A Performance Comparison of Scatternet Formation Protocols for Networks of Bluetooth Devices

Stefano Basagni
Northeastern University
basagni@ece.neu.edu

Raffaele Bruno
IIT, C.N.R. Pisa
bruno@iit.cnr.it

Chiara Petrioli
Università di Roma "La Sapienza"
petrioli@dsi.uniroma1.it

Abstract

*This paper describes the results of an ns2-based comparative performance evaluation among three major solutions presented in the literature for forming multi-hop networks of Bluetooth devices (*scatternet formation*). The three protocols considered in this paper are BlueTrees [12], BlueStars [8], and the "Yao protocol" presented in [7]. We observed that device discovery is the most time-consuming operation, independently of the particular protocol to which it is applied. By means of a thorough performance evaluation we have identified protocol parameters and Bluetooth technology features that affect the duration of this device discovery. We have also analyzed the effect of the different protocols operations on key metrics of the generated scatternets. The comparative performance evaluation showed that due to the simplicity of its operations and to its basic working requirements BlueStars is by far the fastest protocol for scatternet formation which also yields to scatternets with a lower number of piconets, average route length and number of roles per node. However, BlueStars produces scatternets with an unbounded, possibly large number of slaves per piconet, which imposes the use of potentially inefficient Bluetooth operations. A good compromise when interested in forming scatternets whose piconets have a bounded number of slaves is obtained by combining BlueStars and the Yao protocol. Although latency and route lengths are longer than in BlueStars scatternets, with the combined solution we obtain an overall good protocol performance and scatternets with desired characteristics.*

1. Introduction

It is widely anticipated that 4th-generation wireless systems will extensively rely on the unlicensed operations provided by *ad hoc communications*. Allowing spontaneous deployment and self-planning/management, ad hoc networking will play an important role in delivering all kinds of wireless services from the Internet to the very hands of the mobile user, thus realizing the vision of pervasive computing.

The Bluetooth (BT) technology, as described in the Specifications of the Bluetooth System Version 1.1 [5], is expected to be one of the most promising enabling technology for ad hoc networks and pervasive computing. Originally introduced as short-range cable replacement, the BT specifications define ways for which each BT device can set up multiple connections with neighboring devices so that communication can be established in a multi-hop fashion. In this sense, Bluetooth devices spread in a geographic area can provide the missing wireless extension to the various heterogeneous network infrastructures, allowing a more pervasive wireless access.

This paper addresses the fundamental problem of *scatternet formation*, i.e., the problem of the self organization of BT devices into a multi-hop network. In particular, this paper describes and compares the performance of the most promising solutions proposed so far to this problem. Beyond providing the first performance comparison of available solutions, this paper also provides insights on the BT technology and on some limitations of the current specifications with respect to scatternet formation.

According to the current BT specifications, the way in which BT allows multi-hop communication is summarized as follows. When two BT nodes that are into each other communication range want to set up a communication link, one of them must assume the role of *master* of the communication while the other becomes its *slave*. This simple "one-hop" network is called a *piconet* and may include several slaves no more than 7 of which can be actively communicating with the master at the same time. If a master has more than 7 slaves, some slaves have to be "parked." To communicate with a parked slave a master has to "unpark" it, while possibly parking another slave.

The specifications allow each node to assume multiple roles. A node can be a master in one piconet and a slave in one or more other piconets, or a slave in multiple piconets. Devices with multiple roles act as *gateways* to adjacent piconets thus *forming* a multi-hop ad hoc network called a *scatternet*.

A first broader classification of the solutions proposed so far in the literature distinguishes between scatternet formation protocols that require the radio vicinity of *all* nodes (*single-hop* topologies) and protocols that work in the more general *multi-hop* scenario. All the solutions are *localized*, in the sense that the protocols are executed at each node with the sole knowledge of its immediate neighbors (nodes in its transmission range).

Solutions of the first kind are presented in [9], [6] and [10]. The solution proposed in [9] is based on a leader election process to collect topology information at the leader. 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 ≤ 36. When $n > 36$ other centralized schemes could be used such as the one proposed in [1] and [2]. The protocols presented in [6] and [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. We have not considered this first kind of solutions in the performance comparison described in this paper.

Among the solutions that apply to the more general case of multi-hop topologies, the scatternet formation protocol described in [12] requires that the protocol is initiated by a designated node (the *blueroot*) and generates a tree-like scatternet. The blueroot starts the formation procedure by acquiring as slaves its one hop neighbors. These, in turn, start contacting their own neighbors (those nodes that are two hops from the root) trying to acquire them as their slaves and so on, in a "wave expansion" fashion, till the whole tree is constructed. The obtained scatternet has piconets with an unbounded number of slaves. A procedure based on geometric properties of networks of devices scattered in the plane is described to re-configure the tree so that each master has no more than seven slaves. This allows the master to avoid the time and bandwidth consuming operation of parking and unparking of slaves

To the best of the authors' knowledge, the only presented solutions for scatternet formation in multi-hop BT networks that produce topologies different from a tree are those described in [7], [11] and [8].

The main aim of the protocol proposed in [7] is to build up a connected scatternet in which each piconet has no more than 7 slaves. To this purpose, degree reduction techniques are initially applied to the network topology graph so to reduce the number of wireless links at each node to less than 7 without affecting the connectivity of the resulting topology. A scatternet formation protocol (which is left unspecified) is then executed on the resulting topology. 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. After an (unspecified) device discovery phase, some of the nodes enter either page or page scan randomly. Nodes in page mode will be masters and try to invite a bounded number of neighbors to join their piconets as slaves. Nodes that are unable to acquire slaves or to join a piconet within a given amount of time re-execute this process. This provides a statistical guarantee that either a node becomes a member of a piconet (being either the master or a slave of the piconet) or it is isolated (i.e., all its neighbors made a decision on their role but none of its neighboring masters invited it in its piconet). Isolated nodes become masters and try to connect to some already formed piconet. Finally, the master of each piconet instructs its slaves to set outgoing links to neighboring piconets to form a scatternet. The connectivity of the resulting scatternet is not guaranteed (i.e., not all the BlueNets are connected, even when the initial topologies are). The protocol described in [8], termed *BlueStars*, forms connected scatternets without using any extra hardware. Making no assumptions on the nodes placement and with the sole knowledge of a node's one-hop neighbors, BlueStars selects the piconets' masters based on how "fit" a node is to serve as a master. However, BlueStars produces scatternets whose piconets may have an unbounded number of slaves. Whenever positioning information is available at the nodes, BlueStars can be combined with the protocols described in [7] to form scatternets in which each piconet has no more than 7 slaves.

In this paper we compare the performance of three of the four scatternet formation protocols for multi-hop topologies outlined above, namely, BlueTrees, BlueStars and the protocol presented in [7] which makes use of location information. Among the several geometric construction techniques proposed in [7] we implemented the one that the authors consider the most promising for producing scatternets with desirable properties, namely, the *Yao construction*. Therefore, we refer to the solution of [7] as to the "Yao protocol." At this time we were not able to implement BlueNet given the lack of implementation details in [11].

All operations of the chosen protocols require each node to be aware of its neighbors. To this purpose, *device discovery* has to be performed before the actual scatternet formation process takes place.

In this paper we investigate, by means of ns2-based simulations, the impact of device discovery on the performance of the protocols, and compare the protocols with respect to key metrics considered crucial for scatternet formation.

The paper is organized as follows. In the following section we briefly describe the way we solve the problem of device discovery, which is common to all solutions described and compared in this paper. Section 3 describes the three protocols and the problems we encountered in implementing them. In Section 4 we describe in details our simulation environment and we evaluate and comment on the performance of the three protocols with respect to the selected metrics of interest. Finally, Section 5 concludes the paper.

2. Device Discovery in Multi-hop Bluetooth Networks

The device discovery phase should lead each of the network nodes to become aware of all its neighbors in the visibility graph. This neighbor knowledge should be "symmetric," which means that if node v knows node u, u must also know v (this is an assumption upon which all protocols for multi-hop scatternet formation rely).

The mechanisms provided by the BT specifications for device discovery, i.e., the *inquiry* procedures, do not lead to the needed symmetric neighbor knowledge: An *inquirer* that is trying to discover neighboring nodes does not transmit its unique BT identifier, thus remaining unknown to the node that receives the inquiry packet. Furthermore, the BT discovery mechanisms require nodes to be in opposite inquiry modes (called *inquiry* and *inquiry scan* modes) in order to be able to communicate. However, no method is described in the specifications on how to (even statistically) guarantee that two neighboring devices are in opposite modes. Therefore, specifications compliant mechanisms must be defined to ensure that, for each pair of neighboring nodes v and u, they are eventually in opposite modes and that, when node v discovers node u, u is also made aware of v.

The only solution for device discovery in multi-hop networks proposed so far is derived by the mechanism first described in [9]. Each device is allowed to alternate for a predefined device discovery duration between inquiry mode and inquiry scan mode, remaining in each mode for a time selected randomly and uniformly in a given time range. When two nodes in opposite inquiry modes handshake, they set up a temporary piconet. Specifically, the inquirer goes in *page* mode and becomes the master, while the inquired node goes in *page scan* mode, becoming a slave. The two nodes exchange their ID and possibly other information necessary for the following phases of the protocol. As soon as the information has been successfully communicated the piconet is disrupted.

The effectiveness of the described mechanism in providing the needed mutual knowledge to pairs of neighboring devices relies on the idea that, by alternating between inquiry and inquiry scan mode and randomly selecting the length of each inquiry (inquiry scan) phase, we have 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. However, this process can be extremely time consuming [3]. As shown in Section 4.1, we have observed that in networks with high density, after 10s of device discovery, only a fraction of a node's one-hop neighbors have been discovered. However, we have verified that we can keep device discovery reasonably short while still obtaining that the resulting discovered topologies are connected, which is the necessary requirement for generating connected scatternets. The price to pay for a short device discovery is that in the discovered topology it is not guaranteed that two nodes that are in each other transmission range discover each other. As detailed in the next section, this can be a problem for those protocols that use this assumption to form connected scatternets with a bounded number of slaves per piconet.

The duration of the discovery phase is therefore a critical parameter that should be carefully selected taking all these trade-offs into account.

3. Three Scatternet Formation Protocols

We describe here three protocols for scatternet formation and the solutions to the problems we encountered in implementing them.

Given a set of BT nodes, we call *visibility graph* the network topology where there is a link between any two nodes whose Euclidean distance is less than or equal to the nodes transmission radius. (When the BT nodes are scattered in the plane, as it is for the solutions compared in this paper, these topologies are also often referred to as *unit disk graphs*, UDGs.) In the protocol descriptions, the term neighbors refers to the nodes that a node has discovered during the discovery phase.

3.1. BlueTrees

The scatternet formation protocol presented in [12] is the first to solve this problem in a multi-hop topology. The protocol is initiated by a designated node (the *blueroot*) and generates a tree-like scatternet.

The blueroot starts the formation procedure by acquiring as slaves its one hop neighbors. These, in turn, start paging their own neighbors (those nodes that are at most two hops from the root) and so on, in a "wave expansion" fashion, till the whole tree is constructed. In order to limit the number of slaves per piconet, it is observed that if a node in a unit disk graph has more than five neighbors, then there are at least two of them which are in each other transmission range. This observation is used to re-configure the tree so that each master node has no more than seven slaves. If a master v has more than seven slaves, it selects pairs of them which are necessarily in each other transmission range, and instructs one node of each selected pair to be the master of the other node, which is then disconnected from v's piconet. Such "branch reorganization" is carried throughout the network leading to a scatternet where each piconet has no more than seven slaves.

Beyond producing a tree-like topology, which limits the robustness of the obtained scatternet, BlueTrees depends on a selected node to start the formation procedure so that this solution does not work in networks whose topology after the discovery phase is not connected.

We encountered the following two problems in implementing BlueTrees.

In order to limit the number of slaves per piconet to less than eight, the protocol assumes that the topology resulting from the device discovery phase is such that, if two neighboring nodes u and v have discovered each other and they both have a common neighbor z, then either both discovered z or neither of them did. This property, which is necessary for a re-configuration of the BlueTree into a scatternet with no more than 7 slaves per piconet, cannot be guaranteed by the discovery phase as described earlier.

Consider a piconet in which the master v has chosen 7 among its 8 neighbors. Assume neighbor z is the one that has been left out of v's piconet. For the geometric property mentioned above, at least one of v's neighbors, say u, is in the transmission range of z. If u and z did not discover each other, chances are that the formed scatternet is not connected (v cannot reach its neighbor z).

Therefore, for the protocol to correctly work we need to perform extra operations to achieve a consistent knowledge of each node's neighborhood. This can be obtained in the following way.

At the end of a discovery phase all neighboring nodes that have discovered each other exchange the list of the nodes they just discovered. This list exchange can be performed by executing the "pecking" protocol as described below and leads to the construction at each node v of a set A_v of all nodes discovered by all v's neighbors that v did not discover.

Once the list exchange is finished, node v starts contacting the nodes in A_v to see whether they are nodes within its transmission range (i.e., undiscovered neighbors). To this purpose, a node v alternates for a predefined amount of time between page and page scan modes attempting to discover the nodes in its A_v. More specifically, when in page mode v attempts to page one after another (round robin fashion) all the nodes in A_v. Each time two nodes u and v discover each other, they remove each other from their set A_u and A_v and exchange their lists of neighbors. This may lead to new nodes for u and v to be added to their sets A_u and A_v, i.e., to new nodes to page. The length of this phase has to be carefully chosen so to discover all the nodes in A_v that are actually in v's transmission range.

The second problem concerns the way BlueTrees has been defined. According to [12], once a node has been acquired as a slave by a master, it becomes a master itself, and starts contacting (i.e., paging) all its neighbors (except its own master). Assume that a master v has acquired two slaves u and z as slaves, and assume that u and z are nodes that discovered each others during the discovery phase. Then a deadlock situation may arise due to the fact that u starts paging z and vice versa. To solve this problem we have associated to the paging of neighboring devices a time-out. If a discovered neighbor does not reply to a page within a certain time, it is assumed that it is in page mode and thus that it belongs to a piconet already.

The "pecking" protocol. We establish an ordering among the nodes based on their "weight," i.e., according to a number ≥ 0 locally computed at each node. (For the purpose of protocol description, here we can consider the weight the node's unique identifier.) A node v checks whether it has the bigger weight among its neighbors $N(v)$. If this is the case, that node, called in the rest of the paper 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. While the piconet is up, the nodes exchange the information needed for the protocol operation. Then, the piconet is disrupted.

A non-init node u goes to page scan mode and waits for a page from all its neighbors with bigger weight ("bigger neighbors"). As soon as node u has received information from the bigger neighbors (i.e., it has been paged by all of them), it becomes the "bottom of the pecking order," i.e., being now the bigger node among those with which it has to exchange the information, it switches to page mode, and starts setting up temporary piconets with all its smaller neighbors (if any).

3.2. BlueStars

The protocol presented in [8], *BlueStars*, proceeds from the device discovery phase into the following two phases of piconet formation and of the interconnection of the piconets into a connected scatternet. Based on a locally and dynamically computed weight (a number that expresses how suitable that node is for becoming a master) and on the knowledge of the weight of its neighbors (obtained during the discovery phase) each node decides whether it is going to be a master or a slave. This decision is taken at a node depending on the decision of the bigger neighbors, and then communicated to the smaller neighbors. The mechanism through which this is implemented is similar to the pecking protocol described above. In particular, a node that decided to be master is either an init node or a node whose bigger neighbors decided all to be slaves. A node that has been told (via paging) by one or more of its bigger neighbors that they are masters, becomes the slave of the first master who paged it. This phase of the protocol leads to the partition of the topology resulting from the discovery phase into piconets with one master and a number of slaves which is not necessarily bounded (if not by the number of the nodes). Notice that no two masters can be neighbors.

After this phase, each master proceeds to the selection of *gateway devices* to connect multiple piconets so that the resulting scatternet is connected. In order to achieve connectivity it is necessary (and sufficient) that each master establishes a path with (i.e., chooses gateways to) all the masters that are at most three hops away [4]. The knowledge about which nodes are the masters two and three hops apart is achieved during the piconet formation phase. Specifically, each node v communicates its role (and possibly the identity and weight of its master) to all its smaller neighbors and to the bigger neighbors that became slaves. If a node is a slave, it waits for the smaller neighbors to communicate the same information. In this way, at the end of the piconet formation phase each node is aware of all its neighbors ID, and of the ID and weight of their masters, which are the information needed in the piconet interconnection phase.

The process of piconets interconnection is based again on a mechanism similar to the pecking protocol, this time executed only among the masters. The details of the rule adopted for consistent gateway selection and for piconets interconnection can be found in [8].

3.3. Yao protocol

The main aim of the *Yao protocol* proposed in [7] is to build up a connected scatternet in which each piconet has no more than seven slaves.

The protocol assumes that each node knows its own identity, a dynamically computed weight that indicates how much that node is suitable for serving as a master (as in BlueStars), and its own location in the plane (usually provided by an on-board GPS device, or by any suitable inertial positioning system device). It is assumed that, as the outcome of the device discovery phase, a node also knows the identity of its neighbors, their weight and their location. By using the pecking protocol, the discovered devices also exchange information about their neighbors (achieving two-hop neighborhood knowledge).

For the sake of clarity, in the description of the algorithm we assume that nodes are scattered in the plane and that the network graph resulting from the device discovery phase is a connected unit disk graph.

Given that the discovery phase does not produce an UDG, in the simulation experiments that we performed, we run the extra phase described for BlueTrees in order for this protocol to correctly work. Note that in this case, since node v is aware of the location information of its two-hop neighbors, only the nodes expectedly in v's transmission range need to be included in A_v. This reduces consistently the number of nodes to be paged compared to the number of nodes to be paged by a node executing BlueTrees.

The knowledge of the location is exploited for applying to the UDG geometric-based techniques to reduce the degree of the network to at most $k \leq 7$. The Yao construction is executed at each node v and proceeds as follows. Node v divides the plane that surrounds it in k equal angles. In each angle, node v chooses the closest neighbor u, if any. (Ties are broken arbitrarily.)

A link between nodes v and u survives the Yao construction phase if and only if v has chosen u and vice versa. All other links are deleted. To make such decision nodes need to exchange with their neighbors the information on the nodes they selected.

This is performed by using again the pecking protocol described earlier.

Once a connected topology with such a bounded degree has been obtained, the BlueStars algorithm for scatternet formation outlined above uses the nodes' weight for selecting the masters, the slaves and the gateways necessary to form a degree-bounded connected scatternet.

4. Protocols Comparison

We have identified the following desirable properties for a scatternet formation protocol.
1. The produced scatternets should be connected.
2. Resilience to disconnections in the network. The protocol should be able to operate in the connected components of the network.
3. Routing robustness. The scatternets should have multiple routes between any pairs of nodes.
4. Piconet size limited to eight nodes, to avoid the overhead associated with parking and unparking slaves.
5. Resource-based master selection. Master selection should be driven by devices currently available resources as the master role is the most resource-consuming.
6. Distributed and localized operations. A protocol should be executed at each node with information available at the node itself locally (e.g., knowledge of one and two-hop neighbors).
7. Self-healingness. A protocol should react to changes in the network topology to maintain a scatternet that retains all the properties of the scatternet initially formed.

All three protocols we consider in this paper have distributed and localized operations. None of the solutions published so far for scatternet formation in multi-hop topologies address the issue of self-healingness. (That is why we do not take this property into consideration below: we have evaluated the performance of scatternet formation protocols according to the way they have been defined in the corresponding papers.)

The following table shows which of the five remaining properties 1–5 is satisfied by each the three protocols.

$\downarrow$ Protocols/Properties $\rightarrow$	1	2	3	4	5
BlueTrees	$\star$			$\star$	
BlueStars	$\star$	$\star$	$\star$		$\star$
Yao protocol	$\star$	$\star$	$\star$	$\star$	$\star$

Table 1. Scatternet properties.

We notice that the fact that the Yao protocol is a "five stars" protocol relies on the fact the property 5 is brought in by BlueStars, which implements the Yao protocol phases of piconet selection and interconnection. As explained, the tradeoff here is the need for extra hardware at each node (positioning devices).

4.1. Performance evaluation

To evaluate the performance of the three chosen scatternet formation protocols we have developed a Bluetooth extension to the VINT project network simulator ("ns2"). We based our extension on BlueHoc, the ns2-based simulator released by IBM. In particular, in order to implement the operations of the solutions for device discovery and scatternet formation, we have enriched BlueHoc with mechanisms for $a)$ giving to each node the possibility to dynamically assume either the role of master or the role of slave; $b)$ handling collisions that might arise during the establishment of a link, and $c)$ having a node alternating between inquiry and inquiry scan (These functions are not available in BlueHoc.)

In the simulated scenarios, n Power Class 3 BT nodes (i.e., nodes with maximum transmission radius of 10 meters) are randomly and uniformly scattered in a geographic area which is a square of side L. We make the assumption that two nodes are in each other transmission range if and only if their Euclidean distance is $\leq$ 10m. As mentioned, we call *visibility graphs* the topologies generated by drawing an edge between each pair of BT devices that are in each other transmission range (radio visibility).

In the simulation results presented here, the number of BT nodes n has been assigned the values 30, 50, 70, 90 and 110, while L has been set to 30m. This allowed us to test our protocol on increasingly dense networks, from (moderately) sparse networks, where only 95% of the visibility graphs are connected ($n = 30$), to highly dense networks. The average degree ranges from 27.9 for $n = 110$ down to 7.4 when $n = 30$ Figure 1. As the density increases, the average shortest path length in the visibility graph slightly decreases (10%) from 2.37 to 2.14, as shown in Figure 2.

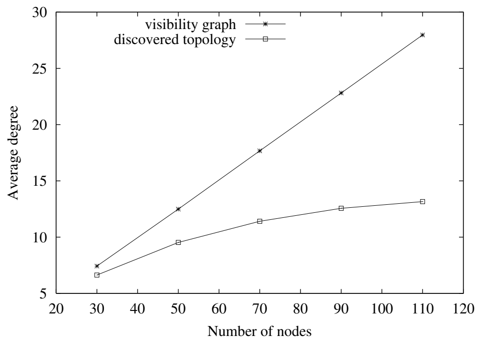

Figure 1. Nodal degree.

The simulated scenarios refer to the case when all the nodes enter the network within a time T_{acc} (set to $100ms$) and run the device discovery and scatternet formation protocols to self-organize themselves in a Bluetooth scatternet. All results presented in this section were obtained by running the three protocols over 300 connected visibility graphs.

Our simulations concern the two main aspects of scatternet formation, namely, device discovery, which is common to all protocols, and piconet formation and interconnection. In particular, simulations have been conducted to compare the performance of the different solutions which include 1. measuring the time needed for scatternet formation (including the phase of device discovery); 2. assessing the effect of the duration of the device discovery phase on the entire scatternet formation process; 3. counting the average number of piconets; 4. counting the average number of slaves per piconet; 5. counting the average number of roles (either master or slave) assigned to each node, and 6. comparing the average length of the routes between any two BT devices in the scatternet with respect to the average shortest path length between any pairs of nodes in the visibility graph.

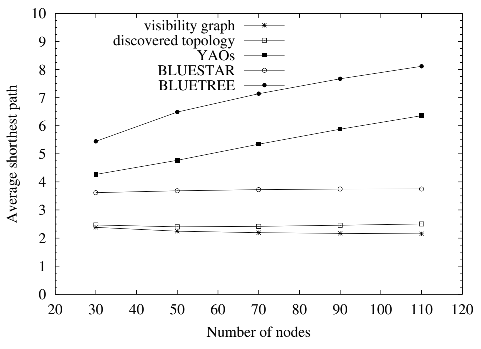

Figure 2. Shortest path length.

• **Device discovery in multi-hop networks.**
We have run the device discovery phase for a predefined time $T_{\text{disc}} \leq$ 10s over each visibility graph. Nodes alternate between inquiry and inquiry scan mode, spending a variable time, uniformly and randomly selected in the interval $(0.01s, T_{\text{inq}})$, in each mode. Unless otherwise specified T_{inq} has been set to 2s. The resulting topology, which we call a BT topology, has links only between those pairs of BT nodes that were able to discover each other during the device discovery phase.

The effect of different discovery phase durations are shown in figures 3 and 4. We notice that within 10s only a small percentage of neighbors is discovered (on average) by each node: The higher the density (and the nodes average degree), the longer the time needed to discover all the neighbors. When $T_{\text{disc}} =$ 10s, the percentage of discovered neighbors ranges from 90% ($n = 30$) down to 47% ($n = 110$) (Figure 3). The number of discovered neighbors increases with the length of the device discovery phase. However, the rate at which new neighbors are discovered decreases significantly with time, as there is a higher chance to handshake again with nodes already discovered. It is therefore impractical to require that each node become aware of all its neighbors.

A necessary condition for a protocol to form a connected scatternet is to start from a connected BT topology. Therefore, we have investigated how long the device discovery must run in order to discover enough neighbors to obtain a connected BT topology. As shown in Figure 4, we can provide statistical guarantees that the BT topologies are connected in case of moderately dense to heavily dense visibility graphs provided that the device discovery runs for at least 6s.

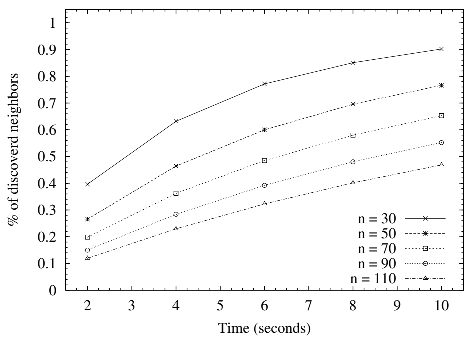

Figure 3. Discovered Neighbors.

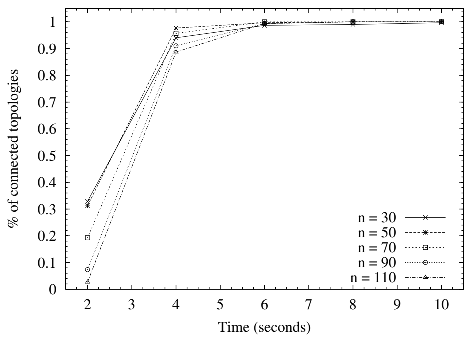

Figure 4. Connected BT topologies (%).

A short device discovery not only reduces the time needed for scatternet formation but results in a significant decrease of the average degree of the BT topologies with respect to the degree of the corresponding visibility graphs. This can be beneficial for protocols like BlueStars that do not form scatternets with limited number of slaves per piconet. When $T_{\rm disc} = 10$s, the average nodal degree decreases from 10% ($n = 30$) to the more significant 51.6% ($n = 110$) (Figure 1).

The price to pay is the increase (from 4% at $n = 30$ to 17% at $n = 110$) of the average shortest path length on the BT topology over the visibility graph, as shown in Figure 2.

To evaluate the impact of the parameter $T_{\rm inq}$, we have performed simulations varying $T_{\rm inq}$ in the set $\{0.5, 1, 2, 4\}$s. The results are depicted in Figure 5, for $n = 30$, 70, and 110. The case where $T_{\rm inq} = 2$s, used in all the remaining simulations, always outperforms the other cases. In particular, we observe that setting $T_{\rm inq}$ to one fourth of the best performing value, significantly reduces the percentage of discovered neighbors. This is due to the fact this parameter is tightly coupled with the length of the backoff interval: if $T_{\rm inq}$ is decreased without decreasing the length of the backoff interval (set to 0.6s according to the standard), then the backoff will often expire after the node switched to inquiry scan mode, making impossible any handshake.

Adopting a long $T_{\rm inq}$ reduces the rate at which nodes alternate between inquiry and inquiry scan, resulting in less chances for two nodes to be in opposite modes and to discover each other (and thus in performance degradation) at low density ($n = 30$). At high density, it is likely that several nodes are in opposite inquiry modes so that a long $T_{\rm inq}$ allows to discover many neighbors within the same interval, and results in performances close to the best case $T_{\rm inq} = 2$s.

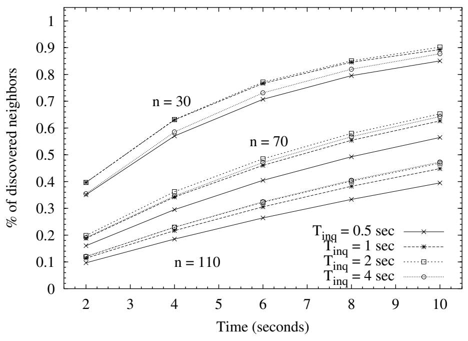

Figure 5. Effect of different inquiry durations on device discovery.

We have identified three features of the BT technology which are responsible for the device discovery for multi-hop BT networks being an extremely time consuming operation: a) the need to adopt (stochastic) mechanisms to have neighboring nodes in opposite inquiry modes, so they can discover each other; b) the overly long duration of the backoff interval as stipulated in the BT specifications (2048 clock ticks), and c) the impossibility of identifying the inquirer, which demands the construction of a temporary piconet (and the amount of time needed for an handshake) between neighbors that discovered each other already.

In our performance evaluation we have attempted to quantify the impact of each of the above effects. First we have run the device discovery using a shorter backoff interval length (one fourth of what specified in the Bluetooth specifications). As shown in Figures 6 and 7 by reducing the backoff interval length the percentage of discovered neighbors improves remarkably. After 8s of device discovery the whole visibility graph is discovered in moderately dense networks ($n = 30$), while at $n = 110$ the improvement ranges from 173% after 2s of device discovery to 74% after 10s.

We have then made a major change in the ID packet format, to allow it to carry the identity of the inquirer. Upon receiving an ID packet, nodes in inquiry scan mode check whether the inquirer is an unknown neighbor. Only in this case they proceed according to the usual procedures (either backing off or answering the packet with an FHS packet that initiates the set up of a temporary piconet). Otherwise, the packet is ignored. The trade-off here is that we had to "slow down" the scan of the inquiry frequencies. Instead of an ID packet every half a slot, since the ID packet carries now the inquirer's address, it takes the whole slot to send the enlarged ID packet. As shown in figures 6 and 7 we have verified that the benefits due to these modifications overcome the possible performance degradation due to the extra time needed for two neighbors to start handshaking. At $n = 30$ and for $T_{\text{disc}} \geq 8$s the whole visibility graph is usually discovered. As the nodes density increases, the curve representing the performance of device discovery with modified inquiry procedures steadily outperforms the performance of device discovery with the standard parameters. As expected, as the time spent in the discovery phase (and thus the number of discovered neighbors) increases, the improvement achieved by the modified inquiry increases (from 25% after 2s at $n = 110$ to 36% after 10s), since it is more and more likely that two neighbors that already discovered each other would do it again. At $n = 110$ and for $T_{\text{disc}} = 10$s, 64% of the neighbors are discovered if the inquirer can be identified as opposed to only 46% of discovered neighbors in the specifications compliant case.

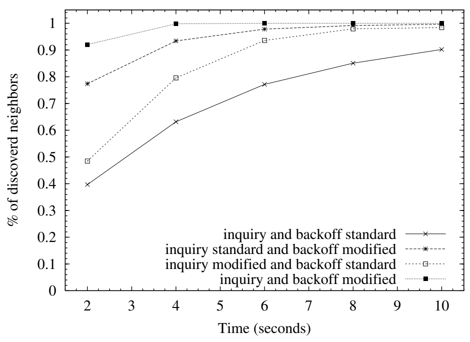

Figure 6. Discovered neighbors (30 nodes).

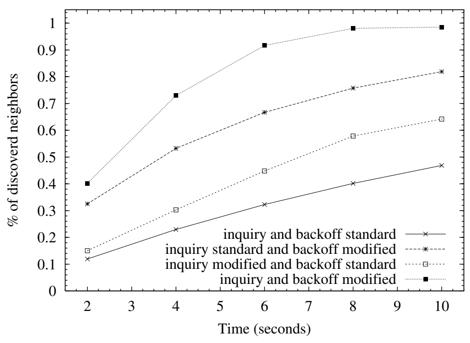

Figure 7. Discovered neighbors (110 nodes).

Finally, we have considered the combined cases of shorter backoff and ID packet carrying the inquirer identifier. This leads to extremely good performance, as all neighbors are discovered within 4s at $n = 30$, and within 8s in the heavily dense case of networks with 110 nodes.

By shortening the backoff interval or allowing the inquirer to be identified we observe also a remarkable decrease in the duration of device discovery necessary to obtain connected BT topologies. Independently of the network density, using a shorter backoff reduce to less than 2s the duration of device discovery necessary to obtain connected BT topologies. When we consider only the identification of the inquirer connected topologies are obtained after no more than 4s, while at least 6s are needed in case we run the standard procedures. We observe that even a very modest change of the standard (reducing the length of the backoff interval) leads to impressive improvements on the average duration of device discovery and therefore of the overall scatternet formation.

• **Performance evaluation comparison of the three scatternet formation protocols.**

We investigate the time needed to complete the three protocols as well as metrics identified in the literature as measures of the "quality" of the generated scatternets. The BlueTrees performance results refer to the 'best case' in which the central node is designated as blueroot.

Figure 8 depicts the average and 99th percentiles of the time needed by BlueStars, BlueTrees and by the Yao protocol to form a scatternet starting from the BT topologies. BlueStars is the fastest protocol, requiring less than 1.8s in the worst case, while the other two protocols may need up to 9.5s (Yao) or 12s (BlueTrees) to generate a connected scatternet. The major reason for BlueStars being faster than the Yao protocol is that the latter needs to run the extra phases with which every pair of nodes u and v that have discovered each other exchange their neighbor lists (via the pecking protocol), start paging all nodes in their sets A_u and A_v (we call this phase "link replenishing" phase) and exchange information (again via the pecking protocol) on the links selected to be part of the Yao topology ("Yao construction" phase). The link replenishing phase needs to be performed for at least 2s to have the statistical guarantee that all the nodes in the set A_v have been contacted by node v and all needed links are set up. Since almost all links in the visibility graph are discovered during this phase, a very high number of neighbors have to be paged in the Yao construction phase, making the pecking-like exchange of information much more time consuming than in the other phases or in the BlueStars case. As expected, the time needed for the Yao protocol to complete its operations significantly increases with the number of nodes as this results in a higher number of neighbors to contact. The last part of the Yao protocol is very fast, since it is implemented by BlueStars now applied to sparser Yao topologies.

BlueTrees also needs to run the link replenishing phase before the actual tree construction can be started. Since no location information are available, no node can be automatically deleted by the set of potential neighbors A_v as in the case of the Yao protocol. This imposes a longer time to successfully complete the link replenishing process. Based on our simulation results we had to set the length of such phase to 4s to have the statistical guarantee of setting up all the links needed for BlueTrees to operate correctly. Furthermore, the protocol duration reflects the need to adopt timeouts to solve possible deadlocks in the formation of the BlueTree.

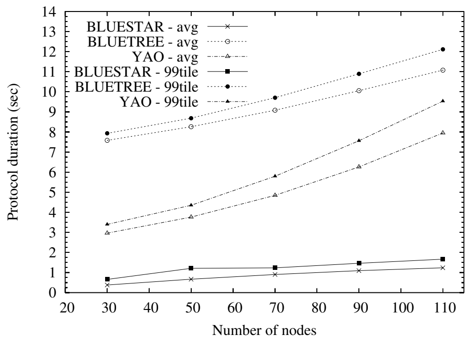

Figure 8. Scatternet formation durations.

Figure 9 shows the average number of piconets in the scatternets generated by the three protocols. In a BlueTree, all nodes but the leaves are masters, resulting in a high number of piconets which varies from 52% to 55% of the total number of nodes.

The number of piconets is much lower in BlueStars, ranging from 30% to 35% of the number of nodes. As the number of nodes increases, a higher number of piconets are needed, as expected, to partition the nodes into piconets. When a bound on the maximum number of slaves per piconet is enforced, like in the Yao case, the number of piconets in the scatternet further increases. This, in turn, results in a higher number of extra piconets needed for interconnecting adjacent piconets, motivating the overall 48%-75% increase in the number of piconets formed by Yao with respect to the number of BlueStars piconets.

The average number of slaves per piconet is depicted in Figure 10, together with the 95th percentile of the number of slaves. Both BlueTree and Yao result in piconets with a very limited number of slaves: 95% of the piconets has less than 4 slaves in BlueTree and less than 5 in Yao. Both the two protocols guarantee that no piconet has more than 7 slaves. The size of the "bigger" piconets instead exceeds the threshold of 7 in BlueStars. As mentioned, we use the duration of the device discovery phase as a "tuning knob" for obtaining discovered topologies with a decreased nodal degree. Doing so we could limit the 95th percentile of the number of slaves per piconet to 13. This, however, still requires parking and unparking of slaves, which may lead to time and bandwidth inefficiencies.

Critical performance measures for Bluetooth scatternets are the (average) number of roles assigned to each node and the number of nodes with master-slave roles. A high number of roles per node translates into reduced throughput performance due to the overhead (an average of 2 slots) associated to piconet switching. Master-slave roles (e.g., the roles assumed by a master-master gateway) are inefficient since all the communications in the piconet of the master that switches to be slave have to be frozen.

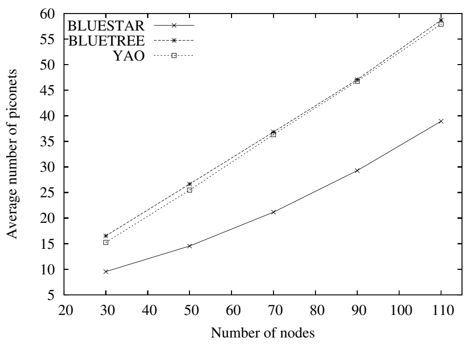

Figure 9. Number of piconets.

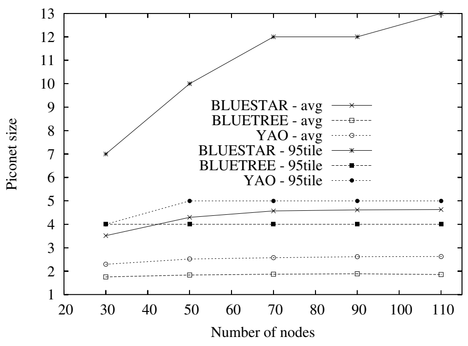

Figure 10. Number of slaves per piconet.

In Figure 11 we show the average number of roles assumed by BT devices in the scatternets produced by BlueTrees, BlueStars, and by the Yao protocol. The average number of roles per node in the BlueStars and Yao protocols slightly increases with n (from 1.4 at $n = 30$ to 2 at $n = 110$ in BlueStars, from 1.6 at $n = 30$ to 1.9 at $n = 110$ in Yao) to take into account the higher number of adjacent piconets that need to be joined (and thus the number of gateways). In BlueTrees all internal nodes have two roles, while the leaves only assume the role of slave. Since the percentage of nodes which are masters do not significantly change with n, the average number of roles per node remain steadily around 1.5, independently of n. A major difference among the protocols is that BlueTrees adopts master-master gateway for piconet interconnection, while BlueStars and the Yao protocol use gateway slaves whenever possible, or intermediate gateways when gateway slaves are not available. The number of master-slave roles is thus much more limited (from over 50% of the nodes in BlueTrees to only 8% and 13-20% in BlueStars and Yao, respectively). At the same time, BlueStars and the Yao protocol have the further advantage of forming scatternets with higher degree of piconet interconnection (a mesh like topology instead of a tree).

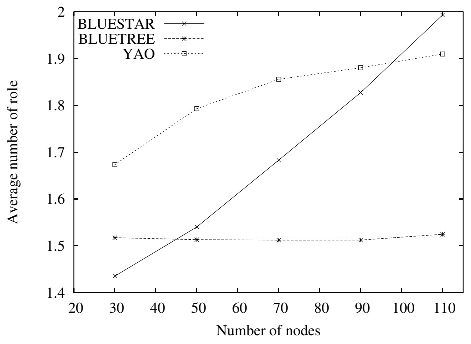

Figure 11. Number of roles per node.

Figure 2 shows the increase of the average length of shortest paths in the formed scatternets with respect to the same metric in the discovered topologies and in the visibility graphs. We first observe that there is little difference between the shortest path lengths in the visibility graph and the lengths of the same paths in the discovered topologies. The highest increase is for networks with 110 nodes, and it is just around 10%. Among the three protocols, BlueStars is the one with the shortest routes between any two nodes. The average increase of route length in the mesh-like scatternets formed by BlueStars with respect to the length of the corresponding routes in the discovered topologies is 50%, independently of the increasing number of nodes. This is essentially due to the piconet-based network organization, which may force nodes that are neighbors in the visibility graph to communicate through their common master (if they belong to the same piconet), or through a possibly long inter-piconets route in case they belong to different piconets. Routes in scatternets formed by the Yao protocol are from 18% ($n = 30$) to 70% ($n = 110$) longer than routes in BlueStars scatternets, to reflect the higher number of piconets that have to be crossed. (A higher number of piconet is formed by the Yao protocol because of the limit imposed on the number of slaves per piconet.) Finally, as expected, in the tree-like scatternets formed by BlueTrees routes are longer—from 30% ($n = 30$) to 94% ($n = 110$) than those in BlueStars scatternets. In this case two nodes that are possibly close to each other have to communicate through the only one route that passes through the first common ancestor.

5. Conclusions

This paper described solutions to the problem of scatternet formation, i.e., the problem of setting up multi-hop networks of Bluetooth devices. Three protocols for scatternet formation have been compared both by listing which of the desirable properties of scatternets are satisfied by the obtained scatternets and by means of thorough simulations.

References

[1] M. Ajmone Marsan, C. F. Chiasserini, A. Nucci, G. Carello, and L. De Giovanni. Optimizing the topology of Bluetooth wireless personal area networks. In *Proceedings of IEEE Infocom 2002*, New York, 23–27 June 2002.

[2] S. Baatz, C. Bieschke, M. Frank, P. Martini, C. Scholz, and C. Kühl. Building efficient Bluetooth scatternet topologies from 1-factors. In *Proceedings of WOC 2002*, 2002.

[3] 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, Hawaii, October 27–30 2002.

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

[5] http://www.bluetooth.com. *Specification of the Bluetooth System, Volume 1, Core*. Version 1.1, February 22 2001.

[6] C. Law, A. K. Mehta, and K.-Y. Siu. A new Bluetooth scatternet formation protocol. *ACM/Kluwer Journal on Mobile Networks and Applications (MONET), Special Issue on Mobile Ad Hoc Networks (A. Campbell, M. Conti and S. Giordano, eds.)*, 8(5), October 2003. to appear.

[7] 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.

[8] C. Petrioli, S. Basagni, and I. Chlamtac. Configuring BlueStars: Multihop scatternet formation for Bluetooth networks. *IEEE Transactions on Computers, special issue on Wireless Internet (Y.-B. Lin and Y.-C. Tseng, eds.)*, 2002. in press.

[9] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire. Distributed topology construction of Bluetooth personal area networks. In *Proceedings of the IEEE Infocom 2001*, pages 1577–1586, Anchorage, AK, April 22–26 2001.

[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, Hawaii, 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.