Performance of a New Bluetooth Scatternet Formation Protocol [*]

Ching Law Amar K. Mehta Kai-Yeung Siu

Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139-4307

ching,amar,siu @list.mit.edu

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 evaluate the performance of a new randomized distributed Bluetooth scatternet formation protocol.

Our simulations validate the theoretical results that our scatternet formation protocol runs in $O(\log n)$ time and sends $O(n)$ messages. The scatternets formed 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 avoid overloading of any single device and lead to low interference between piconets. In addition, the simulations 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 with many masters and slaves in parallel. We investigate the collision rate and time requirement of the inquiry and page processes.

Deducing from the simulations results of scatternet formation and device discovery, we can verify that the total number of packets sent is $O(n)$ and demonstrate that the maximum number of packets sent by any single device is $O(\log n)$. At last, we give estimates of the total time requirement of the protocol and suggest further improvements.

Keywords

Bluetooth, Ad Hoc Networks, Resource Discovery, Topology Construction

[*]This work is supported in part by the MIT Auto-ID Center `<auto-id.mit.edu>`

1. INTRODUCTION

Bluetooth [1, 9, 6, 16] is an emerging low cost and low power short-range radio technology. It has been projected that as many as 200 million Bluetooth devices will be shipped in year 2001 [6]. Thus, Bluetooth is likely to become another important platform for ad hoc networking. Ad hoc networking over Bluetooth can lead to many useful applications. For example, in a conference room, a special announcement can be broadcast to the Bluetooth-enabled mobile phones and hand-held computers through an ad hoc network. Bluetooth ad hoc networks can also be used for rapid deployment of EMID (electromagnetic identification) readers [4].

The area of ad hoc networking has gathered much research interests in the past years. Many studies have concentrated on the routing issues of ad hoc networks [17]. 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 [7, 8] of ad hoc routing protocols have been conducted with a link-layer model based on or similar to the IEEE 802.11b standard.

A Bluetooth ad hoc network, however, brings new challenges. There are specific Bluetooth constraints not present in other wireless networks. A Bluetooth network is composed of *piconets*. Each piconet contains one master and up to seven slaves. Piconets can be connected by sharing slaves. A *scatternet* (Figure 1) is a set of connected piconets. As shown in [15, 21], the configuration of a scatternet has great effects on the performance of the network. For instance, when a scatternet contains more piconets, the rate of packet collisions increases.

Before we can tap the enormous power of Bluetooth ad hoc networking, we must first devise an efficient protocol to form a scatternet from isolated Bluetooth devices.

In this paper, we study the problem of scatternet formation in the situation where the devices are in-range of one another. We adopt a two-layered approach for our investigation of this problem. First, we investigate how these devices can get organized into scatternets. We evaluate the performance of a new scatternet formation protocol. Second, as a subroutine of the protocol, we study how the devices can discover each other efficiently.

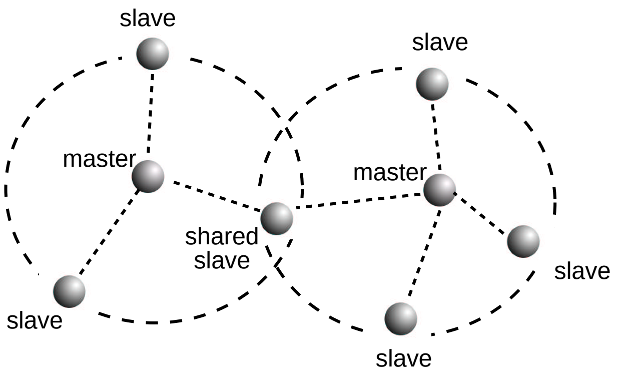

Figure 1: A Bluetooth scatternet.

In Section 2, we discuss the relevant research on Bluetooth scatternets. In Section 3, we introduce the problem of scatternet formation. We then present our protocol in Section 4 with simulation results. We discuss device discovery with simulation results in Section 5. In Section 6 we estimate the overall performance of the protocol. Section 7 concludes with remarks on future work.

2. RELATED WORK

Miklos *et al.* [15] use heuristics to generate scatternets with some desirable properties. They evaluated these scatternets of different characteristics through simulations. Johansson *et al.* [10] perform link-layer simulations of piconets. Raman *et al.* [18] argue for cross-layer optimization in Bluetooth Scatternets.

Aggarwal *et al.* [3] introduce a scatternet 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. Thus, another phase of re-organization is required.

Salonidis *et al.* [19] 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.* [20] also present a scatternet formation algorithm – BTCP (Bluetooth Topology Construction Protocol). BTCP has two phases: first, a leader is elected with a complete knowledge of all devices, and second, this leader will tell other devices how a scatternet should be formed. Our algorithm has only one phase: the scatternet is formed once a leader is elected. Since the topology is determined by a single device, BTCP has more flexibility in constructing the scatternet. A default scheme is presented in [20] for up to 36 devices.

The algorithms in [3, 20] depend on a single device to design the scatternet topology and then notify the 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 scatternet. We proved that our algorithm, which consists of single phase, has $O(\log n)$ time complexity [13]. However, as pointed out in [20], the role determination phase is much shorter than the coordinator election phase. Thus, the advantage of the $O(\log n)$ bound might not be relevant in practice unless the number of devices is very large. Moreover, we note that at least the phase two of [20]'s BTCP protocol 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.

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 within the 2.45 GHz band. Frequency hopping is used to reduce interference and enhance security.

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 and/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. 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 1.28s 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 to k slaves (k is 7 in Bluetooth 1.0b). 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*. A *scatternet* is a set of piconets connected through shared devices.

The problem of scatternet formation: How do 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 [1] of each other. This means that, potentially, any pair of devices can be connected directly.

[1] 10m to 100m in Bluetooth 1.0b

A scatternet formation protocol has two important performance measures:

- time complexity — amount of time to form a scatternet. A scatternet must 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.

In addition, it is also crucial to have scatternets of good quality. It is not very useful to have scatternets that lead to poor network performance. Thus, we should aim to produce scatternet that facilitates inter-piconet communications. It is not easy to quantify the quality of a scatternet, but we believe the following quality 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 [21], 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, this scatternet probably would not perform very well in practice because the shared slave will be overwhelmed, unless the network is small.

4. SCATTERNET FORMATION

In this section, we first present our scatternet formation protocol and then evaluate its performance and properties by simulations. The development of this algorithm was inspired by our research on resource discovery algorithms in general networks [12].

4.1 Algorithm

In the beginning, 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. A component can be a single device, a piconet, or a scatternet. There is one *leader* for 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 scatternet formation algorithm. 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 $\mathcal{S}(v) = \emptyset$. For this algorithm, we assume that $k \geq 2$.

In [13], we proved the following invariances 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 ϕ, such that procedure MAIN and the procedures called by it can be completed in ϕ seconds. A good choice of ϕ can be found by simulations (see Section 5) and by prototyping. We assume that all leaders will call procedure MAIN at time $t_0 + i\phi$, for $i = 0, 1, \ldots,$ where t_0 is the start time. Initially all devices are leaders.

In procedure MAIN, a leader calls SEEK with probability p, $1/3 < p < 2/3$. Otherwise, the leader calls SCAN or asks an unshared slave to call SCAN. During each round, only one device of each component will call SEEK or SCAN.

```
MAIN(leader $u$)
1  $x \leftarrow$ a random number in $[0, 1)$
2  if $x < p$   $(1/3 < p < 2/3)$
3     then SEEK($u$)
4     else  if $\mathcal{S}(u) = \emptyset$
5              then SCAN($u$)
6              else  $v \leftarrow$ 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 fewer 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. Essentially, during each round, a matching is found between the SEEK devices and SCAN devices. The number of connections made (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           // $\mathcal{S}(u) \leftarrow \mathcal{S}(u) \cup \{v\}$
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 5. In general, we make sure that each master is matched to only one slave, and vice versa. The details of this low-level protocol are discussed in Section 5.

When a leader u running SEEK connects to a slave v running SCAN, procedure CONNECTED(u, v) is called. If v's other master is w, the piconets of u and w will try to merge if possible. Essentially, if the piconets of u and w can be fit into a single piconet (with at most $k-1$ slaves), then w and the devices in $\mathcal{S}(w)$ become slaves of u. This is performed by the procedure MERGE. Otherwise, some slaves are moved from $\mathcal{S}(u)$ to $\mathcal{S}(w)$ by the procedure MIGRATE. There are other special cases. See Figures 2, 3, 4, 5, and 6 for illustrations of various cases in procedure CONNECTED.

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

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

Procedure MERGE(u, v, w) makes w and all devices in $\mathcal{S}(w)$ except v become slaves of u.

```
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, $u$)
```

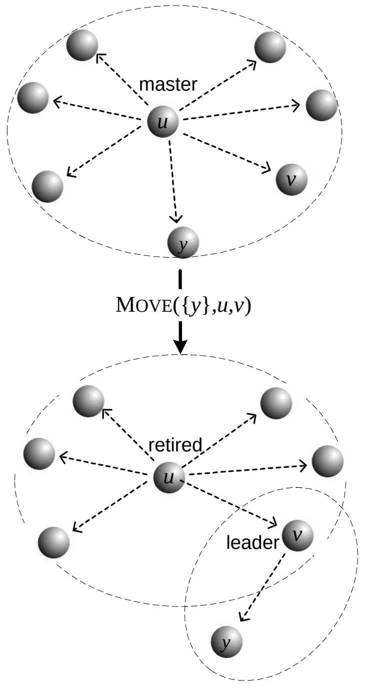

Figure 2: Lines 5-7 in procedure CONNECTED **for $k = 7$.**

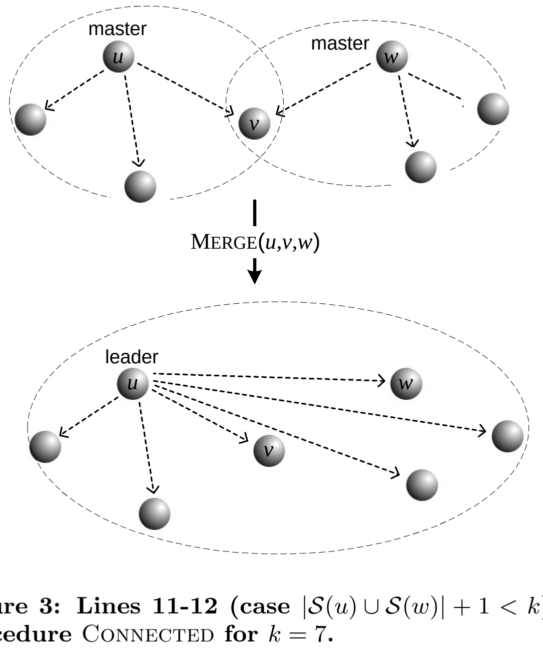

Figure 3: Lines 11-12 (case $|\mathcal{S}(u) \cup \mathcal{S}(w)| + 1 < k$) in procedure CONNECTED **for $k = 7$.**

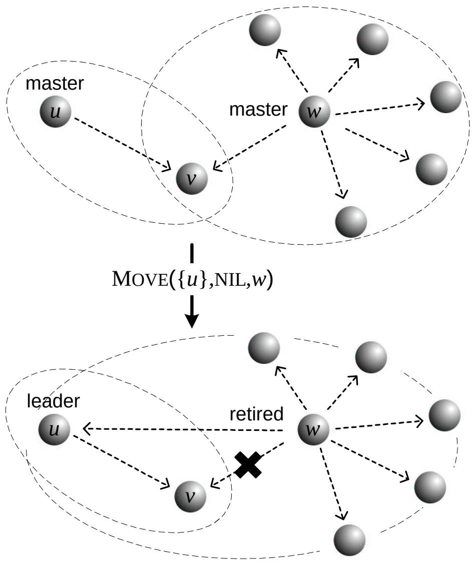

Figure 4: Lines 13-15 (case $|\mathcal{S}(u)| = 1$) in procedure Connected **for $k = 7$.**

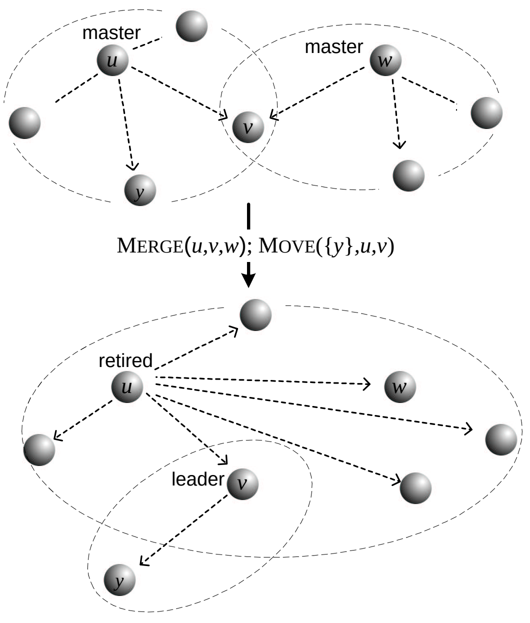

Figure 5: Lines 16-21 (case $|\mathcal{S}(u) \cup \mathcal{S}(w)| + 1 = k$) in procedure Connected **for $k = 7$.**

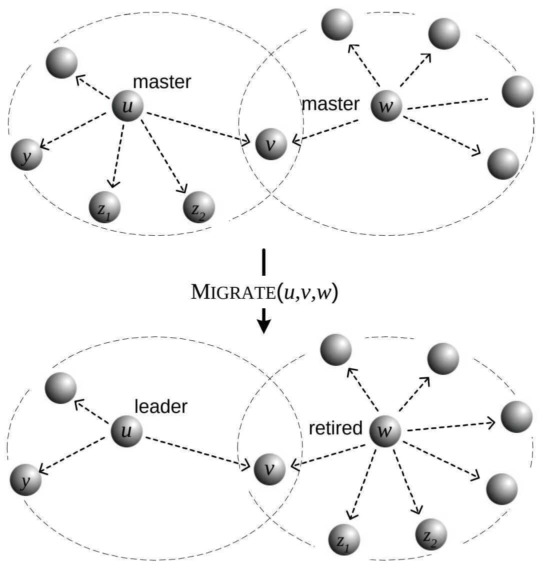

Figure 6: Lines 22-23 (case default) in procedure Connected **for $k = 7$.**

Procedure Migrate(u, v, w) moves slaves from $\mathcal{S}(u)$ to $\mathcal{S}(w)$ until $\mathcal{S}(w)$ is full or when 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 $u$
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
```

The algorithm does not minimize the absolute number of messages passed between devices in all circumstances. The current design is a compromise between simplicity of the algorithm and the constant factors of the time and message complexities of the algorithm. Moreover, Section 5 will show that most of the packets are sent during the inquiry processes.

We summarize the theoretical results proved in [13]:

- The scatternet formed by the algorithm has maximum degree two.
- The scatternet formed by the algorithm has at most $\lfloor (n-2)/(k-1) \rfloor + 1$ piconets.

- Any scatternet contains at least $\lceil (n-1)/k \rceil$ piconets.
- The scatternet formation algorithm has $O(\log n)$ time complexity and $O(n)$ message complexity.

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. We can let the leader stop after it has failed to find any device for several rounds. However, in a dynamic setting, this could prevent new devices from joining the scatternet.

4.2 Simulation Results

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

We simulate our scatternet formation algorithm in with `simjava` [2], a discrete event simulation package for Java. The probability p that each leader chooses to execute SEEK is 1/2 in our simulations. Following Bluetooth 1.0b specification, we set $k = 7$. We start with 2, 4, and 8 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 both linear scale and logarithmic scale. All results are averages of 50 trials.

4.2.1 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 will 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 of a scatternet captures the maximum routing delay between any pair of nodes in the scatternet. Although we did not have a theoretical analysis of the network diameter, Figure 8 shows that the network diameter is linear over the number of devices in log scale. This means that the network diameter is $O(\log n)$ (within the range of 2 to 128 devices).

4.2.2 Performance

First, it is crucial that the scatternet is formed as fast as possible, because this translates to the waiting time 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 result. In Section 5, we will investigate how long it takes to complete 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 energy. We can put the messages into three categories:

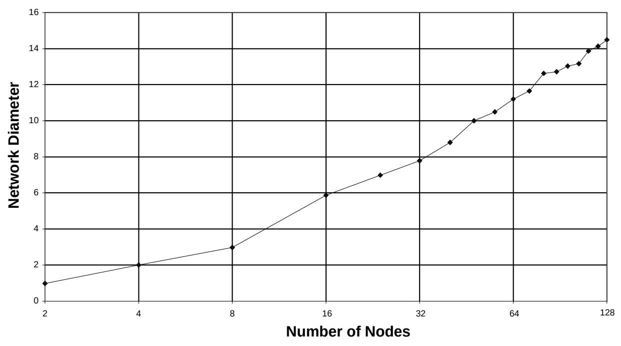

Figure 8: Network diameter of the scatternet formed.

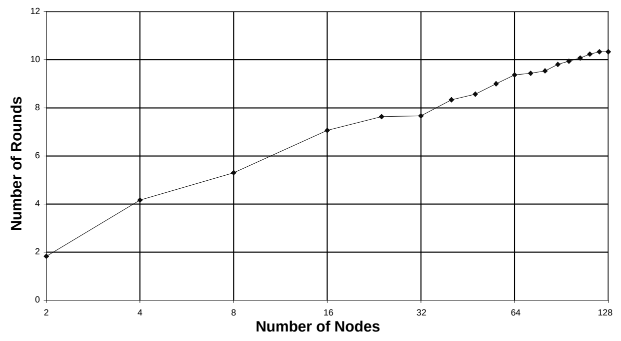

Figure 9: Number of rounds to form a scatternet.

Inquiries – Bluetooth INQUIRY and INQUIRY RESPONSE packets.

Pages – Bluetooth PAGE and PAGE RESPONSE packets.

Algorithmic Messages – rest of the messages used by our scatternet formation algorithm.

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

At last, in Figure 11, we can see that the maximum number of messages sent by any device is $O(\log n)$. This suggests that the power requirement of the unluckiest device is $O(\log n)$. A good 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 $\Omega(\log n)$.

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.

5. DEVICE DISCOVERY

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

As we discussed in Section 2, most previous work on Blue-

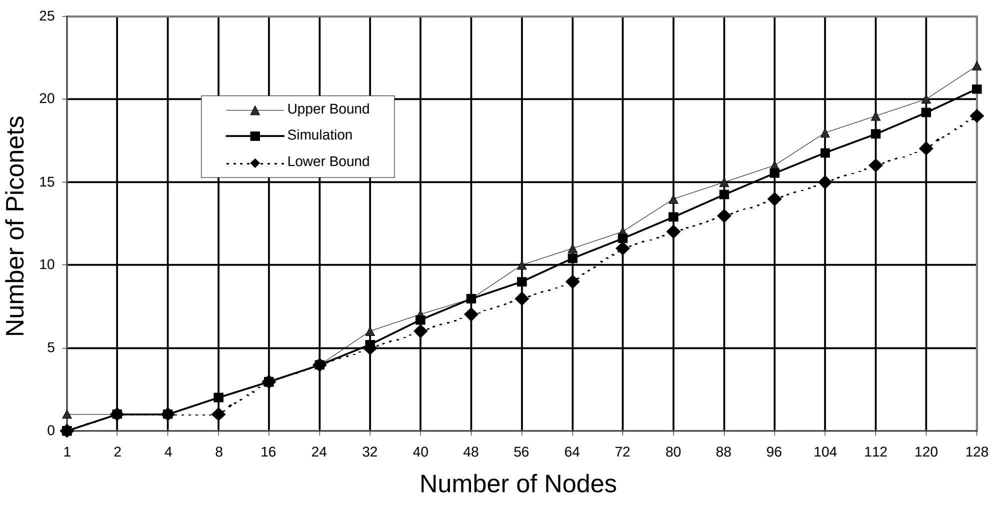

Figure 7: Number of piconets in the scatternet formed, compared to upper bound $\lfloor(n-2)/(k-1)\rfloor+1$ and lower bound $\lceil(n-1)/k\rceil$, where $k=7$.

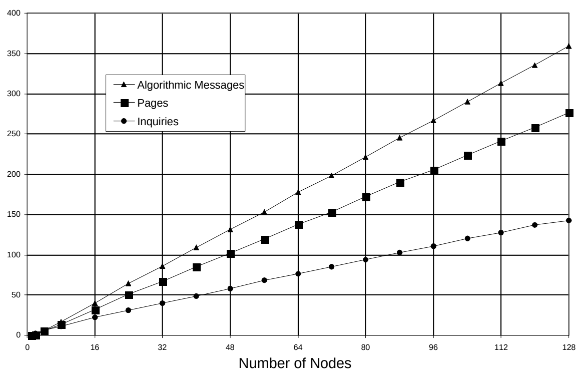

Figure 10: Total number of Algorithmic Messages, Pages, and Inquiries.

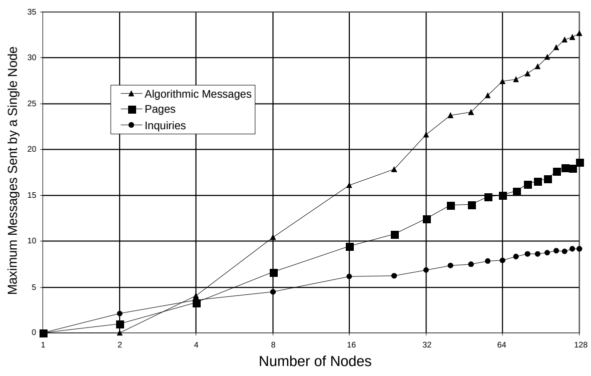

Figure 11: Maximum number of Algorithmic Messages, Pages, and Inquiries sent by any single node.

tooth connection establishment researched the problem of forming a single master-slave connection. However, in the scenario for 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.

5.1 Protocol

We present 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 fewer 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.

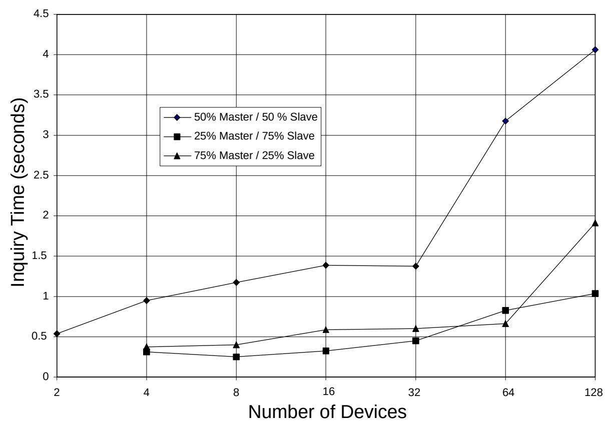

Figure 12: Running time of the inquiry phase with three master-slave ratios.

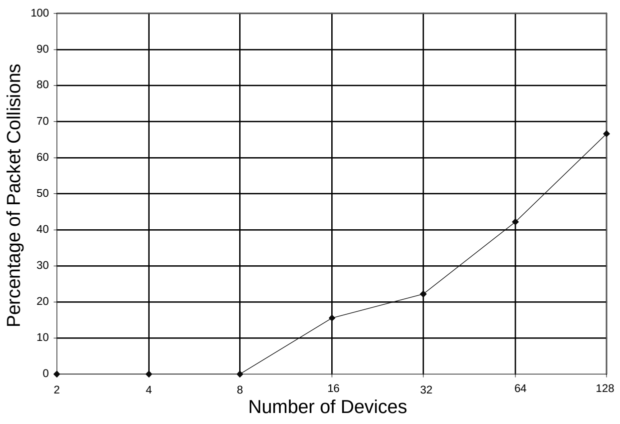

Figure 13: Percentage of packet collisions (over all packages sent) when there are 50% masters and 50% slaves.

5.2 Simulation Results

We also use `simjava` [2] for the simulations of this protocol. IBM's BlueHoc simulator [5] is not used because we started our implementation before BlueHoc was released in public. Frequency hopping sequences are implemented and collisions are detected. Since the overall time scale of the simulation is small, we did not implement clock drift. We present results up to 128 devices. The results are averages of 10 trials.

Figure 12 shows the running time of the inquiry phase, with three different master-to-slave ratios. The number of masters is distributed binomially. For number of nodes more than 8 and $p = 1/2$, the 25%-75% split and 75%-25% split encompass at least 2 standard deviations around the expected number of masters. 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.

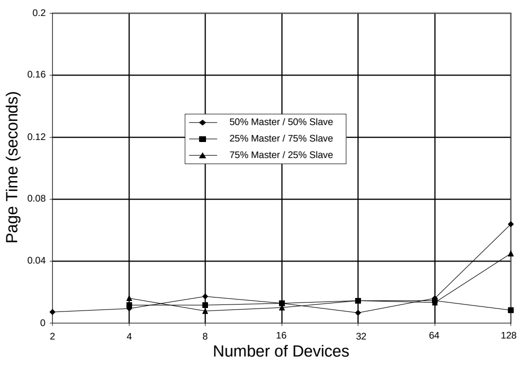

Figure 14: Running time of the page phase with three master-slave ratios.

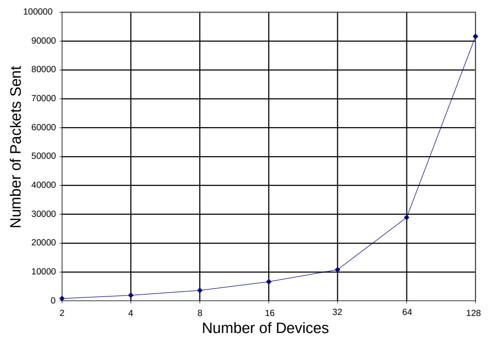

Figure 15: Total number of packages sent when there are 50% masters and 50% slaves.

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 inquiry. This is because the SEEK devices already know the address and clock of their target SCAN devices, thus they are able to contact the SCAN devices quickly. In addition, since their hopping sequences are different, the amount of collisions is much lower in this case. Therefore, the total time ϕ required by each round of the scatternet formation algorithm can approximated by the inquiry time of the largest number of devices to be supported. For example, if we want to support 32 devices, then a round-time ϕ of 1.4 seconds should be sufficient.

At last, 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 $n = 32, 64, 128$. This means that the total number of packets sent is still roughly linear of the number of devices.

6. OVERALL PERFORMANCE

We now estimate the overall performance of our protocol from the results of Subsection 4.2 and Subsection 5.2.

In Subsection 4.2, we learned that the number of Inquiries, Pages, and Algorithmic Messages all increase linearly with the number of devices. And in Subsection 5.2, we found that the number of packets sent during a single round of the protocol is also linear of 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 increase. In Subsection 4.2, we also showed that the number of Inquiries, Pages, and Algorithmic Messages of any single device increases logarithmically. This means that our protocol does not cause a very high load on any single device.

The issue of overall time requirement is more complicated. For an upper bound on the total time taken by the protocol, we can multiply the number of rounds (Figure 9) by the time required for each round (Figure 12). For example, we found that a timeout value of 1.4s is sufficient for each round up to 32 devices, and that the protocol takes about 7.7 rounds to form the piconet. This means that the total time requirement is about $7.7 \times 1.4 = 10.78$ seconds. Similarly, the total time required for 16 devices and 64 devices are at most 9.92 seconds and 29.94 seconds respectively.

The overall time requirement is longer than that of BTCP [20]. However, we believe the actual time requirement of our protocol can be made much shorter because of the following observations:

- The worst-case per-round time happens when there is an even split between the masters and slaves. However, if this happens a lot, the total number of rounds required is smaller. For example, if there is an even split every round, the protocol will only need $\log_2 32 = 5$ rounds to form a scatternet.
- The number of active leaders decreases rapidly. For example, the last round of the protocol will have only two leaders and thus can be finished in a shorter time.

In general, as suggested by [18], we need "cross-layer" optimization to improve the overall performance. In particular, an asynchronous approach should be studied: such that once a connection is established, the involved parties can proceed with the procedure CONNECTED. In practice, it is not necessary for all devices to execute CONNECTED at the same time. The synchronized nature of the current protocol is useful for the theoretical analyses, but it should be possible to consider and evaluate an asynchronous version of the protocol.

7. CONCLUDING REMARKS

In this paper, we introduced a Bluetooth scatternet formation protocol with $O(\log n)$ time complexity and $O(n)$ message complexity. These bounds are validated by our simulation results

We have shown that the algorithm produces scatternet with some desirable properties: small number of piconets for minimizing inter-piconet interference, and low device degrees 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 $O(\log n)$. At last, we also demonstrated that no single device is particularly exhausted by the protocol.

In the following, we suggest three directions of future work.

When there are many devices, 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 an inquiry packet is received, as specified in Bluetooth 1.0b.

In certain situations, some devices might be out of range of one another. The protocol has to be modified for such an environment. Basically, we need to check for feasibility before each MOVE procedure is executed. Depending on the connectivity of the devices, the piconets are likely to be of smaller sizes, implying a larger number of piconets.

Our protocol can be easily extended to work with dynamic environments (with devices joining and leaving the scatternet) and to support fault tolerance. Our current protocol already handles the events of devices joining. The new devices can simply start as leaders and will discover or be discovered by other devices. Additional work is required to deal with the case of devices leaving or failing. We can give an outline of a possible solution:

- 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 master 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 connected to this leader u through the shared slaves should now become leaders again. This will allow the protocol to proceed as usual. Fortunately, this expensive reorganization should be a rare event.

8. REFERENCES

[1] The Bluetooth Special Interest Group. `http://www.bluetooth.com`.

[2] simjava. `http://www.dcs.ed.ac.uk/home/hase/simjava/`.

[3] Alok Aggarwal, Manika Kapoor, Lakshmi Ramachandran, and Abhinanda 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, pages 54–63, Boston, MA, August 2000.

[4] MIT Auto-ID Center. `http://auto-id.mit.edu`.

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

[6] Jennifer Bray and Charles F. Sturman. *Bluetooth: Connect Without Cables*. Prentice Hall, 2001.

[7] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. *Mobile Computing and Networking*, pages 85–97, 1998.

[8] Samir R. Das, Robert Casta neda, and Jiangtao Yan. Simulation-based performance evaluation of routing protocols for mobile ad hoc networks. *Mobile Networks and Applications*, 5:179–189, 2000.

[9] Jaap Haartsen. Bluetooth - the universal radio interface for ad hoc, wireless connectivity. *Ericsson Review*, (3):110–117, 1998.

[10] P. Johansson, N. Johansson, U. Korner, and G. Elg, J.and Svennarp. Short range radio based ad-hoc networking: performance and properties. In *Proceedings of the IEEE International Conference on Communications 1999*, volume 3, pages 1414–1420, 1999.

[11] Ching Law, Amar K. Mehta, and Kai-Yeung Siu. Performance of a new Bluetooth scatternet formation protocol. In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc) 2001*, Long Beach, California, USA, October 2001.

[12] Ching Law and Kai-Yeung 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*, number FIM/110.1/DLSIIS/2000, pages 5–8, October 2000. Available at `http://list.mit.edu/~ching/`.

[13] Ching Law and Kai-Yeung Siu. A Bluetooth scatternet formation algorithm. In *Proceedings of the IEEE Symposium on Ad Hoc Wireless Networks 2001*, San Antonio, Texas, USA, November 2001.

[14] Amar Mehta. Ad-hoc network formation using Bluetooth scatternets. Master's thesis, June 2001.

[15] Gy. 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.

[16] Brent A. Miller and Chatschik Bisdikian. *Bluetooth Revealed: The Insider's Guide to an Open Specification for Global Wireless Communications.* Prentice Hall, 2000.

[17] Charles E. Perkins. *Ad Hoc Networking.* Addison-Wesley, 2001.

[18] Bhaskaran Raman, Pravin Bhagwat, and Srinivasan Seshan. Arguments for cross-layer optimizations in Bluetooth scatternets. In *Proceedings of Symposium on Applications and the Internet, 2001*, pages 176–184, 2001.

[19] Theodoros Salonidis, Pravin Bhagwat, and Leandros Tassiulas. Proximity awareness and fast connection establishment in Bluetooth. In *First Annual Workshop on Mobile and Ad Hoc Networking and Computing*, pages 141–142, 2000.

[20] Theodoros Salonidis, Pravin Bhagwat, Leandros Tassiulas, and Richard 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.

[21] Stefan 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*, volume 2, pages 1315–1319, 2000.