Real-Life Deployment of Bluetooth Scatternets for Wireless Sensor Networks

Michael Methfessel[1], Stefan Lange[1], Rolf Kraemer[1], Mario Zessack[2], Peter Kollermann[3] and Steffen Peter[4]

[1] IHP, Im Technologiepark 25, 15236 Frankfurt(Oder), Germany
[2] lesswire AG, Rudower Chaussee 30, 12489 Berlin, Germany
[3] AE REFUsol GmbH, Uracher Strasse 91, 72555 Metzingen, Germany
[4] Center for Embedded Computer Systems, University of California, Irvine, USA

Abstract. Bluetooth scatternets are constructed from overlapping piconets, allowing any number of nodes to be connected into a multi-hop wireless network. Although the topic has been researched for 15 years, no deployments of self-organized scatternets have been published. Recently we have presented the SFX algorithm, which was implemented on commercial Bluetooth nodes and is an extension of SHAPER from 2003. Here measurements are presented for scatternet trees for a laboratory network of 24 nodes and for deployment in a photovoltaic power plant with 39 nodes. The results demonstrate the effectiveness of the SFX algorithm, which is evidently the only actually implemented scatternet contruction procedure which is distributed and does not assume full node-to-node visibility.

Keywords: Bluetooth, scatternet, wireless sensor network, simulation

1 Introduction and Related Work

Developed in the 1990's, Bluetooth [1] is widely used for short-distance point-to-point wireless links. The Bluetooth standard also introduced the concept of a scatternet. A scatternet is a collection of Bluetooth piconets, each containing up to eight nodes, forming a multi-hop network. Bluetooth is a cheap, robust, and verified technology. Therefore scatternets are good candidates for wireless sensor networks for industrial control and surveillance and similar systems.

Interestingly, the Bluetooth standard left open the procedure by which a scatternet is set up. This has lead to intense research and a number of distinct proposals, in particular for the most useful case: a distributed, self-organizing and self-healing algorithm which does not assume full node-to-node visibility. However, to the best of our knowledge, none of these have made it to a working implementation on actual Bluetooth hardware. Consequently, although scatternets with a fixed predefined topology were used in some applications, self-organized scatternets have not been used in any practical applications to date.

The authors have recently presented a distributed algorithm to build a scatternet with the desired self-organizing, self-healing, and self-optimizing properties, which does not assume full node-to-node visibility. This SFX procedure is

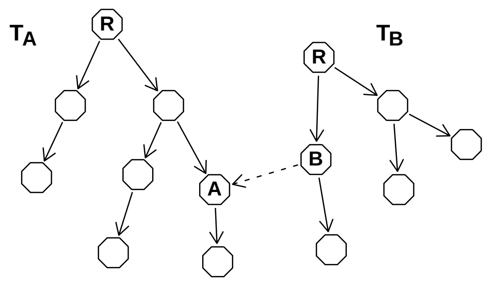


Fig. 1. The basic merge procedure as introduced by SHAPER. Arrows point from master to slave and R labels root nodes. Trees T_A and T_B will merge by building up a new connection (dashed) with node B as master and node A as slave. Prior to connecting, the root of T_A is moved to node A by a sequence of role switches.

based on the previously presented SHAPER algorithm. The basic idea (to move the node of one tree to a suitable position by a series of role switches prior to merging, see Fig. 1) is also used in the SFX algorithm. However, while working on a real-world implementation of SHAPER, key points arose for which solutions were still required. These concern (1) the collection of information about neighboring nodes, (2) overall steering of the algorithm, (3) locking and (4) tree optimization. Solutions were found for these issues and presented in Ref [2].

In this paper, results for tree construction in a laboratory test network of 24 nodes are presented. Furthermore, as a realistic application the SFX algorithm was deployed in photovoltaic power plant, where it has been functioning reliably for several months. The purpose of the installation is to collect diagnostic data from the inverters without need for a cable-based Ethernet network. The measured data demonstrates that the SFX procedure indeed reliably builds, optimizes, and maintains a tree with the required properties.

Comparing to previously proposed scatternet formation algorithms [7], these are generally mesh-based [8] or tree-based, with some discussion of star and ring topologies [9]. A mesh offers redundancy which can quickly compensate for disrupted links, while a tree must be repaired to regain connectivity. On the other hand, a mesh requires extra effort for routing which is avoided in a tree. Real-world considerations argue against some of the proposed procedures. Centralized approaches such as BTCP [10] or also MMPI [11] can construct an optimal piconet structure, but must collect and distribute information from a central site. When all nodes of the system can communicate with each other, elegant techniques such as TSF [4] or the closely similar TreeNet [5] can be used. However, full node-to-node visibility is not given for most actual systems.

The most realistic approach seems to be SHAPER [3], on which the present work is based. In the following, Section 2 briefly characterizes the extensions added by the SFX algorithm. Results for the laboratory test network and the photovoltaic power plant are presented in Sections 2 and 3, respectively, while Section 4 contains the conclusions.

2 Main Points of the SFX Algorithm

Figure 1 illustrates the basic SHAPER procedure to merge two trees. A sequence of role switches is applied to the links between the left-handed root node and A. The result is that A becomes root and is not slave in any piconet. This node is therefore free to become slave in the new connection.

SFX introduced four extensions to obtain a procedure which works reliably and effectively for real-world Bluetooth systems (see [2] for details):

Collecting information about the neighborhood: A central task is to acquire information about other nodes in the neighborhood. The usual Bluetooth procedure (used by SHAPER) is to find a node using inquiry, build a connection, and exchange information over this connection. This turns out to be prohibitively inefficient in practice. SFX packages the relevant information as user-defined data in an extended inquiry response. By this simple device, a node has a full overview of all neighboring nodes and the quality of links to them after each inquiry phase. Effectively, each node is broadcasting its local topology to all nodes in the neighborhood.

Steering the algorithm: When two trees merge, SHAPER reconfigures the smaller tree. Thus each node must know the total number of nodes in its current tree. When multiple distributed merge operations are done, it is a major task to maintain the tree cardinality consistently. Ref. [3] does not adress the issue but presents simulation results; perhaps these simulations were on a higher level which assumes that the cardinality is known *a priori*. In a real-life situation, this sub-task has a similar complexity as the whole tree-building procedure.

SFX uses the following approach to enable a distributed algorithm. First, by means of an asymmetric merge procedure with one "active" and one "passive" tree, any number of merges can be done into the same target tree simultaneously. Secondly, merge operations are only permitted if the active tree can lower its tree identifier by merging. This avoids loop formation by simultaneous antiparallel merges betwen two trees. Furthermore, when a node merges into the ever-growing final tree (with the globally lowest tree identifier) it will be passive in subsequent merges. Therefore the tree which reconfigures is usually the smaller tree.

Tree locking: Antiparallel merging is avoided as described above, but it must still be ensured that each tree does only one active merge at a time. Ref. [3] suggests a solution, which however requires modification in order to cope with delays and possible race conditions arising when actual Bluetooth nodes are used. Details can be found in [2].

Tree optimization: To optimize the tradeoff between tree depth and link quality, SFX introduced two procedures, whereby links are compared according to the measure $X_k = P_k + \beta D_k$ where P_k is the path loss for a link to node k and D_k is depth the node would have when connected via this link. First, after each inquiry phase a node inspects possible links within the same tree and switches its position if X can reduced. Second, a whole subtree can disconnect, "flip" by switching master and slave roles betwen the old and new bridge nodes, and reconnect. This procedure is quite similar to basic tree merging and is required in cases where the simple optimization cannot improve the situation.

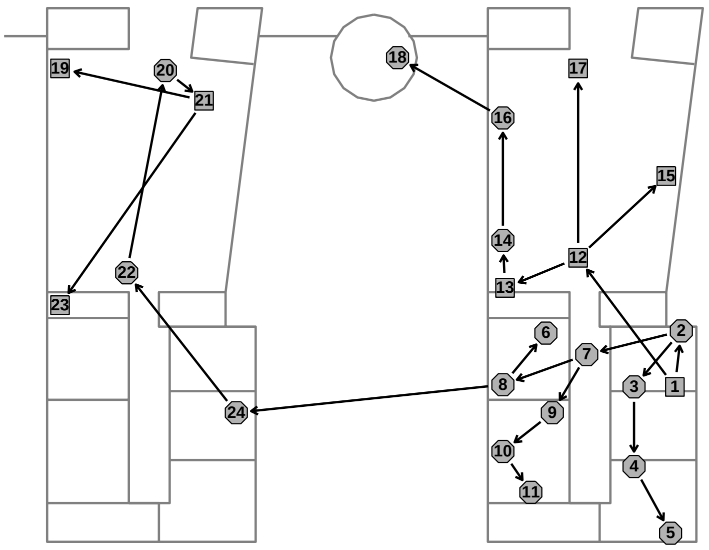


Fig. 2. Sketch of the in-house test network with 24 Bluetooth nodes arranged in two adjacent wings (separation 20 m) over two floors (squares above, circles below). The arrows show a typical stable tree. The root is at node 1.

3 Laboratory Tests

As a test, 24 nodes were distributed over two wings of the IHP building (Fig. 2). The nodes consist of a Bluetooth Class 1 module [1] and a μC unit running a Linux OS (including the full TCP protocol stack) and the SFX algorithm. By means of a dedicated power amplifier and low noise amplifier, distances up to 500 m can be bridged with 6 cm dipole antennas [12]. All nodes were connected to an Ethernet backbone for easy control and monitoring. Nodes typically find 2 to 4 neighbors with a path loss below 85 dB within a wing. Links between the wings are weaker due to a metallic coating on external windows, with a path loss of ∼100 dB from $N8$ to $N24$.

Repeated runs were done to study the performance during tree formation. The nodes booted at random in a ten-second interval, requiring another 18 s to boot. Starting with pairwise connections, nodes coalesce into trees. When a node joins the gateway tree containing $N1$, it optimizes its position within the tree according to the tradeoff between path loss and depth.

In the right-hand wing, most nodes can build robust connections to several nodes in their neighborhood. Therefore, the optimization procedure has enough degrees of freedom to reduce tree depth without sacrificing link quality. As a result, trees tend to look quite different in the right wing but the depth here is never greater than four. In the left-hand wing, links are weaker and the same local tree is constructed in almost all most cases. The link between the wings settles down to $N8 \rightarrow N24$ in the majority of cases.

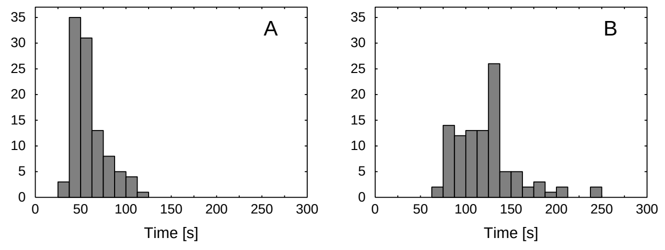


Fig. 3. Histogram showing the elapsed time until connectivity is reached (A) and until the tree is stable (B) for 100 runs of the 24-node IHP test network.

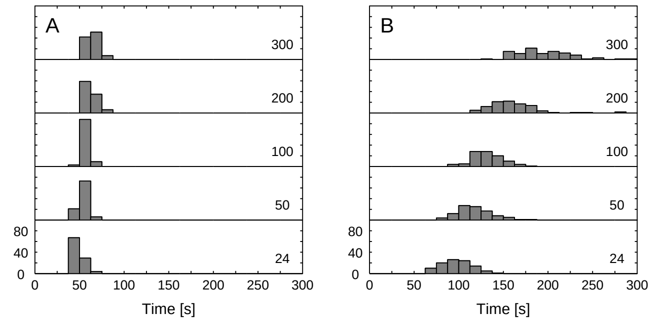


Fig. 4. Histogram showing the elapsed time until connectivity is reached (A) and until the tree is stable (B) for 100 simulation runs each for 24, 50, 100, 200, and 300 nodes.

To characterize the time required to set up the tree, Fig. 3 summarizes the time until all nodes have joined the gateway tree for 100 runs. The time includes the boot delay and lies at around 50 to 100 seconds. Histogram B shows the time until optimization is completed. Stability was defined as the start of the first 60-second period in which the tree remained static. The results show that two to three minutes are adequate to build up an optimized tree.

As described in Ref. [2], development was done in parallel on the actual hardware and in a simulation environment, both running the identical code. These simulations can now be used to study the behavior for larger systems. Nodes were spread randomly over a rectangular area, with parameters similar to those in the test network. The simulation results are shown in Fig. 4. We included the case for 24 nodes to compare with the measurements. Agreement is satisfactory with slightly shorter delays since the simulation uses a static channel model and disregarded the fact that inquiry responses are often corrupt in practice. The simulations show that the SFX algorithm works effectively for node counts in the hundreds. The times to reach connectivity and stability increase somewhat with the node count. The effect is larger for stability since optimization becomes more complex for large systems. In contrast, connectivity is still reached quickly, since merges are performed in all parts of the system in parallel.

4 Deployment in a Photovoltaic Power Plant

As a real-life test, 39 nodes were attached to the inverters of a photovoltaic power plant as shown in Fig. 5. The main issue was to test the effectiveness of the scatternet build-up and the robustness of data transfer under realistic conditions. Previous measurements had shown that radio transmission is made difficult by reflections from the rows of metal struts and from the need to avoid shadows on the photovoltaic modules. The nodes were mounted behind the upper edges of the modules. After finding poor results using small PCB-mounted ceramic antennas, good performance was obtained using 20 cm antennas which protruded above the edge. Information about the system can be found in Ref. [13].

A new tree is built up each morning as the nodes boot. Monitoring this process over six months, a correct tree was constructed in each case, except for several days when snow covered the solar modules. A typical generated tree is shown in Fig. 5. Usually the tree depth was 9 or 10 but depths down to 6 were also observed. Links over distances of 300 m or more occur. The pathloss for any link did not exceed 95 dB, the threshold to trigger subtree optimization. Thus, the algorithm is performing in accordance with the chosen parameter values. Typically, nodes boot within 2 minutes of each other. The tree is stable about one minute after the last node is activated. After that, optimization modifies the tree from time to time (Fig. 6) with a brief loss of connectivity. Since the nodes are located far apart, the algorithm was configured to search for optimal connections aggressively. Reconfigurations are triggered by changes in the transmission properties, *e.g.,* due to weather conditions. In scenarios with shorter distances, parameters can be chosen to reduce the number of reconfigurations.

Measurements were done to characterize the data transfer in the scatternet. Fig. 7 shows the ping times and the times required to transfer data via TCP.

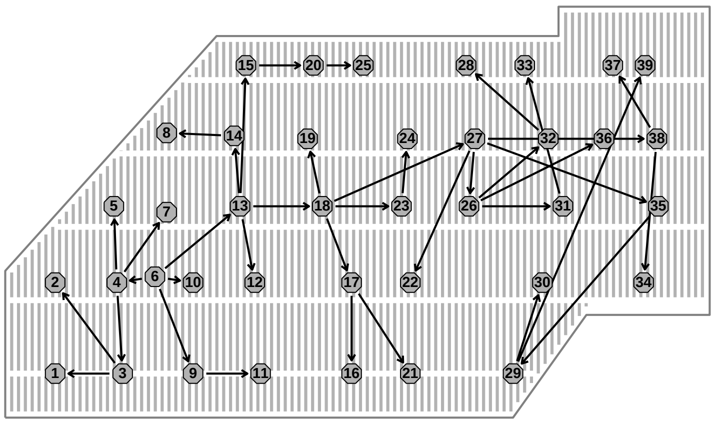


Fig. 5. Map of the photovoltaic power plant, showing the positions of the 39 nodes and a typical tree generated by the SFX procedure. The root is node 6. The field is 1300 by 550 meters with 102 rows of solar modules (courtesy of GP JOULE).

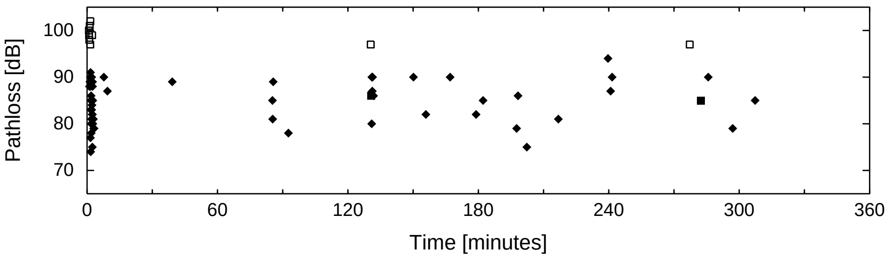
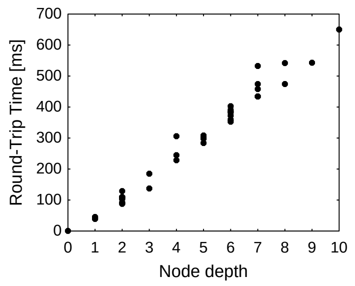


Fig. 6. Tree-maintenance events for six hours after the scatternet boots (diamonds: immediate optimization; open squares: subtree optimization requests; filled squares: subtree optimization). Values show the path loss associated with the event.



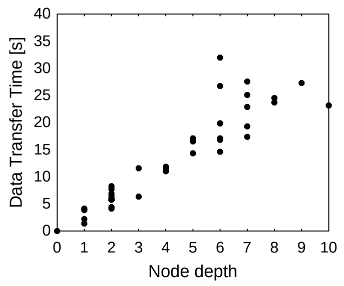


Fig. 7. Round-trip ping times (left) and time needed to transfer one Mbit of data (right) between the root and some other node for the power plant deployment.

Both tests were done one node at a time, avoiding issues of competing data streams through a bottleneck. The results are displayed as function of the depth in the tree of the target node. The evident linear behavior shows that good links were built up between all node pairs, which in the end is the main purpose of the procedure. The ping round-trip times show a delay of 30 ms per hop, as was also observed in the laboratory. A large part of the delay is due to the protocol handling in the Linux operating system and the Bluetooth stack.

5 Conclusions

Bluetooth scatternets have been the subject of research for many years. Ideally, a scatternet should set itself up by a distributed algorithm which does not assume full node-to-node visibility. The scatternet should automatically optimize and respond suitably when nodes enter or fail or when link properties change.

Of the previously proposed distributed algorithms for scatternet set-up and maintenance, none have made it to a deployable system. In this paper, results were presented for the SFX algorithm, which constructs a scatternet tree in accordance with the requirements above and was implemented on commercially available Bluetooth modules. Starting from the basic idea of the SHAPER algorithm, extensions were added to obtain a real-life implementation. Simulations, laboratory tests, and deployment have demonstrated good performance:

– Tree build-up is fast, creating a stable and optimized tree in a few minutes.
– Measurements show a balanced tree with good links along all connections.
– Realistic simulations for hundreds of nodes demonstrate a good scalability.

To our knowledge, these successful results are the only study of self-organized large-scale Bluetooth scatternets in a realistic deployment to date.

Acknowledgements

This work was supported by the German Bundesministerium für Bildung und Forschung (grants KF2123403DF9/KF231001DF9). We thank GP JOULE for access to the photovoltaic power plant.

References

1. Bluetooth SIG: Bluetooth specification v3.0 (2009)
   Networking. pp. 271–278. MobiCom '99 (1999)
2. Methfessel, M., Peter, S., Lange, S.: Bluetooth scatternet tree formation for wireless sensor networks. Mobile Ad-Hoc and Sensor Systems, IEEE International Conference on pp. 789–794 (2011)
3. Cuomo, F., Di Bacco, G., Melodia, T.: Shaper: a self-healing algorithm producing multi-hop bluetooth scatternets. In: IEEE GLOBECOM'03 (2003)
4. Tan, G., Miu, A., Guttag, J., Balakrishnan, H.: Forming scatternets from bluetooth personal area networks. Massachusetts Institute of Techonology, `http://lcs.mit.edu/`, Tech. Rep. MIT-LCS-TR-826 (2001)
5. Beutel, J.: Robust topology formation using btnodes. Computer Communications 28(13), 1523–1530 (2005)
6. Bello, L.L., Mirabella, O.: Communication techniques and architectures for bluetooth networks in industrial scenarios. In: 10th IEEE Conference on Emerging Technologies and Factory Automation (ETFA). IEEE (2005)
7. Whitaker, R., Hodge, L., Chlamtac, I.: Bluetooth scatternet formation: a survey. Ad Hoc Networks 3(4), 403–450 (2005)
8. Yu, C.M., Lin, J.H.: Enhanced bluetree: a mesh topology approach forming bluetooth scatternet. IET Wireless Sensor Systems 2(4), 409–415 (2012)
9. Al-Kassem, I., Sharafeddine, S., Dawy, Z.: Bluehrt: hybrid ring tree scatternet formation in bluetooth networks. In: Computers and Communications, 2009. ISCC 2009. IEEE Symposium on. pp. 165–169. IEEE (2009)
10. Salonidis, T., Bhagwat, P., Tassiulas, L., LaMaire, R.: Distributed topology construction of bluetooth personal area networks. In: IEEE INFOCOM 2001 (2001)
11. Donegan, B.J., Doolan, D.C., Tabirca, S.: Mobile message passing using a scatternet framework. International Journal of Computers, Communications & Control 3(1), 51–59 (2008)
12. BlueBear – industrielles long range HCI-Bluetooth 2.0 Modul mit EDR, Data sheet, lesswire AG, Berlin (2009)
13. `http://europe.refusol.com/en/accessories/refuconnect/`