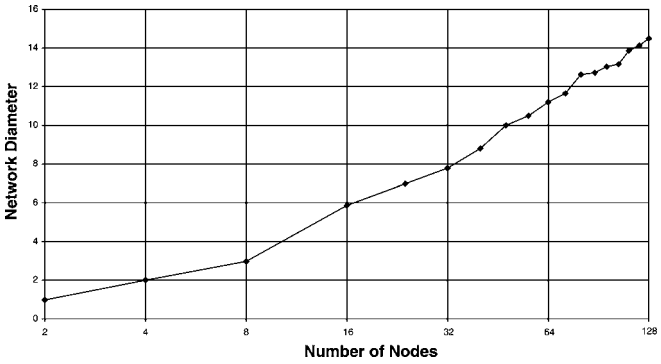


Figure 8. Network diameter of the scatternet formed.

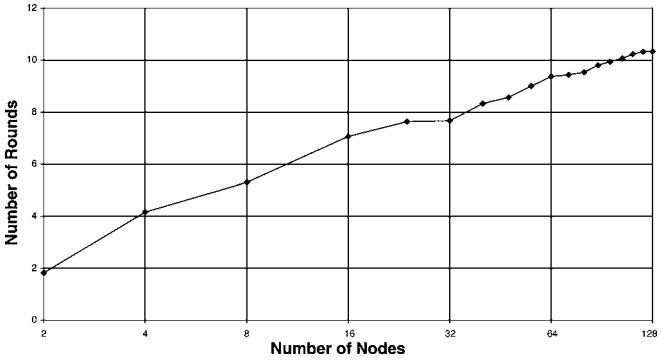


Figure 9. Number of rounds to form a scatternet.

**Inquiries** – Bluetooth INQUIRY and INQUIRY RESPONSE packets.

**Pages** – Bluetooth PAGE and PAGE RESPONSE packets.

**Algorithmic Messages** – other messages used by our scatternet formation algorithm.

Figure 10 presents the total number of the Algorithmic Messages, Inquiries, and Pages. We can verify that the numbers of all three types of messages increase linearly with the number of devices. This agrees with the $O(n)$ message complexity theoretical result.

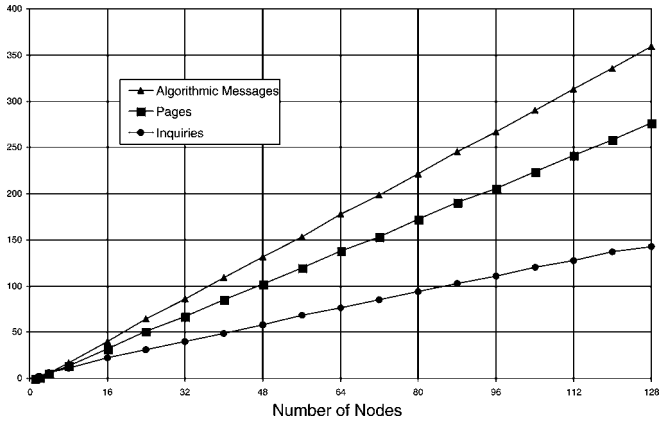


Figure 10. Total number of Algorithmic Messages, Pages, and Inquiries.

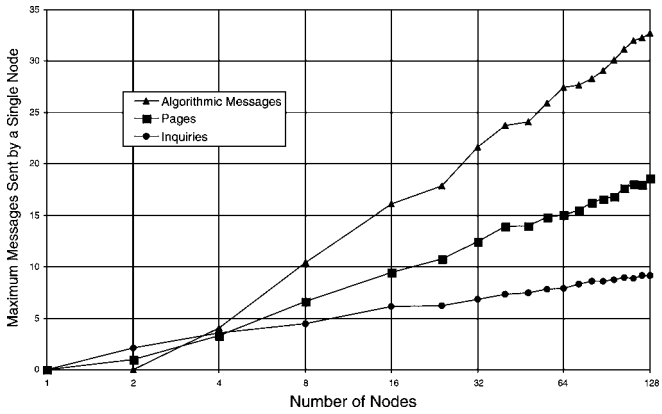


Figure 11. Maximum number of Algorithmic Messages, Pages, and Inquiries sent by any single node.

In figure 11, we can see that the maximum number of messages sent by any device increases logarithmically with the number of nodes. This implies that the power requirement of the "unluckiest" device is $O(\log n)$. A likely candidate of such unlucky device is the last remaining leader in the protocol. Since the last leader is not retired, the number of messages sent by this leader is $\Theta(\log n)$.

In the next section, we will find out how long it takes to finish one round of inquiry and page. We will also see how many packets are sent during the inquiry and page processes.

## 6. Device discovery

In this section, we investigate the performance of the device discovery protocol used during each round of the scatternet formation algorithm.

During scatternet formation, there are many devices trying to get connected at the same time, so the inquiry and page processes will interfere with each other. We call this the problem of device discovery when a set of in-range devices try to connect with each other. In the following, we discuss our approach and present the simulation results.

### 6.1. Protocol

We describe a simple randomized protocol for the problem of device discovery. This protocol is repeated during each round of the scatternet formation algorithm introduced in section 4. We are given $n$ devices that are not aware of each other. Our goal is to establish as many connections as possible. We are not concerned with exactly which of the devices are connected.

First, each device independently decides to be a SEEK node (with probability $p$) or a SCAN node (with probability $1 - p$).

The protocol contains two phases – the inquiry phase and the page phase. In the inquiry phase, all the SCAN devices stay in the INQUIRY SCAN state. Each SEEK device will try to contact a SCAN device. However, a SEEK device may not always succeed in finding a slave because the number of SCAN devices can be smaller than the number of SEEK devices. Therefore, if a SEEK device is not able to contact a slave after certain amount of time, it will simply give up. Similarly, a SCAN device might also fail to be connected. In the page phase, the already paired devices are connected with PAGE and PAGE SCAN.

This protocol makes sure that each SEEK device is connected to at most one SCAN device and each SCAN device is connected to at most one SEEK device. In other words, we obtain a one-to-one matching between the SCAN devices and SEEK devices. The number of connections established is the smaller of the number of SEEK devices and the number of SCAN devices.

### 6.2. Simulation results

We also used `simjava` [8] to simulate this protocol.[2] Each Bluetooth device is simulated by a thread. In each time slot, all devices first send messages, which include the frequency channel numbers, to a special object `Air`. Object `Air` detects the collisions in each of the 79 frequency channels, and only delivers the uncollided messages to those devices listening on the respective frequency channels. Inquiry and page frequency hopping sequences are implemented according to the Bluetooth specification. Since the overall time scale of the simulation is small, we did not simulate the clock drift. Each data point in the figures of this section is an average of 10 trials. In each trial, the devices are assigned addresses and clocks randomly.

Figure 12 shows the running time of the inquiry phase with three different master-to-slave ratios. For example, when there are 16 devices in total, a 50%–50% split leads to 8 masters and 8 slaves, and a 25%–75% split leads to 4 masters and 12 slaves. In the simulations of our algorithm in section 5.2, we set $p$ to $1/2$. Thus, we expect to see 50% masters and 50% slaves in each round. The actual outcomes at each round are distributed according to the binomial distribution. For more

[2] IBM's BlueHoc simulator [3] is not used because we began implementing our simulator before BlueHoc was released in public.

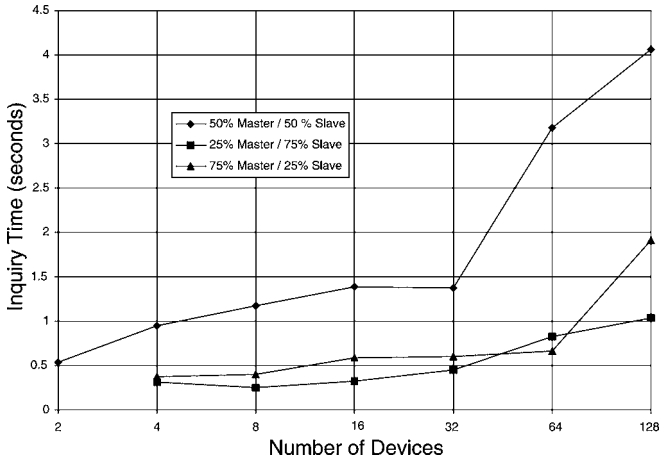


Figure 12. Running time of the inquiry phase with three master-to-slave ratios.

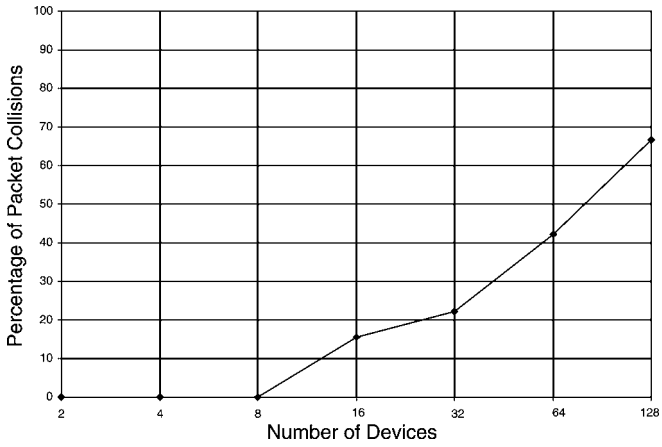


Figure 13. Percentage of packet collisions (over all packets sent) when there are 50% masters and 50% slaves.

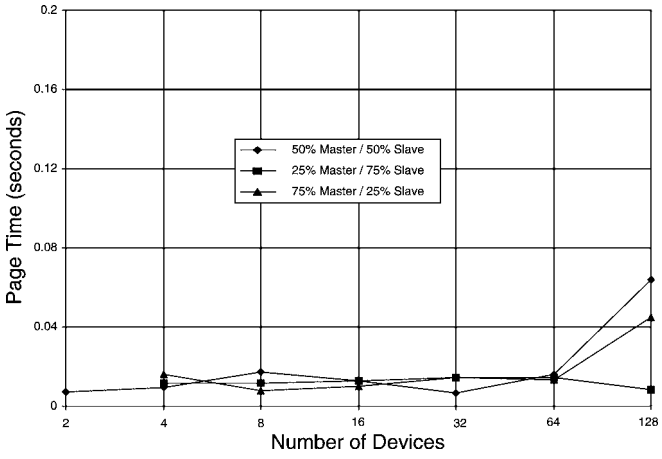


Figure 14. Running time of the page phase with three master-to-slave ratios.

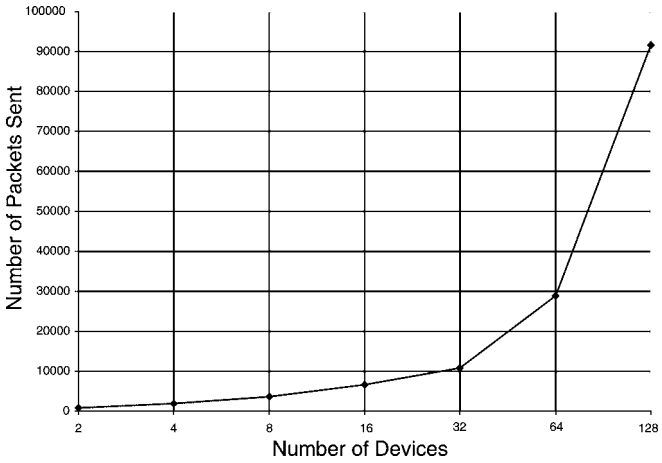


Figure 15. Total number of packets sent when there are 50% masters and 50% slaves.

than 8 devices, the 25%–75% split and 75%–25% split encompass at least 2 standard deviations around the expectation. We observe that the inquiry time of the 50%–50% split case increases sharply when there are around 64 devices. Since all SEEK devices follow the same inquiry hopping sequence (the phase depends on the device's clock), packet collision is a major problem when there are many devices. From the collision graph (figure 13) on the 50%–50% split case, we can deduce that collisions start to hurt the performance severely when there are around 64 devices.

In figure 14, we observe that, for up to 64 devices, the time consumed by the page phase is below 0.02 seconds, which is insignificant compared to the time required for the inquiry phase. This is because the SEEK devices already know the addresses and clocks of their target SCAN devices, thus they are able to contact the SCAN devices quickly. In addition, since they have different hopping sequences, the amount of collisions is lower in this case.

Figure 15 shows the total number of packets sent. Again the number of packets sent rises sharply around the 64-device case, due to collisions. However, we can see that the total number of packets is around $(10000/32)n$ for $n = 32$, 64, and 128. This means that the total number of packets sent increases roughly linearly with the number of devices.

## 7. Overall performance

We now estimate the overall performance of our protocol, using the results of sections 5.2 and 6.2.

In section 5.2, we learned that the number of Inquiries, Pages, and Algorithmic Messages all increase linearly with the number of devices. And in section 6.2, we found that the number of packets sent during a single round of the protocol increases linearly with the number of devices. Therefore, we can conclude that overall message complexity of the protocol is linear. This means that the average power consumed by a device remains constant when the number of devices increases. In section 5.2, we also showed that the number of Inquiries, Pages, and Algorihtmic Messages of any single device increases logarithmically. Thus, our protocol does not cause a very high load on any single device.

To estimate the total time taken by the protocol, we can multiply the number of rounds (figure 9) by the time required for each round. The time taken in each round is the sum of

the time required for the inquiry phase (figure 12), the page phase (figure 14), and the procedure CONNECTED.

We can estimate the time required for CONNECTED. During each round, each device will perform either PAGE or PAGE SCAN at most once as a result of procedure CONNECTED. Procedure CONNECTED does not cause any INQUIRY because the clocks and addresses of the devices are already known. Therefore, the time required for the PAGEs caused by CONNECTED should be more than the time required in the 50%–50% case of the page phase (figure 14). In addition, $O(k)$ messages need to be exchanged among leader $u$, slave $v$, and leader $w$ in procedure CONNECTED. The amount of information to be passed is small, and thus the time required to pass these messages is insignificant compared to the time required for the INQUIRYs and PAGEs.

For example, according to our simulation results, for up to 32 devices, we expect that $1.39 + 0.02 \cdot 2 = 1.43$ seconds are required for each round. The protocol takes on the average 7.7 rounds to form the scatternet. Thus, the total time requirement is about $7.7 \cdot 1.43 < 11.1$ seconds. Similarly, the estimated total time required for 16 devices and 64 devices are at most 10.2 seconds and 30.3 seconds, respectively. Section 8 discusses how the overall performance can be improved with an asynchronous version of our protocol.

## 8. Variations and extensions

In the following, we discuss several limitations of our protocol and suggest techniques for overcoming them.

### *8.1. Inquiry collisions*

When there are many devices, packet collisions among INQUIRY devices can adversely affect the performance. In particular, if two INQUIRY devices happen to have their clocks in phase so that their inquiry sequences are synchronized, then their inquiry packets will collide repeatedly. This effect was observed in our simulations in those cases with large numbers of devices. It is conceivable that this problem can be alleviated if the INQUIRY devices back off randomly during a heavy-collision situation. We note that this back-off by the INQUIRY device is not related to the random back-off by an INQUIRY SCAN device after receiving an inquiry packet, as specified in Bluetooth 1.1.

### *8.2. Asynchronous protocol*

The overall time requirement estimated in section 7 is longer than the phase I of BTCP [19]. To improve the performance, we should consider an asynchronous version of our protocol. We believe that the overall time requirement can be reduced because of the following observations:

- The worst-case time required per round happens when there is a perfect split between the masters and slaves. However, if this happens frequently, the total number of rounds required is small. For example, if there is a perfect split every round, the protocol will only need $\log_2 32 = 5$ rounds to form a scatternet of 32 nodes.
- The number of active leaders decreases rapidly. Thus, the device discovery processes in the later rounds can be completed faster.

We can consider an asynchronous version of our protocol with the following change: once CONNECTED returns, the remaining leader can proceed to MAIN immediately. The synchronized nature of the current protocol is useful for the theoretical analyses of the performance. In practice, it is not necessary for all devices to execute CONNECTED at the same time. The overall performance of the asynchronous version should be better than the synchronous version. Moreover, the analyses on the degrees of devices (lemma 4) and the number of piconets (theorem 5) remain valid in the asynchronous version.

We note that it is not necessary for the devices to start at the same time in practice. Even in the synchronous version, a device can join the scatternet in a later round (see section 8.4 for a discussion of dynamic environments).

### *8.3. Out of range devices*

In some scenarios, some of the Bluetooth devices might be out of range of one another. Given arbitrary device connectivity, it is not possible to maintain the performance and scatternet properties guarantees. Despite such limitations, we can augment the protocol to try to form a scatternet whenever possible. Procedures SEEK and SCAN will not need to be modified because two devices will be connected only if they are in-range. We note that, other than SEEK, the only place that master-slave connections are established (by PAGE and PAGE SCAN) is in procedure MOVE. Therefore, procedure MOVE might fail. Let us consider the places in CONNECTED where MOVE is called:

- *lines 5–7* (figure 2). If $y$ cannot be connected to $v$, then we can try to use other unshared slaves of $u$. If all unshared slaves of $u$ are not able to connect to $v$, then $v$ should become a retired master and have $u$ as its only slave.
- *lines 11, 12* (figure 3). The MERGE call might fail. In this case, we can let $w$ retire with its smaller piconet.
- *lines 13–15* (figure 4). If $u$ cannot be connected to $w$, then we can let $u$ be the slave of $v$. This will be similar to the original outcome except that $v$ will be the new leader, instead of $u$.
- *lines 16–21* (figure 5). If MERGE fails, we will just let $w$ retire.
- *lines 22, 23* (figure 6). The MIGRATE procedure should move as many devices to the retiring master $w$ as allowed by the underlying connectivity.

The above modifications, except the one on lines 5–7, only affect the total number of piconets of the scatternet formed, but not the maximum degree of any device in the scatternet formed.

Each execution of modified lines 5–7 might increase the degree of $u$ by one. Without a distribution assumption of device locations, we cannot bound the probability of such event. However, we can provide some reasons that such event is unlikely. Given that $v$ was still an isolated device before the connection, we can show that it is unlikely that $u$ has more than one shared slaves. If $u$ has at least two shared slaves, then the component led by $u$ has at least $k + (k-1) + (k-1)$ devices, because each retired piconet has at least $k-1$ slaves. This implies that at least $\lceil \log_2(3k-2) \rceil$ rounds have passed before $v$ is able to make a connection. If $p = 1/2$, $v$ has a probability of at least $1/2$ to make a connection in each round. Thus, if $k = 7$, then the probability that $v$ is not connected for $\lceil \log_2 19 \rceil = 5$ rounds and then connect to $u$ as a slave is at most $(1/32)(1/2) = 1/64$. When $u$ has no more than 1 shared slave, it is unlikely that the $k-1$ or $k-2$ unshared slaves are all out of range with $v$.

Depending on the underlying connectivity of the devices, the piconets are likely to have smaller sizes, implying a larger number of piconets in the scatternet formed. Unless the situation discussed in the previous paragraph happens, the maximum degree of any device in the scatternet will still be two.

### *8.4. Joins, leaves, and faults*

Our protocol can be easily extended to work with dynamic environments (with devices joining and leaving the scatternet) and device failures. Our current protocol already handles the events of devices joining – the new devices can simply start as leaders and thus discover or be discovered by other devices. Additional work is required to deal with the case of devices leaving or failing. We give an outline in the following:

- If a master fails (or leaves the network), then a new master can be elected from the slaves. If the failed master was shared, then the new master should become a leader and merge with the rest of the scatternet by the protocol.
- If a shared slave fails, its older master (the master who connected to this slave first) should become a leader again and then it will be connected to the rest of the scatternet by the protocol.
- Nothing needs to be done when an unshared slave fails, unless it is the only unshared slave of an active leader.
- In general, if we end up with a leader $u$ with no unshared slave, then this leader has to disconnect from its shared slaves. Other masters of those shared slaves should now become leaders again. This will allow the protocol to proceed as usual. Fortunately, this expensive reorganization should occur rarely.

## 9. Concluding remarks

In this paper, we introduced a new Bluetooth scatternet formation protocol. We presented both theoretical and simulation results to show that our protocol has $O(\log n)$ time complexity and $O(n)$ message complexity.

We have shown that the algorithm produces scatternet with desirable properties: small number of piconets for minimizing inter-piconet interference, and small degrees for the devices for avoiding network bottlenecks. In addition, according to the simulations, the diameter of the scatternet, which corresponds to the maximum routing distance between nodes, is about $O(\log n)$. At last, we also demonstrated that no single device is particularly exhausted by the protocol.

## Appendix A. Proof of remark 1

We need to show that a scatternet of $n$ devices has at least $\lceil (n-1)/k \rceil$ piconets. Let $p(n)$ be the minimum number of piconets for a scatternet of $n$ devices. We will show that $p(n) \geqslant \lceil (n-1)/k \rceil$ by induction on $p(n)$.

First, if $p(n) = 1$, then $n \leqslant k+1$ by our assumption that each piconet can have at most $k+1$ devices. Therefore

$$p(n) = 1 = \left\lceil \frac{k}{k} \right\rceil \geqslant \left\lceil \frac{(n-1)}{k} \right\rceil.$$

Next, assume that if $p(n) \leqslant m$, then $p(n) \geqslant \lceil (n-1)/k \rceil$. Then we consider the case that $p(n') = m+1$. Given a scatternet of $n'$ devices, we pick a master and remove its piconet so that the rest of the scatternet is still connected. We can at most remove $k$ devices because this piconet was connected with the rest of the scatternet. Therefore, after this removal, we are left with $m$ piconets, and at least $n' - k$ devices. By the inductive hypothesis, we have $p(n) \geqslant \lceil (n-1)/k \rceil$ if $p(n) \leqslant m$. Since $n \geqslant n' - k$, we have $p(n) \geqslant \lceil (n'-k-1)/k \rceil = \lceil (n'-1)/k \rceil - 1$. Thus, $p(n') \geqslant p(n) + 1 \geqslant \lceil (n'-1)/k \rceil$.

## Appendix B. Proof of lemma 4

First, we will show that any device has maximum degree 2. We will verify that the shared slaves and shared masters have degrees at most 2.

*Shared slaves.* We observe that only unshared slaves may participate in SCAN. Thus, a shared slave will not be shared again through SCAN. A shared save might become unshared in a MERGE or become a degree-2 shared master in lines 16–21 of procedure CONNECTED.

*Shared masters.* A shared master can only be a slave of a retired master. Therefore, a shared master will never be shared again with another master through SCAN. In addition, a shared master $v$ is always created from an unshared slave or an isolated master. This means that $v$ had no slave before becoming a shared master. Therefore, a device can only become a shared master once.

We can now consider the topological graph of the scatternet, in which each piconet is a node and each degree-2 device is an edge. We can show that this topological graph is a tree. Initially, each component is a tree (a single node).

During each CONNECTED call, at most one edge is created between the two merging components. And since each component only participates in one CONNECTED process during each round, the components remain trees throughout the protocol.

In a tree of $m$ nodes, there are exactly $m-1$ edges. Therefore, there are $m-1$ degree-2 devices, and $n-(m-1)$ degree-1 devices.

## Appendix C. Proof of theorem 8

We first consider the message complexity of each invocation of the procedures. We note that each of the procedures MAIN, SCAN, SEEK, CONNECTED, MERGE, MIGRATE sends $O(1)$ messages. Procedure MOVE moves at most $k$ devices. Thus it sends $O(k)$ messages.

To analyze the message complexity of MAIN, SEEK, and SCAN, it is sufficient to find the expected number of times that MAIN is called, because each call to MAIN leads to a call to SEEK or a call to SCAN.

First, we can assume that when a leader $w$ chooses SCAN such that if it or its slave is contacted by another leader $u$, then $w$ will retire. This is true except that if $u$ has $k$ slaves, then $u$ will retire instead. See lines 5–7 in procedure CONNECTED. However, for simplicity of the analysis, we can assume that $w$ retires instead of $u$. In other words, we can assume that $w$ and $u$ swap their identities whenever we are in this case. This will not affect our result because we only care about the total number of messages sent by these leaders. The high-level algorithm does not rely on an identifier of the device. (Device address is used by low-level Bluetooth INQUIRY and PAGE. But these processes are independent between different rounds in the algorithm.)

During any round, each leader chooses SCAN with probability $1-p$. Assume that a leader $w$ has chosen SCAN. Leader $w$ or $w$'s unshared slave will definitely be contacted by another leader if the total number of SCAN devices is not more than the number of SEEK devices.

Let $X_i$ be the random variable of the number of SCAN devices over $i$ components. Thus, $[X_i] = (1-p)i$. Let $m \geqslant 2$ be the number of components. We now assume that $p \geqslant 1/2$ because if otherwise, we can switch the roles of SCAN and SEEK. Assume $w$ has chosen SCAN:

$$\begin{aligned}&\Pr\{w \text{ or } w\text{'s slave is contacted by a leader}\}\\ &\quad = \Pr\left\{X_{m-1} \leqslant \frac{m}{2} - 1\right\}.\end{aligned}$$

By Markov inequality, we have

$$\begin{aligned}&\Pr\left\{X_{m-1} > \frac{m}{2} - 1\right\}\\ &\quad = \Pr\left\{X_{m-1} \geqslant \frac{m}{2} - \frac{1}{2}\right\} \leqslant \frac{E[X_{m-1}]}{m/2 - 1/2} = 2(1-p).\end{aligned}$$

Thus,

$$\Pr\left\{X_{m-1} \leqslant \frac{m}{2} - 1\right\} \geqslant 1 - 2(1-p) = 2p - 1.$$

Thus, each leader retires with probability at least $(1-p)(2p-1)$, which is positive except when $p = 1/2$.

We now consider the case where $p = 1/2$. In the proof of lemma 6, we have

$$\Pr\{\text{at least } (1-\alpha)pm \text{ connections}\} > 1 - \frac{1-p}{\alpha^2 pm}.$$

Let $p = 1/2$, $\alpha = 1/2$ and $m \geqslant 5$, then

$$\Pr\{\text{at least } m/4 \text{ connections}\} > 1/5.$$

Therefore, with probability at least $1/5$, at least $m/4$ connections are made. And when that happens, each device has a probability of at least $1/4$ to be the slave of a connection being made. This proves our argument for $m \geqslant 5$. The cases where $m = 2, 3, 4$ can be easily verified.

We have shown that any leader has a constant probability of retiring during each round. This means that each leader is active for $O(1)$ rounds. Thus MAIN is called $O(n)$ times in total, and the overall message complexity for procedures MAIN, SEEK, SCAN is $O(n)$.

Procedure CONNECTED is called exactly $n-1$ times. Thus, the message complexity of CONNECTED is $O(n)$. Each call to CONNECTED could result in at most 1 call to MERGE, at most 1 call to MIGRATE, and at most 3 calls to MOVE. Thus the overall message complexity of MERGE and MIGRATE is $O(n)$, and the overall message complexity of MOVE is $O(kn)$.

## References


[1] A. Aggarwal, M. Kapoor, L. Ramachandran and A. Sarkar, Clustering algorithms for wireless ad hoc networks, in: *Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, Boston, MA (August 2000) pp. 54–63.

[2] Auto-ID Center, `http://autoidcenter.org/`

[3] BlueHoc: Bluetooth performance evaluation tool, `http://oss.software.ibm.com/developerworks/opensource/bluehoc/`

[4] J. Bray and C.F. Sturman, *Bluetooth: Connect Without Cables* (Prentice Hall, New York, 2001).

[5] J. Broch, D.A. Maltz, D.B. Johnson, Y.-C. Hu and J. Jetcheva, A performance comparison of multi-hop wireless ad hoc network routing protocols, Mobile Computing and Networking (1998) 85–97.

[6] S.R. Das, R. Castañeda and J. Yan, Simulation-based performance evaluation of routing protocols for mobile ad hoc networks, Mobile Networks and Applications 5 (2000) 179–189.

[7] J. Haartsen, Bluetooth – the universal radio interface for ad hoc, wireless connectivity, Ericsson Review (3) (1998) 110–117.

[8] F. Howell and R. McNab, simjava: A discrete event simulation library for Java, in: *Proceedings of International Conference on Web-Based Modeling and Simulation*, International Society for Computer Simulation (January 1998).

[9] P. Johansson, N. Johansson, U. Korner, J. Elg and G. Svennarp, Short range radio based ad-hoc networking: performance and properties, in: *Proceedings of the IEEE International Conference on Communications 1999*, Vol. 3 (1999) pp. 1414–1420.

[10] C. Law, A.K. Mehta and K.-Y. Siu, Performance of a new Bluetooth scatternet formation protocol, in: *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing 2001*, Long Beach, CA (October 2001).
[11] C. Law and K.-Y. Siu, An $O(\log n)$ randomized resource discovery algorithm, in: *Brief Announcements of the 14th International Symposium on Distributed Computing*, Technical Report, Technical University of Madrid, No. FIM/110.1/DLSIIS/2000 (October 2000) pp. 5–8.
[12] C. Law and K.-Y. Siu, A Bluetooth scatternet formation algorithm, in: *Proceedings of the IEEE Symposium on Ad Hoc Wireless Networks 2001*, San Antonio, TX (November 2001).
[13] A.K. Mehta, Ad-hoc network formation using Bluetooth scatternets, Master's Thesis, Massachusetts Institute of Technology (June 2001).
[14] G. Miklos, A. Racz, Z. Turanyi, A. Valko and P. Johansson, Performance aspects of Bluetooth scatternet formation, in: *Proceedings of The First Annual Workshop on Mobile Ad Hoc Networking and Computing* (2000).
[15] B.A. Miller and C. Bisdikian, *Bluetooth Revealed: The Insider's Guide to an Open Specification for Global Wireless Communications* (Prentice Hall, New York, 2000).
[16] C.E. Perkins, *Ad Hoc Networking* (Addison-Wesley, Reading, MA, 2000).
[17] B. Raman, P. Bhagwat and S. Seshan, Arguments for cross-layer optimizations in Bluetooth scatternets, in: *Proceedings of Symposium on Applications and the Internet* (2001) pp. 176–184.
[18] T. Salonidis, P. Bhagwat and L. Tassiulas, Proximity awareness and fast connection establishment in Bluetooth, in: *First Annual Workshop on Mobile and Ad Hoc Networking and Computing* (2000) pp. 141–142.
[19] T. Salonidis, P. Bhagwat, L. Tassiulas and R. LaMaire, Distributed topology construction of Bluetooth personal area networks, in: *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies* (2001).
[20] Specification of the Bluetooth System, Version 1.1 (February 2001).
[21] G. Tan, Self-organizing Bluetooth scatternets, Master's Thesis, Massachusetts Institute of Technology (January 2002).
[22] The Bluetooth Special Interest Group, `http://www.bluetooth.com`
[23] C.K. Toh, *Ad Hoc Mobile Wireless Networks: Protocols and Systems* (Prentice Hall, New York, 2001).
[24] Z. Wang, R.J. Thomas and Z. Haas, Bluenet – a new scatternet formation scheme, in: *Proceedings of the 35th Annual Hawaii International Conference on System Sciences* (January 2002).
[25] G.V. Záruba, S. Basagni and I. Chlamtac, Bluetrees–scatternet formation to enable Bluetooth-based ad hoc networks, in: *Proceedings of IEEE International Conference on Communications* (2001) pp. 273–277.
[26] S. Zurbes, Considerations on link and system throughput of Bluetooth networks, in: *Proceedings of the 11th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, Vol. 2 (2000) pp. 1315–1319.

**Ching Law** received his S.B. degree in computer science and engineering and his S.B. degree in mathematics in 1998, and his M.Eng. degree in computer science and electrical engineering in 1999, all from the Massachusetts Institute of Technology. He is currently a Ph.D. candidate at the Department of Computer Science and Electrical Engineering, Massachusetts Institute of Technology. His research interests include ad hoc networks, peer-to-peer networks, and distributed algorithms.
E-mail: ching@list.mit.edu

**Amar Mehta** graduated in June of 2001 from MIT with a S.B. and M.Eng. in electrical engineering and computer science. He graduated with distinction, as a member of both Eta Kappa Nu and Tau Beta Pi. As a graduate student at MIT, he was a research assistant at the MIT Auto-ID Center, simulating the formation of scatternets with Bluetooth, a wireless protocol for short range communication.
E-mail: amar@list.mit.edu

**Kai-Yeung (Sunny)** Siu received the B.S. degree (summa cum laude) in mathematics and computer science from New York University, New York, NY, and the B.Eng. degree (summa cum laude) in electrical engineering from The Cooper Union, New York, NY, both in 1987. He received the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA in 1988 and 1991, respectively. From 1989 to 1990, he was a research student associate at the IBM Almaden Research Center, San Jose, CA. From 1991 to 1995, he was Assistant Professor of Electrical and Computer Engineering at the University of California, Irvine. He joined the Massachusetts Institute of Technology in 1996, and is currently Associate Professor and recipient of the d'Arbeloff Career Development Chair at MIT. He is with the d'Arbeloff Laboratory for Information Systems and Technology of Mechanical Engineering and also affiliated with the Laboratory for Information and Decision Systems of Electrical Engineering and Computer Science. He has published over 100 research papers in the areas of optical networking, wireless communications, Internet routing and congestion control protocols, parallel and distributed algorithms, and computational complexity theory. He has served on the editorial board of the IEEE Transactions on Networking. Dr. Siu received a National Science Foundation Young Investigator Award in 1993, the UC Irvine Distinguished Assistant Professor Award in 1995, the IEEE Browder J. Thompson Memorial Prize Paper Award in 1997, and the Best Paper Award of the SPIE Conference on All-Optical Networking in 1998.
E-mail: siu@list.mit.edu