SHAPER: A Self-Healing Algorithm Producing multi-hop Bluetooth scattERnets

Francesca Cuomo, Guido Di Bacco, Tommaso Melodia
University of Rome "La Sapienza", Via Eudossiana 18, 00184 Rome, ITALY

Abstract— **This paper deals with *scatternet* formation in Bluetooth. A scatternet is an ad hoc network of Bluetooth devices. Some works in the literature rely on the *single-hop* hypothesis, i.e., all devices are in radio visibility of each other. Other works refer to the more likely circumstance that devices are scattered in an area where some of them can not directly communicate. A challenging issue in this latter scenario (often referred to as *multi-hop*) is the design of a formation algorithm that: i) operates in a distributed way; ii) dynamically adapts the topology to the mobility of devices; iii) forms a scatternet with given topological properties. In this paper a distributed algorithm for scatternet formation that gives rise to a tree-like structure is introduced. The algorithm is shown to present three key properties that make it innovative with respect to the literature in the field: i) it is fully distributed and asynchronous; ii) it can be applied in a multi-hop environment; iii) it operates in order to dynamically adapt the topology to nodes' mobility and failures. The key steps and rules of the algorithm are described and performance results obtained by simulation are discussed.**

I. Introduction

Bluetooth (BT) is a promising technology for ad-hoc networking. Eight BT devices are interconnected in a *piconet* and share the same radio channel which supports about 1 Mbit/s [1]. Class 3 devices have a transmission range of about 10 meters. Networks without infrastructure can be realized by means of self-organization and peer to peer communication. In a piconet two BT devices (also *nodes* in the following) exchange information by means of a master-slave relationship. Master and slave roles are dynamic: the device that starts the communication is the master, the other one is the slave. A master can connect with up to 7 slaves. Multiple access in a piconet is centrally regulated by the master that adopts a polling scheme on a slotted time structure. Piconets can coexist in the same area and interconnect in a *scatternet*. A device that joins more than one piconet is named gateway and participates to communications in different piconets on a time-division basis. Each device can be master in only one piconet.

This paper describes SHAPER, a Self-Healing Algorithm Producing multi-hop Bluetooth scattERnets, and is organized as follows. Section II briefly describes works related to the scatternet formation and discusses properties of SHAPER. Section III presents the key rules and steps that constitute the SHAPER mechanism, while Section IV discusses numerical results obtained by simulations. Section V concludes the paper.

II. Bluetooth Scatternet formation

Scatternet formation in BT has recently received a significant consideration [2]. Existing solutions can be classified as single-hop ([3][4][5]) and multi-hop ([6][7][8][9]). Single-hop solutions assume that all nodes are in radio visibility of each other. Multi-hop solutions work also when this hypothesis does not hold. Paper [3] addresses BT scatternet formation with a distributed selection of a leader device which assigns roles to the others. In [4] a distributed formation protocol is defined, with the goal of reducing formation time and message complexity. Both [5] and [6] form tree-shaped scatternets. Tan *et al.* introduce in [5] the TSF (*Tree Scatternet Formation*) protocol; the topology produced is a collection of one or more rooted spanning trees, each autonomously attempting to merge with the others and to converge to a unique tree. TSF assures connectivity only in single-hop scenarios since trees merge only via root nodes; thus, two different trees can merge only if their *root* nodes are in the transmission range of each other. Zaruba *et al.* propose a protocol able to operate also in a multi-hop setting [6]. This latter protocol is based on a process initiated by a unique node (named *blueroot*) and repeated recursively till the leaves of the tree are reached. In order to operate in a distributed way, and to avoid deadlocks, the algorithm is based on time-outs that could affect the overall formation time.

A second class of multi-hop proposals consists of algorithms that produce connected scatternet by exploiting clustering schemes for ad-hoc networks. In [7] and [8] the *BlueStars* and *BlueMesh* protocols are described, respectively. These protocols define rules for device discovery, piconet formation and piconet interconnection so as to achieve suitable properties for the scatternet formed. The generated scatternet is a mesh with multiple paths between any pair of nodes. *BlueMesh* allows each master to select at most 7 slaves. Also [9] defines a protocol that limits to 7 the number of slaves per master by applying degree reduction techniques to the network topology graph. The proposed algorithm assumes that each node knows its position and that of its neighbors.

Finally, we mention a third class of works which concentrate on scatternet topology optimization. This issue is faced for the first time in [10] and [12] by adopting centralized approaches. In [10] the aim is minimizing the load of the most congested node in the network, while [12] discusses the impact of different metrics on the scatternet topology. A distributed approach based on simple heuristics is presented in [11].

As in [5] and [6] our algorithm, named SHAPER, forms a scatternet with a *tree-like* topology. To the best of our knowledge, SHAPER is the first multi-hop algorithm which guarantees a *self-healing* behavior: i.e., it is able to dynamically reconfigure the scatternet after topological variations due to mobility or failure of nodes. The deployment of scheduling algorithms and ad hoc routing schemes becomes easy on a tree topology, as also noted in [5]. Furthermore, the reconfiguration of the network can be obtained by merging different trees. Finally, the distribution of network control information is straightforward. The latter feature deserves

some discussion. Existing scatternet formation algorithms are not specifically concerned with the nature of the information to be exchanged. A tree topology may not be the best choice to transfer user information since there is a single path between any couple of nodes. Some of them can easily become bottlenecks. However, it is needed that nodes self-organize to guarantee a broadcast-like facility as quickly as possible. This broadcast segment can be used to efficiently disseminate control and management information in the network to eventually obtain a more suitable topology. A tree topology may be very convenient to this aim, if two key conditions are met: i) the network is easily reconfigurable after nodes have joined or left the system; ii) the scatternet formation or reconfiguration delay is within a reasonable amount of time. In particular, we have already deployed algorithms that exploit the tree as a platform to transport control information. This information is used to reconfigure and optimize the topology in accordance to user needs expressed by performance metrics. An example of such metrics can be found in [12].

III. THE SHAPER ALGORITHM

The SHAPER algorithm here proposed is based on distributed procedures which run on every network node. Two main phases can be outlined:

1) device discovery and communication;
2) tree formation.

The first phase is dedicated to device discovery and to establishing links that are included in a tree in a second phase. In the second phase, trees that have separately grown are also merged. The network converges to a unique connected tree always and only when the adjacency graph is connected. For every link in the tree, the master is always the parent and the child is the slave. All nodes continuously alternate between phases 1) and 2); this, as will be explained in the following, assures a self-healing behavior.

A. Device discovery and communication

This phase has been designed following the approach proposed in [5]. The status of a SHAPER node can be: *free*, *root* or *non-root*. A node is free (F) when it switches on or when, being in a tree as a *leaf*, it loses its parent. A node already included in a tree is in the *root* (R) or *non-root* (NR) status. Depending on its status, a node moves among five possible *states*: Inquiry (INQ), InquiryScan (INQSC), Page (PG), PageScan (PGSC) and Communication (COMM) (see Figure 1).

At the beginning, an F node can only be in the INQ or INQSC states. In the INQ state the BT Inquiry procedure is performed for a time period of at most T_{inq} seconds. If during this period a node discovers another, then the two connect with the Page and PageScan procedures respectively. The persistence in each of these two states is regulated by timers (T_{pg} and T_{pgsc}). If the Page/PageScan procedures succeed, a master-slave relationship between the two nodes is activated, a link is established, and the two nodes enter the TreeFormation procedure. If the Page/PageScan procedures fail, both nodes randomly enter a new INQ or INQSC. The behavior of a node in INQSC is specular and adopts a T_{inqsc} timeout. After TreeFormation, nodes become R or NR and, on the basis of a random choice, either go back to the INQ or INQSC state, or they enter the COMM state where are involved only in data communications and not in scatternet formation. Data are exchanged for a randomly extracted time period (T_{comm}, uniformly distributed between T_{min} and T_{max} seconds). After this period a node, with a given probability, re-enters the INQ or INQSC states.

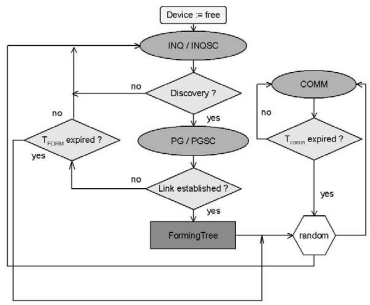

Fig. 1. Main flow chart

B. Tree Formation

The TreeFormation procedure represents the core of the SHAPER algorithm and includes the innovative aspects of the approach. The goals of this procedure are: i) inserting free nodes in a tree; ii) merging, by avoiding loops, trees that have formed in previous iterations; iii) guaranteeing self-healing behaviour.

A non-free node (node i) belongs to the tree T_i; during the TreeFormation it stores and up-dates the following parameters:

1) TREE_ID$_i$, i.e., the BT Device ADDRess (BD_ADDR) of the root of the tree T_i; this information univocally identifies all nodes in the same tree;
2) N_i=current number of nodes in T_i;
3) N_i^{desc}= current number of descendants of node i in T_i;
4) the node's status (F, R and NR).

A node can send four different kind of messages: `Init`, `StartReconf`, `UpdateParameters`, `UpdateNDesc`. Each message includes a sub-set of the previous parameters as reported in TABLE I.

MESSAGE NAME	TREE_ID	N	N^{desc}	node status
`Init`	X	X	-	X
`StartReconf`	X	X	-	-
`UpdateParameters`	X	X	-	-
`UpdateNDesc`	X	X	X	-

TABLE I
SHAPER MESSAGES AND PARAMETERS

To describe the TreeFormation procedure, we refer to a master-slave couple that establishes a link in the initial phase of Figure 1. We call the nodes M_A and S_B. If they both are non-free nodes, then they respectively belong to the T_A and T_B trees. The link between the two is referred to as L_{AB}. After the

connection has been established, M_A sends a `Init` message to S_B. S_B verifies if M_A's TREE_ID equals its TREE_ID. In this case L_{AB} would certainly produce a loop and must be torn down. If the TREE_IDs of the two nodes are different, S_B continues the procedure. The subsequent steps depend on the status of the two nodes. Different actions are foreseen (TABLE II): we will refer to them as *merging procedures*. A pseudo-code for these procedures is given below.

```
TreeFormation
  M_A: send to S_B Init
  If (TREE_ID_{M_A} ≡ TREE_ID_{S_B})
   disconnect(M_A, S_B)
  else
   execute merging procedure
   as a function of (M_A's status, S_B's status)
```

$S_B \backslash M_A$	R	NR	F
R	A_1	A_2	A_3
NR	A_3	A_4	A_3
F	A_2	A_2	A_2

TABLE II
MERGING PROCEDURES

```
A1: if (N_{M_A} ≥ N_{S_B})
     S_B, M_A: send to children UpdateParameters
    else
     LMP_MasterSlaveSwitch (newM_A=S_B, newS_B=M_A)
     newM_A, newS_B: re-enter TreeFormation

A2: S_B: send to children UpdateParameters
    M_A: send to parent UpdateNDesc

A3: LMP_MasterSlaveSwitch (newM_A=S_B, newS_B=M_A)
    newM_A, newS_B: re-enter TreeFormation

A4: if (N_{M_A} ≤ N_{S_B})
     LMP_MasterSlaveSwitch (newM_A=S_B, newS_B=M_A)
     newM_A, newS_B: re-enter TreeFormation
    else
    M_A: send to parent UpdateNDesc
    repeat until initial root of T_B is reached
     S_B: send to F_B StartReconf;
     F_B: update N_{F_B} and TREE_ID_{F_B},
     send to children (excluding S_B) UpdateParameters;
     LMP_MasterSlaveSwitch
     (newF_B=parent of S_B, newS_B=F_B)
```

Case A_1 applies when both M_A and S_B are R. Their two trees can simply merge. The only decision regards which device will be the new root: we assume that it is always the R with the biggest N. If it is M_A, then L_{AB} maintains the initial configuration; S_B acknowledges M_A and sends to its sons a `UpdateParameters` message containing the new $TREE\_ID$ and the new N=N_{M_A}+N_{S_B}; after receiving the ACK, M_A also sends the new N value to its children through a `UpdateParameters` message. Nodes receiving a `UpdateParameters` update their N and $TREE\_ID$ parameters with the values included in the message, and propagate the same message to their children.

If $N_{M_A} \geq N_{S_B}$, L_{AB} has to be switched, i.e. M_A and S_B must exchange their roles; this is done by means of the BT LMP_MasterSlaveSwitch procedure. After the switch, the nodes execute again TreeFormation with the new roles.

In the A_2 case, the link L_{AB} needs not to be switched. S_B simply acknowledges M_A and sends the `UpdateParameters` message to its children; M_A sends a `UpdateNDesc` message to its parent. Whenever a node receives an `UpdateNDesc` message, on its turn it forwards it to its parent and sends an `UpdateParameters` message with the new value for N to its children (excluding the one that sent the `UpdateNDesc`). The `UpdateNDesc` message is thus used to update the value of N^{desc} and is always sent in the child-to-parent direction.

In A_3, nodes simply switch the link L_{AB} and re-execute TreeFormation with the new roles.

Finally, in case A_4 two NR nodes meet. One of the two trees must be reconfigured, i.e., one of the NR nodes must become root of its tree in order to correctly merge with the other tree. Again we choose to reconfigure the tree with the lowest value of N; without loss of generality we assume that the tree to be reconfigured is the one where S_B is. Otherwise the LMP_MasterSlaveSwitch is applied and TreeFormation is executed again.

The reconfiguration of the tree is obtained by recursively applying in T_B the following three steps, performed by a child-parent pair, from the node which starts the reconfiguration up to the root. The node which starts the process is S_B and its parent in T_B is referred to as F_B (see Figure 2). The node S_B sends a `StartReconf` message to its parent F_B and an `UpdateParameters` to its children. Both messages include the new N=N_{M_A}+N_{S_B} and the TREE_ID value of T_A. F_B sends an `UpdateParameters` to all of its children, excluding S_B, and forwards the `StartReconf` message to its parent. An LMP_MasterSlaveSwitch is performed after that between S_B and F_B. These steps are then repeated from F_B towards its parent and the process continues until the initial root of T_B is reached. At this moment S_B is the new root of T_B and T_A is merged with T_B. This gives rise to a unique tree whose root is the initial root of T_A. While T_B is being reconfigured, M_A updates it descendants and propagates a `UpdateNDesc` towards the root of T_A.

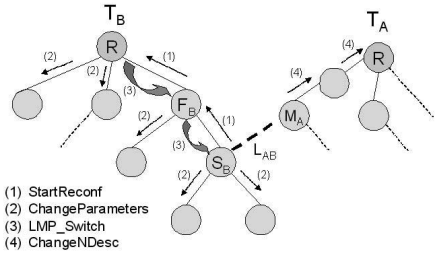

Fig. 2. Tree reconfiguration: S_B becomes the new root of T_B and T_B is included in T_A

To prevent from multiple reconfigurations happening at the same time, we introduced a simple mechanism based on request/permission and lock/unlock messages, managed by the root of the tree. A NR node asks for a permission to start the reconfiguration from the R, by sending a message towards the root. The root sends back an authorization and implicitly locks

all other nodes until the reconfiguration has ended.

In the TreeFormation, whenever a slave tries to connect to a master which already has 7 children, i.e., 7 slaves in its piconet, an adjunctive SHAPER procedure is called, which relies on a property demonstrated in [6]. If a node has more than 5 neighbors, then at least 2 of them are neighbors themselves. Thus, the master temporarily parks the entering slave and forces the set-up of a link between two of its children; then the parked slave is waken up and becomes an active child in the tree. Our simulations showed that this situation is very unlikely to happen.

C. Self-healing behavior

SHAPER guarantees a self-healing behavior of the network by rapidly including nodes entering the network and by re-configuring the scatternet topology when a node abandons the network or moves. In the former case, the incessant iteration of a node in all states of Figure 1 assures that new nodes always have the opportunity to enter the network. As for the second case, when a node loses connectivity with its parent, it assumes that it has moved or switched off and becomes a R node. As a consequence it sends a `UpdateParameters` message to all of its children. This contains its BD_ADDR in the TREE_ID field and its N^{desc} in the N field. On the contrary, when a parent node loses its child, it must update the total number of nodes in the tree. To this aim it requests the N^{desc} values from its active sons. Once N has been updated the parent sends to all its children a `UpdateParameters`.

IV. Performance Results

In this section simulation results of scatternet formation with SHAPER are presented. The algorithm was implemented on Blueware, a public ns-2 simulator developed by MIT's researchers ([13]). We measured the behavior of the protocol in both single-hop and multi-hop scenarios. In the first case we randomly distributed nodes in a square area of side 7 m, thus assuring that all devices are in radio visibility. In this case we also compared our results with [5]. In the multi-hop case we scattered nodes so as to obtain a mean visibility per node around 30% of the total number of generated nodes. Since Blueware does not implement an interference model, results on the formation time must be considered as lower bounds when the nodes' density is high.

As for the parameters adopted in the procedures of Figure 1, we set: T_{inq}=10.24 s, T_{inqsc}=1.28 s, T_{pg}=1.28 s, $T_{pgsc} \simeq 1$ s, T_{min}=2.125 s and T_{max}=5.95 s. The two parameters related to T_{comm} has been selected in order to give a higher weight to the time spent in the discovery states rather than in the communication state. The time spent in the COMM state can be adaptively increased when the network topology becomes more stable.

A. Performance comparison

We compared SHAPER with the single-hop TSF algorithm implemented in Blueware, in the same scenarios. First, we evaluated the mean formation time (*MFT*); that is the mean value of the time needed to obtain a unique connected tree. Figure 3 shows the *MFT* as a function of the number of nodes (N). When TSF is adopted, the *MFT* increases with N; this depends on the fact that only *coordinator* nodes can discover different trees and only *root* nodes can merge them. As N increases, the time spent by a coordinator to find nodes increases too. On the contrary, SHAPER connects trees via both R and NR (as shown in the example of Figure 2). As the node density increases, the probability that nodes meet increases too. This gives rise to a *MFT* that remains around 6.5 seconds for $N \geq 20$.

The scatternets formed when N ranges between 10 and 120 have an average number of piconets ranging from 5 to 75 for SHAPER and from 5 to 65 for TSF. Also the average number of roles assigned to a device is comparable: for $N \geq 40$, each node assumes 1.5 roles with TSF and 1.6 roles with SHAPER. The scatternets generated are similar in terms of number of piconets, number of slaves in a piconet, number of leaves.

B. Multi-hop Scenarios

The *MFT* was evaluated in a multi-hop scenario as a function of the number of nodes for different sizes of the geographical area where the nodes are scattered (Figure 4). As in the single-hop scenario, the *MFT* initially decreases as N increases and then remains quite flat. As expected, *MFT* also increases with the size of the area. This result is also reported in Figure 5 where the trend of the *MFT* as a function of geographical area dimensions is shown. We can conclude that the MFT strongly depends on the node density in a given area. However, after a certain low density threshold has been reached, the formation time is not affected anymore.

In TABLE III the number of `StartReconf` messages exchanged in the network for different values of N are reported for both single-hop and multi-hop scenarios. Although a high number of reconfiguration procedures is time-consuming, it allows the merging procedures to operate regardless of the status of the nodes. This assures a reasonable formation delay even in multi-hop scenarios.

Number of nodes	**40**	**80**	**100**	**120**
Single-Hop case	16	77	107	140
Multi-Hop case	13	67	97	128

TABLE III

NUMBER OF STARTRECONF MESSAGES EXCHANGED

C. Self-Healing behavior

To measure the self-healing behavior of SHAPER we considered the time needed to obtain a unique connected scatternet after an "en mass" arrival of a second group of nodes. All nodes are in radio visibility of each other in an area of 7×7 m^2. With reference to Figure 6, a certain percentage of the nodes on the x axis arrives after a scatternet with the other nodes has formed. We considered different percentages for the second group of nodes (25%, 50%, 75%). The time needed to accommodate the incoming nodes in the network, namely the

incremental formation time, is reported. This is shown to be inversely proportional to the percentage of nodes arriving with the second burst and is less than 8.5 s for $N \geq 20$.

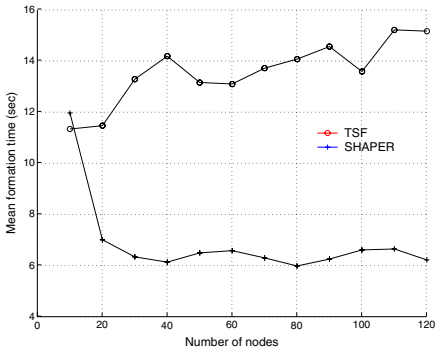

Fig. 3. Formation time: performance comparison between TSF and SHAPER

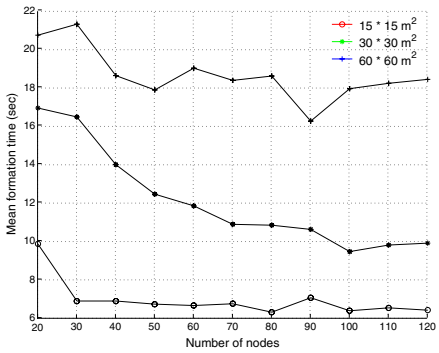

Fig. 4. Formation time, multi-hop scenarios

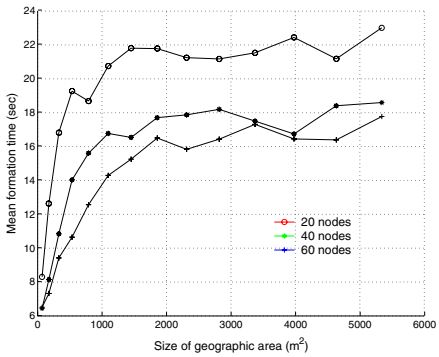

Fig. 5. Formation time as a function of the geographic area, multi-hop scenarios

V. Conclusions

In this paper a distributed algorithm that forms BT scatternets with a tree topology was described. The innovative features of the algorithm with respect to the literature, i.e. the combination of a self-healing behavior and of multi-hop properties, were described. Results show that the scatternet

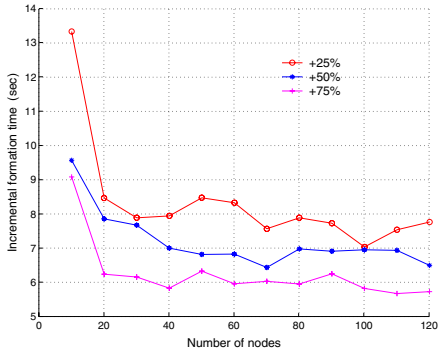

Fig. 6. Incremental formation time in a self-healing scenario

formation time is maintained under reasonable values, even when tree reconfiguration procedures have to be applied to merge trees that meet via non-root nodes. Future work will be dedicated to use the tree as a facility to rapidly propagate control information across the network. These control messages will be aimed at re-arranging periodically the scatternet topology in order to "optimize" the network performance under specific user requirements (e.g, delay, throughput, energy consumption).

References

[1] J. Haartsen, "The Bluetooth Radio System", *IEEE Personal Communications*, Vol. 7, n. 1, pp. 28-36, February 2000.

[2] P. Johansson, R. Kapoor, M. Gerla, M. Kazantzidis,"Bluetooth an Enabler of Personal Area Networking", *IEEE Network*, Special Issue on Personal Area Networks, pp. 28-37, September/October 2001.

[3] T. Salonidis, P. Bhagwat, L. Tassiulas, R. La Maire, "Distributed topology construction of Bluetooth personal area networks", Proc. of the *IEEE Infocom 2001*, pp. 1577-1586, April 2001.

[4] C. Law, A. Mehta, K-Y Siu, "Performance of a new Bluetooth scatternet formation protocol", Proc. of the *Mobihoc 2001*.

[5] G. Tan, A. Miu, J. Guttag, H. Balakrishnan, "An Efficient Scatternet Formation Algorithm for Dynamic Environments", in *IASTED Communications and Computer Networks (CCN)*, Cambridge, November 2002.

[6] G. Zaruba, S. Basagni, I. Chlamtac, "Bluetrees- Scatternet formation to enable Bluetooth-based personal area networks", Proc. of the *ICC 2001*, pp. 273-277, 2001.

[7] S. Basagni, C. Petrioli, "Multihop Scatternet Formation for Bluetooth Networks", Proc. of the *VTC 2002*, pp. 424-428, May 2002.

[8] C. Petrioli, S. Basagni "Degree-constrained Multihop Scatternet Formation for Bluetooth Networks", Proc. of the *IEEE Globecom 2002*, Taipei, November 2002.

[9] I. Stojmenovic, "Dominating set based scatternet formation with localized maintenance", Proc. of the *Workshop on Advances in Parallel and Distributed Computational Models*, April 2002.

[10] M. Ajmone Marsan, C. F. Chiasserini, A. Nucci, G. Carello, L. De Giovanni, "Optimizing the Topology of Bluetooth Wireless Personal Area Networks", *IEEE INFOCOM 2002*, New York, June 2002.

[11] C. F. Chiasserini, M. Ajmone Marsan, E. Baralis, P. Garza, "Towards Feasible Distributed Topology Formation Algorithms for Bluetooth-based WPANs", *36th Hawaii International Conference on System Science* (HICSS-36), Big Island, Hawaii, January 6, 2003

[12] F. Cuomo, T. Melodia, "A general methodology and key metrics for scatternet formation in Bluetooth", Proc. of the *IEEE Globecom 2002*, Taipei, November 2002

[13] G. Tan, "Blueware: Bluetooth Simulator for ns", MIT Laboratory for Computer Science, Cambridge, October 2002