Mobile Networks and Applications 9, 33–47, 2004

BlueMesh: Degree-Constrained Multi-Hop Scatternet Formation for Bluetooth Networks

CHIARA PETRIOLI
Dipartimento di Informatica, Università degli Studi di Roma, "La Sapienza," Italy

STEFANO BASAGNI
Department of Electrical and Computer Engineering, Northeastern University, MA, USA

IMRICH CHLAMTAC
Center for Advanced Telecommunications Systems and Services, The University of Texas at Dallas, TX, USA

Abstract. In this paper we describe *BlueMesh*, a new protocol for the establishment of *scatternets*, i.e., multi-hop wireless networks of Bluetooth devices. BlueMesh defines rules for device discovery, piconet formation and piconet interconnection so to generate connected scatternets with the following desirable properties. BlueMesh forms scatternets without requiring the Bluetooth devices to be all in each other transmission range. BlueMesh scatternet topologies are meshes with multiple paths between any pair of nodes. BlueMesh piconets are made up of no more than 7 slaves. Simulation results in networks with over 200 nodes show that BlueMesh is effective in quickly generating a connected scatternet in which each node, on average, does not assume more than 2.4 roles. Moreover, the route length between any two nodes in the network is comparable to that of the shortest paths between the nodes.

Keywords: Bluetooth technology, scatternet formation, ad hoc networks

1. Introduction

Mobile multi-hop wireless networks (often called ad hoc networks) are networks in which communication do not rely on any kind of fixed infrastructures, such as cables or base stations. In these networks all nodes can be mobile and rely on each other to forward packets to destinations not directly in their transmission range. Applications for this kind of networks range from collaborative and enterprise networking, military/tactical deployment, environmental monitoring, networks of wireless sensors, etc.

Among the new technologies for wireless communication in the unlicensed ISM band (2.4 GHz), the *Bluetooth* (BT) technology [8] is emerging as one of the most promising enabling technologies for ad hoc networks.

When two BT devices (or BT nodes, as we will also refer to them in the paper) come into each other communication range (i.e., they become *neighbors*), in order to set up a communication link one of them assumes the role of *master* of the communication and the other becomes its *slave*. This simple "single-hop" network is called a *piconet*, and may include many slaves, no more than 7 of which can be active (i.e., actively communicating with the master) at the same time. Mechanisms for "parking" and "unparking" of slaves are provided to a master for managing a piconet with more than 7 slaves.

All active devices in a piconet share the same channel (i.e., a frequency hopping sequence) which is derived from the unique ID and Bluetooth clock of the master. Communication to and from a device is always performed through the master of the piconet to which it belongs. Time-Division Duplex (TDD) is used for intra-piconet communications: transmissions occur in pairs of slots, the first of which is for master–slave communication, and the second is for the communication from the polled slave to the master.

A BT device can timeshare among different piconets. In particular, a device can be master of one piconet and slave in other piconets, or it can be slave in multiple piconets. Devices with multiple roles act as *gateways* between adjacent piconets, thus creating a multi-hop ad hoc network called a *scatternet*. Figure 1 shows the case where 11 BT devices have been partitioned into three piconets (A, C and D). Masters are represented by pentagons (surrounded by a large circle that represents their transmission radius), while slaves are depicted as small circles. Adjacent piconets are interconnected through either a common slave, termed a *gateway slave* (this is the case of piconets C and D which are joined by node 3), or through a pair of neighboring slaves, called in the following *intermediate gateways* (this is the case of piconets A and C, joined by nodes 1 and 2). In the latter case, interconnection requires that one of the two intermediate gateways (node 1 in the figure) becomes the master of a new piconet that includes the other intermediate gateway (node 2) as slave. With the creation of piconet B, the four piconets of figure 1 form a connected scatternet.

Although describing methods for device discovery and for the participation of a node to multiple piconets, the BT specification does not indicate any method for scatternet formation. Out of the solutions proposed in the literature so far for scatternet formation, those described in [5,7,10] assume the

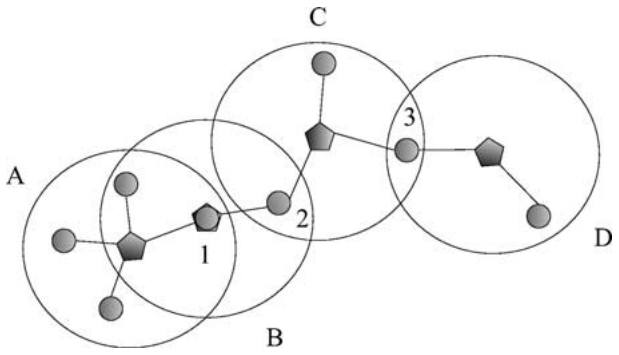

Figure 1. Four piconets forming a connected scatternet.

radio vicinity of all devices ("single-hop" topologies), which is not always the case in realistic scenarios. The solution proposed in [7] is based on a leader election process to collect topology information. Then, a centralized algorithm is run at the leader to assign the roles to the network nodes. In order to achieve desirable scatternet properties, the centralized scheme executed by the leader requires that the number of network nodes is $\leqslant 36$. The scatternet formation protocols presented in [5,10] run over single-hop topologies with no limitations on the number of nodes. However, the resulting scatternet is a tree, which limits efficiency and robustness.

The more general case of multi-hop topology is considered in [12]. The generated scatternet is yet again a tree. Moreover, the proposed protocol relies on a designated device to start the scatternet formation process, which renders the solution dependent on the proper functioning of a single node. To the best of the authors' knowledge, the only three solutions for scatternet formation in multi-hop topologies that produce a mesh are those presented in [2,9,11]. In [2] the generated scatternet is made up of piconets that may have more than 7 slaves. This may result in performance degradation, as slaves need to be parked and unparked in order for them to communicate with their master. The problem of reducing the total number of slaves to less than 7 is solved in [9] (and also in [6]) by first applying degree reduction techniques to the network topology graph, and then executing the actual scatternet formation protocol on a topology in which no node has more than 7 neighbors. These techniques require each node to be equipped with additional hardware that provides to the node its current (geographic) location (e.g., a GPS receiver). Beyond being potentially expensive, this solution is not feasible when such extra hardware is not available. The scatternet formation scheme proposed in [11], *BlueNet*, produces a scatternet whose piconets have a bounded number of slaves. The connectivity of the resulting scatternet is not guaranteed (i.e., not all the BlueNets are connected, even when the initial topologies are). No thorough performance evaluation of the protocols described in [2,9,11] is provided.

In this paper we present and evaluate the performance of a new scatternet formation protocol for multi-hop Bluetooth networks that overcomes the limitations of the solutions listed above. Beyond being fully distributed, our protocol does not require nodes to be in the transmission range of each other and generates a scatternet whose topology is a connected mesh rather than a tree. No piconet in the generated scatternet has more than 7 slaves, thus avoiding the overhead due to the need to park and unpark slaves, and no extra hardware is required, thus making the protocol a viable solution for any network of Bluetooth devices. Given that the topology of the generated scatternet is a mesh, we name the protocol *BlueMesh*.

The idea behind BlueMesh is to generate a connected scatternet by selecting some masters among the network nodes, and allowing each master to select at most 7 slaves. The selection of the Bluetooth masters is driven by the suitability of a node to be the "best fit" for serving as a master. This is realized by equipping each node with a *weight*, i.e., a real number $\geqslant 0$, that each node can dynamically compute locally, depending on some parameters of interest to the prevailing network application (e.g., battery power, mobility, node resources, etc.). The selection of the slaves is performed in such a way that if a master has more than 7 neighbors, it chooses 7 slaves among them so that via them it can reach all the others. Once masters and slaves are selected, i.e., piconets are formed throughout the network, gateways are chosen so that there is an inter-piconet route between all masters that are at most three hops away. This condition ensures the connectivity of the BlueMesh scatternet.

In order to realize the scatternet formation as described, BlueMesh proceeds in two phases:

1. The first phase, *topology discovery*, concerns the discovery of each node's one and two-hop neighboring devices. This phase is executed at the device start of operation.

2. The phase of *scatternet formation* takes care of piconet formation and their interconnection to form a scatternet. This phase starts as soon as the previous phase is over and proceeds in successive iterations through which connectivity is achieved progressively.

Through the use of simulations we have demonstrated that BlueMesh is effective in quickly producing a scatternet which is a connected mesh. We observe that, independently of the number $n \leqslant 210$ of network nodes, the average number of iterations needed to complete the second phase of the protocol never exceeds 4.6. Moreover, BlueMesh scatternets have the desirable properties that nodes have no more than 2.4 roles (on average), and that its routes are not significantly longer than the shortest routes between any two nodes in the *geographic network topology*, i.e., in the topology of the network consisting of all BT devices and the links between the nodes that are in each other communication range.

This work aims at proposing a basic solution to the general problem of scatternet formation, central to the application of the Bluetooth technology to ad hoc networking. In the description and demonstration of our protocol we have made the assumptions that nodes are scattered in the two-dimensional space and form a *unit disc graph*, i.e., a graph in which the presence of a link between two nodes depends on whether their distance is less than the nodes' transmission range or

not. We also assume that the topology of the network does not change during the execution of the protocol.

The rest of the paper is organized as follows. Section 2 describes the basics of the Bluetooth technology. Sections 3 and 4 describe the two phases of BlueMesh and prove their correctness. Simulation results are shown in section 5. Conclusions are finally drawn in section 6.

2. The Bluetooth system: basics

Since piconet formation is a crucial step of our protocol, in this section we briefly describe the procedures of the Bluetooth technology that enable it. For a detailed description of the Bluetooth system, the reader is referred to [8].

Piconet formation is performed in two steps: first, devices are made aware of their neighboring nodes (device discovery). Then, information is exchanged to set up a link between a candidate slave and a candidate master (link establishment). According to the current BT specification, the former step is accomplished by means of the *inquiry* and *inquiry scan* procedures, while the latter requires the *page* and *page scan* procedures.

For device discovery to happen, two neighboring devices have to be in "opposite" modes, namely, one must be the inquirer, the discovering device, and the other device has to be willing to be discovered. These modes are implemented in BT by having the inquirer in inquiry mode, and the other device in inquiry scan mode. The inquirer transmits inquiry ID packets asking neighboring devices to identify themselves and to provide synchronization information needed for link establishment at a later time. To minimize the device discovery time, the BT specification states that ID packets must be very small (i.e., they include only the General Inquiry Access Code, GIAC, and nothing else) and that they must be transmitted over the frequencies of a predefined inquiry/inquiry scan frequency hopping sequence, changing frequencies at a high rate (twice a slot). A device in inquiry scan hops among different frequencies at a very low rate, thus increasing the probability of an handshake on the same frequency of the inquirer. As soon as an ID packet is received at a device in inquiry scan mode, the device computes a backoff interval and starts listening again. Only when an ID packet is received after the backoff phase the unit in inquiry scan mode will send an FHS (Frequency Hop Synchronization) packet containing its identity and synchronization information (its BT clock). (The backoff interval is introduced to decrease the probability of FHS packets collision in case an ID packet is received by two devices scanning on the same frequency.)

The described inquiry procedures lead to an asymmetric knowledge of two neighboring devices: The inquirer identity is not known at the device that received an inquiry ID packet. After successful reply from the device in inquiry scan mode, instead, the inquirer knows the identity and clock of the neighbor that just replied.

The inquiry procedures allow a discovering device v to collect the BT addresses and clocks of all devices u that respond to its inquiry message. This enables v to estimate the frequency hopping sequence used by its neighbors and thus to invite them to join its piconet as slaves. This invitation is accomplished by means of the page procedures. As before, in order for two neighboring devices to successfully communicate, one must be in page mode, and the other in page scan mode. By definition, the device that goes to page mode is the master. For a device v to enroll a neighbor u in its piconet, two things must be ensured: Device u has to be in page scan mode, and v must be aware of u's ID and clock. If this is the case, v transmits a page ID packet on u's frequencies, containing u's address. When u, which is in page scan, receives such a packet, it immediately acknowledges it. At this point v transmits to u an FHS packet that carries all the required information for u to synchronize on v's own frequency hopping sequence. Finally, the two devices exchange all the information for setting up a link and a piconet is formed with v as the master and u as its slave.

3. Topology discovery

The first phase of BlueMesh, the topology discovery phase, concerns the discovery of the two-hop neighborhood of a node. Specifically, by the end of this phase, each node knows the ID and the weight of all its neighbors up to two hops away, and through which one-hop neighbors it can reach its two-hop neighbors.

This phase is performed at each node in two steps. In the first step each node discovers its one-hop neighbors. In the second step each pair of neighboring devices exchange information on their one-hop neighbors, thus getting to know all the nodes which are at most two hops away.

For the correct functioning of BlueMesh we need a *symmetric knowledge* among neighbors, namely, we need that if a node u is aware of a node v at most two hops away, then v also knows about u.

We start by describing a method for the discovery of one-hop neighbors (first step). The inquiry procedure described in the specification indicates how a device in inquiry mode can trigger a peer device in inquiry scan mode to send its ID and the synchronization information needed for link establishment (see section 2). However, no indication is given on how to guarantee that neighboring devices are in opposite inquiry modes which is the needed condition for them to communicate. Furthermore, the inquiry message broadcast by the inquirer device does not contain any information about the inquirer itself, thus, once two neighboring devices complete an inquiry handshake, only the inquirer knows the identity of the device in inquiry scan mode, not viceversa (asymmetric knowledge).

To overcome these drawbacks and attain mutual knowledge of ID and weights among one-hop neighbors, we use a mechanism similar to that introduced in [7].

The following procedure describes the operations performed at each device v as it enters the topology discovery phase of the protocol.

```
DISCOVERY(v)
 1  T_disc ← ℓ_td
 2  while T_disc > 0
 3    do if RAND(0,1) < 0.5
 4         then INQUIRYMODE
 5              COMPUTE(T_inq)
 6              INQUIRY(min(T_inq, T_disc))
 7              INQUIRYSCANMODE
 8              COMPUTE(T_scan)
 9              INQUIRYSCAN(min(T_scan, T_disc))
10         else INQUIRYSCANMODE
11              COMPUTE(T_scan)
12              INQUIRYSCAN(min(T_scan, T_disc))
13              INQUIRYMODE
14              COMPUTE(T_inq)
15              INQUIRY(min(T_inq, T_disc))
16  EXIT
```

The generic device v that executes procedure DISCOVERY, sets a timer T_{disc} to the selected length ℓ_{td} of the topology discovery phase. This timer is decremented at each clock tick (namely, T_{disc} keeps track of the remaining time till the end of this phase).

Device v then randomly enters either inquiry or inquiry scan mode, and computes the length of the phase it is entering (T_{inq} or T_{scan}). While in a given mode, device v performs the inquiry procedures as described by the BT specification. The procedures that implement the inquiry mode (procedure INQUIRY) or the inquiry scan mode (procedure INQUIRYSCAN) are executed for a time which is randomly computed (T_{inq} and T_{scan}, respectively), never exceeding T_{disc}. Upon completion of an inquiry (inquiry scan) phase, if $T_{\text{disc}} > 0$, a device switches to the inquiry scan (inquiry) mode. To allow each pair of neighboring devices to achieve a mutual knowledge of each others' ID and weight, the scheme requires that whenever a device in inquiry (inquiry scan) mode receives (sends) an FHS packet, a temporary piconet is set-up by using the BT standard page procedures, and devices exchange their ID and weight, together with the synchronization information required for further communication. As soon as this information has been successfully communicated the piconet is disrupted.

The method of setting up two-node temporary piconets for information exchange is used throughout the paper whenever two neighbors need to exchange any kind of information regarding BlueMesh operation.

The effectiveness of the described mechanism in providing the needed mutual knowledge to pairs of neighboring devices relies on the idea that by alternating inquiry and inquiry scan mode, and randomly selecting the length of each inquiry (inquiry scan) phase, there is high probability that any pair of neighboring devices will be in opposite mode for a sufficiently long time, thus allowing the devices to discover each other. Setting up a two-node temporary piconet provides the means of exchanging information in a symmetric way.

Once a node v exits the procedure DISCOVERY, it locally builds a *neighbor list* $N(v)$ of the discovered neighbors. This marks the beginning of the second step of the topology discovery phase. Node v then exchanges its list $N(v)$ with all its neighbors, so that each node attains knowledge also of its two-hop neighbors. The mechanism we use for neighbor list exchange is again the temporary set up of a piconet between every pair of neighboring nodes. To set up a piconet between two neighbors, one of the nodes has to page the other one. According to the BT specification, this requires the initiator to be in *page* mode, while the other one has to be in the opposite *page scan* mode. In order to achieve the complete exchange of the neighbor lists we have to guarantee that every pair of nodes are in opposite modes. This is obtained by having the nodes executing the following *list exchange* protocol. Upon completing the one-hop neighbors discovery process, a node v checks whether it has the bigger weight among its neighbors in $N(v)$. If this is the case, that node, called in the rest of the paper an *init* node, executes the following procedure.

```
PECKORDER(v)
1  PAGEMODE
2  for each smaller u in N(v)
3     do PAGE(u, v, N(v), N(u))
4  EXIT
```

An init node goes to page mode and starts paging all its neighbors (which, by definition, are "smaller neighbors", i.e., nodes with a smaller weight) setting up temporary piconets with each one of them so that they can exchange the lists of neighbors. Here the parameter of the page (i.e., the information exchanged via the temporary piconets set up by v and each of its neighbors u, one at a time) are the destination node u, the initiator of the page v and their neighbor lists. Symmetrically, a non-init node u executes the following procedure.

```
SUBNODE(u)
1  PAGESCANMODE
2  for each bigger v in N(u)
3     do WAIT PAGE(u, v, N(v), N(u))
4  PECKORDER(u)
```

Node u goes to page scan mode and waits for a page from all its neighbors with bigger weight ("bigger neighbors"). As soon as all the bigger neighbors have exchanged the list of their neighbors with it, node u itself becomes the "bottom of the pecking order", i.e., being now the bigger node among those with which it has to exchange the neighbor list, it switches to page mode, and starts setting up temporary piconets to exchange $N(u)$ with all its smaller neighbors (if any).

3.1. Topology discovery correctness

To prove the correctness of the topology discovery phase we show that this phase terminates with each node having a symmetric knowledge of its one-hop and two-hop neighbors. The correctness of this phase is shown by proving correct its two steps separately.

As a result of the first step of the device discovery phase each device v has the list of the IDs, weights and synchro-

nization information of all the devices it was able to discover within T_{disc}. Only statistical guarantee can be provided on a device being able to discover all its one-hop neighbors. Given that all devices enter the topology discovery phase in a given time interval (t_0, t_1), $t_1 < T_{\text{disc}}$, the greater the value of T_{disc}, the higher the probability for a device to discover all its neighbors, the longer the discovery phase duration.

To prove the correctness of the first step, we prove that it terminates with all devices having a symmetric view of their one-hop neighbors (i.e., for each pair of neighbors v and u, v knows u's ID and weight, and vice versa).[1]

Proposition 1. Each device v terminates the execution of the first step of the topology discovery phase. If device v has discovered a neighbor u, it knows its ID and weight. Also, if device v discovered device u, then u discovered v.

Proof. The proof of the first part of the claim is straightforward as each device exits phase one after timer T_{disc} has expired (line 2 of procedure DISCOVERY). The second part of the claim can be derived from the protocol operations: whenever a device discovers a neighbor, the two devices create a temporary piconet. The two neighbors then exchange ID and weight and the piconet is torn down. The construction of the piconet and the information exchanged during its lifetime guarantee the symmetric knowledge of the piconet nodes. □

The correctness of the neighbors list exchange protocol is proven by the following result.

Proposition 2. Each node v terminates the execution of the second step of the topology discovery phase knowing all its one-hop and two-hop neighbors. If node v knows the ID and weight of a node u which is at most two hops away, then node u also knows the ID and weight of node v. If u and v are two hops apart, then they also know which nodes interconnect them.

Proof. We start by noticing that the two steps can interfere in the sense that a node executing the procedure PECKORDER, i.e., a node v that is executing the second step of the topology discovery phase, may attempt to page a node u that is still executing the first step. However, this interference does not compromise the correctness of this phase, since, for this to happen, node v and u must already have discovered each other, so that the page from v does not have any effect at u. Eventually node u exits the execution of the first step (proposition 1) of the protocol and having a bigger neighbor v, it starts executing the procedure SUBNODE, i.e., it will be ready to accept pages.

[1] In case packets are lost or corrupted during transmissions, by the end of the topology discovery phase some devices may have an "asymmetric view" of their neighborhood. This does not affect the correctness of the following step of the topology discovery phase, as well as the correctness of the following phase of the protocol, provided that timeouts are introduced in those phases to re-establish the symmetry.

This said, the proof of correctness of this step proceeds by induction on the weight-based ordering of the networks nodes that lists the init devices first and the non-init devices then, all ordered by decreasing weight. We assume that there are no transmission errors, i.e., that all attempted pages are successful and lead to the exchange of all the information needed for the formation of a (temporary) piconet.

Let B be the set of all Bluetooth devices, $|B| = n$, and let $I \subseteq B$ be the set of the init devices. We enumerate the devices in B according to the following one-to-one correspondence $\text{pecking} : B \to \{1, \ldots, n\}$, defined as follows:

$$\text{pecking}(x) = \begin{cases} |I| - \text{ord}_I(x) + 1 & \text{if } x \in I; \\ n - \text{ord}_{B \setminus I}(x) + 1 & \text{if } x \in B \setminus I, \end{cases}$$

where $\text{ord}_S : S \to \{1, \ldots, |S|\}$ is defined so that $\text{ord}_S(x) = i$ if x is the ith *order statistics* in the set S according to the devices' weight (as customary, ties are broken by using the devices' unique ID) [4]. Function pecking orders the devices in such a way that the init devices come first, sorted in decreasing order with respect to their weight, and all the other devices come after, sorted in decreasing order according to their weight.

We now proceed by induction on the pecking order $\text{pecking}(x) = k \leqslant n$, $x \in B$, i.e., the number of consecutive (with respect to pecking) devices.

The induction base case is comprised of the init devices, i.e., those devices i such that $\text{pecking}(i) \leqslant |I|$. Since there is always at least an init device, the set I is never empty. The proof for the base case is based on the code of the procedure PECKORDER: having no bigger devices, an init node immediately goes to page mode (line 1, procedure PECKORDER) and exchanges its neighbor list with each of its neighbors (which are smaller, hence in page scan mode: line 1, procedure SUBNODE).

It finally exits the execution of this step of the topology discovery phase, i.e., it terminates the execution of the phase itself, now aware of all neighbors at most two hops away (line 4, procedure PECKORDER) and of all the one-hop neighbors through which a given two-hop neighbor can be reached.

Let us now assume that all nodes x such that $\text{pecking}(x) \leqslant h$, $h > |I|$, have exchanged their $N(x)$ with all their bigger neighbors, and, by executing the procedure PECKORDER (possibly from executing the procedure SUBNODE) have also exchanged their neighbor list with all their smaller neighbors and have exited the execution of this phase of BlueMesh (knowing all neighbors at most two hops away and, for the two-hop neighbors, the one-hop neighbors through which they can be reached).

Consider the device y such that $\text{pecking}(y) = h + 1$. By definition of pecking all y's bigger neighbors z are such that $\text{pecking}(z) < \text{pecking}(y)$ (bigger neighbors come first). Having at least a bigger neighbor, node y is executing the procedure SUBNODE, i.e., it is in page scan mode (line 1, procedure SUBNODE).

By inductive hypothesis, every neighbor z of y has exited the device discovery phase (line 4, procedure PECKORDER) after having paged all its smaller neighbors (which include y;

line 2, procedure PECKORDER) and exchanged with them their neighbor lists (line 3, procedure PECKORDER).

This implies that node y has exchanged $N(y)$ with all its bigger neighbors. At this point, as the bottom of the pecking order, y executes the procedure PECKORDER (line 4, procedure SUBNODE) and pages its smaller neighbors (which having y as bigger neighbor are in page scan mode) to exchange with them the neighbor list. Node y finally exits the topology discovery phase (line 4, procedure PECKORDER) knowing all neighbors at most two hops away. Via the neighbor list each node v also knows which are the nodes that connect it to all nodes u that are two hops away, and vice versa. □

4. Scatternet formation

In this section we describe the scatternet formation phase of the BlueMesh protocol. Aim of this phase is to divide the BT devices into piconets and to interconnect them through gateways to form a connected scatternet. Scatternet connectivity is guaranteed by establishing an inter-piconet route between any two masters that are at most three hops away [3, theorem 1].

This phase of BlueMesh proceeds in successive iterations. Each iteration is executed by the network nodes that have not yet exited the execution of the protocol at some previous iteration. Let us call $G_i = (V_i, E_i)$ the network topology graph at iteration $i, i \geqslant 1$. G_1 is simply the geographic network topology. Each of the $G_i, i > 1$, is the subgraph of G_1 that spans the nodes of V_1 that did not exit the execution of BlueMesh in one of the previous iterations. In each iteration i, piconets are formed from the nodes in the topology graph G_i. The interconnection of two piconets is achieved either via a common slave (a *gateway slave*) or via a pair of neighboring slaves, called *intermediate gateways*, one of which belongs to one piconet and the other to the other piconet. In the following we call *adjacent piconets* those piconets that can be interconnected through a gateway slave or through a pair of intermediate gateways. Gateway slaves are selected in the current iteration so that the piconets they belong to are joined. Masters then proceeds to select the intermediate gateways between adjacent piconets not yet interconnected.

The intermediate gateways are the nodes that proceed onto the next iteration. All masters, slaves that have not been selected as gateways, and the gateway slaves exit the execution of BlueMesh at this time. BlueMesh terminates when all nodes have exited the execution of the protocol.

We now describe in detail the operations performed by a node v which is executing BlueMesh at the generic iteration i, $i \geqslant 1$. (In describing this phase, we assume that BT devices know their ID, their weight and the ID and the weights of all their one-hop and two-hop neighbors. See section 3.)

Each iteration of the protocol is performed locally at each node v and it is made up of two parts: *Role selection* (for piconet formation) and *gateway selection*.

First part: Role selection

Role selection is executed by every node at the very beginning of each iteration i (in the case of the first iteration role selection is performed as soon as the topology discovery phase is over). Based on its weight and the weight of its one-hop neighbors, a node determines whether it is an init node in G_i. Only init nodes go to page mode. All the other nodes go to page scan mode.[2] An init node v executes the procedure MASTER(v) whereas every non-init node u executes the procedure NONINIT(u). In all procedures, $N(v)$ denotes the set of all v's one-hop neighbors, $S(v)$ is the set of at most 7 slaves selected by each master v and $C(v)$ denotes the set of v's bigger neighbors that are slaves and v's smaller neighbors.

```
MASTER(v)
1  myRole ← master
2  PAGEMODE
3  COMPUTES(v)
4  for each u in S(v)
5     do PAGE(u, v, master, true, NIL)
6  for each u in C(v) \ S(v)
7     do PAGE(u, v, master, false, NIL)
8  EXIT
```

Each master v chooses as slaves those neighbors in $C(v)$ that "cover" all the other neighbors in the sense that if a neighbor u is not selected as v's slave, then at least one of u's neighbors has been selected by v. Such a coverage is always possible by selecting at most 5 slaves (Lemma 1).

Slave selection is performed at each master as follows.

```
COMPUTES(v)
1  S(v) ← ∅
2  U ← C(v)
3  while U ≠ ∅
4     do x ← bigger in U(v)
5        S(v) ← S(v) ∪ {x}
6        U ← U \ N(x)
7  S(v) ← S(v) ∪ GET(7 − |S(v)|, C(v) \ S(v))
```

The function GET(m, W) returns a set of m nodes from the set W (for instance, the m smaller ones, or randomly chosen, or the bigger ones, etc.).

Procedure COMPUTES implements a greedy approach for computing $S(v)$. One of v's neighbors in $C(v)$ (e.g., the one with the biggest weight) is selected as its slave, say node x. The procedure is then executed again on the set of all nodes in $C(v) \setminus \{x\}$, which are not covered by x. This rule allows node v to select up to 5 among its neighbors in $C(v)$ through which all other neighbors can be reached. Function GET is finally used for selecting additional slaves to a maximum of 7 ($|S(v)| \leqslant 7$).

Once an init node executing MASTER(v) has selected the neighbors in $S(v)$, it starts paging all its neighbors u in $C(v)$ to tell them whether it has selected them or not. The parameters of the page are the destination node u, the initiator of the

[2] Given the definition of weight and given the total ordering of the set of all nodes' weights, there is always at least an init node.

page, v, the initiator role r (either master or slave), a Boolean j which is *true* if and only if v selected u for its piconet and, finally, the list of masters M that have selected v (if v is a master, M is the empty list). Once all the paging is done, node v exits the execution of the role selection part of this iteration.

Every non-init node v executes the following procedure NONINIT(v).

```
NONINIT(v)
 1  PAGESCANMODE
 2  for each bigger neighbor w
 3     do WAIT PAGE(v, w, r, j, NIL)
 4        if r = master and j = true
 5           then myRole ← slave
 6                JOIN(w)
 7                M ← M ∪ {w}
 8  if myRole = slave
 9     then PAGEMODE
10        for each w in C(v)
11           do PAGE(w, v, slave, false, NIL)
12        PAGESCANMODE
13        for each w in C(v)
14           do WAIT PAGE(v, w, r, j, NIL) from smaller w
15              WAIT PAGE(v, w, sslave, false, M)
                from bigger w
16        PAGEMODE
17        for each neighbor slave w
18           do PAGE(w, v, slave, false, M)
19        PAGESCANMODE
20        for each smaller slave w
21           do WAIT PAGE(v, w, slave, false, M)
22        EXIT
23  else MASTER(v)
```

Node v starts by waiting for all bigger neighbors w to page it and communicate their role. If node w is a master and it has selected node v, then v joins w's piconet. Once all bigger neighbors have paged, node v knows whether it has already joined the piconet of some bigger masters or not. In the first case, node v is the slave of the bigger masters that selected it. In the latter case node v is going to be a master itself. In any case, node v goes to page mode and starts paging smaller neighbors that may need the information about v's role to decide their own. It will also page all the bigger neighbors that are slaves, since they need to know about v's role for interconnecting piconets later in the iteration. (Bigger masters already exited the execution of this part of the iteration.) The rest of the procedure depends on whether node v has decided to be a master or to be a slave. In the former case, v acts exactly as if it was an init node (line 23). If node v is a slave, after having informed its smaller neighbors and bigger neighbors that are slaves (i.e., nodes in $C(v)$) of its decision, it goes back to page scan mode. It then waits for smaller neighbors to communicate the decision they make on their role, and for bigger neighbors that are slaves to inform it about their complete list of masters. Once this exchange has been completed, node v knows the role of all its neighbors, and which (if any) of its neighbor masters have invited it to join their piconet. In order for the role selection to be completed, v's list of masters has to be distributed to all its slave neighbors, and v has to become aware of their list of masters. Therefore, v goes to page mode and communicates its list of masters to all its slave neighbors, and finally it switches again to page scan mode waiting for its smaller neighbors that are slaves to communicate their list of masters. When all these operations have been completed, each node has a knowledge of its neighbors ID, role, and, in case they are slaves, of all the piconets they belong to. Node v then exits the role selection part of this iteration and moves to the gateway selection part.

Second part: Gateway selection

When the role selection part of an iteration is over, the nodes start the gateway selection phase of the iteration. In this part all slaves communicate to their master(s) information about the roles of their neighbors, their neighbors' list of masters, and whether some of their neighboring masters selected them as slaves. Based on this information each master decides which slaves to select as gateways and to which piconet in order to obtain a connected scatternet. If a pair of masters have selected common slaves, they choose the bigger one among them as gateway slave. This is the preferred way to interconnect adjacent piconets. Whenever no gateway slave can be selected to interconnect adjacent piconets, intermediate gateways are selected, again based on their weight (e.g., so that the sum of their weights is maximized, or the minimum weight is maximized, or any other unambiguous selection rule).

Upon completion of these operations, the gateway slaves, together with the masters and the non-gateway slaves, exit the execution of the BlueMesh protocol. The intermediate gateways proceed to iteration $i+1$ to form new piconets that interconnect them, hence providing connectivity between the piconets they affiliated to in iteration i. The functioning of the second phase of BlueMesh is illustrated by the following example. Let us consider the network of figure 2, where a link between two devices indicates that the two nodes have discovered each other during the topology discovery phase and the number aside each node indicates its weight.

Being the nodes with the bigger weight in their neighborhood, devices 30 and 36 are init nodes. They are masters and switch to page mode (procedure MASTER, lines 1 and 2). All the other nodes go to page scan mode (procedure NONINIT,

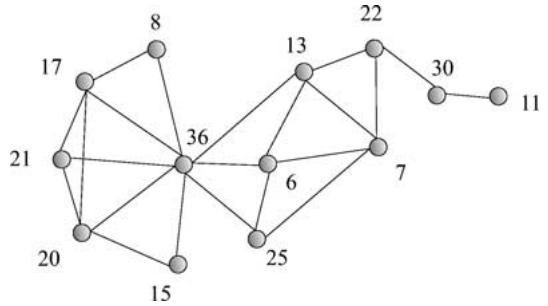

Figure 2. A network of Bluetooth devices.

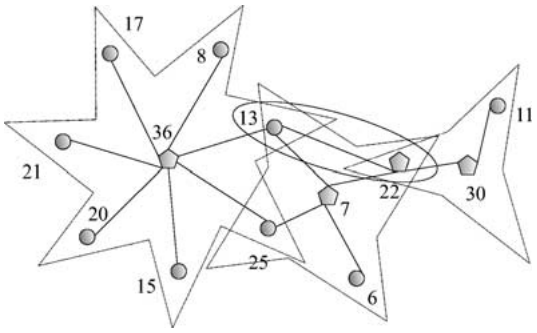

Figure 3. A BlueMesh scatternet.

line 1). Device 30, which has less than 7 neighbors, selects all of them (nodes 11 and 22) as its slaves and pages them to communicate its decision. "Piconet 30" is then formed by nodes 11, 22 and 30. Device 36 has eight neighbors. By executing the procedure COMPUTES locally, it selects seven of them. In particular, it first selects five nodes, devices 8, 13, 15, 21 and 25, through which 36 can reach all its remaining neighbors. Then, by selecting nodes 17 and 20 it reaches the limit of seven slaves (procedure GET, line 7). At this point, node 36 pages *all* its neighbors. It will communicate to node 6 that it is not invited to join its piconet, and it will invite all the other selected nodes. The resulting piconet 36 is made of eight devices: master 36, and the seven slaves 8, 13, 15, 17, 20, 21, and 25. In piconet 30, slave 22 has been paged by all its bigger neighbors. It switches to page mode and starts paging the nodes in $C(22)$, namely, devices 7 and 13, communicating its role. Similarly, device 25, which has received the page from node 36, pages nodes 6 and 7, which are in $C(25)$. Device 13, received the pages from all its bigger neighbors, and it is now ready to communicate its role (slave of master 36) to its smaller neighbors 6 and 7. Device 7, which now knows that all its bigger neighbors (nodes 13, 22 and 25) are slaves, becomes master and pages all its four neighbors inviting them to join its piconet, which they do: Piconet 7 is thus formed by its master and the slaves 6, 13, 22 and 25. At this point, the network has been divided into three piconets that need to be interconnected into a connected scatternet. Devices 22 and 25, common slaves of piconet 7 and 36, and 7 and 30, respectively, are selected as gateway slaves to interconnect such piconets. Piconets 30 and 36 can be interconnected only through the pair of intermediate gateways node 13 and 22. These two nodes are the only two nodes that move to the next iteration of the protocol. All the other nodes quit the execution of BlueMesh at this time: Some of these nodes are masters (of a single piconet), some are just slaves in one piconet (as node 6) and some have multiple roles (e.g., gateway slaves 13, 22 and 25). Realizing that it is now an init node, node 22 goes to page mode and pages its smaller neighbor 13 which is waiting for its page. The two-node piconet 22 is thus formed which implements the interconnection between master 36 and 30. The scatternet resulting from the execution of BlueMesh on the network of figure 2 is displayed in figure 3. Masters are depicted as pentagons, the piconets generated in the first iteration are star shaped, and the piconet generated in the second iteration has an "oval" shape.

We end this section with some notes related to the actual implementation of the protocol.

- During the protocol execution some nodes may have assumed the role of non-gateway slave in multiple piconets. Since these extra roles are not needed and since they lead to unnecessary switching of those slaves among their masters the following rule is adopted before exiting the protocol execution. Each slave checks whether it has been selected as gateway or not. In the former case, it leaves all the piconets that it is not interconnecting. If, instead, it is a non-gateway slave, it affiliates to the bigger master that selected it and leaves all the others.
- At the beginning of the ith iteration every non-master node must inform its neighbors whether it has proceeded to the current iteration or not. This information is necessary for the intermediate gateways that have proceeded to the ith iteration and need to know about their neighbors in G_i. To implement this communication of information, nodes selected as slaves and gateway slaves in the $(i-1)$th iteration participate to the first page/page scan phase of the ith iteration. In this way, once they are contacted by their bigger neighbors they make them aware that they are no longer involved in the protocol execution. They then proceed to communicate it to all their smaller neighbors. At this point, they can finally exit the execution of the protocol.

4.1. Correctness of the scatternet formation phase

The correctness of the BlueMesh protocol is based on the following facts: (1) Each device terminates the execution of the protocol. (2) Upon termination each device v has assumed one or more roles (if v is master, it is master only in one piconet, and a node can be slave in multiple piconets). (3) No piconet has more than 7 slaves. (4) For each pair of BT devices u and v, $u \neq v$, if there is a path between u and v in the geographic network topology, then there is also a path between u and v in the scatternet generated by BlueMesh.

In order to prove the first three claims, we proceed as follows. We first show that the first iteration terminates (i.e., each node either exits the protocol or proceeds to the next iteration), with all nodes in G_1 assuming a role (either slave in multiple piconets or master in one piconet), and with all the piconets generated in the first iteration not having more than 7 slaves. We then prove these results for successive iterations.

In all the proofs, we assume that there are no transmission errors, i.e., that all attempted pages between nodes in symmetric page modes are successful and lead to the exchange of all the information needed for the creation of a piconet.

We start by proving the following result.

Proposition 3. In the first iteration, each device $v \in G_1$ decides which role to assume *(*either master of one piconet or slave, possibly in multiple piconets*)*, and communicates it to each neighbor $u \in C(v)$. Device v communicates its role

only after all its bigger neighbors have communicated their role to it.

Proof. Let B_{G_1} be the set of BT devices in G_1. Similarly to proposition 2, the claim can be proven by induction on pecking$(x) = k \leqslant |B_{G_1}|$, $x \in B_{G_1}$, i.e., the number of consecutive (with respect to pecking) devices in G_1. Let I_{G_1} denote the set of init devices in G_1.

The induction base case is comprised of the init devices. Since there is always at least an init device, the set I_{G_1} is never empty. The proof of the base case is based on the code of the procedure MASTER: an init node immediately decides its role, goes to page mode and communicates its role to its neighbors which are in page scan mode (proposition 2).

Let us now assume that all nodes x such that pecking$(x) \leqslant h$, $h > |I_{G_1}|$ have decided their role, and have communicated it after all their bigger neighbors did.

Consider the device y such that pecking$(y) = h + 1$. By definition of pecking all y's bigger neighbors z are such that pecking$(z) <$ pecking(y) (bigger neighbors come first).

By inductive hypothesis, every such a neighbor z has decided its role, and, as stated by the code of the procedures MASTER and NONINIT, it has paged each of its smaller neighbors (which include y) to communicate it. This implies that device y has received a page from every bigger neighbor and only one of the following cases can occur: either (a) no bigger master invited y to join its piconet, or (b) node y has joined the piconet of one or more bigger masters that paged it. In case (a) device y decides to be a master (line 23 in procedure NONINIT), goes to page mode and communicates its role to all its neighbors in $C(v)$, as described in procedure MASTER. In case (b), device y affiliated with the masters that paged it (i.e., it becomes a slave), goes to page mode and communicates to all the smaller neighbors and bigger neighbors that are slaves that it is a slave. In both cases, the smaller devices are either still executing the topology discovery phase (which they will eventually exit), or are in page scan mode since they are not allowed to switch to page mode until every bigger neighbor has successfully paged them. The bigger neighbors that are slaves are still executing the first iteration and, by inductive hypothesis, they will complete the execution of lines 1–11 of procedure NONINIT and proceed to line 12 (i.e., they will go to page scan mode). Node y's role will therefore be successfully communicated.

Note that procedure NONINIT prevents a node from communicating its role unless it has received a page from all its bigger neighbors. □

The following proposition is proven similarly to the previous one.

Proposition 4. In the first iteration, each device $v \in G_1$ that becomes a slave communicates its list of masters to each of its slave neighbors. Device v communicates its list of masters only after all its neighbors have communicated their role to it.

Proof. The proof proceeds by induction on the weight-based ordering of the slaves, ordered by decreasing weight.

The induction base case is provided by the bigger slave node x. Upon communicating its role to the neighbors in $C(x)$, and after having received the information on the role of its smaller neighbors (see proposition 3), node x knows which of the neighbors which are masters selected it as slave. No page received later on in the iteration can change its list of masters. Node x then goes to page mode and communicates its list of masters to its smaller neighbors that are slaves (lines 16–18 in procedure NONINIT). These neighbors will eventually go into page scan mode and receive the page (proposition 3).

Let us now assume that the first h bigger slaves have communicated their list of masters, after having received the role from all their neighbors.

Consider the $(h + 1)$th bigger slave y. By proposition 3, slave y has communicated its role to its neighbors in $C(v)$, and knows the role of its neighbors (lines 1–14 of procedure NONINIT). It therefore knows its list of masters. By inductive hypothesis, every bigger neighbor z which is a slave has also communicated its list of masters to y. Node y then proceeds to execute lines 16–18 in procedure NONINIT (i.e., it goes to page mode and communicates its list of masters to all its neighbors that are slaves). Smaller neighbors that are slaves are guaranteed to enter page scan mode so to correctly receive the page (proposition 3). Bigger slaves successfully communicate their list of masters to slave neighbors (by inductive hypothesis), after which they go to page scan mode (line 19 in procedure NONINIT) to receive the list of master of smaller neighbors. □

We now prove that the piconets generated do not have more than 7 slaves. Let us denote with r the nodes' transmission range. As mentioned, we assume that two nodes x_i and x_j are neighbors if and only if their Euclidean distance $d(x_i, x_j)$ is $\leqslant r$ (i.e., the topology graph is a so-called *unit graph*). We make use of the following useful property of unit graphs.

Lemma 1. Let v be a node of a unit graph, and let S be a subset of its neighbors such that $|S| > 5$. Then there must be at least two nodes x_i and x_j in S which are in each other transmission range (i.e., $d(x_i, x_j) \leqslant r$).

Proof. Let us assume, by contradiction, that there exist 6 neighbors of v, $x_1, \ldots, x_6$, such that for each i and j, $i \neq j$, $d(x_i, x_j) > r$.

If $x_1, \ldots, x_6$ are all on the circle R_v of radius r centered in v, then the "best way" of placing them (so that the minimum distance between pairs of devices is maximized) is on the vertices of a perfect hexagon inscribed into the circle. In a regular hexagon, adjacent vertices have distance r, leading to a contradiction. Let us consider the case in which a node x_i is not on the circle R_v. Let p_i be the intersection point (closer to x_i) between R_v and the line interconnecting v and x_i. Let us also denote with R_{x_i} and R_{p_i} the circles of radius

r centered in x_i and p_i, respectively. By simple geometric reasoning it is possible to prove that $R_v \cap R_{p_i} \subseteq R_v \cap R_{x_i}$. By moving each inner node x_i to the corresponding point p_i on the circle we would therefore find six nodes on the circle which are not in each other transmission range, leading to a contradiction. □

The following proposition proves the consistency of the gateway selection phase of an iteration.

Proposition 5. The gateway selection part of an iteration terminates with masters of adjacent piconets having selected either one gateway slave, or two neighboring intermediate gateways. Masters of adjacent piconets have a consistent view of the gateways between them: If they selected a gateway slave, they selected the same one. If intermediate gateways are chosen, each master knows the identity of both of them.

Proof. Upon receiving information on the role of its neighbors, and on the list of masters of its neighboring slaves (propositions 3 and 4), a slave x knows the identity, the weight and the role of all its neighbors. It is also aware of all the piconets it has joined as slave, and of the list of masters of all its slave neighbors.

By gathering such information from all their slaves, masters get to know all the potential gateways toward adjacent piconets. As in the device discovery phase nodes acquire a symmetric knowledge of their one and two-hop neighbors (i.e., if a node u knows its one (two)-hop neighbor v, v also knows that u is one of its one (two)-hop neighbors), every pair of masters of adjacent piconets have a consistent view of the slaves that can act as gateways. They then adopt a weight-based unique rule to select the gateways (the bigger gateway slave is selected whenever possible; if no gateway slave is available, neighboring slaves are selected as intermediate gateways). Possible unique rules for the selection of intermediate slaves are those that maximize the sum of the weights of the intermediate gateways, or that minimize the minimum weight, etc. Therefore, given a pair of adjacent piconets, their masters either select the same gateway slave, or select the same pair of intermediate gateways. □

The termination of the first iteration is stated by the following result.

Corollary 1. Each device terminates the execution of the first iteration of BlueMesh having been assigned either the role of master in one piconet or the role of slave in multiple piconets. Piconets never have more than 7 slaves. Each pair of adjacent piconets is always interconnected by either a common gateway slave or by two neighboring intermediate gateways.

Proof. The termination of the first iteration is proven in propositions 3–5.

Each master terminates the execution of the role selection part of the first iteration as soon as it has made its decision and it has paged its neighbors to communicate it (proposition 3).

It is not possible for a master device to re-enter the execution of this part of the iteration and assume another role. Similarly, upon making the decision to be slave (line 5 in procedure *NonInit*), there is no way for a node to assume a different role. Hence, a node is either the master of a piconet, or a slave, possibly in more than one piconet.

Each master v selects its slaves by executing the procedure COMPUTES. By lemma 1, the inner while of the procedure cannot be executed more than 5 times, which results in the selection of at most 5 slaves. Procedure GET then selects $7 - |S(v)|$ additional slaves, which brings the nodes in $S(v)$ to a maximum of 7.

Finally, a consistent gateway selection is guaranteed by proposition 5. □

The correct termination of the generic ith iteration is proven similarly.

Corollary 2. Each device in G_i terminates the execution of BlueMesh ith iteration having being assigned either the role of master in one piconet or the role of slave in multiple piconets. Piconets in G_i are made up of at most 7 slaves. Each pair of adjacent piconets is always interconnected by the means of either a gateway slave or two neighboring intermediate gateways.

Once the termination of the iteration $i-1$ has been proven, the correct termination of the ith iteration is obtained by following exactly the same reasoning shown above for the first iteration.

To prove the correctness of BlueMesh we now provide a formal proof that the protocol terminates with nodes and piconets meeting the requirements 1, 2 and 3 listed at the beginning of this section. We finally use this result to prove that the scatternet generated by BlueMesh is connected.

Proposition 6. Each device terminates the execution of the BlueMesh protocol. Upon termination each device v has assumed one or more roles. If v is a master, it is the master of only one piconet. If v is also a slave, it can be slave in multiple piconets. No generated piconet has more than 7 slaves.

Proof. Given the definition of weight and given the total ordering of the set of all nodes' weights, there is always at least an init node in G_i, which will be master and that will not proceed to the next iteration (i.e., $|V_{i+1}| < |V_i|$, where V_i is the set of nodes that are active at iteration i, $i \geqslant 1$). The BlueMesh protocol will therefore terminate after a finite number of iterations.

All nodes that assume the role of master in a given iteration exit the protocol execution at the end of that iteration. This fact, together with the fact that nodes can only be master of one piconet in each iteration (corollary 2), proves that, over the entire protocol execution, no node can be master in more than one piconet.

We also note that no piconet can have more than 7 slaves, as this property is verified by all the piconets generated during the protocol iterations (corollary 2). □

We now prove that if the geographic network topology is connected, the scatternet generated by BlueMesh is also connected. Let $G_i^1 = (V_i^1, E_i^1)$ denote the subgraph of the ith iteration topology graph $G_i = (V_i, E_i)$ such that $V_i^1 = V_i$ and $E_i^1 = E_i \setminus \{(x, y) | x, y \in V_i, x \text{ master and } y \in N(x) \setminus S(x)\}$. G_i^1 is the graph obtained by G_i by deleting the links between a master x and a neighbor y which has not been selected as one of its slaves.

Two useful properties of G_i^1 are the following.

Lemma 2. For each pair of nodes i and j, if there exists a path between i and j in G_i, then there is also a path between i and j in G_i^1.

Proof. Let us assume for the sake of contradiction that there exist two nodes i and j which are connected through a path p: $n_0 = i, n_1, \ldots, n_k, n_{k+1} = j$ in G_i but not in G_i^1. This implies that there must be two adjacent nodes in the path p, n_l and n_{l+1}, $0 \leqslant l \leqslant k$ such that $(n_l, n_{l+1}) \in E_i \setminus E_i^1$. At least one of the two nodes n_l and n_{l+1} must be a master. Without loss of generality we assume node n_l to be the bigger master.

Since n_{l+1} has not been selected as n_l's slave, a neighbor n_x of n_{l+1} has (see procedure COMPUTES). If node n_{l+1} is a slave, this immediately implies that there is a path p^1: n_l, n_x, n_{l+1} in G_i^1, which leads to a contradiction. If n_{l+1} is a master, it either selected n_x as one of its neighbors (in which case n_l and n_{l+1} are connected through path p^1, resulting again in a contradiction), or there must be a slave n_y in n_{l+1}'s piconet which covers n_x. In this case p^2: n_l, n_x, n_y, n_{l+1} is yet again a path between nodes n_l and n_{l+1} in G_i^1, leading to a contradiction. □

Lemma 3. No two masters in G_i^1 are neighbors.

Proof. The proof proceeds by contradiction: We assume that there exist two neighbors u and v in G_i^1 which have decided to be masters. Without loss of generality, let u be the one with bigger weight. Since node v decides to be a master, v has not been selected as one of u's slaves (i.e., $u \notin S(u)$). Then, by construction of G_i^1, $(u, v) \notin E_i^1$, which means that u and v cannot be neighbors. □

The proof that the scatternet built by BlueMesh is connected relies on the following result, first proven in [3].

Theorem 1 [3, theorem 1]. If the masters form a dominant independent set in the current network graph, and the graph is connected, then a connected backbone is guaranteed to arise if each master establishes connections to all other masters that are at most three hop away. These connections are all needed to ensure that the resulting scatternet is connected, in the sense that if any of them is missing the scatternet may be not connected.

In order to prove the connectivity of the scatternets generated by BlueMesh, we prove the following corollary of theorem 1.

Corollary 3. Given the piconets resulting from the execution of the ith iteration of BlueMesh, the scatternet generated interconnecting them is guaranteed to have a path between each pair of nodes which were connected in G_i if each master establishes multi-hop connections with all the masters that are two and three hops away in G_i^1, $i \geqslant 1$.

Proof. By lemma 2 if there is a path between two nodes in G_i, then there is also a path between them in G_i^1. The claim then directly follows from applying theorem 1 to each connected component in G_i^1 whose masters form a dominant independent set (lemma 3). □

Masters which are two hop away in G_i^1 have selected a common slave and can be interconnected through a gateway slave, while masters that are three hops away will be interconnected through intermediate gateways.

We are finally able to prove that nodes which are connected by a path in the geographic network topology are also connected in the BlueMesh scatternet.

Proposition 7. For each pair of BT devices u and v, $u \neq v$, if there is a path between u and v in the geographic network topology, then there is also a path between u and v in the scatternet generated by BlueMesh.

Proof. We prove this claim by induction on the number of iterations which are still needed to complete the protocol. The induction base case is the last iteration k (BlueMesh has been proven to require a finite number of iterations to complete its operations in proposition 5). All adjacent piconets generated during the kth iteration have masters that are two hops away in G_i^1 so that they can be interconnected by means of common slaves. Hence, corollary 3 guarantees that if there is a "geographic path" between u and v, there is also a path in G_k.

Let us now assume that for each $i > h$, it is true that if there is a path between a pair of devices $u, v \in G_i$, then there is also a path between u and v in the scatternet generated by BlueMesh in iterations $i, \ldots, k$.

Consider the hth iteration. From corollary 3 we know that the claim also holds for devices in G_h, provided that we can prove that the adjacent piconets generated in this iteration will be interconnected before BlueMesh termination. Masters that are two hops away in G_h^1 select the same slave gateway (proposition 5) through which they are interconnected. Masters that are three hops away select a pair of neighboring intermediate gateways (proposition 5). All the selected intermediate gateways (and only them) proceed to the next iteration (they are the nodes in G_{h+1}). By inductive hypothesis, they will be interconnected in iterations $h + 1, \ldots, k$. □

5. Simulation results

We have simulated BlueMesh to demonstrate its effectiveness in generating a connected scatternet. We used a simulator of BT-based ad hoc networks with 210 nodes, implemented in C++.[3] The Power Class 3 BT nodes (i.e., those nodes with a maximum transmission radius of 10 meters) are randomly and uniformly scattered in a geographic area which is a square of side L. The resulting geographic network topology graph is a *unit graph*, i.e., it is assumed that two nodes are in each other transmission range if and only if their distance in the plane (Euclidean distance) is $\leqslant 10$ m.

In our simulations, the number of BT nodes n has been varied in the range $30, 60, 90, \ldots, 210$, while L has been set to either 40 m or 60 m. This allowed us to test our protocol on moderately ($L = 60$ m) to heavily ($L = 40$ m) dense networks.

Table 1 shows the average and maximum degrees of the generated geographic network topologies. Dense networks with average (maximum) degree up to 32.8 (51.67) are generated in the simulated scenarios by choosing $L = 40$, while for $L = 60$ the node density is much lower, varying from an average (maximum) degree equal to 4.4 (8.8) when $n = 60$ to 15.8 (27.4) for networks with 210 nodes. (The case $L = 60$ and $n = 30$ is not taken into consideration as it corresponds to geographic network topologies which are not connected.)

The measures investigated in our simulations concern the average number of iterations needed to complete the protocol, i.e., to obtain a connected scatternet, and the set of metrics identified in the literature as measures of the "quality" of a scatternet. These metrics are (a) the average number of piconets in the scatternet, (b) the average number of roles (master or slave) assigned to each node, (c) the average number of slaves per piconet, and (d) the average length of the routes in the scatternet, and its comparison to the average length of the routes in the geographic network topology. All experiments have been performed on a number of (geographically) connected topologies that allow us to achieve a confidence level of 95% and a precision within 5%.

The average number of iterations required to complete the scatternet formation is depicted in figure 4. We observe that BlueMesh scales well: as the number of nodes increases linearly from 30 to 210, the average number of iterations only increases from 2.3 to 4.6 (2.7–4.3) when $L = 40$ ($L = 60$). The number of nodes involved in the ith iteration quickly decreases with i as depicted in figure 5, which shows the average number of nodes that are active in the first five iterations for $n = 30$, 90, 150, 210 and $L = 40$ (the case $L = 60$ shows a similar behavior). Let us consider the case $n = 210$ and $L = 40$. The average number of iterations is 4.6. However, only 37% of the nodes (on average) proceed to the second iteration, 14% to the third iteration, and only 4% are still executing the protocol in the fourth iteration. Therefore, the operations of BlueMesh are faster and faster as the protocol proceeds through the various iterations. As n increases, the number of adjacent piconets to be interconnected through intermediate gateways also increases, resulting in a higher number of nodes and links which proceed to successive iterations, and therefore in an increased number of iterations required by BlueMesh to complete its operations.

Figure 6 shows that the average number of piconets in the scatternet generated by BlueMesh grows linearly with the number of nodes in the network. The percentage of network nodes which are masters (i.e., the number of piconets) varies from 42% (44%) when $n = 30$ and $L = 40$ ($n = 60$ and $L = 60$) to 37% (41%) when $n = 210$. A significant part

Table 1
Average and maximum degrees for networks in constrained areas.

Number of BT devices	30	60	90	120	150	180	210
$L = 40$, avg degree	4.6	9.2	13.9	18.6	23.3	28	32.8
$L = 40$, max degree	8.4	15.9	23.5	30.8	37.7	44.7	51.7
$L = 60$, avg degree		4.4	4.6	8.9	11.2	13.5	15.8
$L = 60$, max degree		8.8	12.7	16.7	20.3	23.9	27.4

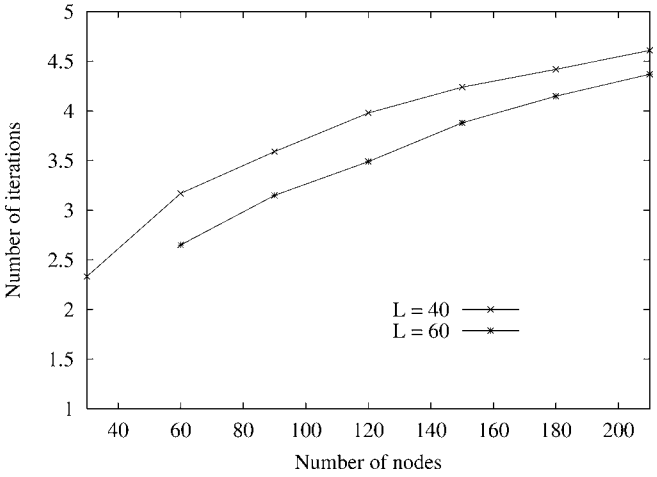

Figure 4. Average number of iterations.

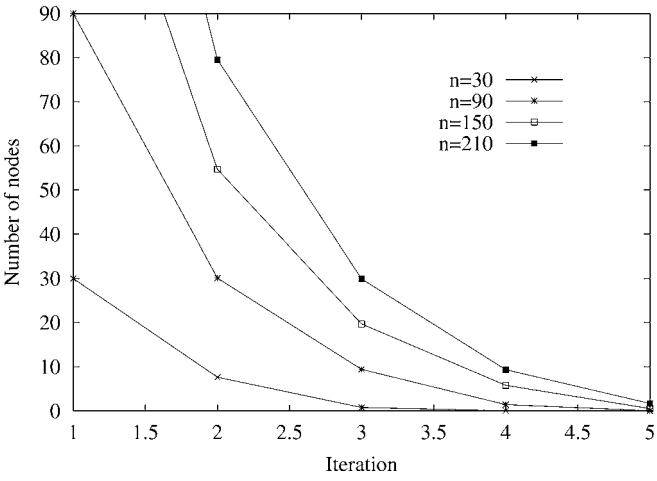

Figure 5. Average number of nodes in each iteration. $L = 40$.

[3] Currently, our study is limited to network-layer details, thus the BT stack is not implemented. A detailed performance evaluation providing insights on device discovery and other aspects of scatternet formation can be found in [1]. In this paper, simulation results are obtained by means of a ns2-based simulator that implements the BT stack.

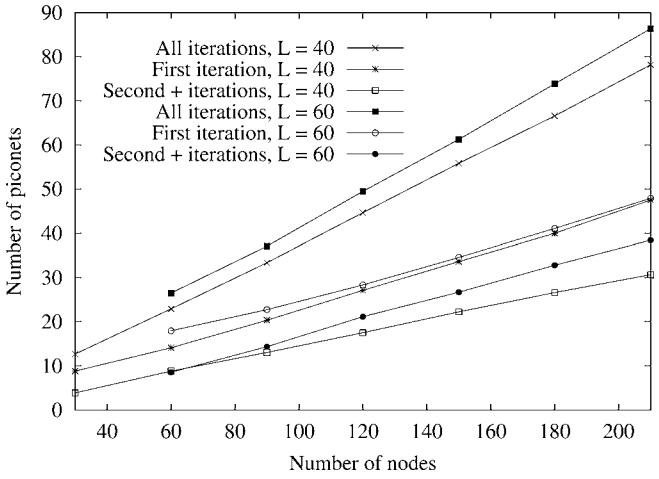

Figure 6. Average number of piconets generated in different iterations.

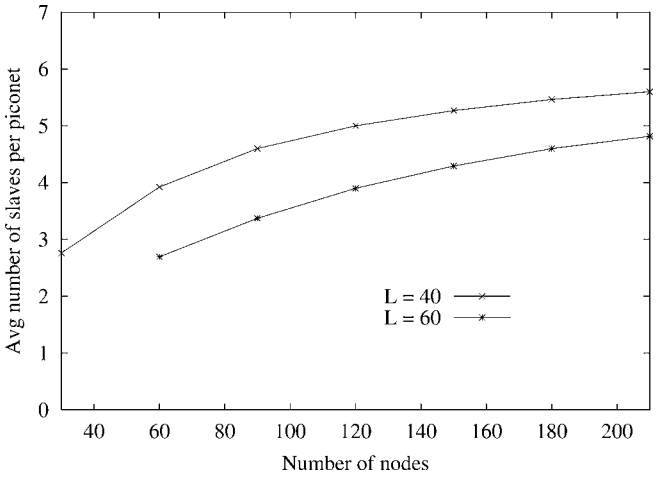

Figure 7. Average number of slaves per piconet.

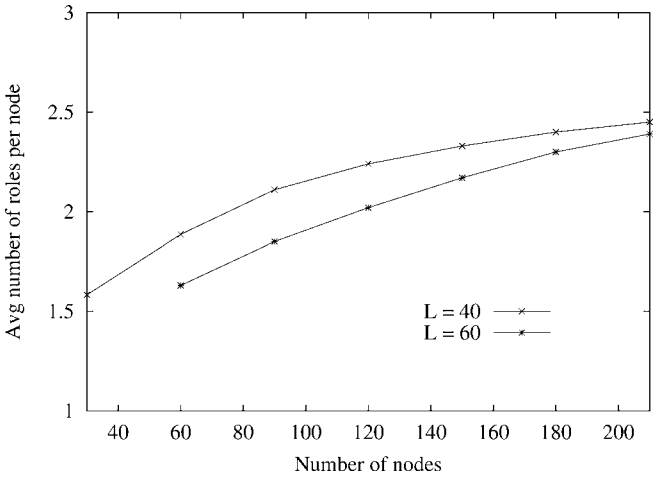

Figure 8. Average number of roles assumed by BT nodes.

of the generated piconets (up to 39% (44%) when $n = 210$ and $L = 40$ ($L = 60$)) are generated after the first iteration, namely, they are needed for piconet interconnection. This depends on the fact that in BT, whenever two piconets are interconnected through intermediate gateways, an extra piconet needs to be created in which one of the two slaves is the master, and the other is the slave (see piconet B in figure 1).

BlueMesh generates a higher number of piconets on moderately dense networks with respect to the number of piconets it generates in denser networks. This is mostly due to the fact that low density networks are divided into piconets with a smaller number of slaves, resulting in an overall bigger number of piconets. When the average node degree is very low, e.g., less than 7 (as it happens for networks with 60 and 90 nodes, $L = 60$) the number of piconets generated in the first iteration is significantly higher than the number we would obtain for bigger degrees.

When the number of nodes (and hence the density) increases, this problem applies again to the piconets generated in the following iterations. The subgraph spanning the nodes which have not exited the protocol yet is sparse, so that it is partitioned in a higher number of piconets.

One of the characteristics of the scatternets generated by BlueMesh is that each of its piconets has no more than 7 slaves. As shown in figure 7, on average, this limit is seldom approached, given that the average number of slaves per piconet is between 2.7 and 5.6.

A critical performance measure for Bluetooth Scatternets is the (average) number of roles assigned to each node. A high number of roles per node translates into reduced throughput performance since nodes can be active only in one piconet at a time. The worst case scenario is when a node is master in one piconet and slave in multiple piconets, as no communication can be performed in the piconet in which it is the master while it acts as a slave in another piconet.

In figure 8 we show the average number of roles assumed by BT devices in a BlueMesh scatternet. The average number of roles per node slightly increases with n, and is higher in case of heavily dense networks ($L = 40$). As the network density increases, the number of adjacent piconets which have to be joined (and thus the number of gateways) increases. Also, the same gateway is likely to be selected for the interconnection of a higher number of piconets.

However, the average number of roles assumed by each node is always very low, and never exceeds 2.4. Around 60–65% of the nodes assume only one role. Master–slave roles are assumed only by intermediate gateways for the sole purpose of interconnecting piconets. The percentage of BT devices with master-slave roles is always within 12–17% of the total number n of nodes.

Figure 9 shows the increase of the average length of the shortest paths between BT devices in the scatternet, with respect to the average length of the shortest paths between BT nodes in the geographic network topology. The increase in the average route length is quite limited, ranging from 21% (14%) for $n = 30$ ($n = 60$) and $L = 40$ ($L = 60$), to 54% (31%) when $n = 210$ and $L = 40$ ($L = 60$). Increased route lengths are motivated (i) by the need for all communications in a BT piconet to pass through the master (two neighboring slaves that belong to the same piconet cannot communicate directly), (ii) by the piconet-based network organization (which may force nodes which are one hop

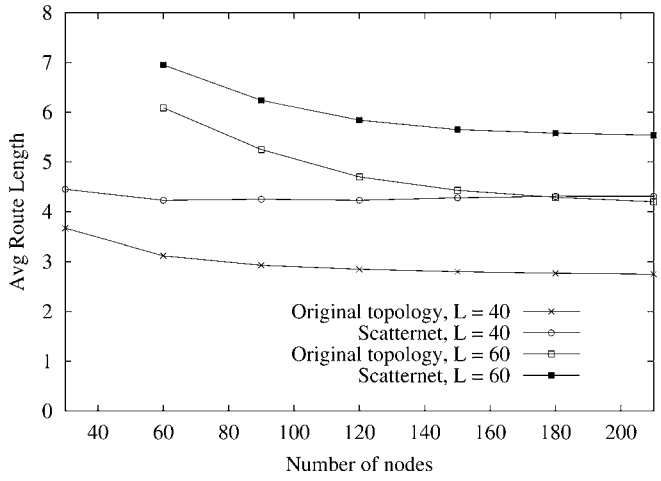

Figure 9. Comparison of the average route length in the geographic topology and in the scatternet.

away but do not belong to the same piconet to communicate through a much longer inter-piconet route), (iii) by the protocol choice to always interconnect masters that are neighbors through gateway slaves, and (iv) by possible extra hops introduced by BlueMesh for piconet interconnection. To better understand the impact of each of the above effects on the route length increase, we have separately measured the increase in route length of routes interconnecting (a) slave nodes belonging to the same piconet, (b) master nodes, and (c) slave nodes belonging to different piconets.

Simulation results showed that the intra-piconet average route length experiences a similar increase in all the simulated scenarios, ranging from 28–32% increase in dense networks to 25–26% increase in moderately dense networks.

The *backbone* (i.e., between masters) route lengths are only 4–5% higher than the shortest routes in the geographic network topologies with $n = 30$, and suffer only a moderate increase as the network density increases (up to 20% (11%) when $n = 210$ and $L = 40$ ($L = 60$)). Despite the significant increase of the length of routes joining neighboring masters, its effect on backbone route length is almost negligible, as the number of such pairs is very low. Non-neighbor masters only experience up to a 17% (10%) increase of the length of the routes interconnecting them, when $L = 40$ ($L = 60$). This, in turn, suggests that the possible extra hops introduced by BlueMesh because of piconet interconnection has only little effect on route length.

The routes between slaves belonging to different piconets, finally, suffer a higher length increase, ranging from 33% (21%) when $n = 30$ ($n = 60$) when $L = 40$ ($L = 60$), to a 78% (45.3%) increase when $n = 210$. The piconet-based network organization thus appears to have a strong impact on route length performance.

6. Conclusions

In this paper we have presented BlueMesh, a new protocol for the establishment of a Bluetooth-based multi-hop ad hoc network. Considering all aspects of the BT technology, BlueMesh describes how to perform topology discovery, piconet formation and piconet interconnection so that, starting from a network of devices geographically connected, a connected scatternet is always generated and each of its piconets does not have more than 7 slaves. Through the use of extensive simulations we have demonstrated that BlueMesh is effective in quickly producing a scatternet, which is a connected mesh. In particular we have observed that the BlueMesh scatternet has the desirable properties that nodes have no more than 2.4 roles (on average), and that its routes are not significantly longer than the shortest routes between any two nodes in the geographic network topology. Current research is mainly aimed at extending the operations of BlueMesh to generate scatternets in situations more commonly expected in real deployment situations, namely, where the network topology may change dynamically and a realistic physical channel model is considered.

References

[1] S. Basagni, R. Bruno and C. Petrioli, Performance evaluation of a new scatternet formation protocol for multi-hop Bluetooth networks, in: *Proceedings of the 5th International Symposium on Personal Wireless Multimedia Communications, WPMC 2002*, Honolulu, HI (October 27–30, 2002).

[2] S. Basagni and C. Petrioli, A scatternet formation protocol for ad hoc networks of Bluetooth devices, in: *Proceedings of the IEEE Semiannual Vehicular Technology Conference, VTC Spring 2002*, Birmingham, AL (May 6–9, 2002).

[3] I. Chlamtac and A. Faragó, A new approach to the design and analysis of peer-to-peer mobile networks, Wireless Networks 5(3) (1999) 149–156.

[4] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, *Introduction to Algorithms*, 2nd ed. (MIT Press/McGraw-Hill, Cambridge, MA and New York, 2001).

[5] C. Law, A.K. Mehta and K.-Y. Siu, A new Bluetooth scatternet formation protocol, Mobile Networks and Applications (MONET), Special Issue on Mobile Ad Hoc Networks 8(5) (2003).

[6] X. Li and I. Stojmenovic, Partial Delaunay triangulation and degree-limited localized Bluetooth scatternet formation, in: *Proceedings of AD-HOC NetwOrks and Wireless (ADHOC-NOW)*, Fields Institute, Toronto, Canada (September 20–21, 2002).

[7] T. Salonidis, P. Bhagwat, L. Tassiulas and R. LaMaire, Distributed topology construction of Bluetooth personal area networks, in: *Proceedings of the IEEE Infocom 2001*, Anchorage, AK (April 22–26, 2001) pp. 1577–1586.

[8] Specification of the Bluetooth System, Vol. 1, Core, Version 1.1, `http://www.bluetooth.com` (February 22, 2001).

[9] I. Stojmenovic, Dominating set based Bluetooth scatternet formation with localized maintenance, in: *Proceedings of the Workshop on Advances in Parallel and Distributed Computational Models*, Fort Lauderdale, FL (April 2002).

[10] G. Tan, A. Miu, J. Guttag and H. Balakrishnan, An efficient scatternet formation algorithm for dynamic environment, in: *Proceedings of the IASTED Communications and Computer Networks (CCN)*, Cambridge, MA (November 4–6, 2002).

[11] Z. Wang, R. J. Thomas and Z. Haas, Bluenet – a new scatternet formation scheme. in: *Proceedings of the 35th Hawaii International Conference on System Science (HICSS-35)*, Big Island, HI (January 7–10, 2002).

[12] G. Záruba, S. Basagni and I. Chlamtac, Bluetrees – Scatternet formation to enable Bluetooth-based personal area networks, in: *Proceedings*

of the IEEE International Conference on Communications, ICC 2001, Helsinki, Finland (June 11–14, 2001).

Chiara Petrioli received the Laurea degree with honors in computer science in 1993, and the Ph.D. degree in computer engineering in 1998, both from Rome University "La Sapienza," Italy. She is currently Assistant Professor at the Computer Science Department at Rome University "La Sapienza." Her current work focuses on ad-hoc and sensor networks, Bluetooth, energy-conserving protocols and QoS in IP networks. Prior to Rome University she was research associate at Politecnico di Milano, and was working with the Italian Space Agency (ASI) and Alenia Spazio. Dr. Petrioli is an area editor of the ACM Wireless Networks journal, and of the Wiley Wireless Communications and Mobile Computing journal. She has served in the organizing committee and technical program committee of several leading conferences in the area of networking and mobile computing including ACM Mobicom, ACM MobiHoc, IEEE ICC. Dr. Petrioli was a Fulbright scholar, and is member of ACM, IEEE, ACM SIGMOBILE, IEEE Communication Society.
E-mail: petrioli@dsi.uniroma1.it

Stefano Basagni holds a Ph.D. in electrical engineering from the University of Texas at Dallas (December 2001) and a Ph.D. in computer science from the University of Milano, Italy (May 1998). He received his B.Sc. degrees in computer science from the University of Pisa, Italy, in 1991. Since Winter 2002 he is on faculty at the Department of Electrical and Computer Engineering at Northeastern University, in Boston, MA. From August 2000 to January 2002 he was assistant professor of computer science at the Department of Computer Science of the Erik Jonsson School of Engineering and Computer Science, The University of Texas at Dallas. Dr. Basagni's current research interests concern research and implementation aspects of mobile networks and wireless communications systems, Bluetooth and sensor networking, definition and performance evaluation of network protocols and theoretical and practical aspects of distributed algorithms. Dr. Basagni has published over two dozens of referred technical papers. He is also co-authors of book chapters. Dr. Basagni served as a guest editor of the special issue of the Journal on Special Topics in Mobile Networking and Applications (MONET) on Multipoint Communication in Wireless Mobile Networks as well as guest editor of the special issue on mobile ad hoc networks of the Wiley's Interscience's Wireless Communications & Mobile Networks journal, for which he serves in the editorial board. Dr. Basagni also served and serves on Technical Program Committees, including the ACM/SIGMOBILE MobiCom TPC, and ACM/SIGMOBILE MobiHoc TPC, IEEE Globecom and IEEE ICC. Dr. Basagni is a member of the ACM (including the ACM SIGMOBILE) and of the IEEE (Computer and Communication Societies).
E-mail: basagni@ece.neu.edu

Imrich Chlamtac holds a Ph.D. degree in computer science from the University of Minnesota. Since 1997 he is the Distinguished Chair in Telecommunications at the University of Texas at Dallas. Dr. Chlamtac also holds the titles of the Sackler Professor at Tel Aviv University, Israel, The Bruno Kessler Honorary Professor at the University of Trento, Italy, where he is currently on sabbatical, and University Professor at the Technical University of Budapest, Hungary. Dr. Chlamtac is a Fellow of the IEEE and ACM societies for, a Fulbright Scholar and an IEEE Distinguished Lecturer. He is the winner of the 2001 ACM Sigmobile annual award and the IEEE ComSoc TCPC 2002 award for contributions to wireless and mobile networks, and of multiple best paper awards in wireless and optical networks. Dr. Chlamtac published close to three hundred papers in refereed journals and conferences, and is the co-author of the first textbook on Local Area Networks (Lexington Books, 1981, 1982, 1984) and of Mobile and Wireless Networks Protocols and Services (Wiley, 2000) – an IEEE Network magazine's 2000 Editor's Choice. Dr. Chlamtac serves as the founding Editor in Chief of the ACM/URSI/Kluwer Wireless Networks (WINET), the ACM/Kluwer Mobile Networks and Applications (MONET) journals and the SPIE/Kluwer Optical Networks (ONM) Magazine.
E-mail: chlamtac@utdallas.edu