AN ALGORITHM FOR ENERGY-EFFICIENT BLUETOOTH SCATTERNET FORMATION AND MAINTENANCE

Canan Pamuk and Ezhan Karasan

Department of Electrical and Electronics Engineering
Bilkent University, 06800, Ankara, Turkey, {canan, ezhan}@ee.bilkent.edu.tr

Abstract - We discuss an energy-efficient, distributed Bluetooth *S*catternet *F*ormation algorithm based on *Devi*ce and *L*ink characteristics (SF-DeviL). SF-DeviL forms multihop scatternets with tree topologies and increases battery lifetimes of devices by using device types, battery levels and received signal strengths. The topology is dynamically reconfigured in SF-DeviL by depleting battery levels and it is shown through simulations that the network lifetime is increased by at least 32% compared to LMS algorithm [1].

Keywords – Bluetooth, scatternet formation and maintenance, energy-efficient topology construction.

I. INTRODUCTION

Bluetooth is a short-range (10-100m) wireless ad-hoc network technology, that supports both voice and data communication. Bluetooth operates in the unlicensed 2.4 GHz ISM band and employs fast frequency hopping spread spectrum (FHSS). The basic network architecture of Bluetooth is a *piconet*, which consists of a master and up to 7 active slave nodes. The master controls intra-piconet communication by polling the slaves. Bluetooth also enables inter-piconet communication by forming scatternets. *Scatternet* is the network formed by interconnecting piconets through shared nodes called *bridges*. A bridge node can be master in one piconet and slave in the other (M/S), slave in both piconets (S/S) or M/S/S, etc.

The Bluetooth standard enables formation of scatternets, but it does not define an exact method [2]. The problem of scatternet formation can be stated as the assignment of master, slave and bridge roles to Bluetooth nodes and the assignment of links between nodes. Some factors that make scatternet formation more challenging are: mobility of devices, low computational and energy resources of devices, devices with no prior knowledge about other nodes, necessity to form the scatternet within a tolerable delay, requirement to set up each link before data is exchanged (due to frequency hopping channel).

Energy efficiency is one of the most important aspects of Bluetooth operation since mobile devices rely on batteries. Energy efficiency can be measured in terms of the *lifetime of a scatternet*, which is defined as the duration until one of the Bluetooth devices exhausts its battery.

In this paper, we present a multi-hop, distributed scatternet formation and maintenance algorithm called SF-DeviL, that efficiently manages battery powers of devices in order to increase scatternet lifetime. SF-DeviL uses device characteristics (class of device, battery capacity and level) and link features (received signal strength) together with power control, in order to achieve energy efficiency. Master, slave and bridge roles are based on device types. Established links in the topology are chosen such that links with lower transmit power requirements are given priority over potential links with higher transmit powers. The minimum transmit power for each candidate link is obtained from quantized measurement of the received signal strength. One of the important features of SF-DeviL is that slave nodes select their masters. SF-DeviL reconfigures the scatternet topology as the battery levels deplete and/or positions of devices change in order to maintain energy-efficiency. Simulations show that SF-DeviL increases scatternet lifetime by at least 32% compared to the LMS algorithm [1] while forming scatternets within reasonable delays.

The rest of the paper is organized as follows. Proposed solutions for Bluetooth scatternet formation are reviewed first and SF-DeviL is introduced next as an energy-efficient algorithm for scatternet formation. Simulation results are presented for comparing performances of SF-DeviL with another scatternet formation algorithm in Section IV.

II. SCATTERNET FORMATION ALGORITHMS

Proposed methods for Bluetooth scatternet formation show differences in their approaches. A centralized approach [3] needs extensive messaging between nodes and is impractical in dynamic environments. Distributed techniques provide the most appropriate solution for constructing scatternets. In single-hop scatternet formation algorithms, it is assumed that all nodes are within communication range of other nodes [1,4]. LMS [1], which tries to minimize the number of piconets, and TSF [4] are distributed single-hop scatternet formation algorithms that result in tree topologies and are appropriate for maintaining topology changes such as node additions and deletions (failures). Algorithms with multi-hop scatternets [5-7] do not require the assumption that all nodes are within communication range of other nodes, and thus have a wider application range.

Algorithms also differ in the resulting scatternet topology: some with tree [1,4-6] and some with mesh topologies [7]. Two distributed, multi-hop scatternet formation protocols resulting in tree topologies called Bluetrees are proposed in [6]. A multi-hop solution that results in a mesh topology is proposed in [7].

An energy-efficient, multi-hop scatternet formation algorithm, called SF-DeviL, which forms scatternets with tree topologies, is proposed in [5]. The resulting algorithm is shown to form energy-efficient scatternets with increased lifetime. Energy-efficient techniques for routing in Bluetooth scatternets have been investigated, and it is shown

0-7803-8523-3/04/$20.00 ©2004 IEEE.

that a considerable gain in network lifetime can be achieved by using distance based power control and battery level based master/slave switch [8].

III. SF-DEVIL

SF-DeviL forms a scatternet such that efficient usage of device batteries throughout scatternet operation is maintained [5]. Battery capacities of devices and transmission powers of potential links are considered in forming the scatternet. In SF-DeviL, each device selects the best master for itself. The selection of a single master by each device results in a tree topology with leaf nodes undertaking slave roles, intermediate nodes become M/S type bridges and the root node undertake the master role. SF-DeviL quantifies device and link specific features using two parameters: Device Grade and Received Signal Strength Grade.

A. Algorithm Parameters

Each device is assigned a Device Grade (DG) using the 'class of device' and battery level information. The class of a device can expose many features of the node such as mobility, traffic generation rate and battery capacity. For example, a laptop has a larger battery capacity than a mobile phone, and it most likely generates more traffic. Each Bluetooth unit calculates its DG by:

$$DG = BatteryCapacity * BatteryLevel + TrafficGenerationGrade \quad (1)$$

where BatteryCapacity indicates the power capacity of the device battery, BatteryLevel represents the fraction of remaining battery and TrafficGenerationGrade is a prediction of amount of traffic generated by the device.

The device class is known to Bluetooth modules, and it is exchanged with neighboring devices during connection establishment by using the 24-bit class of device/service (CoD) field in the FHS packet [2]. We assume that the BatteryLevel information is also embedded using some of the reserved bits in the FHS packet. Thus, two devices that establish a connection know DGs of each other.

Bluetooth supports power control, where the transmission power can be lowered as long as reliable communication is assured. Power control can be used for optimizing the system interference and energy-efficiency. The Bluetooth transceiver has a Receiver Signal Strength Indicator (RSSI) that measures the strength of the received signal [2]. In SF-DeviL, each device assigns a Received Signal Strength Grade (RSSG) to each neighboring device, based on the measured RSSI for each link. RSSG is quantized according to the strength of the received signal as: weak (W), medium (M), strong (S) and very strong (VS).

B. Best Master Selection

Using DG and RSSG, each device chooses itself a master, i.e., slaves choose their master based on DG and RSSG. The selection of the 'best master' is done by comparing DG and RSSG of a newly discovered neighbor with the current master. The flow chart for the BestMaster selection procedure is given in Fig. 1 for a generic node X. The BestMaster selection is done based on the following observations:

1. A device with high DG is more appropriate to be a master since it has higher battery capacity, battery level and/or traffic generation rate.
2. Establishing links with lower path loss provides advantages since transmission power and interference can be reduced by using power control.

BestDevice(master, neighbor) is the procedure which determines the most suitable master for X. The BestMaster selection procedure chooses the better node between the current master and a newly discovered neighbor. A discovered neighbor is selected as the master only if it has a larger or equal DG compared to X. When DGs are equal, the device with larger number of slaves or larger BD_ADDR is selected as the master.

A link with RSSG = VS has priority over other links. This ensures that links between devices receiving strong signals from each other are established, so that less power is consumed for transmitting signals, thereby increasing the lifetime of the scatternet and reducing interference. The node with the largest sum of RSSG and DG is preferred as the master. Using this rule, a closer PDA can be chosen to be the master than a far away laptop.

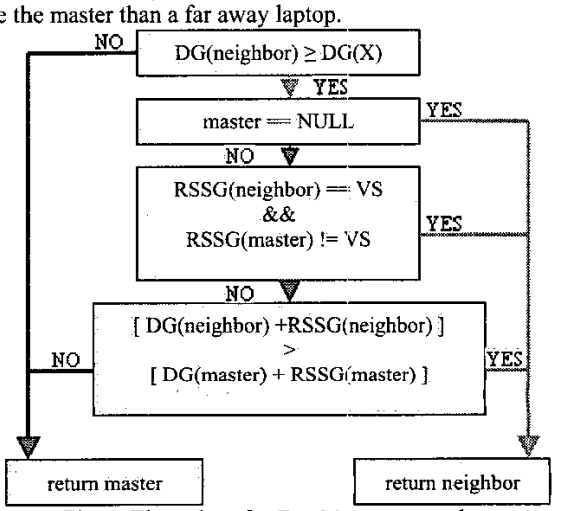


Fig. 1. Flow chart for BestMaster procedure

C. Algorithm for Scatternet Formation

SF-DeviL is a two-phase algorithm:

I. During the first phase, each node continuously tries to discover other devices. Each time a new neighbor is discovered, the *better* master for the node is determined by choosing between its current master and the newly discovered neighbor according to the *BestMaster* procedure. This phase ends when the discovery timeout (discTO) is reached. At the conclusion of this phase, each slave should have chosen a master and connected to it.

II. In the beginning of the second phase, each device has either found a master, or it has declared itself as the root of the scatternet. In the second phase, paging procedures are initiated by the nodes that have no assigned master, so

that disconnected trees resulting after the first phase, are merged. This phase ends when all disconnected trees are united into a scatternet.

SF-DeviL is a distributed algorithm where each device upon initialization starts the MAIN procedure given in Table 1. The generic device X calculates its DG and starts device discovery by alternating between inquiry and inquiry scan modes [2,3] until a neighboring device Y is found and a link is established. X executes the ArrangeRoles(X,Y) procedure given in Table 2, by which either the master-slave roles for the established X-Y link are chosen or the link is deleted. Node X gets the BD_ADDR, DG of Y through exchange of FHS packets and RSSG from RSSI measurements.

In line 2 of by the ArrangeRoles procedure, X uses the BestMaster procedure to select the best master for itself. If Y is chosen, X frees itself from its current master and becomes the slave of Y. If X is the inquirer (which implies that X becomes the master of Y automatically after connection establishment), X and Y switch master/slave roles for X to become the slave of Y. This is done to ensure that 'better nodes' are masters. If Y is not a better master for X, the newly established X-Y link is broken.

Node X continues with the discovery of neighbors, seeking the best master for itself by comparing newly discovered neighbors with its current master. X forms a list of its discovered neighbors by adding the discovered devices to its neighbor_list(X). The first phase continues until the discovery timeout (discTO) is reached.

If X has not found a master in the first phase, it declares itself as the *semiroot* (this term is used for a node that may potentially be the root of the scatternet) and runs the Semiroot procedure given in Table 3. In this procedure, the semiroots accessible by a single hop are merged first (line 1), and the semiroots accessible in multiple hops are then merged (line 4). The detection of being a semiroot or the actual root, and merging of disconnected trees in case of being a semiroot, is done by the P-PS(X, B) procedure given in Table 4. In the second phase of SF-DeviL, at least one device runs the Semiroot procedure.

Through exchanging messages with the tree members, the semiroot discovers if there exists any *unconnected but heard node*, that appears in one of the neighbor_lists of the descendants but is not connected to the tree. At line 3 of P-PS procedure, X either learns that it is the actual root (in which case SF-DeviL terminates), or X gets the list of all unconnected but heard devices so that it can initiate paging procedures to connect to these disconected nodes/trees. If any unconnected node exists, X starts alternating between page and page scan modes, paging all the unconnected nodes with DG ≥ DG(X) (line 5 of P-PS procedure). If the unconnected nodes have DG < DG(X), X only does page scanning. This is done, to ensure that other semiroots are paged rather than their members (since tree descendants have smaller DG).

Each time X connects to any of the other semiroots, it executes the ArrangeRoles procedure. At the conclusion of

Table 1. Main procedure of SF-DeviL

```
MAIN:
Phase 1:
1  Upon initialization, calculate DG(X)
2  counter ● discTO
3  while counter > 0 and no neighbor discovered yet
4        Alternate between inquiry and inquiry scan modes
5        if a device Y is discovered
6           then establish link to Y
7                 ArrangeRoles(X,Y)
8                 counter ← discTO
Phase 2:
9  if X has no master
10    then execute Semiroot(X)
11    else exit
```

Table 2. ArrangeRoles procedure

```
ArrangeRoles( device X, device Y)
1  Add BD_ADDR(Y).DG(Y).RSSG(Y) to neighbor_list(X)
2  if (Y==BestMaster(current master of X, Y) )
3     then delete link to the current master
4           if X is the inquirer
5              then do master/slave switch
6     else request Y to disconnect from itself
```

Table 3. Semiroot procedure

```
Semiroot (device X):
1  P-PS (X, true)
2  if any unconnected node exists
3     then if X has no master
4              then for each descendant D of X,
                       execute P-PS (D, false)
5           exit
6     else exit
```

Table 4. P-PS procedure

```
P-PS (device X, boolean B):
1  timer ←pagingTO
2  while timer > 0
3     Check if any unconnected but heard node exists
4     if any unconnected node exists
5        then alternate between page and page scan
6              if connection established with any Y
7                 then if (B == true)
8                         then ArrangeRoles(X,Y)
9                 else ReverseLinks(X,Y)
10                      timer ← pagingTO
11       else exit
```

Table 5. ReverseLinks procedure

```
ReverseLinks (device X, device Y):
1  if (Y==BestMaster(X, Y))
2     then Y becomes the master of X
3          do master/slave switch at nodes
           from X to semiroot of X
4     else X becomes the master of Y
5          do master/slave switch from Y to semiroot of Y
```

ArrangeRoles, X decides whether it will keep the link and if so which node will become the master. After each merging of trees, X goes to line 2 and 3 of P-PS procedure, checking unconnected nodes, since the new link may have brought new unconnected but heard devices.

X may not be able to establish a link with any of the paged devices and may not be paged for a specific paging timeout (pagingTO), meaning that *X is not in the communication range of the seeked devices* or *the once heard devices are gone*. In this case, if X has a master (X may have a master after merging trees), it gives up Semiroot procedure and reports the unconnected devices to the semiroot of its tree and exits. If X has no master, it orders its descendants to page the unconnected but heard devices (line 4 of Semiroot procedure). This is done to provide multihop connectivity among disconnected trees, the semiroots of which are not in communication range of each other. All the members of the tree page these devices in cooperation (also do PS alternatively) for pagingTO. If an unconnected device is detected by a member of the tree, a connection is established through that member with the newly discovered node by using the ReverseLinks procedure given in Table 5. The slave, s, of this connection, determined by BestMaster selection, requests the semiroot for master/slave switching at intermediate nodes from the semiroot of s upto itself. If no device is found, X declares itself as the root and SF-DeviL scatternet formation terminates. The proof that SF-DeviL always generates connected scatternets is given in [5].

D. Topology Maintenance in SF-DeviL

SF-DeviL handles topology maintenance by reconfiguring the scatternet topology when battery levels are depleted. If BatteryLevel of a device (except leaf nodes) reaches a threshold value, a scatternet update request is sent to the root. The root orders all members to re-calculate their DGs and collects the updated DG information from all descendants. The root sends a packet, which includes BD_ADDR and DG of all tree members, to its descendants. Upon receiving this packet, each tree member starts paging devices with higher or equal DG and enters page scan mode alternatively. This way the devices with decreasing battery levels are pushed downward towards the leaf positions in the tree to increase their battery lifetimes.

IV. SIMULATION RESULTS

A C++ based simulator based on the Bluetooth specifications [2] is developed in order to evaluate the performance of SF-DeviL. The lifetime, number of piconets, and formation delay of SF-DeviL and LMS scatternets are compared. The effects of changing discTO on SF-DeviL performance are also investigated. Two different networking scenarios are considered: a network with identical devices (corresponding to a homogeneous sensor network) and a network with devices of different classes (corresponding to multiple PANs or a heterogeneous sensor network).

In the simulations, nodes are randomly distributed in an area of 10mx10m. Although SF-DeviL supports multi-hop operation, nodes are positioned such that all nodes can communicate with each other since this is required by the reference LMS algorithm. For a given number of nodes, the averages of the results for five randomly generated node locations and traffic patterns are reported.

The following classes of devices are used: laptops, mobile phones, PDAs, headsets, peripherals and sensors. The devices are initially assigned with full batteries. At each node, traffic is generated randomly with a rate proportional to the TrafficGenerationGrade that is assigned to each device based on the kind of traffic it generates.

Power consumed for transmission/reception at each slot is taken as $P_{transmit}$ for transmission and $P_{receive}$ for reception. Power consumed in standby mode is ignored. Based on the specifications of Bluetooth chips currently in the market, the maximum transmit power and $P_{receive}$ are assumed to be equal. Power control is done at each node assuming a receiver sensitivity of -60dBm. $P_{transmit}$ can be reduced by at most 30% by the power control. The following path loss model is used:

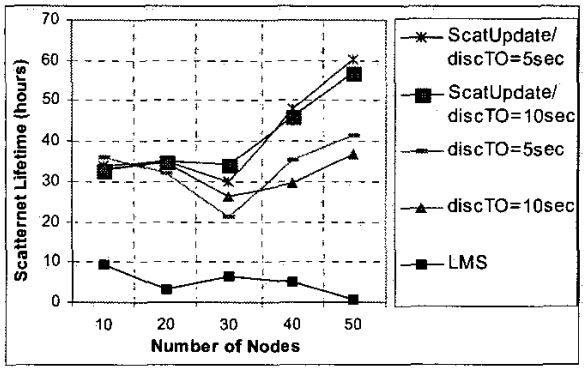
$$PL(d) = PL(d_0) + 10\,\gamma \log(d/d_0) + X_\sigma ,$$

where PL(d) denotes the path loss, in dB, for a path of length d, $PL(d_0=1m)=30dB$, $\gamma=2.5$, $X_c=N(0,\sigma)$ with $\sigma=5dB$.

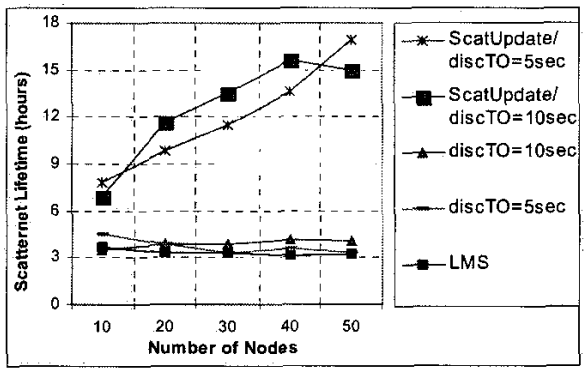
We assume that nodes are fixed, and topology is reconfigured only in response to battery level depletions. Each device, other than leaf nodes, initiates a scatternet update when its battery is halved, i.e., BatteryLevel $\leq$½.

The average scatternet lifetimes, as a function of the network size, are given in Fig. 2 for scatternets consisting of different device classes and all sensors, respectively. Different values of discTO are used for SF-DeviL in order to analyze the trade-off between optimality of the topology and formation delay. For different device types, the lifetime is increased substantially with and without topology maintenance with respect to the LMS algorithm. When all devices are sensors, the lifetime is increased by up to 32% without, and by 86-443% with the topology maintenance. The effect of maintenance is more pronounced in the homogeneous sensor network since batteries of some nodes deplete very rapidly. These results show that SF-DeviL increases lifetime for both cases, but more when different device classes exist. It is observed that batteries of leaf nodes deplete first for different device classes, since devices with higher DGs are assigned as the root and bridge nodes. Thus scatternet updates in response to decreasing battery levels may not significantly increase the lifetime until the network becomes larger so that the nodes in the upper layers of the tree topology are heavily loaded with traffic.

Average lifetimes of SF-DeviL scatternets for different values of discTO exhibit similar behavior. Large discTO means longer I/IS intervals and more battery dissipation during discovery, which lessens the lifetime for some cases. On the other hand, with a small discTO, e.g., discTO<5 sec, a smaller fraction of neighboring devices can be discovered



a)



b)

Fig. 2. Scatternet lifetime when devices are of: a) different types, b) same type

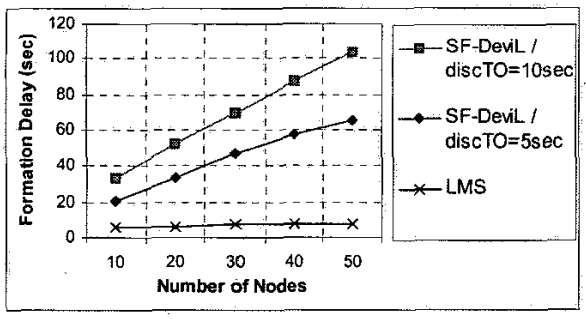


Fig. 3. Number of piconets

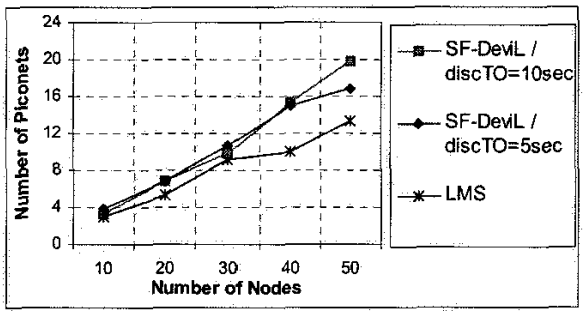


Fig. 4. Scatternet formation delay

and a connected scatternet topology cannot always be formed. Simulations show that with discTO=5 sec, about half of the neighbors are discovered, whereas with discTO=10 sec almost all neighbors can be discovered.

SF-DeviL, unlike LMS, does not have the explicit goal of forming scatternets with small number of piconets. As shown in Fig. 3, the number of piconets with LMS is smaller than SF-DeviL, and there is not a significant difference between different values of discTO for SF-DeviL. In Fig. 4, the formation delay for SF-DeviL is shown as a function of the network size for different values of discTO. We observe that the formation of the scatternet takes longer than LMS. The connection delay for SF-DeviL increases with the network size due to the increase of number of discovered neighbors. Increasing discTO increases connection delay, thus there is a trade-off between formation delay and discovering more neighbors. Simulations show that discTO=5 sec provides a good compromise between these two trends.

V. CONCLUSION

SF-DeviL is a Bluetooth scatternet formation and maintenance protocol that is optimized for low-power consumption in multi-hop wireless networks. SF-DeviL reconfigures the topology in response to depleting battery levels. It produces scatternets where a node participates in just two piconets so that bridge nodes do not become bottlenecks between multiple piconets.

Simulations are used to show that using class and link characteristics during scatternet formation, network lifetimes are substantially prolonged. SF-DeviL forms topologies with number of piconets close to the minimum within reasonable formation delays.

REFERENCES

[1] C. Law, A. K. Mehta, K.-Y. Siu, "A New Bluetooth Scatternet Formation Protocol," Mobile Networks and Applications, vol. 8, no. 5, pp. 485-498, 2003.

[2] Bluetooth SIG, "Specification of the Bluetooth System," Version 1.1, http://www.bluetooth.com.

[3] T. Salonidis, P. Bhagwat, L. Tassiulas, R. LaMaire, "Distributed topology construction of Bluetooth personal area networks," INFOCOM'01, Anchorage, Alaska, pp. 1577-1586, April 2001.

[4] G. Tan, A. Miu, J. Guttag and H. Balakrishnan, "An Efficient Scatternet Formation Algorithm for Dynamic Environments," CCN'02, Cambridge, November 2002.

[5] C. Pamuk, E. Karasan, "A Tree-Based Energy-Efficient Distributed Algorithm for Forming Bluetooth Scatternet Topologies," Med-Hoc-Net'04, Bodrum, Turkey, June 2004.

[6] G. V. Zaruba, S. Basagni, I. Chlamtac, "Bluetrees-Scatternet Formation to Enable Bluetooth Based Ad Hoc Networks," ICC'2001, pp. 273–277, Helsinki, Finland, June 2001.

[7] C. Petrioli, S. Basagni, I. Chlamtac, "BlueMesh: Degree-Constraint Multi-Hop Scatternet Formation for Bluetooth Networks," Mobile Networks and Applications, vol. 9, no. 1, pp. 33-47, 2004.

[8] B.J. Prabhu, A. Chockalingam, "A Routing Protocol and Energy Efficient Techniques in Bluetooth Scatternets," ICC'02, pp. 3336-3340, New York, 2002.