Distributed Self-Healing and Variable Topology Optimization Algorithms for QoS Provisioning in Scatternets

Francesca Cuomo, Tommaso Melodia, and Ian F. Akyildiz, *Fellow, IEEE*

***Abstract*—Bluetooth is an enabling technology for Personal Area Networks. A *scatternet* is an ad hoc network created by interconnecting several Bluetooth *piconets*, each with at most eight devices. Each piconet uses a different radio channel constituted by a frequency hopping code. The way the devices are grouped in different piconets and the way the piconets are interconnected greatly affect the performance of the scatternet in terms of capacity, data transfer delay, and energy consumption. There is a need to develop distributed scatternet formation algorithms, which guarantee full connectivity of the devices, reconfigure the network due to mobility and failure of devices, and interconnect them such a way to create an optimal topology to achieve gainful performance. The contribution of this paper is to provide an integrated approach for scatternet formation and quality-of-service support (called SHAPER-OPT). To this aim, two main procedures are proposed. First, a new scatternet formation algorithm called self-healing algorithm producing multihop Bluetooth scatternets (SHAPER) is developed which forms tree-shaped scatternets. A procedure that produces a meshed topology applying a distributed scatternet optimization algorithm (DSOA) on the network built by SHAPER is then defined.**

Performance evaluation of the proposed algorithms, and of the accordingly created scatternets, is carried out by using ns2 simulation. Devices are shown to be able to join or leave the scatternet at any time, without compromising the long term connectivity. Delay for network setup and reconfiguration in dynamic environments is shown to be within acceptable bounds. DSOA is also shown to be easy to implement and to improve the overall network performance.

***Index Terms*—Bluetooth, multihop ad hoc networks, scatternet formation, topology optimization.**

I. Introduction

UNTETHERED networks of small hand-held electronic devices, such as cellular phones, personal digital assistants, notebooks, digital cameras and mp3 players are very likely to be part of our daily lives in the near future. These networks are usually referred to as personal area networks (PANs). Different PANs can be interconnected to enable sharing of information or seamlessly integrated with networks of sensor/actuator devices, to allow interaction with the physical environment. The IEEE 802.15.1 working group has recently released a standard for PANs based on the *Bluetooth* Industrial Specifications. Thanks to the *scatternet* concept [1], which allows more than eight devices to be interconnected in a multihop network, Bluetooth is considered an enabling technology for these scenarios [2].

A scatternet is composed of different *piconets*. Each piconet has an overall 1 Mbit/s gross data rate, to be shared by at most eight active devices (also, *nodes* in the following). Inside each piconet, the medium access control (MAC) is contention free and centrally regulated by a *master* device, which periodically polls the other devices (*slaves*). In a piconet setup, roles of the participating devices (master and slaves) are dynamic: the device that starts a communication in a piconet becomes the master. The way piconets are interconnected to form scatternets is a key issue in deploying high performance and efficient Bluetooth ad hoc networks.

Multihop ad hoc wireless networking, however, has been extensively studied in the past few years (and a few hardware test beds have been implemented, e.g., [3]) mostly based on the IEEE 802.11 standard. The 11 Mbit/s 802.11b is the most widespread standard for wireless local area networks in its *infrastrucured* mode. However, the standard also specifies an ad hoc peer-to-peer communication mode which can be used to realize PANs. Some recent works, however, questioned the suitability of 802.11 to support multihop ad hoc networking [4]. Others, e.g., [5], presented a comparison between the IEEE 802.11b and Bluetooth technologies for the support of PANs.

Bluetooth is known to present many appealing characteristics.

- As the density of nodes increases, Bluetooth has a higher capacity than IEEE 802.11b; the latter, due to its higher link rate, presents good performance when the network is sparse, but, in Bluetooth, the capacity increases as new piconets are included in the network [5], [6].
- Currently, output power of Bluetooth devices is around 0 dBm, as compared with the 13–20 dBm for IEEE 802.11b devices.
- Energy efficiency in Bluetooth remains the same as the network becomes denser; on the other hand, it drops in IEEE 802.11 because of the increased number of MAC collisions [5].
- Interactions between the TCP congestion control mechanism and the random access MAC mechanisms in 802.11 lead to serious unfairness between different connections and instability [4], [5]; on the contrary, the centrally regulated Bluetooth access in a piconet is suitable to avoid

Manuscript received June 10, 2003; revised March 22, 2004. This paper was presented in part at the IEEE Globecom 2003, San Francisco, CA, November 2003, and in part at the IEEE ICC 2004, Paris, France, June 2004 .

F. Cuomo is with the Infocom Department, Università degli Studi di Roma "La Sapienza," Rome 18-00184, Italy (e-mail: cuomo@infocom.uniroma1.it).

T. Melodia and I. F. Akyildiz are with the Broadband and Wireless Networking Laboratory, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: tommaso@ece.gatech.edu; ian@ece.gatech.edu).

Digital Object Identifier 10.1109/JSAC.2004.829341

useless transmission control protocol (TCP) time-outs expirations.
- Bluetooth has native mechanisms for quality-of-service (QoS) negotiation [1]; these could be useful in supporting voice and other real-time applications. IEEE 802.11 natively lacks a similar mechanism; however, efforts are under way in the 802.11e working group to provide a support for QoS.

On the other hand, the following main drawbacks have been pointed out.

- The scatternet structure adds complexity; mechanisms such as interpiconet scheduling [5] are not likely to scale well when the number of nodes increases.
- Bluetooth is connection oriented at the data link layer, whereas 802.11 is connectionless. This means that data link connections have to be explicitly setup. The issue of setting up data link connections among a set of Bluetooth devices, so as to guarantee network connectivity, is usually referred to as *Scatternet Formation*. Scatternet formation algorithms are aimed at forming and maintaining the topology of Bluetooth scatternets, starting from nodes without any knowledge of one another. In Bluetooth, the topology is determined by the way piconets are formed and interconnected. Different topologies lead to different performance [6]–[8]. On the other hand, in 802.11 the topology is univocally determined by the set of distances among nodes.
- Device discovery procedures of Bluetooth are extremely time-consuming (in the order of seconds) [9] and require asymmetric behavior of the involved devices in the inquiry and inquiry scan phases.

For the reasons discussed above, we believe that Bluetooth is the most suited technology for PAN applications, for networks of small and moderate size, i.e., when scalability does not become a major issue. The inherent energy efficiency and the possibility of deploying simple mechanisms for QoS support are two important features for the applications we are interested in.

The contribution of this paper is to provide an integrated approach for scatternet formation and QoS support (called SHAPER-OPT). To this aim, two main procedures are proposed. First, a new scatternet formation algorithm called self-healing algorithm producing multihop Bluetooth scatternets (SHAPER) is developed to form tree-shaped scatternets. SHAPER is a fully distributed and an asynchronous algorithm, works even for cases when not all devices are within radio range and provides self-healing capabilities. A procedure that produces a meshed topology applying a distributed scatternet optimization algorithm (DSOA) on the network built by SHAPER is then defined.

This paper is organized as follows. Section II discusses the scatternet formation issue and briefly describes the state of the art. Section III reports the main constraints of a Bluetooth scatternet formation algorithm and gives the problem formalization. Section IV overviews our approach for scatternet formation and topology optimization. The key rules and steps that constitute the proposed SHAPER algorithm are reported in Section V. Section VI sets an analytical framework for topology optimization and discusses the integration of DSOA and SHAPER in order to achieve a fully distributed optimization procedure. Section VII provides an extensive performance analysis of the proposed approach. Finally, conclusions are given in Section VIII.

II. Scatternet Formation: Related Work

Although Bluetooth was initially designed for cable replacement between small devices, it is currently considered as a potential enabler for ad hoc networking applications due to the *scatternet* concept. Different piconets can coexist in the same area with low mutual interference [10] by using different frequency hopping sequences. A scatternet is an ad hoc network of Bluetooth devices which consists of an interconnection of overlapping piconets. A Bluetooth device joining more than one piconet is called a *gateway*. The gateway participates in communications within different piconets on a time division basis and can be master in only one piconet. Scatternet formation has been extensively discussed in the last few years. Existing solutions can be classified as single-hop [11]–[15] and multihop [16]–[20]. The former operate only when nodes are in radio visibility. Reference [11] addresses Bluetooth scatternet formation with a distributed selection of a leader device, which assigns roles to the others. In [12], a distributed formation protocol is defined, with the goal of reducing formation time and message complexity. In [13], *loop scatternet formation* is presented, which minimizes the number of piconets and other parameters such as maximum node degree, network diameter, and node contention. Reference [14] proposes a scatternet topology called *BlueRing*, which connects piconets as a ring. The works in [15]–[17] form tree-shaped scatternets. In [15], Tan *et al.* introduce the tree scatternet formation (TSF) 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. Reference [16] presents an on-demand Bluetooth scatternet formation algorithm (called ODBT). The scatternet formation is initiated by a tree root and progressively extended to all devices. Dynamic changes in the scatternet topology are supported. Zaruba *et al.* [17] propose a multihop protocol, *Bluetree*, based on a process initiated by a unique node and repeated recursively till the leaves of the tree are reached.

A second class of multihop proposals consists of algorithms that produce connected scatternets by exploiting clustering schemes for ad hoc networks. In [18] and [19], 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 of the formed scatternet. The generated scatternet is a mesh with multiple paths between any pair of nodes. *BlueMesh* allows each master to select at most seven slaves. Also, [20] defines a protocol that limits to seven the number of slaves per master. It is based on the *Yao construction*, which applies a degree reduction techniques to the network topology graph.

The proposed algorithm assumes that each node knows its position and that of its neighbors.

The work in [22] proposes a new approach to scatternet formation. Route discovery and construction is performed on-demand on the basis of real-traffic conditions and traffic requests. However, the proposed approach requires substantial modifications of the Bluetooth standard to guarantee acceptable route-setup delay.

Some other works discuss scatternet topology optimization. This issue is faced in [23] and [6] by adopting centralized approaches. In [23], the aim is minimizing the load of the most congested node in the network, while [6] discusses the impact of different metrics on the scatternet topology. A distributed approach based on simple heuristics is presented in [24]. In [7], an analytical model of scatternet based on queueing theory is introduced, aimed at determining the number of nongateway and gateway slaves to guarantee acceptable delay characteristics. The paper in [8] investigates the relationship between network capacity and topology.

III. Problem Statement and Bluetooth Constraints

In 802.11-based ad hoc networks a single broadcast channel is shared by all devices using the carrier sense multiple access/collision avoidance (CSMA/CA)-based MAC. Thus, the distance among nodes defines the network topology. In Bluetooth, since multiple communication channels are available, the topology is not univocally defined by distances among nodes. The way devices are grouped in different piconets and the way piconets are interconnected greatly affect the scatternet performance in terms of capacity, delay and energy consumption.

Given a set of devices and their positions, we refer to the problem of finding the set of piconets and the set of interconnections among them, which optimizes a given performance metric. A scatternet must be formed by means of distributed algorithms running on each device. By means of these algorithms, nodes must discover their neighbors, establish links, and interconnect piconets, leading to a suitable topology.

To clearly define the problem, given an area which respects the conditions:

- each node should have another node in its transmission range

$$\forall i \in N \; \exists j \in N : \mathrm{dist}(i,j) \leq TR \tag{1}$$

  where $\mathrm{dist}(i,j)$ is the Euclidean distance between node i and node j, N is the set of nodes in the considered area, and TR the typical Bluetooth transmission range (equal to 10 m for a class 3 device);
- there is a possible path between any couple of nodes (i.e., the system is *geographically connected*, as defined in [17]).

There is a need to develop distributed algorithms that form scatternets, which respect the following **topological constraints**:

- no more than seven slaves belong to a piconet

$$\forall \mathcal{P}_k \in \mathcal{S}, \quad S_k \leq 7 \tag{2}$$

  where $\mathcal{S}$ represents the scatternet, $\mathcal{P}_k$ represents the kth piconet in the system, and S_k is the number of slaves of $\mathcal{P}_k$;
- a node can be master in only one piconet

$$\forall \mathcal{P}_k, \quad \exists\,!\, i \equiv \text{ master and } i \in \mathcal{P}_k \tag{3}$$

- each slave is connected at least to a master

$$\forall i \equiv \text{slave}, \quad \exists j \equiv \text{master} : \; i,j \in \mathcal{P}_k \tag{4}$$

- each piconet in the scatternet has a node shared with another piconet

$$\forall \mathcal{P}_k, \quad \mathcal{P}_h \in \mathcal{S} \quad \exists i : \; i \in \mathcal{P}_k \text{ and } i \in \mathcal{P}_h \tag{5}$$

  this shared node is called *gateway*;
- for each node, at least one path exists that connects it to any another node

$$\forall i, \quad j \in \mathcal{S} \quad \exists \,\text{path}(i,j). \tag{6}$$

Furthermore, a scatternet formation algorithm should have the following **properties**:

a) *guarantee full connectivity* to all the involved devices even in multihop scenarios, when devices are scattered in an area where not all of them are within radio range of each other (also referred to as *multihop* scenario);
b) achieve full connectivity after *limited time*;
c) guarantee a *self-healing behavior* in variable network conditions; in particular the algorithm should handle:
   i) entrance of new nodes in the network;
   ii) mobility or failure/deactivation of nodes;
d) guarantee *multiple* and *short paths* between any pair of nodes;
e) *optimize the topology* according to suitable performance metrics (e.g., overall network capacity).

IV. Overview of SHAPER-OPT: An Integrated Approach for Scatternet Formation and Topology Optimization

As previously discussed, no solutions have been proposed, thus far, that satisfy all the aforementioned properties of a scatternet formation algorithm. In our approach, we perform three main steps to form a Scatternet satisfying properties (2)–(6) and a)–e).

A. Step 1: Forming a Connected Tree via SHAPER

The first set of procedures, named SHAPER, has been preliminary presented in [21] and creates a tree-shaped scatternet. SHAPER quickly forms a connected scatternet respecting the topological constraints of (2)–(6) and the properties a)–c). A set of *merging procedures* allow SHAPER to interconnect randomly formed links (and piconets) in a loop free topology: different trees are merged and converge to a unique tree.

SHAPER is the first scatternet formation algorithm, which is able to dynamically adapt the topology to the mobility and failures of nodes in a multihop scenario. We will refer to this physical tree-shaped scatternet as primitive topology (PT).

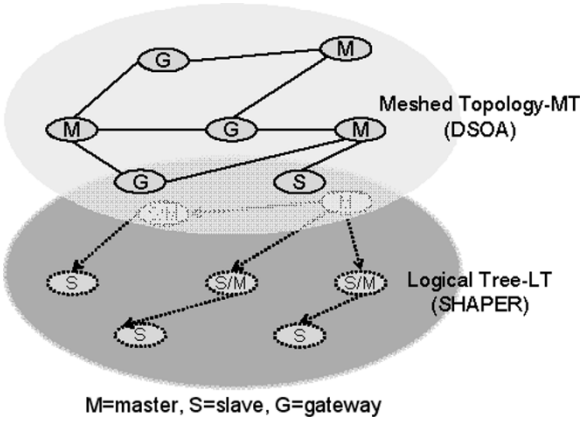


Fig. 1. Physical and logical topology.

B. Step 2: Optimizing the Topology via DSOA

Since a tree topology may not be the best choice to transfer user information (nodes easily become bottlenecks when data are exchanged), we enhance SHAPER by combining it with another distributed procedure (DSOA) that aims at optimizing the topology according to a selected performance metric. The DSOA heuristic and its performance in static scenarios were discussed in [6]. At the end of DSOA, the network presents an optimized meshed topology (MT). With DSOA also, the properties d) and e) are satisfied.

C. Step 3: Maintaining the Logical Tree for the Self-Healingness

After setup of the links that compose that optimized topology, the physical links of the tree become *logical links*. This means that nodes maintain information about their parent and children as established at the end of SHAPER, even if the relevant physical links have been torn down by DSOA in the MT. The set of these logical relationships will be referred to as logical tree (LT).

Purposes of this tree-shaped logical structure are:

1) allowing to periodically reapply DSOA to optimize the scatternet topology after dynamic events (mobility, failure of nodes, etc.);
2) simplifying the procedures that guarantee the connectivity of the network and the entrance of new nodes;
3) handling failure/mobility of nodes (SELF-HEALING procedures).

In general, after the first execution of DSOA the network has (see Fig. 1):

- a meshed physical topology;
- a tree logical topology.

The combination of SHAPER and DSOA, together with the SELF-HEALING procedures, is called SHAPER-OPT. Thanks to the LT, SHAPER-OPT is able to merge networks that have autonomously formed. These networks could either be in a PT or in a MT configuration. Periodically, DSOA is applied to reoptimize the overall physical topology.

To the best of our knowledge, SHAPER-OPT is the first scatternet formation algorithm that contemporarily works in a multihop environment, presents self-healing properties (e.g., dynamically reconfigures the network after node entrance/exit) and optimizes the scatternet topology. Moreover, a dynamic behavior is extensively studied in the framework of scatternet formation algorithms.

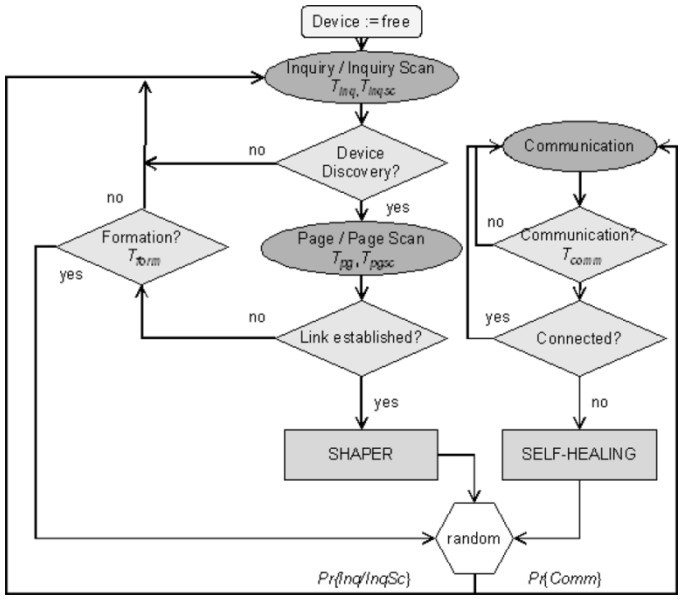


Fig. 2. Evolution of states for a Bluetooth device.

V. Description of the SHAPER-OPT Distributed Procedures

In a first phase, nodes perform device discovery and establish links. A node alternates among five possible states: *inquiry*, *inquiry scan*, *page*, *page scan*, and *communication* (see Fig. 2).

At the beginning, a node is free (i.e., not already connected to a scatternet) and enters the *inquiry* or *inquiry scan* states. In the *inquiry* state, the Bluetooth Inquiry procedure is performed for a time period of at most $T_{\rm inq}$ seconds. If during this period a node discovers another, then the two connect with the *page* and *page scan* procedures, respectively. The persistence in each of these two states is regulated by timers ($T_{\rm pg}$ and $T_{\rm pgsc}$). If the *page/page scan* procedures succeed, a master-slave relationship between the two nodes is activated, a link is established, and the two nodes enter the SHAPER procedure. If the *page/page scan* procedures fail, each node randomly enters again the *inquiry* or *inquiry scan* if its $T_{\rm form}$ timer has not already expired. Otherwise, if $T_{\rm form}$ has expired, the node goes in a *communication* state with probability $\Pr\{Comm\}$ or reenters the *inquiry* or *inquiry scan* states with probability $\Pr\{Inq/InqSc\}$. A node in the *inquiry scan* state performs the Bluetooth Inquiry Scan procedure with a $T_{\rm inqsc}$ timer.

In general, each node alternates, throughout its lifetime, between performing *network formation tasks* and *communication tasks*. Network formation tasks are dedicated to form and maintain a connected scatternet by running SHAPER, DSOA and the SELF-HEALING procedures. During communication tasks, user data are exchanged for a randomly extracted time period ($T_{\rm comm}$, uniformly distributed between $T_{\rm comm}^{\rm min}$ and $T_{\rm comm}^{\rm max}$ s). After this period a node, with probability $\Pr\{Inq/InqSc\}$, reenters the *inquiry* or *inquiry scan* states. It should be clear, though, that there is no other way to guarantee network connectivity than assuring that every single node in

the network periodically performs network formation tasks. In fact, every node could be the unique *trait d'union* of otherwise disconnected components. The percentage of time a node spends performing communication tasks, i.e., $\Pr\{Comm\}$, should be adaptively set in order to tradeoff between a faster responsiveness to dynamics in the network and a higher network transport capacity. This happens because a Bluetooth device cannot participate in data communication when it is performing connection establishment or device discovery procedures.

A. State Variables and Messages

To better explain the SHAPER-OPT procedures, we introduce a number of state variables that define the behavior of each node when these procedures are executed. Let us refer to a generic node i. It maintains and stores the following state variables.

- $comp_i$ is the component i is affiliated with. A component can be: 1) a *primitive topology* (PT, tree-shaped, not yet optimized) and 2) a *meshed topology* (MT, shaped by DSOA).

T_i represents the tree node i is affiliated with. We refer to the root node of a tree T as $root_T$. When a node i is affiliated with a tree T_i in a PT, it stores and updates the following variables:

- $stat_i$ is the *status* of the node i, that can be *free* (F), *root* (R) or *nonroot* (NR);
- $tree\_ID_i$ is the Bluetooth device address (BD_ADDR) of the root of the tree T_i (or of itself if free); this information univocally identifies all nodes in the same tree;
- num_i is the current number of nodes in T_i;
- $desc_i$ is the current number of descendants of node i in T_i;
- par_i is the BD_ADDR of the parent node of i in T_i;
- $ch_i = \{ch_i^1, ch_i^2, \ldots, ch_i^7\}$ is the set of children nodes of node i.

We always assume that a free node has $comp = \mathrm{PT}$.

When a node i is affiliated with a MT, it belongs also to a *logical tree* LT_i; it stores and updates the following parameters:

- $Lstat_i$ is the *status* of node i, F, R or NR in the logical tree;
- $tree\_ID_i$ is the BD_ADDR of the root of LT_i;
- $Lpar_i$ is the BD_ADDR of the parent node of i in LT_i;
- $Lch_i = \{Lch_i^1, Lch_i^2, \ldots, Lch_i^7\}$: is the set of children nodes in LT_i;
- $role_i$ is the role i assumes in the MT, either master (M), slave (S) or gateway (G).

When DSOA is performed for the first time, every node sets its logical state variables equal to the respective PT variables.

A node can send different types of messages; messages containing the information used for merging and reconfiguring trees are reported in Table I.

TABLE I
SHAPER-OPT MESSAGES AND RELEVANT PARAMETERS

MESSAGE NAME	*comp*	*tree_ID*	*num*	*desc*	*stat*	*role*
MERGE	×	×	×	-	×	×
STARTRECONF	-	×	×	-	-	-
UPDATEPARAMETERS	-	×	×	-	-	-
UPDATENDESC	-	×	×	×	-	-
UPDATETREEID	×	×	-	-	-	-

B. SHAPER Algorithm

The SHAPER procedures allow two nodes that meet during the network formation tasks to merge the components they belong to. We always refer to the two meeting nodes, through which the two components merge, as M, the master, and as S the slave of a newly formed link (named L_{MS}). After the link has been established, M sends a MERGE message to S (see also, the pseudocode of Procedure 1). This message contains the component $comp$ and the $tree\_ID$ of M as well. S verifies if the $tree\_ID$ of M equals to its $tree\_ID$. In this case, the two nodes already belong to the same scatternet and the two components need not to be merged. The link L_{MS} can be torn down. If the $tree_ID$s are different, different actions are taken depending on the components the two nodes are affiliated with.

The following situations can occur:

- Case 1) a free node meets a PT-node: the SHAPER procedures are applied to include the F node in the PT tree;
- Case 2) a PT-node meets a PT-node: one of the two PTs is included in the other, thus forming a unique tree;
- Case 3) a PT-node meets an MT-node: the SHAPER procedures are applied with the aim of forming a unique connected scatternet (giving rise to a hybrid PT–MT topology);
- Case 4) an MT-node meets an MT-node: the SHAPER procedures are applied with the aim of forming a unique connected scatternet.

Cases 1) and 2) refer to the nodes belonging to PT and will be discussed into details in the following subsection. The other two cases deal with an MT that merge with another component (either PT or MT), and are discussed in Section V-D.

Procedure 1 MainSHAPER
$M \rightarrow S$: MERGE
if $(tree\_ID_M \equiv tree\_ID_S)$ **then**
  disconnect (M, S)
else
  execute SHAPER as a function of $comp_M$, $comp_S$, $stat_M$, $stat_S$, $role_M$, $role_S$
endif

C. SHAPER on Primitive Topologies

In this section, we explain how SHAPER works on a primitive topology, i.e., on nodes that have never executed DSOA. Different links are merged in trees; trees that have separately grown are subsequently merged. The network converges to a unique connected tree always and only when the adjacency graph is connected. In every branch of the final tree, the parent is always the master and the child is always the slave. Thus, an intermediate node in the tree (non root and non leaf) is a gateway between two piconets.

Let us refer to the M and S nodes in the previous section, which belong to different PTs and have established a link

L_{MS}. If both are nonfree nodes, then they respectively belong to the T_M and T_S trees. After the first message exchange in Procedure 1, when the two components are equal to PT, subsequent steps depend on the *stat* values of the two nodes. Different actions are foreseen (Table II). A pseudocode for these procedures is given next.

Procedure 2 PT-SHAPER
$A1:$
if $(num_M \geq num_S)$ **then**
  $par_S = M;\ tree\_ID_S = tree\_ID_M$
  $desc_M = num_S + desc_M$
  $num = num_S + num_M$
  $S \rightarrow ch_S$: UPDATEPARAMETERS
  $M \rightarrow ch_M$: UPDATEPARAMETERS
else
  LMP_MasterSlaveSwitch(M, S)
  $\{\mathrm{new}M = S, \mathrm{new}S = M\}$
  new$M$, new$S$: execute MainSHAPER
end if
$A2:$
$par_S = M;\ tree\_ID_S = tree\_ID_M$
$desc_M = num_S + desc_M$
$num = num_S + num_M$
$S \rightarrow ch_S$: UPDATEPARAMETERS
$M \rightarrow ch_M$: UPDATEPARAMETERS
$M \rightarrow par_M$: UPDATENDESC
$A3:$
LMP_MasterSlaveSwitch(M, S)
$\{\mathrm{new}M = S, \mathrm{new}S = M\}$
new$M$, new$S$: execute MainSHAPER
$A4:$
if $(num_M \leq num_S)$ **then**
  LMP_MasterSlaveSwitch(M, S)
  $\{\mathrm{new}M = S, \mathrm{new}S = M\}$
  new$M$, new$S$: execute MainSHAPER
else
  $desc_M = num_S + desc_M$
  $tree\_ID_S = tree\_ID_M$
  $num = num_S + num_M$
  $M \rightarrow par_M$: UPDATENDESC
  $S \rightarrow par_S$: STARTRECONF
  S: wait for RECONFACK
  $P = par_S$
  $par_S = M$
  $S \rightarrow ch_S$: UPDATEPARAMETERS
  repeat
    $P \rightarrow ch_P$ (excluding S): UPDATEPARAMETERS
    LMP_MasterSlaveSwitch(P, S)
    $S = P$
    $P = par_P$
  until $S = initialroot_{T_S}$
end if

Let us refer to Fig. 3. Action A_1 (upper left box in the Figure) applies when both M and S are roots. The two trees they belong to can simply merge. The only decision to be taken regards which device will be the new root. We assume that it is always the R with the highest value of num. If it is M, then L_{MS} maintains the initial configuration; S acknowledges M and sends an UPDATEPARAMETERS message to its children, which contains the new $tree\_ID = tree\_ID_M$ and the new value of $num = num_M + num_S$; after receiving an ACK from S, M also sends the new num value to its children through an UPDATEPARAMETERS message. All nodes receiving the message update their num and $tree\_ID$ parameters with the values included in the message, and propagate the same message to their children until the leaves of the tree are reached.

TABLE II
DIFFERENT ACTIONS AS A FUNCTION OF THE $stat$ VARIABLE

$S \backslash M$	R	NR	F
R	A_1	A_2	A_3
NR	A_3	A_4	A_3
F	A_2	A_2	A_2

If $num_M < num_S$, L_{MS} is switched, i.e., M and S exchange their roles; this is done by means of the Bluetooth LMP_MasterSlaveSwitch primitive. After the switch, the nodes execute the procedure MainSHAPER again with the new roles.

In the A_2 case (upper right box in Fig. 3), the T_S is simply included in the tree, independently of the number of nodes of the two trees. S acknowledges M and sends the UPDATEPARAMETERS message to its children to update the number of nodes. M updates its $desc$ value ($desc_M = desc_M + num_S$) and sends an UPDATENDESC message to its parent to propagate the $desc$ updating for nodes on higher levels. Whenever a node receives an UPDATENDESC message, it updates the $desc$ value, forward it to its parent and sends an UPDATEPARAMETERS message with the new value of num to its children (excluding the one that sent the UPDATENDESC). The UPDATENDESC message is, thus, used to update the value of $desc$ and is always sent in the child-to-parent direction.

In A_3, nodes simply switch the link L_{MS} and reexecute MainSHAPER with the new roles.

Finally, in case A_4, two NR nodes meet (lower box in Fig. 3). 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 num; without loss of generality, we assume that the tree to be reconfigured is T_S. Otherwise, the LMP_MasterSlaveSwitch is applied and MainSHAPER is executed again.

The reconfiguration process is started by S that requests a permission from the root node of its tree. This permission is granted by the root if no other reconfigurations are occurring on the tree itself. Node S requests the permission by sending a RECONFREQ message to its parent. The parent node forward this message to its own parent. This is repeated until the root of T_S is reached. The root enables the reconfiguration by replying with a RECONFACK message. Once enabled, S sends a STARTRECONF to its parent and an UPDATEPARAMETERS to its children. Both messages include the new $num = num_M + num_S$ and the $tree\_ID$ value of M. The parent of S sends an

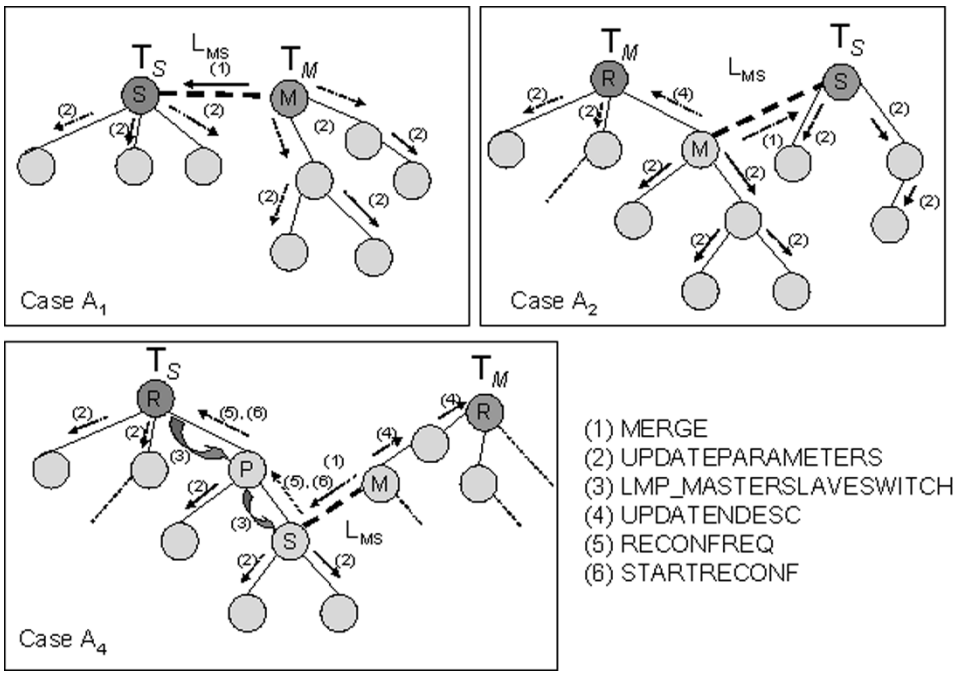


Fig. 3. Different actions performed by SHAPER on primitive topologies.

UPDATEPARAMETERS to all of its children, excluding S, and forward the STARTRECONF message to its parent. An LMP_MasterSlaveSwitch is performed after that between S and its parent. These steps are then repeated from the parent of S (P in the Figure) toward its own parent and the process continues until the initial root of T_S is reached. At this point, S is the new root of T_S, and T_S is included in T_M. This gives rise to a unique tree whose root is $root_{T_M}$. M updates its descendants and propagates an UPDATENDESC message toward the root of T_M.

In SHAPER, whenever a slave tries to connect to a master which already has seven children, i.e., seven slaves in its piconet, an adjunctive procedure is called (named ReconfSlaves), which relies on a property demonstrated in [17]. If a node has more than five neighbors, then at least two of them are neighbors themselves. Thus, the master temporarily parks the entering slave and forces the setup of a link between two of its children, then the parked slave is waken up and becomes an active child in the tree.

All the above rules guarantee that trees can always merge and a unique tree can be obtained even in a multihop scenario.

D. SHAPER With Mixed Primitive and Meshed Topologies

In this section, we generalize SHAPER to the following two cases: 1) when a PT and an MT have to be merged (e.g., when new nodes are activated in an area where an already optimized scatternet exists) and 2) when two MT scatternets meet.

As for the first case, thanks to the link setup procedure, a link is activated between a node with $comp = \mathrm{PT}$ and a node with $comp = \mathrm{MT}$. If the master of the new link has also master $role$ in the MT, the link is kept as it is. Otherwise, the link is switched and the procedure is reentered. In the following description, we assume that the master M is in the MT and the slave S is in the PT. In accordance to MainSHAPER, the two meeting nodes exchange their $tree\_ID$. It is to be noticed that, for the slave node, this $tree\_ID$ corresponds to the address of its physical tree. On the contrary, for M, this $tree\_ID$ corresponds to the address of the logical tree associated to the MT. SHAPER connects the two scatternets with the newly formed link and includes the physical tree in the logical one. At the end of the procedure, a unique tree is available at the LT level while, at the physical level, the scatternet will present an hybrid topology with a meshed zone and a tree zone.

As for the inclusion of the PT tree in the LT_M, the same procedures of PT-SHAPER are applied on the basis of the status of the two meeting nodes as in Procedure 2. However, there are three main differences: 1) at the beginning of the procedure a message, UPDATETREEID, is broadcast to all nodes in the PT; this message substitutes UPDATEPARAMETERS and aims at changing the $tree\_ID$ of the PT with $tree\_ID_M$; 2) the switch between parent and child is only logical (there is no need for an LMP_MasterSlaveSwitch); and 3) the number of nodes and descendants is not updated any more. This way, we are not able anymore to include the smallest logical tree in the biggest tree, which would minimize the number of switches, since nodes do not have this information. However, logical switches are less time-consuming than physical switches, while we avoid all the signaling traffic associated to updating the number of descendants.

As for the merging of the logical trees underlying two MTs, we try to keep the roles of the two meeting nodes in their respective MT. This means that, if M in the new link has also a $role = M$ in the MT, while S has also a slave role in the MT, the link is kept as it is and the LT_S is included in the LT_M, regardless of the number of nodes of the two LTs. In the opposite case, the link is switched and the procedure is reexecuted with the new roles. Finally, in case the two nodes both have $role = M$ or G in their MT the link is not switched, for it would bring no advantage.

In both the above cases (MT–PT and MT–MT), the link between the two scatternets is always setup with the aim of keeping the role of nodes in MT equal to the role in this new link. An exception occurs when the node that should become the master M already has seven slaves in its original piconet. In this case, the link is switched and the second node (initially S) becomes master, thus neglecting previous rules. If both (M and S) have seven slaves the ReconfSlaves procedure of Section V-C is called.

E. Self-Healing Behavior

SHAPER-OPT guarantees a self-healing behavior of the network by reconfiguring the scatternet topology when a node abandons the network or moves. This is accomplished in two steps.

- Step 1) The SELF-HEALING procedures update all the state variables characterizing the tree (either logical or physical.
- Step 2) SHAPER merges scatternets that could have disconnected by means of the mechanisms described in Sections V-C and V-D.

In case of PTs, when a node loses connectivity with its parent (e.g., when it does not receive any answer in a given timeout), it assumes that the parent has moved or switched off and becomes a R node. It subsequently sends an UPDATEPARAMETERS message to all its children. This contains its BD_ADDR in the $tree\_ID$ field and the current value of $desc$ in the num field. When a parent node loses its child, it must update the number of nodes in the tree. To this aim it requests the $desc$ values from its active sons and compares it with its current value of $desc$ to figure out how many nodes have disconnected. Once num has been updated it sends an UPDATEPARAMETERS to all its children and an UPDATENDESC to its parent. At the end of this process, we obtain two independent PT trees. Procedure 3 is periodically performed by a generic node in the PT (named X). Lines 1–12 tackle the disconnection of children nodes, while lines 13–18 tackle the disconnection of a parent.

Procedure 3 PT-SELF-HEALING
1: **for** each ch_X^i **do**
2: **if** ch_X^i not connected **then**
3: X: delete ch_X^i
4: **end if**
5: **end for**
6: **for** each ch_X^i **do**
7: $X \rightarrow ch_X^i$: query $desc_{ch_X^i}$
8: **end for**
9: X: update $num = num - (desc - \sum_{i \in ch_X} desc_{ch_X^i})$
10: X: update $desc = \sum_{i \in ch_X} desc_{ch_X^i}$
11: $X \rightarrow ch_X^i$: UPDATEPARAMETERS
12: $X \rightarrow par_X$: UPDATENDESC
13: **if** par_X not connected **then**
14: X: $stat_X = R$
15: **for** each ch_X **do**
16: $X \rightarrow ch_X^i$: UPDATEPARAMETERS with $tree\_ID = BD\_ADDR_X$ and $num = desc_X$
17: **end for**
18: **end if**

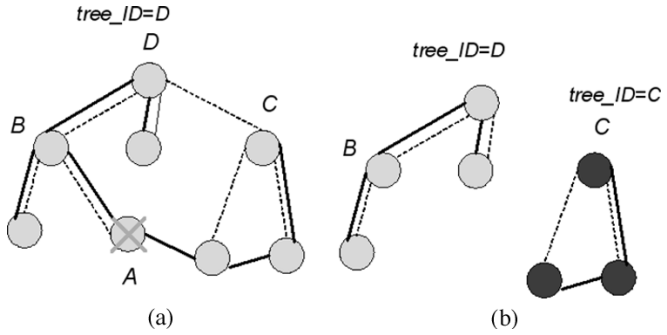


Fig. 4. Self-healing behavior when the network gets disconnected.

The SELF-HEALING procedures for MT rely on a periodical check of the connectivity of the logical tree. Each node controls whether its logical parent and its logical children are physically reachable. In case a node discovers that its logical children are not reachable, it simply expunges it from the list Lch. In case a node loses it logical parent, the LT is reconfigured. It is to be noticed that a node can lose its parent in the following two situations:

- Case 1) the physical MT becomes disconnected, the parent is still active but no path toward the parent exists any more;
- Case 2) the parent abandons the network.

Case 1) is reported in Fig. 4. Solid lines represent physical links, while dashed lines represent logical relationships in the LT. Node A disconnects and, being a leaf in the logical tree, it is simply deleted by its parent B from the Lch_B. Due to the periodical check, C discovers that its logical parent D is not physically reachable. As a consequence, C becomes R and assigns its BD_ADDR to its descendants. At this point, the two scatternets are completely separated (at a physical and logical level). SHAPER will reconnect these two scatternets if at least two nodes of them are in radio visibility.

In Case 2) a node [E in the Fig. 5(a)] abandons the network. Its child in the logical tree (A), that, as describe before, periodically controls the presence of its $Lpar$, discovers the node disconnection and starts a reconfiguration on its tree. This reconfiguration terminates with A as root of a new logical tree (with $tree\_ID = BD\_ADDR_A$). As for the parent of node E (C in the figure), it deletes the child from the Lch_C set. A difference with respect to the previous case is that A already has some physical links with other nodes (B) that are not its children. This means that separate logical trees exist in the same MT. Node A then verifies, for each neighbor connected to it via a physical link, if it has the same $tree\_ID$. If the $tree\_IDs$ are different, A establishes a logical relationship with its neighbor [B in the Fig. 5(c)] as a child. For each logical descendant, A updates the $tree\_ID$ with the $tree\_ID$ of B.

Procedure 4 reports the self-healing behavior of a generic node (named X in the pseudocode) when it belongs to a MT. It is to be noticed that Procedure 4 is executed periodically; moreover, it is executed whenever a node loses its physical neighbor, since it is more likely that a node loses also its logical parent or child. For example, with reference to Fig. 4, lines 1–15 correspond to actions performed by node C, while node B executes lines 1–7.

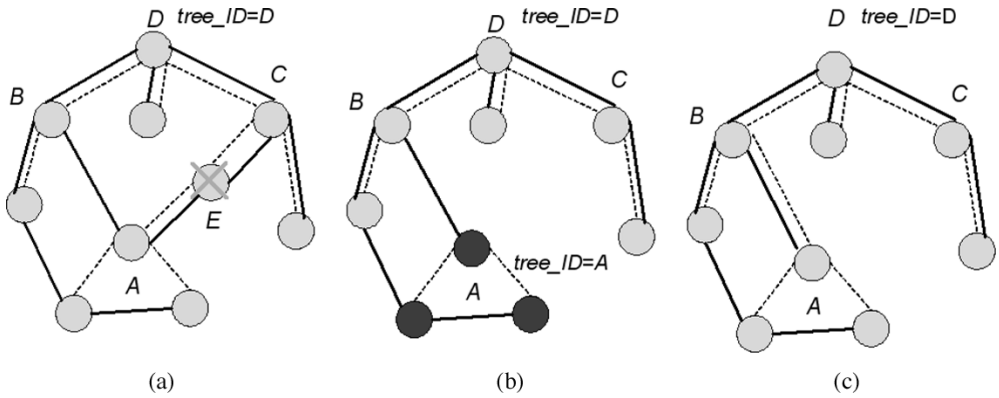


Fig. 5. Self-healing behavior when the network is still connected and two logical trees merge.

As for Fig. 4, when node E disconnects, C as parent of E executes lines 1–7 while A, as child executes lines 8–25.

Procedure 4 MT-SELF-HEALING
1: **for** each Lch^i_X **do**
2: $X \rightarrow Lch^i_X$: CHECK
3: X: wait for ACK from Lch^i_X
4: **if** TimeOut **then**
5: X: delete Lch^i_X
6: **end if**
7: **end for**
8: $X \rightarrow Lpar_X$: CHECK
9: X: wait for ACK from $Lpar_X$
10: **if** TimeOut **then**
11: X: delete $Lpar_X$
12: X: $Lstat = R$
13: **for** each Lch^i_X **do**
14: $X \rightarrow Lch^i_X$: UPDATEPARAMETERS with $tree\_ID = BD\_ADDR_X$
15: **end for**
16: **for** each neighbor Y of X **do**
17: **if** $tree\_ID_X \neq tree\_ID_Y$ **then**
18: $Lpar_X = Y$
19: **for** each Lch^i_X **do**
20: $X \rightarrow Lch^i_X$: UPDATEPARAMETERS with $tree\_ID = BD\_ADDR_Y$
21: **end for**
22: **exit**
23: **end if**
24: **end for**
25: **end if**

VI. TOPOLOGY OPTIMIZATION

In this section, we discuss the proposed topology optimization algorithm. We describe the DSOA and explain how the physical or logical tree structure can be exploited to provide topology optimization. First, we give a general formulation of some scatternet topology optimization problems, each defined by a cost function, which aims at optimizing different performance targets. Then, we discuss DSOA and its integration with SHAPER. In [6], the impact of different metrics on the topology was shown. Here, we generalize the formulation and discuss the implementation aspects related to our integrated approach to scatternet formation.

A multihop scenario, constituted by N nodes, can be modeled with an undirected graph $G(\mathcal{V}, \mathcal{A})$, where $\mathcal{V}$ is the set of nodes and an edge a_{ij} belongs to $\mathcal{A}$ iff $dist(v_i, v_j) \leq TR$, i.e. if v_i and v_j are in each other's *transmission range*. The graph can be represented with the corresponding $N \times N$ adjacency matrix $\mathbf{A} = [a_{ij}]$.

A scatternet in the above scenario can be represented with another square matrix $\mathbf{B}$ whose element b_{ij} equals 1 if node j is a slave in master i's piconet. Thus, in this second matrix, which is not symmetric, we associate row i with the master role for node i and column j with the slave role for node j. For example, $b_{13} = 1$ means that node 1 is master of a piconet in which node 3 is a slave.

Scatternet topology optimization is related to the selected scheduling policies and routing schemes. Given a scatternet topology, the capacity of any link depends on the adopted intrapiconet and interpiconet scheduling policies [2]. Thus, given a certain scheduling policy $\mathfrak{S}$ it is possible to define a rule to assign an amount of capacity to every link of the scatternet

$$\mathfrak{S} : \mathbf{B} \rightarrow \mathbf{C} \tag{7}$$

where $\mathbf{C}$ is a square matrix, whose element c_{ij} represents the *normalized capacity* (with respect to the capacity of a Bluetooth piconet) of the link between master i and slave j according to the adopted scheduling rule $\mathfrak{S}$. In [6], a simple method to calculate the capacity of all links was defined, given a simple rule $\mathfrak{S}$ and assuming bipartite scatternets. Furthermore, by assuming a model for interference between co-located piconets, given $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ it is possible to calculate the matrix $\mathbf{C^I}$, whose element c^I_{ij} represents the capacity of the link (i, j), taking into account interference effects. We can, thus, refer to

$$c = \sum_{(i,j)} c^I_{ij} \tag{8}$$

as the overall *normalized capacity* of the scatternet, where $\sum_i c^I_{ij} + \sum_j c^I_{ij} < 1$ holds.

For what concerns routing, we can describe a path between any two nodes (n, m) with another matrix $\mathbf{P}^{n,m}$, whose element $p^{n,m}_{ij}$ equals 1 if link (i, j) is part of the path, defined by the routing scheme, between node n and m. Thus, for each possible pair of nodes (n, m), we have

$$\mathfrak{R} : \mathbf{B} \rightarrow \mathbf{P}^{n,m}. \tag{9}$$

This *routing rule* ($\mathfrak{R}$) defines the path between the two nodes and depends on the routing algorithm and on the link metrics applied. For example, in [6], a link metric which assigns the same cost to every link was applied, which results in the path with the lowest number of hops being selected between any two nodes.

We can now define

$$h^{n,m} = \sum_{(i,j)} p_{ij}^{n,m} \tag{10}$$

which represents the *number of hops* for the path between node n and node m, according to $\mathfrak{R}$, and

$$\tilde{h} = \sum_{(n,m)} \frac{h^{n,m}}{N \cdot (N-1)} \tag{11}$$

which represents the *average path length* of the scatternet $\mathbf{B}$.

Optimization problem 1 ($\mathbf{O1}$)—Maximize Normalized Capacity

Given: $\mathbf{A}$, $\mathfrak{S}$

$$\mathbf{O1}: \max_{\mathbf{B}}\{c\} \tag{12}$$

with the following constraints:

1) $b_{ij} \leq a_{ij}, i = 1,2,\ldots,N; j = 1,2,\ldots,N;$
2) $\sum_{j=1}^{N} b_{ij} \leq 7\ i = 1,2,\ldots,N;$
3) $\sum_{j=1}^{N} b_{ij} + \sum_{j=1}^{N} b_{ji} \geq 1, i = 1,2,\ldots,N;$
4) $\mathbf{B}$ does not have a block structure, row permutations notwithstanding.

These constraints impose that the matrix $\mathbf{B}$ represents a connected scatternet compliant with the Bluetooth Specifications. These are the equivalent of the topological constraints in (2)–(6) in Section III with the matrix representation. Constraint 1 imposes that links exist only between adjacent nodes. Constraint 2 imposes that no piconet has more than seven slaves; constraint 3 imposes that every node is at least connected with another node, while constraint 4 is a sufficient condition to guarantee full connectivity of the scatternet.

O2—Minimize Average Path Length

Given: $\mathbf{A}$, $\mathfrak{S}$, $\mathfrak{R}$

$$\mathbf{O2}: \min_{\mathbf{B}}\{\tilde{h}\} \tag{13}$$

subject to constraints 1), 2), 3) and 4).

O3—Maximize Equivalent Normalized Capacity

Given: $\mathbf{A}$, $\mathfrak{S}$, $\mathfrak{R}$

$$\mathbf{O3}: \max_{\mathbf{B}}\left\{\frac{c}{\tilde{h}}\right\} \tag{14}$$

with constraints 1), 2), 3) and 4). The cost function in (14), which we refer to as *equivalent normalized capacity*, can be thought as the capacity the scatternet would have if all nodes were one hop distant from one another.

It is easy to see that each objective function defines a *metric* on the space of possible solutions, i.e., we can order the scatternets according to those metrics.

The above optimization problems were resolved by means of state space enumeration for small topologies in [6]; moreover, in [6] a distributed greedy heuristic, called DSOA, was introduced. DSOA assumes that the nodes in $\mathcal{V}$ are ordered in a set $\mathbf{w} = [w_1, w_2, \ldots, w_N]$. At step k of DSOA, the node w_k receives the matrix $\mathbf{B^{k-1}}$, which describes the topology selected by the nodes $w_1, w_2, \ldots, w_{k-1}$. The node w_k must then select the role to assume in the scatternet (master, slave, or gateway) and which connections to establish with nodes $w_1, w_2, \ldots, w_{k-1}$. The configuration is selected that results in the best local solution for the $\mathbf{O}$ problem. The algorithm evaluates the different configurations by means of a function $\mathcal{M}(\mathbf{B^k})$. This function is calculated on the all matrixes representing the possible topological alternatives at step k. In accordance to the selected optimization problem $\mathbf{O}$, this metric should be maximized or minimized. Node w_k selects the topology that optimizes the target metric value $\mathcal{M}(\mathbf{B^k})$; after that, the new topology is signaled to node w_{k+1}, by means of the matrix $\mathbf{B^k}$ that describes the new topology. Nodes cannot modify decisions previously taken by other nodes.

The ordering of the nodes $[w_1, w_2, \ldots, w_N]$ must respect the property that at least one node in $w_1 \ldots w_{k-1}$ is in the transmission range of w_k, for every $k \geq 2$. Under this assumption, in [6], DSOA was shown to be correct and to create near-optimal scatternets.

For the convenience of the reader, DSOA is reported in the following pseudocode. For the sake of simplicity it is assumed that the nodes are divided in two disjoint sets, the set of the masters ($\mathcal{V}_M$) and the set of the slaves ($\mathcal{V}_S$), thus leading to a bipartite graph representing the scatternet. In this case, the matrix is rectangular, the rows are associated with master nodes and the columns with slaves. The algorithm can easily be extended to the general case at the expense of complexity.

Procedure 5 DSOA
Input: $\mathbf{w}$, $G(\mathcal{V},\mathcal{A})$, $\mathcal{M}(\mathbf{B})$, $\mathbf{O}$
Output: Optimized Topology $\mathbf{B}^*$
begin
$\mathcal{V}_M = \emptyset$
$\mathcal{V}_S = \emptyset$
$\mathcal{V}_M = \mathcal{V}_M \cup w_1$
$\mathcal{V}_S = \mathcal{V}_S \cup w_2$
$\mathbf{B^2} = [1]$
for k = 3: N **do**
  case 1: consider w_k in $\mathcal{V}_M$
  derive all Bluetooth-compliant matrices
  $\mathbf{B^k}$ with
  $|\mathcal{V}_M| + 1$ rows and $|\mathcal{V}_S|$ columns
  calculate values of $\mathcal{M}(\mathbf{B^k})$
  case 2: consider w_k in $\mathcal{V}_S$
  derive all Bluetooth-compliant matrices
  $\mathbf{B^k}$ with
  $|\mathcal{V}_M|$ rows and $|\mathcal{V}_s| + 1$ columns
  calculate values of $\mathcal{M}(\mathbf{B^k})$
  select the $\mathbf{B^k}$ with optimal $\mathcal{M}(\mathbf{B^k})$ in accordance
  to $\mathbf{O}$
  if optimum in case 1 **then**
    $\mathcal{V}_M = \mathcal{V}_M \cup w_k$
  else
    $\mathcal{V}_S = \mathcal{V}_S \cup w_k$
  end if
end for
$\mathbf{B}^* = \mathbf{B^N}$
end

A. Combining SHAPER and DSOA: SHAPER-OPT

As previously described, nodes willing to form a connected ad hoc scatternet execute the SHAPER algorithm. After a few seconds (see Section VII), the nodes are connected in a tree-shaped physical scatternet. In [6], it was shown that the ordering of the nodes required by DSOA can be provided by a tree shaped scatternet. This is accomplished in the following way. All nodes are sequentially visited, starting from the root. The root also starts the optimization process by broadcasting an INIT message on the tree. This means that an optimization phase is starting, during which no other network formation tasks are performed. A generic node w_k receives the matrix $\mathbf{B^{k-1}}$ from its parent; the matrix describes the topology selected by the nodes $w_1, \ldots, w_{k-1}$. Then, w_k takes its decision and sends the new matrix $\mathbf{B^k}$ to a child (node w_{k+1}). When w_k receives the matrix back from w_{k+1}, the matrix contains the decisions taken by w_{k+1} and all its descendants in the tree. Node w_k sends the new matrix to another child. After w_k receives the matrix from its last child, it sends it back to its parent. When the root node receives the final $\mathbf{B^N}$, this matrix describes the topology that has been selected by all the nodes. The decision phase is finished and the setup of the new topology can start. The matrix is then broadcasted and all nodes start establishing the connections that constitute the selected network topology. Each master connects its slaves with the Bluetooth *page* procedure, following the order defined by the matrix, while each slave listens for incoming connections from the master with the Bluetooth *page scan* procedures. Slaves with multiple masters follow the order defined by the matrix. Physical links produced by SHAPER, when different from DSOA links, are torn down after the new links are established, while the parent-child relationships of the physical tree become logical ones.

B. Periodical Reoptimization of the Network

Once the first optimized topology has been obtained, a timer T_{DSOA} is started by the root node. This timer regulates the periodical reoptimization of the network that is applied to assure that new nodes (or networks) that have merged with the original one are reconfigured in accordance with DSOA. When the timer expires, the root node broadcasts an INIT message on the tree, thus starting a new optimization process.

To conclude, we observe that the advantages of using and maintaining the tree structure are:

1) allowing periodical execution of DSOA, by sequencing the nodes in the network;
2) assuring that the network can self-reconfigure during its lifetime in dynamic conditions.

VII. PERFORMANCE RESULTS

In this section, we report extensive simulation results of our scatternet formation and optimization procedures. The SHAPER-OPT procedures have been implemented on Blueware, an ns-2 simulator which simulates most aspects of the Bluetooth protocol stack [25].

The WirelessPhy module of Blueware simulates the radio aspects of Bluetooth. The FhChannel module simulates the frequency hopping shared medium. The baseband module of the simulator implements the pseudorandom frequency hopping technique and the *inquiry*, *inquiry scan*, *page*, *page scan*, and LMP_MasterSlaveSwitch procedures as specified in the Bluetooth Baseband Specifications. Furthermore, the LMP module implements the link manager protocol and link control operations such as automatic repeat request (ARQ) as defined in the specifications. The TaskScheduler module allows to implement different schemes to handle intra/interpiconet communication and topology formation schemes, by relying on the primitives offered by the lower layers. At the TaskScheduler level, we implemented all the procedures which constitute SHAPER-OPT as described in Sections V and VI. Several primitives and messages are exchanged among the various modules to perform the desired procedures.

We measured the behavior of the proposed procedures in both single-hop and multihop scenarios. In the first case, the nodes were randomly distributed in a square area of side 7 m. This way, with a $TR = 10$ m it is assured that all devices are in radio visibility. In this case, we also compared our results with those in [15].

In the multihop case, nodes were scattered 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 density of nodes is high. We adopted the interpiconet scheduling mechanism proposed in [25].

As for the parameters of the procedures in Fig. 2, we set: $T_{\text{inq}} = 10.24$ s, $T_{\text{inqsc}} = 1.28$ s, $T_{\text{pg}} = 1.28$ s, $T_{\text{pgsc}} \approx 1$ s, while T_{form} is randomly extracted between 10.24 s and 1.28 s. The two parameters related to T_{comm} are $T_{\text{comm}}^{\min} = 2.125$ s and $T_{\text{comm}}^{\max} = 5.95$ s, and have been selected so that each node spends a higher percentage of time performing network formation tasks rather than communication tasks. The time spent in the *communication* state can be adaptively increased after the initial topology formation phase. It can also be adjusted in order to adapt to the mobility and other dynamics of the network.

All figures reported in this section show the average value of the experiments performed with 95% confidence intervals.

A. Performance Comparison With Other Scatternet Formation Algorithms

As a first performance evaluation, we compared PT-SHAPER with the single-hop TSF algorithm implemented in Blueware, in the same scenarios. Both protocols give rise to a tree structure. First, we evaluated the mean formation time (MFT); that is the mean value of the time needed to obtain a unique connected tree. Fig. 6 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 needed for a coordinator to find neighboring nodes increases too.

On the contrary, one of the assets of PT-SHAPER is that it connects trees via both R and NR nodes (as shown in the example of Fig. 3). As the node density increases, the probability that two different nodes meet increases too. This gives rise to a

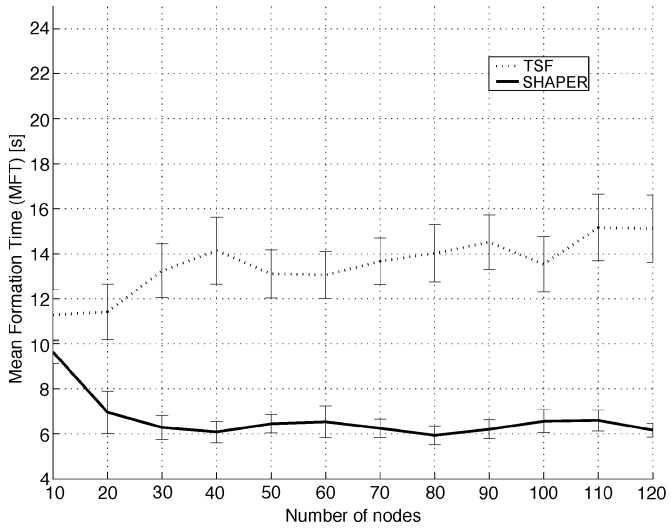


Fig. 6. Formation time: Performance comparison between TSF and SHAPER.

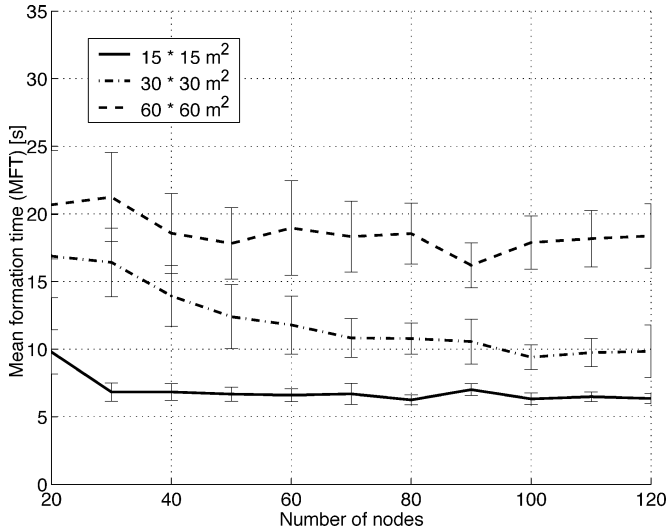


Fig. 7. Formation time, multihop scenarios.

MFT that remains around 6.5 s for $N \geq 20$. The *MFT* is slightly higher ($\leq$10 s) for low density networks.

The scatternets formed when N ranges between 10 and 120 have an average number of piconets ranging from 5 to 75 for PT-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 PT-SHAPER. The scatternets generated are similar in terms of number of piconets, number of slaves in a piconet, and number of leaves.

Since SHAPER operates also in multihop scenarios, the *MFT* has been evaluated as a function of the number of nodes for different sizes of the geographical area where the nodes are scattered (Fig. 7). As in the single-hop scenario, the MFT initially decreases as N increases and then remains flat. As expected, the MFT also increases with the size of the area. This result is also reported in Fig. 8, where the trend of the MFT as a function of the size of the geographical area is shown. We can conclude that, though the MFT depends on the node density in a given

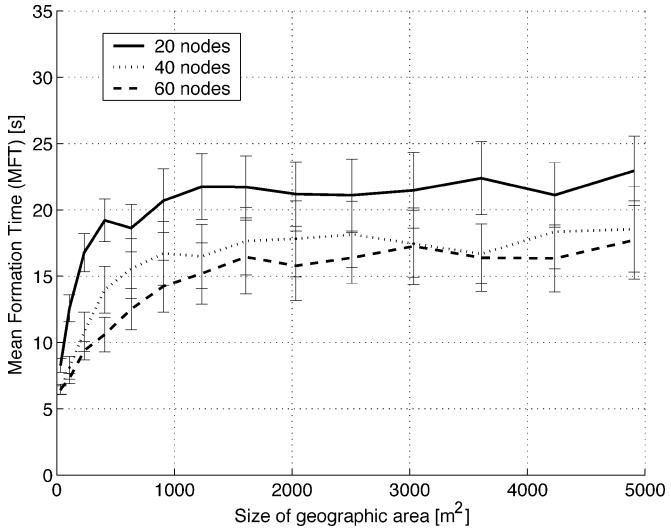


Fig. 8. Formation time as a function of the geographical area, multihop scenarios.

TABLE III
NUMBER OF STARTRECONF MESSAGES EXCHANGED

NUMBER OF NODES	40	80	100	120
SINGLE-HOP CASE	16	77	107	140
MULTI-HOP CASE	13	67	97	128

area, after a certain 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 is reported, both for single-hop and multihop scenarios. Although the reconfiguration procedures are time-consuming, they are necessary to merge different trees regardless of the status of the nodes (i.e., to merge trees also via NRs), as explained in Section V-C. This is the mechanism that makes SHAPER work also in multihop scenarios.

It is to be pointed out that, due to the time consuming *inquiry/inquiry scan* procedures in Bluetooth, the duration of the scatternet formation is a crucial performance metric. However, most of the solutions, thus far, proposed in the literature do not explicitly calculate the time needed to complete the algorithm. Only in [9], the formation time of three multihop scatternet formation algorithms is compared, namely, *Bluetree*, *Bluestar*, and *Yao*. The main problem with the three protocols above is that all of them rely on synchronized operations, i.e., on a discovery phase that must be executed by all nodes starting at the same time. This is not realistic in a distributed setting and is time consuming, since during the discovery phase no user data communications are possible among nodes. The authors of [9] show that a device discovery phase of at least 6 s is needed for the devices to acquire enough neighborhood information to guarantee a connected topology with probability close to 1, when the devices are scattered in an area of 900 m^2. According to our results in multihop settings, the time needed is probably higher when the size of the area increases. Moreover, additional time is needed for the distributed protocols to complete their operations and establish the scatternet links after the device discovery phase. The

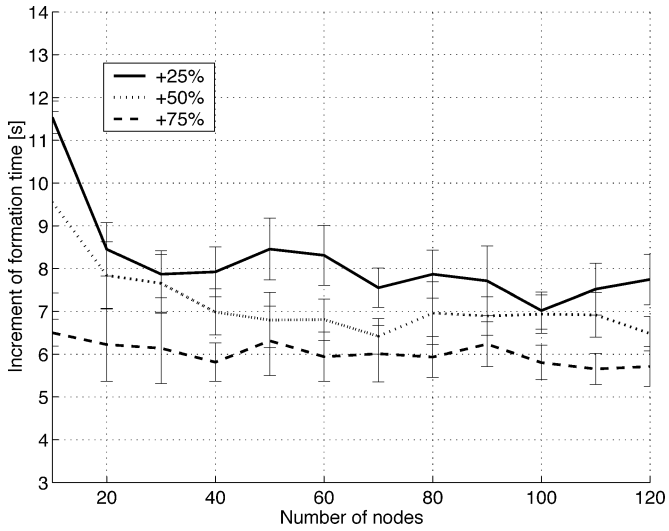


Fig. 9. Incremental formation time.

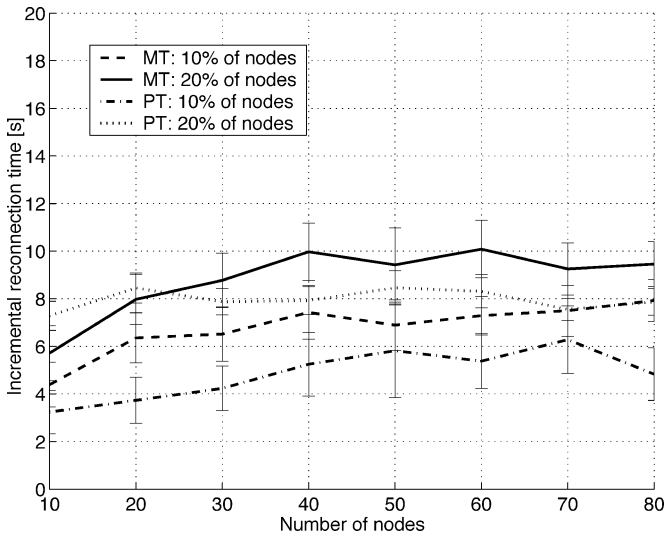


Fig. 10. Reconfiguration time in primitive and meshed topologies.

fastest of the protocols evaluated in [9] has additional formation time in the order of 1 s, but it does not guarantee that every piconet has seven slaves maximum [i.e., constraint in (2)]. The additional delays for the *Yao* and *Bluetree* protocols are in the order of 3–6 and 8–11 s, respectively.

B. Dynamic Behavior on Primitive Topologies

To measure the dynamic 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 Fig. 9, 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. Counterintuitively, 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 $N \geq 20$. This happens because when the density of nodes performing network formation tasks is higher, it is more likely that two nodes meet and establish a link and connect to the already formed scatternet quickly. The incremental formation time can be thought as the time needed for the last node to be included in the existing network.

C. Self-Healing Behavior of SHAPER-OPT

To evaluate the self-healing procedures of SHAPER-OPT, we measured the time needed to reconfigure the scatternet topology after an "en mass" arrival in case of both PT and MT and the behavior when nodes switch off. All nodes are in reciprocal visibility in an area of 7×7 m^2. We suppose that a node performs a communication task for a period of T_{comm}, uniformly distributed between 2.25 and 6.3 s. With reference to Fig. 10, we assume that a certain quota of the nodes arrive at time t_0, while the remainder arrives after the initial scatternet (PT or MT) has been formed. We considered 10% and 20% of the initial group of nodes. The figure reports the reconfiguration time due to this second burst of arrivals. Also in this case, the time to reconfigure both topologies is shown to be reasonably limited, though being in general higher for meshed topologies.

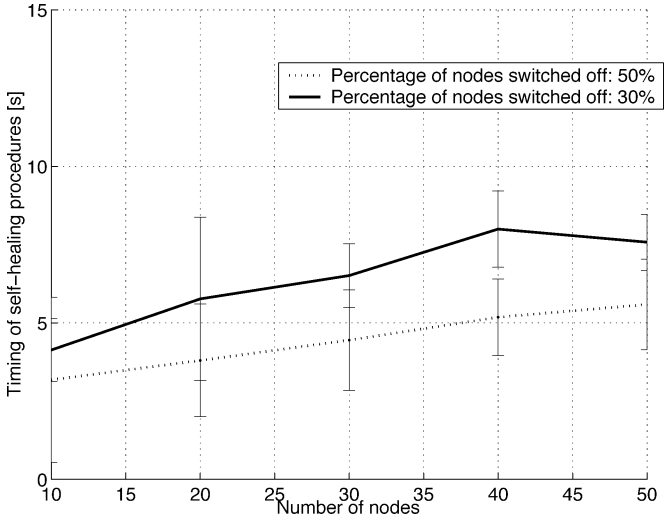


Fig. 11. Time needed for the self-healing procedures in meshed topologies.

In Fig. 11, we consider the opposite situation: given an MT scatternet, a certain quota of nodes switches off. We considered 30% and 50% of the initial amount of nodes. The remaining nodes execute the MT-SELF-HEALING procedure described in Section V-E. The figure shows the delays for updating the $tree\_ID$ of the LTs, as a function of the number of nodes that are switched off. The procedure to update the $tree_ID$s can be accelerated by using the updated version of the $\mathbf{B}$ matrix that contains the current scatternet topology.

D. Delay of the Optimization Procedures

Fig. 12 plots the time needed to execute different distributed procedures of SHAPER-OPT, as a function of the number of nodes, and starting from disconnected and neighborhood unaware nodes. All nodes are randomly distributed in an area of 15×15 m^2. The SHAPER curve shows the *MFT* needed to setup a connected tree-shaped scatternet. The other three curves report the time needed to broadcast the INIT message and the final

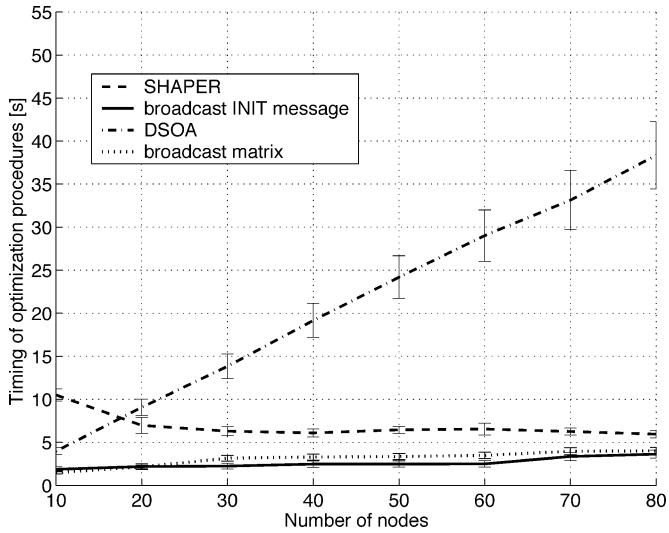


Fig. 12. Time needed by the SHAPER-OPT procedures.

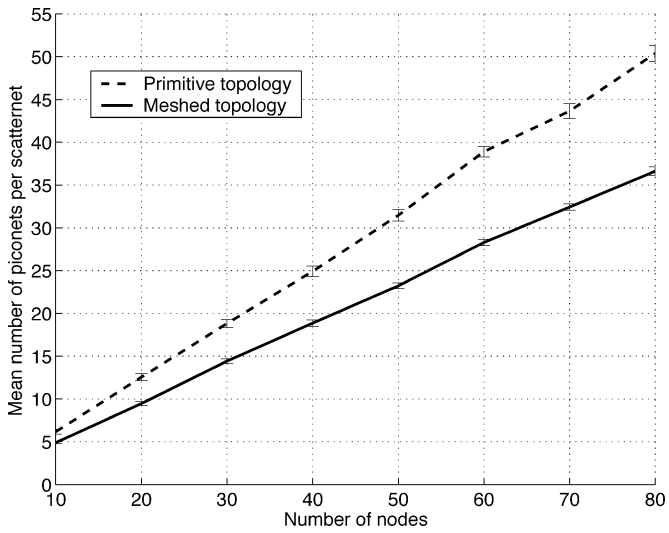


Fig. 13. Mean number of piconets for primitive and meshed topologies.

$\mathbf{B}^{\mathbf{N}}$ matrix, respectively. DSOA needs an amount of time that is directly proportional to the number of involved nodes and represents, in the overall SHAPER-OPT, the most time consuming procedure. It is however to be noticed that once SHAPER ends (after a few seconds), the network is fully connected and, as a consequence, DSOA optimization can be executed in parallel with user data communications. The time needed by DSOA can be reduced by assigning a higher priority to the packets that carry DSOA information in the scheduling mechanism and by defining ad hoc interpiconet scheduling mechanisms.

E. Topological Properties of Primitive and Meshed Topologies

A second set of simulations refers to the topological properties of PT and MT, respectively. It is to be noticed that these properties have a key role for the support of QoS. We consider a multihop scenario. The optimization metric used in DSOA is *equivalent normalized capacity* (as in problem **O3**), which aims at jointly maximizing the overall scatternet capacity and minimizing the number of hops between a generic source-destination pair (see also [6]). In Fig. 13, the mean number of piconets per scatternet is reported, as a function of the number of nodes. As the number of nodes increases, the mean number of piconets increases, thus enhancing the overall capacity of the scatternet. However, in the MT, the number of piconets per scatternet is reduced with respect to the PT. This happens because the optimization metric tends to tradeoff between increased capacity and increased interpiconet interference due to a higher number of piconets [6].

The number of piconets in our MT is almost exactly the same as that obtained by the *Bluetree* and *Yao* algorithms in [9]. *Bluestar* leads to a significantly lower number of piconets but, as stated above, piconets are not constrained to contain at most 7 slaves.

Other topology characteristics are reported in Fig. 14, where the number of links and the number of roles per node are compared for PT and MT. The mean number of links, which measures how meshed the scatternet is, changes from a value of 1.6

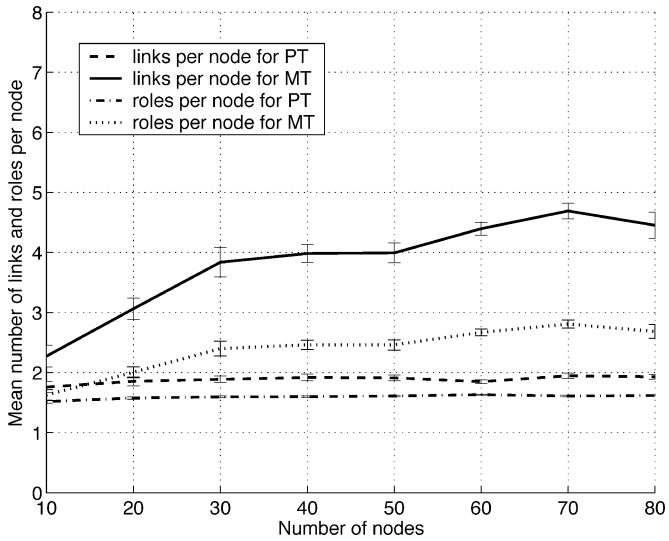


Fig. 14. Number of links and number of roles per node for primitive and meshed topologies.

for the PT to values ranging from 2.4 to 4.6 for the MT. If, for example, a scatternet has an average number of links equal to 3, this means that in average every master has three slaves and every slave belongs to three piconets. A more meshed topology leads to scatternets with shortest paths among nodes. Moreover, intuitively, when the topology is more meshed multiple paths are available among different piconets, which improves the resiliency and the reliability of the network.

The average number of roles per node is also an important measure of performance. Low values indicate that nodes spend their time in a few piconets (thus reducing the interpiconet scheduling delay). However, a more meshed topology (which is a desirable property) is usually associated to higher values of the average number of roles per node.

In Table IV, a summary of the comparison of the leading scatternet formation protocols is reported for the convenience of the reader, when 50 nodes participate in the scatternet formation

TABLE IV
COMPARISON OF DIFFERENT SCATTERNET FORMATION PROTOCOLS

	AVERAGE NUMBER OF PICONETS	AVERAGE NUMBER OF SLAVES PER PICONET	MEAN FORMATION TIME [s]	SELF-HEALING TIME (20 % OF NODES) [s]
PT-SHAPER	32	2	6	8
TSF [15]	30	1.8	13	N/A
ODBT [16]	17	3.4	5	N/A
BLUETREE [17]	25	2	14.5	N/A
SHAPER-OPT	22	4	6 + (26)	10
BLUESTAR [18]	15	4.3	7.5	Not Supported

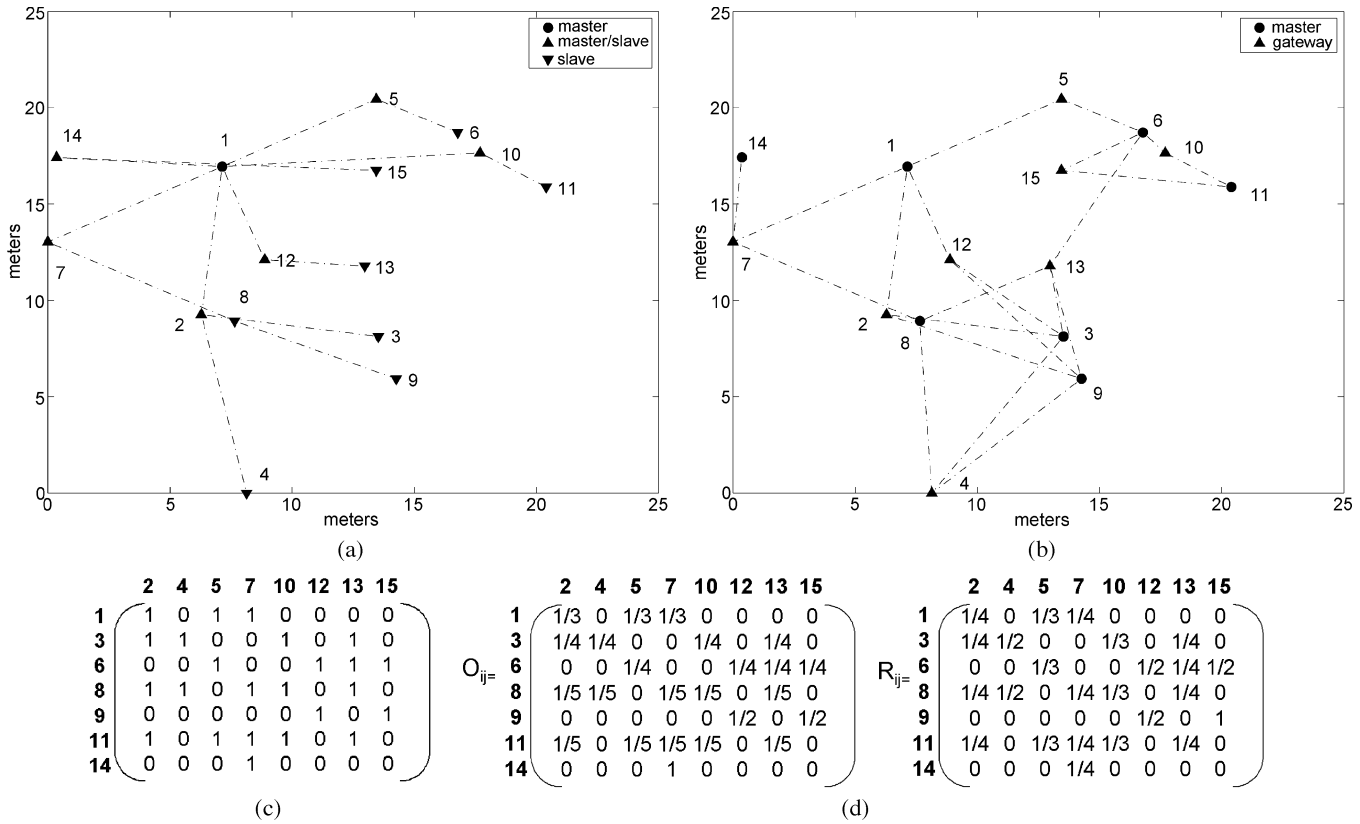


Fig. 15. Illustrative example. (a) Primitive topology. (b) Meshed optimized topology. (c) Matrix describing the optimized topology. (d) Matrices used to calculate the average normalized capacity.

process. The first four protocols give rise to a tree topology; last two create meshed topologies.

F. An Illustrative Example of Scatternet Optimization

Fig. 15 reports an optimization example of the scatternet topology. Fifteen nodes are scattered in an area of 25×25 m^2. The primitive topology is shown in Fig. 15(a) and the meshed topology, optimized with the equivalent normalized capacity metric, is shown in Fig. 15(b). The matrix that describes the locally optimal topology is shown in Fig. 15(c). All nodes are numbered according to the order they execute DSOA. By observing the matrix, it is easy to understand how the optimized topology is obtained by means of subsequent choices made by the nodes. Every node k decides its role and its connections only on the basis of what the previous $1, \ldots, k-1$ nodes have decided. The tree structure is used to convey information on decisions taken by the nodes. Node 1 is by default the master of node 2. Node 2 sends the matrix $\mathbf{B^2}$ to node 3. Node 3 also chooses to be a master for node 2 since it has no other choice (it is not a neighbor of node 1). Node 4 receives the matrix after node 3 sends it back to node 2 (see the tree structure) and selects to be a master for both nodes 2 and 3. The matrix is sent back to node 2 and then to node 1, that forward it to node 5. Node 5 simply selects to be a slave of node 1. Note that node 5 will also be selected as a slave by nodes 6 and 11. This way, the matrix reaches nodes 6, 7, etc., which take their decisions, until it reaches node 15. After node 15 has selected its role (slave) and the connections to establish (nodes 6 and 9), the new topology is defined. The final $\mathbf{B}^{15}$ is sent back to node 1 that broadcasts it to all the nodes in the tree. The links in the new topology are then established. The average capacity in (8) was calculated as described in [6]. As a scheduling rule [(9)], it

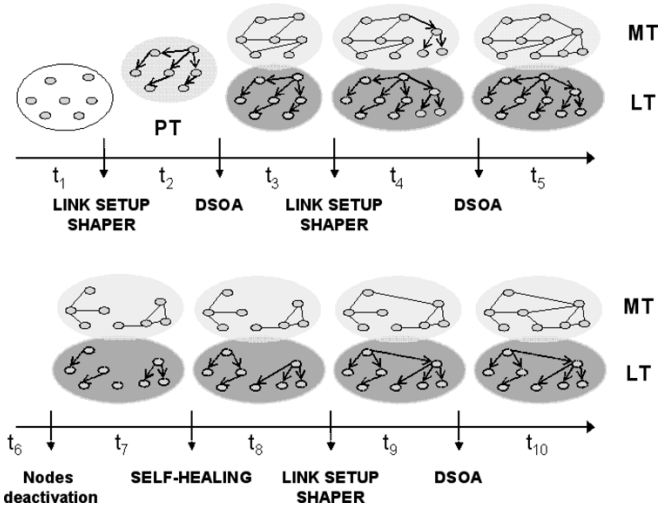


Fig. 16. Scatternet evolution with SHAPER-OPT.

is assumed that every master can offer the same capacity to any of its slaves (fair scheduling). Thus, the matrix O_{ij} in Fig. 15(d) is calculated, whose generic element o_{ij} represents the capacity the master i can offer to the slave j. Similarly, it is assumed that every slave spends the same amount of time in any of the piconets it belongs to. The matrix R_{ij} can be calculated whose element r_{ij} represents the capacity the slave j requires from the master i. The capacity of the link c_{ij} is then calculated as the minimum between r_{ij} and o_{ij}. With this model, the average capacity in (8) is 5.8833. For what concerns the routing rule in (9), a shortest path algorithm was applied in this example. This way, for this scatternet the average path length [(11)] is 2.1220. The final equivalent normalized capacity [(14)] of the scatternet is 2.7725.

VIII. Conclusion

Bluetooth has the potential to become a viable and competitive wireless technology for PANs. For this reason there is a need for suitable and efficient solutions for scatternet formation. This article proposed a scatternet formation paradigm that combines a quick tree scatternet setup with a distributed scatternet optimization algorithm, which gives rise to meshed topologies. A logical tree-structure is maintained to perform dynamical reconfiguration of the network. The procedures that constitute the building blocks of our proposed approach (SHAPER-OPT) are briefly summarized in the following. Fig. 16 reports an example of evolution of the scatternet topology. At time t_1 the nodes are all disconnected. Thanks to the *inquiry/inquiry scan* and *page/page scan* procedures links are setup and SHAPER connects them in a tree (t_2). DSOA is then executed on this PT giving rise to a LT=PT and to an optimized MT (t_3). New devices are included in the original scatternet after subsequent link setups by means of MT-SHAPER. At time t_5, DSOA is executed again in order to reoptimize the scatternet topology.

A qualitative example of the evolution when nodes disconnect is given in the bottom part of Fig. 16. At time t_6 some nodes disappear (e.g., due to mobility or deactivation). The scatternet gets disconnected (t_7) and the SELF-HEALING procedures are called. The logical trees are appropriately reconfigured (t_8), so that SHAPER can merge again the two scatternets (t_9) and DSOA (t_{10}) can reoptimize the new topology.

We implemented all these procedures in the Blueware platform and we performed an extensive simulation analysis. Results show how the network can be setup and dynamically reconfigured with limited delay. The optimization delay was compared with the time required to form a tree-shaped connected scatternet. A comparison of the topological properties of tree-shaped and meshed topologies was also provided.

Acknowledgment

The authors are thankful to the Editors of this JSAC Special Issue and to the anonymous referees for their helpful insights, which improved the quality of the paper. The authors would also like to thank G. Di Bacco for his support in deriving the simulation results.

References

[1] J. Haartsen, "The Bluetooth radio system," *IEEE Pers. Commun.*, vol. 7, pp. 28–36, Feb. 2000.

[2] P. Johansson, R. Kapoor, M. Gerla, and M. Kazantzidis, "Bluetooth an enabler of personal area networking," *IEEE Network*, pp. 28–37, Sept./Oct. 2001.

[3] G. Anastasi, E. Borgia, M. Conti, and E. Gregori, "IEEE 802.11 ad hoc networks: Performance measurements," in *Proc. 23rd IEEE Int. Conf. Distributed Computing Systems Workshops (ICDCSW'03)*, May 2003, pp. 758–763.

[4] S. Xu and T. Saadawi, "Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks?," *IEEE Commun. Mag.*, vol. 39, pp. 130–137, June 2001.

[5] P. Johansson, R. Kapoor, M. Kazantzidis, and M. Gerla, "Personal area networks: Bluetooth or IEEE 802.11?," *Int. J. Wireless Inform. Networks*, Apr. 2002. Special Issue on Mobile Ad Hoc Networks (MANETs) Standards, Research, Applications.

[6] T. Melodia and F. Cuomo, "Ad hoc networking with Bluetooth: Key metrics and distributed protocols for scatternet formation," *Ad Hoc Networks*, vol. 2, no. 2, pp. 109–202, Apr. 2004.

[7] R. Kapoor, M. Y. M. Sanadidi, and M. Gerla, "An analysis of Bluetooth scatternet topologies," in *Proc. IEEE Int. Conf. Communications (ICC 2003)*, 2003, pp. 266–270.

[8] D. Miorandi, A. Trainito, and A. Zanella, "On efficient topologies for Bluetooth scatternets," in *Lecture Notes in Computer Science*, vol. 2775, Proc. 8th IFIP TC6 Int. Conf. (PWC 2003), Sept. 2003, pp. 726–740.

[9] S. Basagni, R. Bruno, G. Mambrini, and C. Petrioli, "Comparative performance evaluation of scatternet formation protocols for networks of Bluetooth devices," *ACM Wireless Networks*, vol. 10, pp. 197–213, 2004.

[10] S. Zurbes, "Considerations on link and system throughput of Bluetooth networks," in *Proc. 11th IEEE Int. Symp. Personal Indoor Mobile Radio Communications*, vol. 2, 2000, pp. 1315–1319.

[11] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. La Maire, "Distributed topology construction of Bluetooth personal area networks," in *Proc. IEEE INFOCOM 2001*, Apr. 2001, pp. 1577–1586.

[12] C. Law, A. Mehta, and K.-Y. Siu, "Performance of a new Bluetooth scatternet formation protocol," in *Proc. Mobihoc 2001*, 2001, pp. 183–192.

[13] H. Zhang, J. C. Hou, and L. Sha, "A Bluetooth loop scatternet formation algorithm," in *Proc. IEEE Int. Conf. Communications (ICC 2003)*, May 2003, pp. 1174–1180.

[14] T.-Y. Lin, Y.-C. Tseng, K.-M. Chang, and C.-L. Tu, "Formation, routing, and maintenance protocols for the Bluering scatternet of Bluetooths," in *Proc. 36th Hawaii Int. Conf. System Science (HICSS-36)*, Jan. 2003, pp. 313–322.

[15] G. Tan, A. Miu, J. Guttag, and H. Balakrishnan, "An efficient scatternet formation algorithm for dynamic environments," in *Proc. IASTED Communications Computer Networks (CCN)*, Cambridge, MA, Nov. 2002.

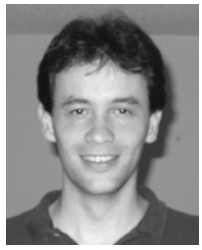
[16] E. Pagani, G. P. Rossi, and S. Tebaldi, "An on-demand Bluetooth scatternet formation algorithm," in *Proc. 1st IFIP TC6 Working Conf.—Wireless On-Demand Network Systems (WONS 2004)*, Jan. 2004, pp. 130–143.

[17] G. Zaruba, S. Basagni, and I. Chlamtac, "Bluetrees—scatternet formation to enable Bluetooth-based personal area networks," in *Proc. IEEE Int. Conf. Communications (ICC 2001)*, 2001, pp. 273–277.
[18] C. Petrioli, S. Basagni, and M. Chlamtac, "Configuring BlueStars: Multihop scatternet formation for Bluetooth networks," *IEEE Trans. Comput.*, vol. 52, pp. 779–790, June 2003.
[19] C. Petrioli and S. Basagni, "Degree-constrained multihop scatternet formation for Bluetooth networks," in *Proc. IEEE GLOBECOM 2002*, Nov. 2002, pp. 222–226.
[20] X.-Y. Li, I. Stojmenovic, and Y. Wang, "Partial delaunay triangulation and degree limited localized Bluetooth scatternet formation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, pp. 350–361, Apr. 2004.
[21] F. Cuomo, G. Di Bacco, and T. Melodia, "SHAPER: A self healing algorithm producing multihop Bluetooth scatternets," in *Proc. IEEE GLOBECOM 2003*, San Francisco, CA, Dec. 2003, pp. 236–240.
[22] Y. Liu, M. J. Lee, and T. N. Saadawi, "A Bluetooth scatternet-route structure for multihop ad hoc networks," *IEEE J. Select. Areas Commun.*, vol. 21, pp. 229–239, Feb. 2003.
[23] M. A. Marsan, C. F. Chiasserini, A. Nucci, G. Carello, and L. De Giovanni, "Optimizing the topology of Bluetooth wireless personal area networks," in *Proc. IEEE INFOCOM 2002*, vol. 2, June 2002, pp. 572–579.
[24] C. F. Chiasserini, M. A. Marsan, E. Baralis, and P. Garza, "Toward feasible distributed topology formation algorithms for Bluetooth-based WPANs," in *Proc. 36th Hawaii Int. Conf. System Science (HICSS-36)*, Jan. 2003, pp. 313–322.
[25] G. Tan, "Blueware: Bluetooth Simulator for NS," MIT Lab. Comput. Sci., Cambridge, MA, Oct. 2002.

Francesca Cuomo received the Laurea degree (*magna cum laude*) in electrical and electronic engineering and the Ph.D. degree in information and communications engineering from the University of Rome, "La Sapienza," Rome, Italy, in 1993 and 1998, respectively.

Since 1996, she has been an Assistant Professor in the Infocom Department, University of Rome. She participated in the European ACTS INSIGNIA Project dedicated to the definition of an Integrated IN and B-ISDN network, the IST WHYLESS.COM Project focusing on adoption of the ultrawideband radio technology for the definition of an open mobile access network, and the RAMON Project, funded by the Italian Public Education Ministry, which focused on the definition of a reconfigurable access module for mobile computing applications. She is now participating in the European IST ePerSpace Project focusing on the support of personalized audio/video services at home and everywhere. She is also involved in FIRB project virtual immersive communications (VICOM), where she is responsible for the research activities on the BAN and PAN networks. Her main research interests focus on broadband integrated networks, intelligent networks, architectures and protocol for wireless networks, mobile and personal communications, quality-of-service guarantees, and real-time service support in the wired and wireless Internet.

Dr. Cuomo is on the Editorial Board of the Elsevier *Computer Networks Journal* and she has served on technical program committees and as reviewer for several international conferences and journals including ACM Wireless Mobile Internet Workshop, IEEE ICC and GLOBECOM, the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, and the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS.



Tommaso Melodia received the Laurea degree in telecommunications engineering from the University of Rome, "La Sapienza," Rome, Italy, in 2001. He is currently working toward the Ph.D. degree as a Research Assistant in the Broadband and Wireless Networking Laboratory, Georgia Institute of Technology, Atlanta.

In 2001, he was a Research Assistant on wireless personal area networks with the University of Rome. His main research interests are in wireless sensor networks, wireless ad hoc networks, and personal and mobile communications.

Ian F. Akyildiz (M'86–SM'89–F'95) received the B.S., M.S., and Ph.D. degrees in computer engineering from the University of Erlangen, Nuremberg, Germany, in 1978, 1981, and 1984, respectively.

Currently, he is the Ken Byers Distinguished Chair Professor with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, and Director of the Broadband and Wireless Networking Laboratory. He is an Editor-in-Chief of *Computer Networks* (Elsevier) and for the newly launched *Ad Hoc Networks Journal* (Elsevier). His current research interests are in sensor networks, interplanetary Internet, wireless networks, and satellite networks.

Dr. Akyildiz is an Association for Computing Machinery (ACM) Fellow (1996). He served as a National Lecturer for ACM from 1989 until 1998 and received the ACM Outstanding Distinguished Lecturer Award for 1994. He received the 1997 IEEE Leonard G. Abraham Prize Award (IEEE Communications Society) for his paper entitled "Multimedia Group Synchronization Protocols for Integrated Services Architectures" published in the IEEE JOURNAL OF SELECTED AREAS IN COMMUNICATIONS (JSAC) in January 1996, the 2002 IEEE Harry M. Goode Memorial Award (IEEE Computer Society) with the citation "for significant and pioneering contributions to advanced architectures and protocols for wireless and satellite networking," the 2003 IEEE Best Tutorial Award (IEEE Communications Society) for his paper entitled "A Survey on Sensor Networks," published in the *IEEE Communications Magazine* in August 2002, and the 2003 ACM SIGMOBILE Award for his significant contributions to mobile computing and wireless networking.