Distributed Construction and Maintenance of Bandwidth-Efficient Bluetooth Scatternets

Metin Tekkalmaz, Hasan Sözer, İbrahim Körpeoğlu
Department of Computer Engineering
Bilkent University, Ankara, Turkey
{metint, hsozer, korpe}@cs.bilkent.edu.tr

***Abstract*— Bluetooth networks can be constructed as piconets or scatternets depending on the number of nodes in the network. Although piconet construction is a well-defined process specified in Bluetooth standards, scatternet construction policies and algorithms are not well specified. Among many solution proposals for this problem, only a few of them focus on efficient usage of bandwidth in the resulting scatternets. In this paper, we propose a distributed algorithm for scatternet construction problem, that dynamically constructs and maintains a scatternet based on estimated traffic flow rates between nodes. The algorithm is adaptive to changes and maintains a constructed scatternet for bandwidth-efficiency when nodes come and go or when traffic flow rates change. Based on simulations, the paper also presents the improvements in bandwidth-efficiency provided by the proposed algorithm.**

I. Introduction

Bluetooth [1] is a short range wireless RF technology designed initially for cable replacement at indoor places, but also supports usage scenarios for personal area or local area networking. Its low cost, low power consumption and ad-hoc connectivity features make it a good wireless connectivity choice for mobile devices due to their anytime-anywhere connection requirement and limited battery power.

Bluetooth devices are able to form small networks, which are called piconets, with up to eight nodes. A piconet consists of a master node and one or more slave nodes. The scheduling of packets into the common FHSS radio channel in a piconet is coordinated by the master node via a polling based TDMA scheme. No direct communication between any two slave nodes is allowed. Data traffic among slave nodes has to go through the master node.

Bluetooth standards also specify some set of mechanisms to construct larger networks called scatternets. A scatternet is actually nothing but a network consisting of multiple piconets with common nodes between them, called bridges, to forward traffic between those piconets. Since all members of a piconet follows the same pseudorandom hopping sequence over some set of predefined RF channels specified in Bluetooth standards, and since each piconet has a different hopping sequence, a bridge node follows the hopping sequence of different piconets, which it belongs to, at different times.

Although piconet construction is a well-defined process described in current Bluetooth standards, scatternet construction algorithms and policies are not specified in detail. Therefore, this is an open research area and many methods have been proposed so far for scatternet construction problem. Once the scatternet is constructed, maintaining it in case of new node arrivals and node departures is another issue, which is not addressed much in the literature. Additionally, despite numerous works on scatternet construction, there is only a few studies ([2], [3] and [4]) that focus on constructing a scatternet with the aim of having the bandwidth capacity of the resulting scatternet utilized as efficiently as possible considering the traffic demands of nodes. These studies all propose static (i.e. does not handle new node arrivals and node departures) and centralized solutions.

In this paper, we provide an algorithm that dynamically constructs a scatternet and concurrently modifies it to make it more bandwidth-efficient. In order to reduce the traffic load on a scatternet and thereby effectively utilize the capacity of the scatternet, our approach is based on reducing the path lengths between highly communicating nodes as in [4]. Different from [4], our algorithm works in a distributed manner with no central coordination and without requiring the availability of global topology and traffic demand information. Furthermore, it is dynamic (i.e. new node arrivals and node departures are handled appropriately), and it can adapt to changes in the number of traffic flows and their rates, consequently preserve bandwidth-efficiency. Moreover, messaging overhead imposed by the algorithm is kept as low as possible.

The remainder of this paper is organized as follows. In the next section, related previous studies are summarized. In Section III, scatternet construction algorithm is described. In Section IV, evaluation criteria for bandwidth efficiency are explained and simulation results, based on these criteria, are presented. Finally, some future work issues are discussed and the paper is concluded in Section V.

II. Related Work

There have been many solution proposals for Bluetooth scatternet formation problem. Most of these proposals focus on the efficiency of the scatternet formation algorithm, considering the duration of the construction process and number of messages exchanged during the construction as it is the case in [5], [6] and [7]. Some of them also apply heuristics in order to make the constructed scatternet efficient in terms of some metrics. One such heuristic that is followed by some studies

This work is supported by The Scientific and Research Council of Turkey (TÜBİTAK), grant number 103E014.

is to keep the number of piconets as small as possible in the resulting scatternet with the goal of decreasing the amount of inter-piconet traffic and interference. There exist other studies, which focus on different aspects of the problem. In [8] and [9], for instance, tree-shaped topologies are formed for the ease of routing. Some studies like [10], on the other hand, aim to form fault-tolerant topologies by means of constructing alternative paths between nodes.

Although various proposals have been made to form scatternets with various objectives, algorithms that favor efficient usage of bandwidth are not widely studied. In [11], as one of the earliest studies, traffic patterns among the piconet members is taken into consideration while selecting the master of the piconet. Some studies on constructing efficient scatternet topologies exist also: [2], [3] and [4]. All these three proposals are single-hop, centralized, and static solutions. [2] proposes a graph theoretical solution which uses 1-factors to construct a scatternet. [3] approaches to the problem as a min-max optimization. [4], on the other hand, uses a set of heuristics to designate the locations and the roles of the nodes, assuming that the traffic flow information is known a priori.

Along with the scatternet construction proposals considering the relation between topology and efficiency, there are studies that try to reveal the relation between them such as [12], [13] and [14]. [13] applies an analytical approach where [14] tries to establish a mathematical background to determine the relations between the topology and network capacity. On the other hand, [12] applies a statistical approach in order to investigate the effects of topology on performance.

III. The Construction Algorithm

Two *concurrently* executed main procedures constituting the algorithm are *Maintenance* and *Link Establishment* procedures. The *Maintenance* procedure maintains the bandwidth-efficiency of a scatternet by modifying a given topology. On the other hand, the *Link Establishment* procedure handles new connections and constructs a base topology that the *Maintenance* procedure can work on.

The *Maintenance* procedure constantly collects traffic flow information and runs a set of operations based on this information when *significant* changes in traffic patterns are observed. These operations are performed in order to modify the scatternet topology so that the node pairs whose communication demands between each other are relatively higher are placed closer to each other in the scatternet.

The *Maintenance* procedure depends on some assumptions. The first assumption is that all the nodes, which are going to be part of the scatternet, are in the communication range of each other. Another assumption is that the topology which the *Maintenance* procedure works on can only contain slave/slave bridges, where a slave/slave bridge connects at most two piconets together. The topology cannot contain any other type of bridge, such as master/slave bridge. The *Link Establishment* procedure takes these restrictions into account, while establishing new links. The *Maintenance* procedure also preserves these properties. The restrictions on the properties of the topology can be eliminated with slight modifications on the operations of the *Maintenance* procedure. However, they are preferred to exist for the efficiency of the resulting scatternet topology. As studied in [15], master/slave bridges constitute a burden for the load balancing and simultaneous communication in different piconets. When a bridge master node switches to be a slave, all its slaves become stalled and their resources are wasted. On the other hand, switching time of bridges cannot be underestimated and sharing a bridge's time for more than two piconets would make that bridge a bottleneck. We also assume that the routing protocol that is run over the scatternet uses shortest paths between any two communicating nodes. The proposed algorithm can still work with a routing protocol that does not satisfy this condition, but such a routing protocol contradicts with the basic approach of the algorithm and cost metrics used for evaluation.

The method to estimate the flow patterns is explained in the next section. Section III-B explains how the operations use the traffic flow information, once it is acquired and Section III-C describes how the operations are combined to construct the *Maintenance* procedure. Section III-D provides the details of *Link Establishment* procedure.

A. Estimating the Traffic Patterns

Since the scatternet topology is modified according to the communication demands between nodes, traffic flow patterns must be estimated. There exist three types of traffic in a piconet about which information is collected:

- **Intra-piconet:** Traffic between members of the piconet (i.e. master-slave and slave-slave communication).
- **Incoming/Outgoing:** Traffic between members of the piconet and neighboring piconets.
- **Relayed:** Traffic forwarded to a neighboring piconet, which is received from another neighboring piconet.

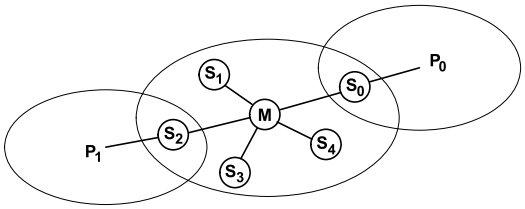

Fig. 1. A sample piconet on which traffic data is gathered.

Consider the piconet shown in Figure 1. It consists of a total of six members: A master node M; two slave/slave bridges, S_0 and S_2, connected to neighboring piconets P_0 and P_1, respectively; and three slave nodes, S_1, S_3 and S_4. For such a piconet, a traffic table, as illustrated in Figure 2, is constructed and maintained at the master node M.

Each cell in this table represents the amount of traffic flow rate observed between the entities that match the corresponding row and column. As indicated with the shaded cells in the figure, half of the table is used, since we are interested in *sum* of two one-way traffic demands between any two nodes.

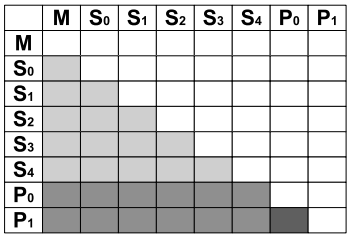

Fig. 2. Traffic table that is maintained at the master node of a piconet.

Also, the three types of traffic that are previously introduced are indicated with different shadings.

Traffic flow inside a piconet passes through the master node and therefore, almost all information required to generate the traffic table is available without extra messaging. Information exchange is only needed for traffic flow between bridge nodes and corresponding neighboring piconets. In the case shown in Figure 1, for instance, S_0 and S_2 should send traffic flow information between themselves and piconets P_0 and P_1, respectively, to the master node M.

The rate of traffic flow may change rapidly, which would lead to constant alteration of the scatternet topology. In order to prevent that, a discrete-time aging method shown in Equation 1 can be used while collecting traffic data. This method smoothly integrates changes in the instantaneous traffic rate to the average traffic rate, and the effect of instantaneous traffic rate to the average traffic rate can be adjusted by varying the α parameter between 0 and 1.

$$avgRate(t) = \alpha \times avgRate(t-1) + (1-\alpha) \times instRate(t) \quad (1)$$

Please note that, as new nodes join or existing nodes leave, the structure of the table should be changed. When a new node joins or an existing slave connects to another master resulting a new neighboring piconet, corresponding rows and columns should be added to the traffic table, after which monitoring related to the new entries begins. Similarly, when a node leaves the piconet or one of the bridges looses its connection to its other master, corresponding rows and columns should be deleted from the table.

B. Operations

The *Maintenance* procedure is composed of a set of operations, which are explained in subsequent sections.

1) Master/Bridge Reassignment: This operation reassigns master and bridge roles to the members of a piconet so that the cost function introduced in Equation 2 is minimized. In the equations, n is the number of slaves in the piconet (including the bridges), and m is the number of piconets around the piconet of interest.

$$C_p = C_{intra-p} + C_{inter-p} \quad (2)$$

$$C_{intra-p} = \sum_{i=0}^{n-1} \mathbf{T}_{S_i M} + \sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} 2 \times \mathbf{T}_{S_i S_j} \quad (3)$$

$$C_{inter-p} = \sum_{i=0}^{m-1} \mathbf{T}_{P_i M} + \sum_{i=0}^{m-1} \sum_{\substack{j=0 \wedge \\ S_j \neq bridgeto_{P_i}}}^{n-1} 2 \times \mathbf{T}_{P_i S_j} \quad (4)$$

The cost function has two components: cost of intra-piconet traffic ($C_{intra-p}$) and cost of inter-piconet traffic ($C_{inter-p}$). $\mathbf{T}$ represents the traffic table, while subscripts identify row and column indices. In order to calculate the cost, traffic flow rates between communicating entities multiplied by the length of corresponding communication paths are summed up. Since, all traffic flow passes through the master node, traffic flow rates of slave/slave and slave/piconet communication are multiplied by two. Since, there is no cost of communication incurred on the piconet of interest for traffic flowing between a neighboring piconet and the corresponding bridge node ($bridgeto_P$), such traffic amounts are excluded in the calculation of $C_{inter-p}$. The cost of relayed traffic is not affected by the selection of different master and bridge nodes. Hence, it is also not included in the cost function.

Although this operation rearranges roles inside a piconet, it has a global effect as illustrated in Figure 3. Suppose that nodes labeled as A and B in the figure communicate heavily with each other. Master node of P_0 will assign node A as the bridge node for P_1, since cost function will be minimized in this way because of high traffic flow rate between node A and piconet P_1. Afterwards, master node of P_1 will notice a high traffic flow rate between the new bridge node A and piconet P_2. In result of the *Master/Bridge Reassignment*, node A will now be given role as the bridge node between piconets P_1 and P_2. After node A becomes a member of piconet P_2 and master node of P_2 executes the *Master/Bridge Reassignment*, A will be the new master node in piconet P_2, and the length of the communication path between nodes A and B will be reduced to one.

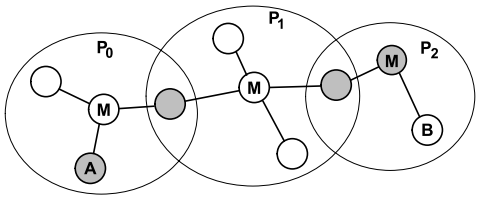

Fig. 3. Global effect of Master/Bridge Reassignment

This is just a sample scenario and it might have been the case that B approaches to A, or they could have met at piconet P_1 depending on the order of execution of the *Master/Bridge Reassignment* operation by the master nodes. At the end, however, A and B will be closer to each other in the scatternet as a result of successive executions of *Master/Bridge Reassignment* operation in different piconets.

2) Piconet Division: This operation splits a piconet into two provided that the number of slaves of the piconet is at least two more than the number of neighboring piconets of that piconet. Such a constraint is necessary because as a

result of the operation, there will be a need for a node to be the master in the offspring piconet (number of piconets increased from one to two) and a need for a node to be the bridge to connect the two piconets. Other than these, there must be enough number of slave nodes to communicate with neighboring piconets. In Figure 4, a piconet is split into two piconets.

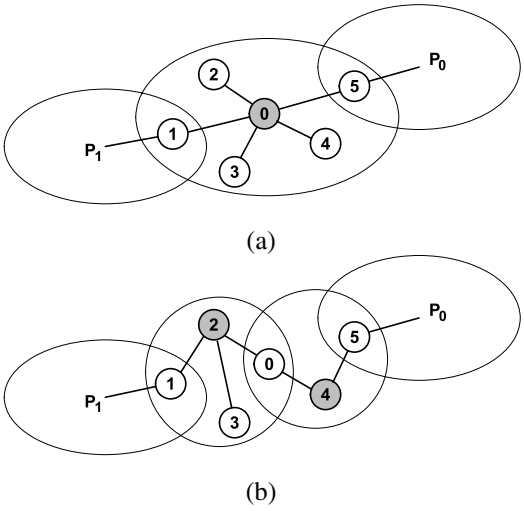

Fig. 4. Piconet Division operation

Similar to *Master/Bridge Reassignment*, in *Piconet Division*, for all possible role assignment configurations in a candidate resulting topology, the value of cost function is calculated and the configuration minimizing the cost is selected. Cost function in *Piconet Division* is similar to equations 3 and 4 in the sense that it sums up the end-to-end traffic flow rates between node pairs multiplied by the corresponding communication path lengths for all such pairs. However, resulting topology of *Piconet Division* may have either one-, two-, three- or four-hop paths, whereas hop distances are either one or two in *Master/Bridge Reassignment*.

3) Slave Transfer: This operation transfers a slave node of a piconet to a neighboring piconet if it is beneficial to do so and if the neighboring piconet can accept it. In order to determine whether a transfer of a slave is beneficial or not, first, gain and cost of the transfer have to be computed. Equations 5 and 6 shows how the gain, G_{ST}, and cost, C_{ST}, of a slave transfer is calculated. In the equations, n denotes the number of slave nodes in the piconet and m denotes the number of neighboring piconets of that piconet.

$$G_{ST} = \mathbf{T}_{S_s P_p} \tag{5}$$

$$C_{ST} = \mathbf{T}_{S_s M} + \sum_{\substack{i=0 \wedge \\ i \neq p}}^{m-1} \mathbf{T}_{S_s P_i} + \sum_{\substack{j=0 \wedge \\ j \neq s \wedge \\ j \neq bridgeto_p}}^{n-1} \mathbf{T}_{S_s S_j} \tag{6}$$

The gain and cost of transfer is calculated for each non-bridge slave node of the piconet of interest, which we denote as s, and for each neighboring piconet that has less than eight members, which we denote as p. The gain, G_{ST}, is the traffic flow rate between s and p. The cost of transferring a slave, C_{ST}, is the summation of traffic flow rates between the slave to be transferred and

- master node (of the piconet the slave belongs to before the transfer),
- neighboring piconets other than the candidate piconet,
- other slave nodes except the bridge node connected to the candidate piconet, which we denote as $bridgeto_p$.

In fact, G_{ST} and C_{ST} should be multiplied by two, because transfer of a slave node increments or decrements routing paths by two hops. Since this is the case for both cost and gain however, they are omitted.

Transfer of s to a p for which $G_{ST} - C_{ST}$ is positive, is marked as a beneficial transfer. The greater this value is, the more the transfer is beneficial. After determination of beneficial transfers, each transfer is performed one by one starting from the most beneficial one down to the least beneficial one. It might be the case that a transfer cannot be performed because p happens to have already eight members. Such cases may occur when beneficial transfers with common p exist. In these cases, transfers with more benefit are performed, others are skipped.

Although this operation affects two piconets, successive execution of it has a global effect as depicted in Figure 5. Suppose nodes labeled as A and B communicate heavily with each other. Master node of P_0 will notice that A communicates with P_1 more than it communicates with piconet members, and eventually it will transfer A to P_1. After that, P_1 will notice a high traffic flow rate between A and P_2. At the end, A will be transferred to P_2, where B is also located.

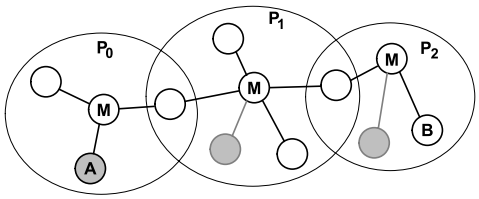

Fig. 5. Global effect of Slave Transfer operation

4) Piconet Merge: This operation can be thought as the reverse operation of *Piconet Division*. The topology shown in Figure 4(b), for example, turns into the topology shown in Figure 4(a) as a result of the execution of this operation. *Piconet Merge* tests all neighbors for merge feasibility until one such piconet is found. If total number of nodes does not exceed 8 and if total traffic flow rate does not exceed the raw capacity of 1 Mbps (or a practical upper bound that is set) of a Bluetooth piconet, two piconets are merged.

As the first action, traffic tables are merged. Actually, this is not a straightforward task because the traffic table of the resulting piconet cannot be generated out of the traffic tables of two neighboring piconets. As an example, in Figure 4(b), traffic flow rate between nodes 1 and 5 cannot be obtained

from traffic tables stored at nodes 2 and 4. Such information could be obtained by keeping and maintaining extra information and by exchanging extra messages. However, we apply a heuristic for completing the unknown parts of the merged traffic table. Envision two neighboring piconets P_0 and P_1 that are going to merge. From the traffic table of P_1, we know the amount of traffic flow between P_1 and P_0. However, we do not know how much of it is destined to (or originated from) P_0, and how much of it is relayed at P_0. But, from the traffic table of P_0, we also know what ratio of traffic flow between P_0 and P_1 is generated or consumed by the members of P_0. Combining these information we can estimate the amount of traffic flow between P_1 and the members of P_0 assuming the traffic is equally shared by the piconet members.

After traffic tables are merged, if total traffic flow rate does not exceed 1 Mbps, all nodes of the neighboring piconet are assigned as slave nodes to the master node executing the operation. *Master/Bridge Reassignment* is executed according to the traffic table thereafter in order to assign roles in the newly constructed piconet.

C. Maintenance Procedure

As indicated in the previous sections, the *Maintenance Procedure* is a combination of a set of operations which we have described already. In this section, we present a fusion algorithm that combines these operations and establishes an execution priority among them.

Algorithm 1 The Fusion Algorithm

```
1: Execute Master/Bridge Reassignment operation
2: if Current node is still master then
3:     Execute Slave Transfer operation
4:     if Total piconet traffic > 1 Mbps then
5:         Execute Piconet Division operation
6:     else
7:         Execute Piconet Merge operation
8:     end if
9: end if
```

The fusion algorithm, shown in Algorithm 1, is executed at a master node whenever a *significant* change is observed in the traffic table maintained at that master node or the total traffic that has to flow in the piconet of that master exceeds 1 Mbps. Such a bound is necessary because the capacity of a piconet is 1 Mbps. When total traffic demand in a piconet is more than 1 Mbps, the piconet should be split into two in order to increase the capacity. When this is not the case, however, merging piconets as much as possible is beneficial for the sake of shortening paths and decreasing the number of bridges that switch between piconets.

Significant change is supposed to occur when the value of *totalDiff* calculated as in Equation 7 exceeds a threshold value. Adjustment of the threshold would affect the sensitivity of the *Maintenance* procedure to the changes in the traffic flow rate.

$$totalDiff = \sum_{i=0}^{t-1} \sum_{j=i+1}^{t-1} \left| T_{ij} - T_{last_{ij}} \right| \qquad (7)$$

This equation gives the sum of differences between traffic flow rates for each pair of communicating entities (i.e. master node, slave nodes and piconets) represented in the current traffic table and the traffic table that was used in the last execution of the fusion algorithm. Initial execution of the algorithm at a master node is performed after a warmup period when enough information is ready in the traffic table of the master node.

D. Link Establishment Procedure

The links in Bluetooth are established after inquiry and page steps. Inquiry and inquiry-scan modes of inquiry step play a key role for device discovery and determination of master/slave roles. *Link Establishment* procedure in our proposal basically specifies whether a node having a certain role (i.e. master, bridge, slave, or free-node) can switch to inquiry or inquiry-scan modes. This specification controls the discoverability of nodes, and in this way the established links are forced to meet the restrictions described in Section III. Table I shows which modes of inquiry step are allowed in which roles of nodes in a scatternet.

TABLE I
MODES ASSIGNED TO ROLES AT INQUIRY STEP

Role	Inquiry	Inquiry-Scan
Free-Node	Yes	Yes
Master	Yes	No
Slave	No	Yes
Bridge	No	No

Because of the mode assignments shown in Table I, link establishments are restricted to happen only between pairs of nodes having the following role combinations: master/slave, where slave becomes bridge; master/free-node, where free-node becomes slave; slave/free-node, where free-node becomes master; and free-node/free-node, where one of them becomes master and the other becomes slave. Bridge nodes cannot establish new links. As a result, there is no master/slave bridges and a bridge node has exactly two masters as required by the *Maintenance* procedure.

Unlike *Maintenance* procedure, *Link Establishment* procedure runs on any node. Nodes, except bridges and masters that already have seven slaves, are scheduled to enter one or both of the inquiry and inquiry-scan modes, which is determined by the rules listed above, at certain intervals. These intervals are determined by the master node for itself and for its slaves. Once a new node is discovered in accordance with the restrictions on the topology of scatternet, *Link Establishment* procedure further decides to establish the link or not to. Link establishment can be restricted in order to limit the average node degrees. One such restriction would be to prevent

establishing links between a slave and the master of a one-hop distance piconet. This information can be passed to slave nodes each time they are instructed to enter inquiry-scan mode by the master, which already have one-hop distance master information in its traffic table. Other controls can be imposed whether to continue with page/page-scan mode after the discovery, using probabilistic approaches or acquiring 2-, 3-hop information by extra messaging if desired.

Link Establishment procedure enables dynamic handling of new link establishments. Hence, a scatternet topology, that can further be modified by *Maintenance* procedure for bandwidth-efficiency, is constructed on the fly. During *Maintenance* procedure the set of nodes constituting the piconet should not be changed, requiring coordination between *Link Establishment* and *Maintenance* procedures. This can be achieved if the masters do not schedule slave nodes for device discovery before starting a *Maintenance* procedure.

IV. Simulation Results

In this section, we present the evaluation for our proposed algorithm. For evaluation we use a performance metric called weighted average shortest path (WASP) proposed in [4]. WASP aims to reflect the load on the network for a given network topology and traffic demand distribution among nodes. WASP relates the end-to-end *traffic demand* to the *traffic load* imposed on a network with a given topology. The amount of end-to-end traffic demand between two nodes is expressed as the sum of two one-way traffic that has to flow between these two nodes. The load imposed on the network due to the traffic demand between two nodes is computed by multiplying the amount of traffic demand by the number of hops between the two nodes. Equation 8 shows how to compute the WASP of a flow between a node-pair x (WASP_x), where d_x is the traffic demand of node pair x, L_x is the number of hops between these two nodes, and n is the number of all node-pairs demanding some amount of traffic to flow in between:

$$WASP_x \quad = \quad (d_x \times L_x) \quad / \quad \sum_{i=1}^{n} d_i \qquad (8)$$

The WASP of a network is defined as the sum of WASPs of all end-to-end flows demanded in the network:

$$WASP \quad = \quad \sum_{i=1}^{n} (d_i \times L_i) \quad / \quad \sum_{i=1}^{n} d_i \qquad (9)$$

The WASP value in a network will be equal to one when communicating nodes in the network are separated from each other by direct links (i.e. single-hop communication). In this case, the traffic load imposed on the network by a certain traffic demand pattern will be minimum. WASP value gets closer to one as the traffic load on the network decreases, meaning that pairs with higher traffic demands are getting closer.

In order to test the performance of the proposed algorithm, a simulation environment, which uses WASP as the performance metric, is developed. Simulations are run for different traffic characteristics and for scatternet sizes varying between 10 and 100 nodes. For each scatternet size, 100 different initial scatternet topologies are generated and the proposed algorithm is run on arbitrary master nodes until no significant improvement on total bandwidth usage is obtained. Numerically speaking, if the improvement is less than 1% compared to the previous topology, the scatternet is assumed to be stabilized. For each scatternet size, the WASP values for each of 100 initial and final topologies are gathered and their average is calculated.

Three different traffic characteristics are used in simulations. In the first type of traffic characteristic (TC-1), given a pair of nodes (out of all possible pairs of nodes), the nodes

- do not communicate with a probability of 0.3,
- communicate at 5 Kbps with a probability of 0.2,
- communicate at 10 Kbps with a probability of 0.2,
- communicate at 15 Kbps with a probability of 0.2,
- communicate at 20 Kbps with a probability of 0.1.

In the second type of traffic characteristic (TC-2), given a pair of nodes, they

- do not communicate with a probability of 0.4,
- communicate at 5 Kbps with a probability of 0.3,
- communicate at 30 Kbps with a probability of 0.3.

In the third type of traffic characteristic (TC-3), nodes are grouped into three groups as Group-A, Group-B and Group-C, which constitute 30%, 30% and 40% of all nodes in a scatternet, respectively. Given a pair of nodes

- both from Group-A, they communicate at 20 Kbps with a probability of 0.7;
- both from Group-B, they communicate at 15 Kbps with a probability of 0.8;
- both from Group-C, they communicate at 15 Kbps with a probability of 0.6;
- one from Group-A and one from Group-B, they communicate at 4 Kbps with a probability of 0.2;
- one from Group-A and one from Group-C, they communicate at 2 Kbps with a probability of 0.3;
- one from Group-B and one from Group-C, they communicate at 3 Kbps with a probability of 0.3.

In TC-1 and TC-2, although there are demands with different communication rates, the demands are evenly distributed among node pairs. Different from TC-1 and TC-2, in TC-3, however, there are groups, where we have higher rate traffic demands between the nodes belonging to the same group and lower rate demands between the nodes belonging to different groups. The initial and final WASP values for these three different traffic characteristics on varying scatternet sizes are listed in Table II. As it can be clearly seen, whatever the traffic characteristics is, the initial WASP values are very similar. However, as the scatternet-wide communication relations among nodes get weaker, the final WASP values decrease.

In Figure 6, improvements in WASP for TC-1, TC-2 and TC-3 are shown for varying scatternet sizes. For TC-3 the improvements are higher than TC-1 and TC-2, and improvements up to 46% is reached, since heavily communicating nodes can be grouped together in the scatternet for TC-3. On the other hand, in TC-1 and TC-2 as a node is moved towards another

TABLE II
INITIAL AND FINAL WASP VALUES FOR TC-1, TC-2 AND TC-3

Node Count	TC-1 Initial	TC-1 Final	TC-2 Initial	TC-2 Final	TC-3 Initial	TC-3 Final
10	2.913	1.931	2.927	1.797	2.774	1.563
20	4.202	2.999	4.126	2.866	4.161	2.264
30	5.060	4.087	4.932	3.924	4.891	2.971
40	5.670	4.917	5.686	4.672	5.656	3.813
50	6.331	5.577	6.189	5.255	6.146	4.748
60	6.688	6.041	6.626	5.752	6.633	5.372
70	7.063	6.365	6.903	6.084	7.292	6.091
80	7.467	6.827	7.332	6.602	7.402	6.396
90	7.755	7.149	7.801	6.987	7.783	6.820
100	7.995	7.417	8.029	7.329	8.245	7.348

node it highly communicates, it gets further from the other nodes it communicates at the same rate, because every node has a similar probability of communication with another node at certain rate. TC-2 has better improvement than TC-1, since the ratio of non-communicating nodes is higher in TC-2 and it has a more diverse traffic distribution than TC-1. Figure 6 also shows that as the scatternets get larger, the improvements decrease. This is due to the higher load on the scatternets, where piconets have to relay more traffic. According to the proposed algorithm, a piconet is divided if the traffic flow rate within the piconet exceeds 1 Mbps. Hence, as the scatternet gets larger, the piconets tend to divide, which increases the path lengths between communicating nodes and consequently the value of WASP metric. Although improvement in WASP decreases, since the average available bandwidth per piconet on the way from source to destination increases, the data packets are not dropped due to high congestion, which, in fact, means improvement in general performance.

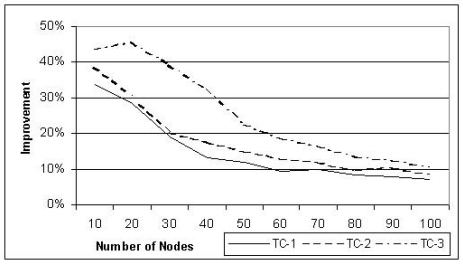

Fig. 6. Improvement in WASP

V. CONCLUSION AND FUTURE WORK

In this paper, we propose a Bluetooth scatternet formation and maintenance algorithm. The algorithm is different from previous proposals due to its basic motivation, which is to minimize the bandwidth usage in the resulting scatternet topology. The proposed algorithm works in a distributed fashion and it is adaptive, that is, it preserves bandwidth efficiency as traffic demands change. It is also dynamic so that it handles arrival of new nodes and departures of existing nodes. As the bandwidth usage is reduced in the scatternet, available bandwidth for new communication demands increases. Furthermore, since the average number of hops between communicating nodes and traffic load on the links is reduced, the average end-to-end latency is also reduced. Another benefit is that the average power consumption per node is expected to decrease, since the amount of traffic which is not directly related with a node but still needs to be forwarded by that node is reduced. The proposed algorithm is also evaluated and the results are observed to be promising.

Although the proposed scatternet formation and maintenance algorithm is designed to meet the requirements of a practical solution for wireless ad-hoc networks, such as distributed approach with as less messaging overhead as possible and adaptivity to changing conditions, it has one major restriction which is the assumption that all nodes are in the Bluetooth range of each other. This assumption may not be valid for some application scenarios. Therefore, as a future work, we consider enhancing the proposed algorithm to handle this kind of situations as well.

REFERENCES

[1] *Bluetooth*, Bluetooth Special Interest Group. [Online]. Available: http://www.bluetooth.com

[2] S. Baatz, C. Bieschke, M. Frank, C. Kuhl, P. Martini, and C. Scholz, "Building efficient bluetooth scatternet topologies from 1-factors," in *Proceedings of the IASTED International Conference on Wireless and Optical Communications, WOC 2002*, 2002.

[3] M. A. Marsan, C. F. Chiasserini, A. Nucci, G. Carello, and L. D. Giovanni, "Optimizing the topology of bluetooth wireless personal area networks," in *IEEE INFOCOM 2002*, 2002.

[4] T. Topal, "Constructing efficient bluetooth scatternets," Master's thesis, Department of Computer Engineering, Bilkent University, 2004.

[5] 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, MobiHoc*, 2001.

[6] C. Law and K. Y. Siu, "A bluetooth scatternet formation algorithm," in *Proceedings of the Symposium on Ad Hoc Wireless Networks*. IEEE, 2001.

[7] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire, "Distributed topology construction of bluetooth personal area networks," in *Proceedings of the INFOCOM*. IEEE, 2001.

[8] G. Tan, A. Miu, J. Guttag, H. Balakrishnan, T. Berners-Lee, L. Masinter, and M. McCahill, "Forming scatternets from bluetooth personal area networks," MIT Laboratory for Computer Science, Tech. Rep. Mit-lcs-tr-826, 2001.

[9] G. Zaruba, S. Basagni, and I. Chlamtac, "Bluetrees-scatternet formation to enable bluetooth-based personal area networks," in *Proceedings of the IEEE International Conference on Communications*, 2001.

[10] C. Petrioli, S. Basagni, and I. Chlamtac, "Configuring bluestars: Multihop scatternet formation for bluetooth networks," *IEEE Transactions on Computers, Special Issue on Wireless Internet*, vol. 52, no. 6, pp. 779–790, 2003.

[11] D. Miorandi and A. Zanella, "On the optimal topology of bluetooth piconets: Roles swapping algorithms," in *Proceedings of Mediterranean Conference on Ad Hoc Networks, Med-Hoc-Net*, 2002.

[12] G. Miklos, A. Racz, Z. Turanyi, A. Valko, and P. Johansson, "Performance aspects of bluetooth scatternet formation," in *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, 2000.

[13] R. Kapoor, M. Sanadidi, and M. Gerla, "An analysis of bluetooth scatternet topologies," in *ICC 2003*, 2003.

[14] D. Miorandi, A. Trainito, and A. Zanella, "On efficient topologies for bluetooth scatternets," *Lecture Notes in Computer Science*, vol. 2775/2003, pp. 726–740, 2003.

[15] M. Kalia, S.Garg, and R. Shorey, "Scatternet structure and inter-piconet communication in the bluetooth system," in *IEEE National Conference on Communications*, 2000.