CHAPTER 4

SCATTERNET FORMATION IN BLUETOOTH NETWORKS

STEFANO BASAGNI, RAFFAELE BRUNO, and CHIARA PETRIOLI

4.1 INTRODUCTION

It is widely anticipated that fourth-generation wireless systems will extensively rely on the unlicensed operations provided by *ad hoc communications* [1]. 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.

The Bluetooth (BT) technology, as described in the Specifications of the Bluetooth System, Version 1.1 [2], is expected to be one of the most promising enabling technologies for ad hoc networks. Originally introduced as short-range cable replacement, the BT specifications define ways that each BT device can set up multiple connections with neighboring devices so that communication can be established in a multihop fashion. In this sense, Bluetooth devices spread out in a geographic area can provide the missing wireless extension to the various heterogeneous network infrastructures, allowing a more pervasive wireless access.

This chapter describes solutions to the fundamental problems that need to be addressed for the self-organization of Bluetooth devices into an ad hoc network.

According to the specifications, when two BT nodes that are in each others' 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 seven of which can be actively communicating with the master at the same time. If a master has more than seven slaves, some slaves have to be "parked." To communicate with a parked slave, a master has to "unpark" it, while possibly parking another slave.

Mobile Ad Hoc Networking. Edited by S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic.
ISBN 0-471-37313-3

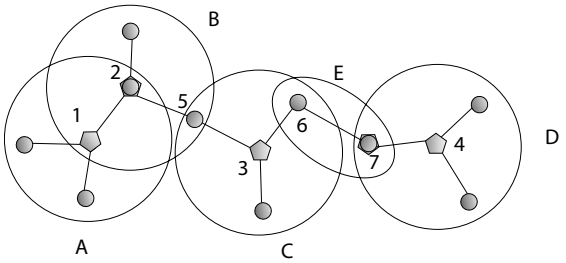

Figure 4.1. Five piconets forming a connected scatternet.

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 creating a multihop ad hoc network called a *scatternet*.

Figure 4.1 shows the case in which 13 BT devices have been partitioned into four piconets (A, B, C, and D). Masters are represented by pentagons (surrounded by a large circle that represents their transmission radius), whereas slaves are depicted as small circles. Adjacent piconets can be interconnected in different ways. Piconets A and B depict the *master–master* case, when two masters are neighbors and interconnection is achieved by having one of the two masters joining the piconet of the other as a slave (in the figure, node 2 became the slave of node 1). Two piconets can be joined by a common slave, termed a *gateway slave*. This is the case of piconets B and C, which are joined by node 5. The third case is when piconets are interconnected through a pair of neighboring slaves, called in the following *intermediate gateways,* as in the case of piconets C and D, joined by nodes 6 and 7. In the latter case, interconnection requires that one of the two intermediate gateways becomes the master of a new piconet that includes the other intermediate gateway as a slave (in the figure, node 7 becomes the master of the extra piconet). With the creation of piconet E, the five piconets of Figure 4.1 form a connected scatternet.

In this chapter, we describe the solutions proposed so far for scatternet formation. The next section introduces to the basics of the Bluetooth technology. In Section 4.3, we define the problems underlying scatternet formation, and we also sum up the desirable properties that should be satisfied by a generated scatternet. Section 4.4 surveys solutions that have appeared so far in the literature. Sections 4.5 and 4.6 describe in detail two protocols for generating connected scatternets. Section 4.7 describes some concerns raised by the implementation of some of the protocols according to the current version of the Bluetooth specifications. Finally, Section 4.8 concludes the chapter.

4.2 BLUETOOTH BASICS

In this section, we briefly describe the procedures of the Bluetooth technology that are needed to describe solutions for scatternet formation. This section is not intended to provide a detailed description of the Bluetooth system, for which the reader is referred to [2].

Bluetooth operates in the 2.4 GHz, unlicensed ISM band. Frequency-hopping spread spectrum technology is adopted to reduce interference both among BT nodes and with technologies that operate in the same band such as IEEE 802.11b.

In order to establish a connection between two BT nodes, one of them assumes the role of *master* of the communication and the other one becomes its *slave*. This simple "one hop" network is called a *piconet* and may include many slaves, no more than seven of which can be active at the same time. All 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. In particular, a Time-Division Duplex (TDD) scheme is employed for intrapiconet communications: transmissions occur in pairs of 625 μs slots, the first of which is for master–slave communication, and the second for the communication from the polled slave to the master.

A BT device can time share among different piconets. In particular, a device can be either the master of one piconet and a slave in other piconets or a slave in multiple piconets. A node with multiple roles acts as *gateway* between the piconets to which it belongs. Piconets can be interconnected through gateways into a multihop ad hoc network called a *scatternet*.

Piconet formation is performed in two steps: first, devices must become aware of their neighboring nodes, that is, the nodes in their transmission range (device discovery); then, information must be exchanged to set up a link between a candidate slave and a candidate master (link establishment). According to the current BT specifications, the former step is accomplished by means of the *inquiry* and *inquiry scan* procedures, whereas 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 be in inquiry mode, and the other device be 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 specifications state 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 (one frequency every 1.28 s), thus increasing the probability of a handshake on the same frequency of the inquirer. As soon as an ID packet is received at a device in the 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 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 the inquiry scan mode, the inquirer instead, knows the identity and the clock of the neighbor that just replied. This enables the inquirer v to estimate the frequency hopping sequence used by its neighbor and thus to invite it to join its piconet as a slave. This invitation is accomplished by means of the paging procedures.

In order for two neighboring devices u and v to establish a link, one must be in page mode (for instance, node v) and the other in page scan mode (node u). By definition, the device that is in page mode is the master. Node v transmits a page ID packet on u's fre-

quencies, containing *u*'s address. When *u*, which is in page scan mode, receives such a packet, it immediately acknowledges it. At this point, *v* transmits to *u* a FHS packet that bears 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.

It may happen that device *u*, which is in page scan, is already the master of another piconet and that it could host *v* as one of its slaves. In this case, once a piconet has been established between *v* and *u*, with *v* as the master, the slave *u* can request a *switch* of role. This situation is explicitly addressed by the BT specifications, and it is implemented via exchanging a specific Link Manager Protocol (LMP) packet that instructs the two devices to switch to the frequency-hopping sequence of the new master.

To save the energy of BT devices, "low-power operation modes" have been included in the specifications that allow BT nodes to "go to sleep" when they are not actively involved in communication. This feature is also used to let a master "release" a slave so that the slave can perform protocol-related operations in another piconet. Among the several modes provided in the specifications for low-power operations, we outline here the functioning of the *park mode*. A slave that has been put in park mode by its master cannot be actively involved in communication with that master. However, parked slaves periodically wake up in predefined beacon slots to listen to their master communication. Unparking of (possibly multiple) devices is achieved by transmitting an LMP unpark Protocol Data Unit (PDU) in the beacon slot. This packet carries the ID of the devices to be unparked and their new active slave addresses. Parked slaves can trigger an unpark LMP PDU by sending explicit requests during preallocated slots (access window). Similarly, active devices can ask to be parked (or they can be parked by their master) by exchanging an LMP park packet with their master.

4.3 PROBLEM DESCRIPTION

The problem of scatternet formation concerns the grouping of the network nodes into piconets (*piconet formation*), and the joining of the piconets into a connected scatternet (*piconet interconnection*). These operations require each node to be aware of its neighbors. A phase of *device discovery* must, therefore, be performed before the actual scatternet formation process takes place. For these operations, we describe here their desirable features and the barriers to their implementation.

In what follows, given a set of BT nodes, we call a *visibility graph* the network topology in which there is a link between any two nodes whose Euclidean distance is less than or equal to the nodes' transmission radius (for the sake of protocol description, we assume that all nodes have the same transmission radius). These topologies are also often referred to as *unit disk graphs* [3].

4.3.1 Device Discovery

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*. In general, unless simplifying assumptions are made on the visibility graph (e.g., the visibility graph is a clique, as in "single-hop" topologies), this is the least information required for performing the follow-

ing phases of piconet formation and interconnection. As described in Section 4.2, the mechanisms provided by the BT specifications for device discovery (inquiry procedures) do not lead to the needed symmetric neighbor knowledge and require nodes to be in opposite inquiry modes in order to be able to communicate. 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 implementation of the device-discovery mechanisms as outlined above is challenged by a number of BT standard features. We mentioned already the asymmetry introduced by the inquiry procedures: since the inquirer does not transmit its unique BT address, a node that receives an inquiry ID packet cannot discern if this packet comes from an already discovered node. This leads to useless inquiry handshakes, and may compromise the possibility of knowing the entire neighborhood. In addition, ensuring that two nodes are in opposite modes (for instance, by having them alternating between inquiry and inquiry scan mode) can be guaranteed only statistically, and it is a time-consuming process. This is also exacerbated by the duration of the backoff interval, which makes the inquiry handshake long.

4.3.2 Piconet Formation

The piconet formation phase concerns the assignment of roles—either master or slave—to all the network nodes. As dictated by the specifications, if a node is a master, it is master only in one piconet. In general, a slave can be enrolled in more than one piconet.

Desirable properties for piconet formation include:

- Distributed operations. Networks of Bluetooth devices are characterized by high dynamics (e.g., because of mobility and nodes joining the network at different times), and gathering complete information about the changing network topology can be infeasible. Therefore, piconet formation should be executed at each node with limited knowledge of the node's surrounding topology. One-hop or two-hop neighborhood knowledge is what can be known at each node in reasonable time and with limited resource consumption.
- Piconet size limited to eight nodes. Since no more than seven slaves can be actively communicating with a master, a master of a piconet with more than seven slaves is forced to park and unpark its slaves in order for all of them to be able to communicate. Parking and unparking have an associated overhead both in terms of induced delay and bandwidth. The throughput available to the nodes is significantly reduced in the case of large piconets, leading to inefficient operations.
- Resource-based master selection. Being a master is more resource consuming than being a slave, as masters have to handle and coordinate all the communications to and from all the nodes in the piconet. Piconet formation should, therefore, be performed, taking into account the different types of devices and their available resources when assigning the role of master.

4.3.3 Piconet Interconnection

The final phase concerns the selection of *gateway devices* to interconnect multiple piconets into a scatternet. There are three ways of interconnecting two piconets:

1. *Master–master*. In the case in which the masters of the two piconets are neighbors, interconnection can be achieved by having one of the two masters join the other piconet as a slave.
2. *Gateway slaves*. When a slave is neighbor of two masters, whether that slave belongs to only one of them or to both, it may be used to interconnect the two piconets by joining the piconet to which it does not belong (if any). When this is the case, that slave is called a gateway slave.
3. *Intermediate gateways*. This is the case in which two masters have two slaves that are neighbors. The interconnection of the two piconets can be achieved by requiring that one of the two gateways becomes the master of a new piconet that includes the other intermediate gateway as slave.

Desirable characteristic of piconet interconnection (which are properties of the resulting scatternet) are:

- Connected scatternet. If the topology resulting from the device discovery phase is connected, the scatternet generated by a scatternet formation protocol should also be connected.
- Resilience to disconnections in the network. If the topology resulting from the device discovery phase is not connected, a scatternet formation protocol should be able to correctly operate in the connected components of the network.
- Routing robustness. The scatternet should have multiple routes between any pairs of nodes.
- Limited route length. Scatternet routes are longer than the corresponding routes in the topology resulting from the device discovery phase for two main reasons:
  1. All intrapiconet communications pass through a master (two neighboring slaves that belong to the same piconet cannot communicate directly).
  2. The piconet-based network organization, which, for instance, may force nodes that are one hop away but that do not belong to the same piconet to communicate through a much longer interpiconet route.

Scatternet formation protocols should carefully select gateways so that the increase in the route length is limited.

- Selection of gateway slaves. Whenever possible, gateway slaves are to be preferred for piconet interconnection. This is due to the fact that when a master is also a slave for interconnection purposes, when it acts as a slave, all the communications to and from all the nodes in its own piconet are "frozen," which clearly detrimentally affects throughput performance.
- Small number of roles per node. A node should have the minimum number of roles. If a node has x roles, then it belongs to x different piconets. Switching between two piconets has an overhead due to synchronization to the frequency hopping sequence of the current master.
- Self-healing. When the network topology varies dynamically (due to nodes mobility, arrival of new nodes, failures of links/nodes, etc.), a scatternet formation protocol should be able to converge to a scatternet that retains all the properties of the initial scatternet.

Papers that have addressed desirable properties of scatternet formation protocols along with the identification of key metrics for their performance evaluation are [4], [5], [6], and [7].

4.4 OVERVIEW OF PROPOSED SOLUTIONS

The BT specifications describe methods for device discovery and for the participation of a node in multiple piconets. However, solutions for scatternet formation are not provided.

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 *multihop* scenario. The solutions are usually distributed and localized, in the sense that the protocols are executed at each node with limited knowledge of the surrounding topology.

4.4.1 Single-Hop Solutions: Device Discovery and Scatternet Formation

In some of the scatternet formation protocols for single-hop topologies, the device-discovery phase takes place concurrently with the phases of piconet formation and interconnection. Therefore, in this section the three phases are described together. In all cases, the nodes involved in the device discovery process keep switching between the inquiry and inquiry scan modes. This mechanism allows two nodes to eventually be in opposite modes at the same time, which is the needed condition for them to meet each other.

Among the solutions that work only in single-hop topologies with n nodes, the first algorithm for scatternet formation has been presented in [8]. The *Bluetooth Topology Construction Protocol* (BTCP) described in that paper is based on a distributed leader-election process. The leader election is performed by means of the inquiry and inquiry scan procedures so that when two neighboring nodes discover each other, one of the two nodes "wins," that is, it keeps participating in the discovery of other nodes, while the loser quits this phase, letting the winner know about its identity, its clock, and the identities and the clocks of all the nodes it was made aware of via previous confrontations. The elected leader (final winner) will eventually know the number, identities, and clocks of *all* the nodes in the network, based on which it decides the role that each node performs in the final scatternet. The computed roles are then communicated by the leader to all nodes. Designated masters are informed of the list of their designated slaves. The specific centralized algorithm, performed locally by the leader aims at minimizing the number of piconets while generating a mesh-like connected scatternet whose piconets have no more than seven slaves and are interconnected by gateways of degree two. The centralized algorithm executed by the leader generate a connected scatternet that satisfies the required properties only when the number of nodes in the network is ≤ 36. When $n > 36$, other centralized schemes could be used such as the one proposed in [9], which also takes traffic into account. The problem of scatternet formation is formulated as an integer linear programming problem where the objective function to be minimized is the traffic load at the most congested node (bottleneck node). Another centralized scheme has been presented in [10], which works for networks with any number of nodes. The aim of the protocol is to design a scatternet topology for which optimal interpiconet scheduling can be defined and that obtains max–min fairness. Once it has gathered all the information about all the network

nodes, the leader computes a k-regular topology (i.e., a topology in which all nodes have k links). The final topology is obtained by selecting and combining k different disjoint sets of edges that are (near-) perfect matchings (also called 1-factors), $2 \leq k \leq 2n - 1$. The authors describe a selection of the 1-factors that leads to a connected topology.

A randomized distributed algorithm for single-hop topologies is described in [11]. The protocol proceeds in rounds (i.e., it is a synchronous protocol). The devices are grouped into components (that may be either a single device, a piconet, or a connected scatternet), each of which has a leader. They are progressively joined to form the final connected scatternet. In every round, leaders of different components attempt to discover each other. This is performed by having each leader randomly enter either the inquiry or inquiry scan mode for that round. Leaders in opposite modes that discover each other decide which of the nodes in their components they can be interconnected with, thus forming a larger component. The protocol terminates when all the nodes belong to the same component. The resulting scatternet has the following properties: Each piconet has no more than seven slaves, and every gateway has degree two. The number of generated piconets is close to the theoretical minimum $\lceil n/7 \rceil$. The scatternet, which is a tree, is formed in $O(\log n)$ rounds with high probability.

A tree-like scatternet is also produced in the Tree Scatternet Formation (TSF) protocol described in [12]. The idea behind the protocol is very similar to the one presented in [11], although TSF is an asynchronous protocol. At any point in time, the scatternet being generated by TSF is a forest of connected trees. As the protocol proceeds, trees are joined by their roots that, by periodically alternating between inquiry and inquiry scan mode, try to discover each other. Since the trees' interconnection happens only via the roots, and since any node could be a root, all nodes must be in the transmission range of each other for the generated scatternet to be connected (single-hop solution). Newly arriving nodes can be included in already formed trees, which render this solution self-healing, that is, able to cope with changes in the network topology. The solution aims at minimizing the number of piconets, and at keeping the number of nodes per piconet below seven. However, this cannot be always guaranteed. In general, generating a tree-like scatternet topology simplifies routing but introduces limits in terms of robustness and efficiency.

4.4.4 Multihop Solutions

Protocols for general multihop topologies rely on the assumption that each node is aware of its neighbors and this knowledge is symmetric. This knowledge is provided by the device discovery phase, which is, therefore, performed before the actual piconet formation and interconnection phases.

4.4.2.1 Device discovery. The solution for device discovery in multihop networks uses a mechanism introduced in [13] and [14] that is similar to that described in [8]. Each device alternates between inquiry mode and inquiry scan mode, remaining in each mode for a time selected randomly and uniformly in a predefined time range. The operations while in each of the two modes are those described in the specifications. When two nodes in opposite inquiry modes handshake, they set up a temporary piconet that lasts only the time necessary to exchange their ID and possibly other information necessary for the following phases of the protocol. The formation of temporary piconets and the exchange of information achieves the required mutual knowledge.

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_scan)
15             INQUIRY(min(T_inq, T_disc))
16  EXIT
```

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

Device v then randomly enters either the inquiry or inquiry scan mode, and computes the length of the selected phase (T_{inq} or T_{scan}). This computation is performed by randomly and uniformly selecting the phase duration in a predefined interval. While in a given mode, device v performs the inquiry procedures as described by the BT specifications. The procedures that implement the inquiry mode (procedure INQUIRY) or the inquiry scan mode (procedure INQUIRYSCAN) are executed for the computed time (T_{inq} and T_{scan}, respectively), not to exceed T_{disc}. Upon completion of an inquiry (inquiry scan) phase, a device switches to the inquiry scan (inquiry) mode. A node keeps alternating between the two opposite modes until $T_{\text{disc}} > 0$. As mentioned, to allow each pair of neighboring devices to achieve a mutual knowledge of each other, our scheme requires that whenever a device in inquiry (inquiry scan) mode receives (sends) an FHS packet, a temporary piconet is set up by means of a page phase, and devices exchange their ID and possibly other information. As soon as this 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.

The duration of the discovery phase should be chosen so that each node is made aware of enough of its neighbors to guarantee network connectivity. This implies that, as opposed to single-hop solutions, in multihop topologies when two nodes discover each other they both have to keep performing device discovery as there might be other nodes that need to discover them to be connected to the rest of the network (and vice versa).

4.4.2.2 Scatternet Formation. Among the solutions that apply to the more general case of multihop topologies, the scatternet formation protocol described in [15] requires that the protocol be initiated by a designated node (the *blueroot*) and generate 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, until the whole tree is constructed. In order to limit the number of slaves, it is observed that if a node in a unit disk graph has more than five neighbors, then at least two of them must be connected. This observation is used to reconfigure the tree so that each master node has no more than seven slaves. If a master v has more than seven slaves, it selects two of them that are necessarily connected and instructs one of the two to be the master of the other, which is then disconnected from v's piconet. Such branch reorganization is carried throughout the network, leading to a scatternet in which each piconet has no more than seven slaves. Depending on a selected node to start the formation procedure, this solution does not work in the case of networks whose topology after the discovery phase is not connected. Furthermore, the implementation of the protocol requires the use of time-outs to solve possible deadlocks in the piconet formation phase.

Solutions for scatternet formation in multihop BT networks that produce topologies different from a tree are those presented in [14], [16], [17], [18], and [19].

The protocol presented in [13] and [14] proceeds from the device discovery phase as described above into the *BlueStars* (i.e., piconet) formation phase, and the configuration of the BlueStars into the connected scatternet. The phase of piconet formation deploys a clustering-based approach for master selection [20]. Based on a locally and dynamically computed weight (a number that expresses how suitable that node is for becoming a master), each node decides whether it is going to be a master or a slave. This phase starts at some dynamically selected nodes and terminates with the formation of disjoint piconets, each with one master and possibly multiple slaves. The final phase concerns the selection of *gateway devices* to connect multiple piconets so that the resulting scatternet is connected.

This solution has the following features. It works for general multihop BT networks. The generated scatternet is a mesh with multiple paths between any pair of nodes. The selection of the BT masters is driven by the suitability of a node to be the "best fit" for serving as a master. The generated scatternet is connected whenever the network resulting from the device discovery phase is connected. Finally, even in case of a disconnected discovered topology, the protocol generates a connected scatternet over each connected component.

The scatternet formation scheme proposed in [16], *BlueNet,* produces a scatternet whose piconets have a bounded number k of slaves. After the device discovery phase, each node randomly enters the page or the page scan mode with probability p (phase 0). When a node succeeds in getting at least one slave, it proceeds to phase 1 and tries to acquire up to k neighboring nodes as slaves. Otherwise, it keeps randomly entering the page or page scan mode and executing phase 0 until all neighboring nodes have communicated that they joined some other node's piconet. (The fact that a node in phase 0 can actually contact all its neighbors can be guaranteed only statistically.) In case a phase 0 node remains isolated, it enters phase 2, goes to page mode, and tries to interconnect to neighboring piconets by acquiring as slaves one node from each such piconet (up to k). After having accomplished this task, a phase 2 node exits the protocol. A master in phase 1 that has contacted all its neighbors and acquired at most k nodes in its piconet, proceeds to phase 3, the piconet interconnection phase. In this phase, the slaves of the piconets formed in

phase 1, by alternating between page and page scan mode, attempt to set up links with neighboring slaves of other phase 1 piconets as instructed by their masters. The connectivity of the resulting scatternet is not guaranteed (i.e., not all the BlueNets are connected, even when the topologies resulting from the discovery phase are).

The main aim of the protocol proposed in [17] and [18] is to build up a connected scatternet in which each piconet has no more than seven slaves. To this purpose, degree reduction techniques are initially applied to the network topology graph to reduce the number of wireless links at each node to less than seven without disconnecting the network. Any (multihop) scatternet formation protocol can then be executed on the resulting topology, yielding to a scatternet whose piconet size is at most eight (one master and at most seven slaves). These techniques require each node to be equipped with additional hardware that provides the node with its current (geographic) location (e.g., a GPS receiver). Details of this solution, combined with the BlueStars protocol outlined above, are given in Section 4.5.

The idea behind *BlueMesh,* the scatternet formation protocol presented in [19] and [21], is to generate a connected scatternet by selecting some masters among the network nodes, and allowing each master to select *at most* seven slaves. The selection of the slaves is performed in such a way that if a master has more than seven neighbors, it chooses seven 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 interpiconet route between all masters that are at most tree hops away (i.e., all adjacent piconets are interconnected). This condition ensures the connectivity of the BlueMesh scatternet [22]. Further details about BlueMesh are given in Section 4.6. BlueMesh improves previous solutions in that: a) as opposed to BlueTrees, the generated scatternet is a more robust mesh and it also works in connected components of a possibly disconnected network; b) all generated scatternets are connected, which is not the case with BlueNet; c) as opposed to BlueStars, no piconet in a BlueMesh scatternet has more than seven slaves; and finally, d) no extra hardware is required as in the geometric-based solutions of [18].

Thorough performance evaluation of BlueStars has been presented in [23]. A performance comparison of the solutions for multihop scatternet formation presented in [14], [16], and [18] is given in [24].

4.5 GEOMETRIC TECHNIQUES AND SCATTERNET FORMATION

In this section, we describe the details of a scatternet formation protocol that produces scatternets that are connected and whose piconets have a bounded number k of slaves (usually it will be $k = 7$).

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, 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 also assumed that, as the outcome of the device discovery phase, a node also knows the identity of its neighbors, their weight, and their location.

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

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. Once a connected topology with such a bounded degree has been obtained, the BlueStars algorithm for scatternet for-

mation outlined above uses the nodes' weight for selecting the masters, the slaves, and the gateways necessary to form a degree-bounded connected scatternet.

In [18], several degree reduction techniques are described, and it is proven that the resulting degree-bounded topologies are connected. Here we describe only one of those techniques, namely the one that [18] deems the most promising for the Bluetooth technology, termed *Yao construction*. This technique was first proposed by Yao to construct the minimum spanning tree of the graph originated by a set of points in high dimension efficiently [25]. The reader is referred to [18] for a description of other geometric techniques for degree reduction in wireless networks modeled by UDGs.

The Yao construction is executed at each node v and proceeds as follows. Node v divides the plane that surrounds it into 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 contruction 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. The mechanism we use for information exchange is the temporary setup of a piconet between every pair of neighboring nodes. For this to be possible, we have to guarantee that every pair of nodes are in opposite page modes. This is obtained by having the nodes execute the following protocol. Upon completing the local selection of links, a node v checks whether it has the biggest weight among its neighbors $N(v)$. If this is the case, that node, called in the rest of the chapter an *init* node, executes the following procedure:

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

An init node goes into 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. The information exchanged in the temporary piconet concerns whether the two nodes have chosen each other or not.

Symmetrically, a noninit node u executes the following procedure:

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

Node u goes into page scan mode and waits for a page from all its neighbors with bigger weights ("bigger neighbors"). As soon as node u has decided about all the links with the bigger neighbors (i.e., it has been paged by all of them), it goes to the "bottom of the pecking order"; that is, being now the biggest node among those with which it has to exchange the link information, it switches to page mode, and starts setting up temporary piconets with all its smaller neighbors (if any).

The topology resulting from the Yao construction, as described, is connected and has the property that no node has more than k neighbors [18].

Once a connected topology with such a bounded degree has been obtained, the BlueStars algorithm for scatternet formation outlined in Section 4.4 uses the nodes' weight for selecting the masters, the slaves and the gateways necessary to form a degree-bounded connected scatternet.

Let us consider the network of Fig. 4.2 as the network resulting from the device discovery phase. The only node with more than seven neighbors is node 36. Therefore, node 36 executes the Yao construction procedure to discard one of its eight neighbors. Assuming that nodes 20 and 21 fall in the same seven angles in which node 36 has partitioned the plane around itself, the result of the Yao construction phase is the cancellation of the link between node 36 and node 21. At this point, a connected scatternet is obtained by executing BlueStars over the "Yao topology" just obtained (where now nodes 36 and 21 are no longer neighbors). BlueStars, described in [13] and [14], proceeds from the Yao topology to the following two phases of piconet formation and 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 and the Yao construction 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 is then communicated to the smaller neighbors. The mechanism through which this is implemented is similar to the pecking order protocol described above. In particular, a node that decided to be master is either an init node or a node whose bigger neighbors all decided 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 that paged it. This phase of the protocol leads to the partition of the topology resulting from the discovery phase and the Yao construction into piconets. BlueStars does not guarantee a bounded number of slaves per piconet. However, its combination with the Yao contraction (which limits the nodal degree to k) also obtains this desirable property. The execution of the piconet formation phase of BlueStars over the Yao topology obtained from the topology depicted in Fig. 4.2 is shown in Fig. 4.3.

Nodes 21, 30, and 36, being init nodes, start paging their neighbors, which are all in page scan mode. As depicted in Figure 4.3, node 21 is successful in paging nodes 17 and 20, which become part of its piconet. Node 30 has no competitors in having nodes 11 and 22 join its piconet. Node 36 forms the largest piconet by acquiring nodes 6, 8, 13, 15, and 25 as slaves. Once nodes 13, 22, and 25 have communicated their decision to become slaves, node 7, whose bigger neighbors are all slaves, decides to be a master,

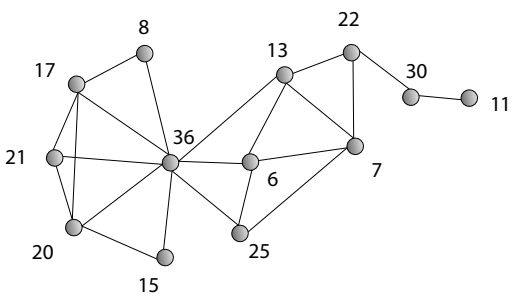

Figure 4.2. A Bluetooth network after the discovery phase.

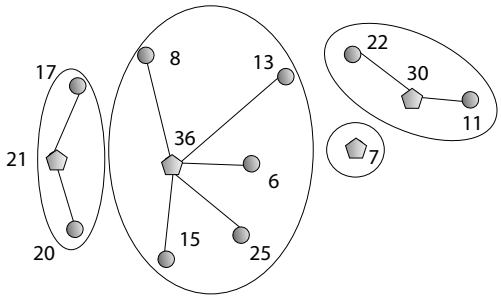

Figure 4.3. BlueStars' piconet formation.

and since node 6 is already affiliated to node 36, it will be a master of a piconet with one node.

After the piconet formation 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 [22]. 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 the identity of all its neighbors, and of the identity and weight of its masters, which is the information needed in the piconet interconnection phase. The process of piconet interconnection is based again on a mechanism similar to the pecking order protocol, this time executed only among the masters. The result of the BlueStars piconet interconnection phase over the network of piconets of Figure 4.3 is depicted in Figure 4.4.

By the end of the piconet formation phase, node 36 knows the identity and the weight of all the master at most three hops away, and the slaves through which it can reach them. For instance, one (or more) among nodes 6, 13, and 25 could be used as gateway slaves to interconnect to node 7. Once it has chosen one of these nodes (say, the biggest one: node

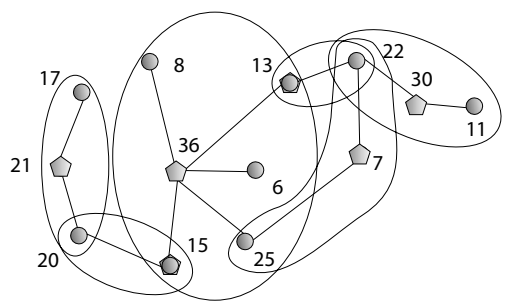

Figure 4.4. The scatternet produced by combining the Yao construction with BlueStars.

25), it instructs it to wait for a page from node 7. Node 7 also knows that nodes 6, 13, and 25 are the nodes through which the two piconets can be interconnected, and it adopts the same gateway selection rule of node 36. Therefore, it pages node 25 to enroll it as its slave. The same thing happens between masters 7 and 30, which select as intermediate slave node 22 (the sole node that can fulfill this purpose). The piconets of nodes 36 and 21 must interconnect via intermediate slaves. To this purpose, since both masters know each other and which of their slaves are neighbors of the slaves of the other, they can consistently choose two neighboring slaves and instruct them to page each other in order to form the needed new piconet. In the case depicted in Fig. 4.4, for instance, node 36 instructs node 15 (the biggest of its slaves that are neighbors of slaves of node 21) to page node 20, which is its neighbor that is a slave of node 21. Consistently, node 21 instructs its slave 20 to go into page scan mode and waits for a page from node 15. Similarly, the piconets of masters 36 and 30 are joined by the new piconet formed by node 13 and 22, which act as intermediate slaves. (The details of the rule adopted for consistent gateway selection and for piconet interconnection can be found in [14].)

4.6. BLUEMESH

In this section, we describe BlueMesh, a protocol for forming connected scatternets whose piconets have a bounded number of slaves. As with BlueStars, scatternet connectivity is guaranteed by establishing an interpiconet route between any two masters that are at most three hops away [22, Theorem 1]. Unlike the protocol presented in the previous section, BlueMesh does not need location information.

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, $i \geq 1$. G_1 is simply the topology after the device discovery phase (as before, we assume that this topology is a UDG). 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 gateway slave or via a pair of intermediate gateways, one of which belongs to one piconet and the other to the other piconet. Gateway slaves are selected in the current iteration so that the piconets they belong to are joined. Masters then proceed 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.

The functioning of BlueMesh is illustrated by the following example. We assume that BT devices know their identity, their weight, and the identities and weights of all their one-hop and two-hop neighbors. Two-hop neighbor knowledge can be achieved after the device discovery by executing the "pecking order" protocol described in Section 4.5, where the information exchanged via the temporary piconet between the nodes v and u are the lists of neighbors $N(v)$ and $N(u)$.

Each iteration of the protocol is performed locally at each node v and is made up of two parts: *role selection* (for piconet formation) and *gateway 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 two-hop neighbor discovery process has been completed). 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 into page mode. All the other nodes go into page scan mode.

Let us consider again the network of Fig. 4.2. Being the nodes with the biggest weight in their neighborhood, devices 30 and 36 are init nodes. They are masters and switch to page mode. All the other nodes go into page scan mode. Device 30, which has less than seven 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. Since we want to form piconets each with a number of slaves $\leq k = 7$, node 36 must select only seven of its eight slaves. Slave selection is performed at a master node v by executing the following procedure, where $S(v)$ is the set of selected neighbors and $C(v)$ denotes the set of v's bigger neighbors that are slaves and v's smaller neighbors.

```
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))
```

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 coverage is always possible by selecting at most five slaves [19].

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 five of its neighbors in $C(v)$ through which all other neighbors can be reached.

The function GET(m, W) returns a set of m nodes from the set W (for instance, the m smaller ones, or randomly chosen ones, or the bigger ones, etc.). It is used by COMPUTES for selecting additional slaves to a maximum of seven ($|S(v)| \leq 7$).

By executing COMPUTES(36), node 36 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. As opposed to what happens in BlueStars, when a node is invited to join a piconet, it always accepts the invitation even if it already belongs to other piconets. Piconet 36 is made up 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 of 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. This marks the start of the gateway selection part of the current 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. The information is obtained with an extra network-wide "wave" of messages exchanged in a pecking-like fashion during the role selection phase. Based on this information, each master decides which slaves to select as gateways 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 the 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 nongateway 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 with in iteration i. Going back to our example, 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 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 is 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 into 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 masters 36 and 30. The scatternet resulting from the execution of BlueMesh on the network of Fig. 4.2 is displayed in Fig. 4.5. 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.

4.7 IMPLEMENTATION CONCERNS

The main concerns about the implementation of most of the described protocols for scatternet formation in multihop networks are due to the discovery phase as we have described it in Section 4.4.2.1. In particular, we observed the following two problems:

First, it is extremely time consuming, and therefore impractical, to discover all the neighbors of a given node. We have simulated the device discovery phase by using a BT extension to the ns2 network simulator, which implements all the details of the BT protocol stack. The device discovery was run for a predefined time T_{disc} over each visibility graph generated by distributing uniformly and randomly $n = 30, 50, 70, 90, 110$ nodes over a square area of side $L = 30$ meters. The transmission range of each node was that of Power Class 3 BT nodes (10 m). Nodes alternate between inquiry and inquiry scan mode,

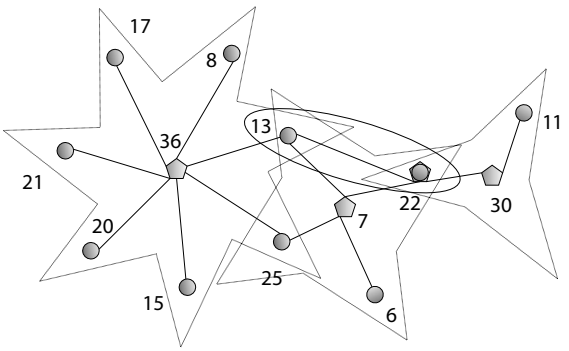

Figure 4.5. A BlueMesh scatternet.

spending a variable time, uniformly and randomly selected in the interval $(0.02s, 2s)$, in each mode. 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.

We observe that it may take a long time to discover all the neighbors of a node (e.g., only 47% of a node's neighbors have been discovered after 10 s of device discovery in networks with 110 nodes). However, a shorter time suffices for discovering enough neighbors so that the resulting BT topologies are connected, which is the needed requirement for obtaining connected scatternets. This is shown by Figure 4.6. 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 6 s.

Three are the features of the Bluetooth technology that we have identified as having a strong impact on the device discovery duration: a) The need to adopt (stochastic) mechanisms to have neighboring nodes in opposite inquiry modes, so they can discover each other; b) the impossibility of identifying the inquirer, which demands the construction of a temporary piconet between neighbors that have discovered each other already; and c) the overly long duration of the backoff interval as stipulated in the BT specifications (2048 clock ticks). In our performance evaluation, we have quantified the impact of each of these features on the performance of device discovery [24]. We observe that, by just decreasing the backoff duration to one-fourth of the value specified by the standard, a significant increase in the number of discovered neighbors is achieved that leads to connected topologies in less than 2 seconds. For instance, in networks with 110 nodes—by far the worst case in the simulated scenarios—over 80% of a node's neighbors are discovered within 10 seconds.

Second, the topology resulting from the device discovery phase may not be a UDG graph. This violates the assumption on which all protocols that provide connected scatternets with piconets with less than $k = 7$ slaves rely. For instance, it might no longer be true that given a node v with more than five neighbors, it is possible to find two among v's neighbors that are physically neighbors and have discovered each other. For the protocol to work correctly, after the device discovery phase, it is, therefore, necessary to perform an extra phase leading to a consistent knowledge of each node's neighborhood. Basically, we want two nodes that have discovered a common neighbor and are neighbors themselves to discover each other. This can be obtained in the following way. At the end of the

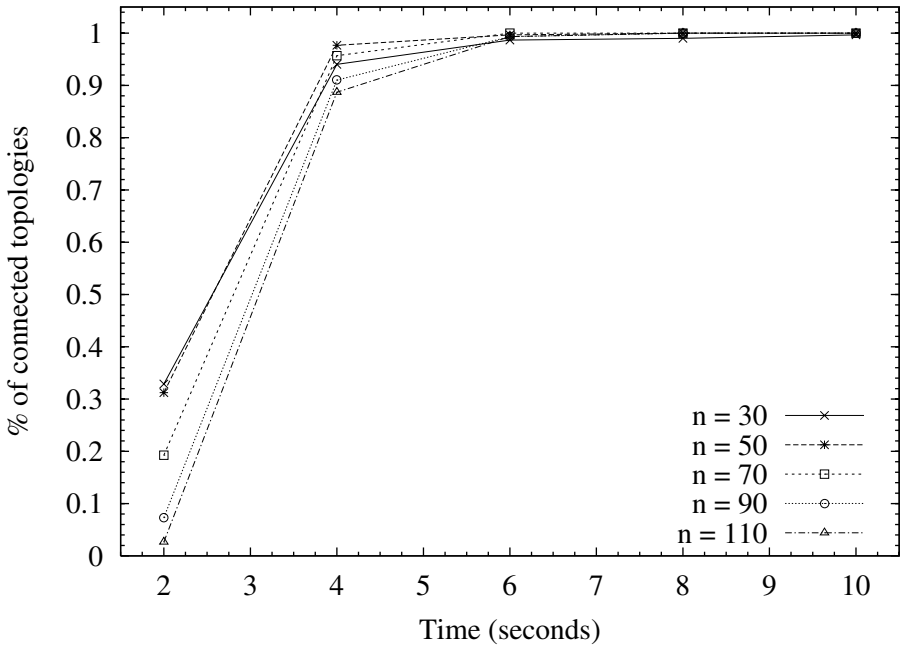

Figure 4.6. Percentage of connected BT topologies versus T_{disc}.

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 order" protocol as described above (Section 4.5) 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 (in round-robin fashion) of the nodes in A_v. Each time two nodes u and v discover each other, they remove each other from their sets 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, that is, to new nodes to page. The length of this phase has to be carefully chosen so as to discover all the nodes in A_v that are actually in v's transmission range.

4.8 CONCLUSIONS

This chapter described solutions to the problem of scatternet formation, namely, to the problem of setting up multihop networks of Bluetooth devices. Solutions for the two major cases of when the devices are all in each others' transmission range and the more general case of multihop topologies have been illustrated. Two approaches for generating connected scatternets whose piconets have no more than seven slaves for multihop networks have been illustrated in detail. Observations and comments were given that described con-

cerns arising while implementing scatternet formation protocols by following the current specifications (Version 1.1).

ACKNOWLEDGMENTS

The authors wish to thank Carla Fabiana Chiasserini, Imrich Chlamtac, Francesca Cuomo, Gabriele Mambrini, and Ivan Stojmenovic for valuable comments and useful discussions on the topics of this chapter.

REFERENCES

1. W. Kellerer, H.-J. Vögel, and K.-E. Steinberg, "A communication gateway for infrastructure independent 4G wireless access," *IEEE Communications Magazine, 40,* 3, pp. 126–131, March 2002.
2. http://www.bluetooth.com, *Specification of the Bluetooth System, Volume 1, Core,* Version 1.1, February 22, 2001.
3. B. N. Clark, C. J. Colburn, and D. S. Johnson, "Unit disk graphs," *Discrete Mathematics, 86,* 165–167, 1990.
4. G. Miklós, A. Rácz, Z. Turanyi, A. Valkó, and P. Johansson, "Performance aspects of Bluetooth scatternet formation," in *Proceedings of ACM MobiHoc 2000,* Boston, MA, August 11 2000.
5. P. Bhagwat and S. P. Rao, "On the characterization of Bluetooth scatternet topologies," http://www.winlab.rutgers.edu/pravin/bluetooth/index.html, 2000.
6. R. Guerin, E. Kim, and S. Sarkar, "Bluetooth technology: Key challenges and initial research," in *Proceedings of the SCS Conference on Networks and Distributed Systems, CNDS 2002,* San Antonio, TX, January 27–31 2002.
7. F. Cuomo and T. Melodia, "A general methodology and key metrics for scatternet formation in Bluetooth," in *Proceedings of IEEE Globecom 2002,* Taipei, Taiwan, 17–21 November 2002.
8. 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.
9. 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.
10. 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.
11. 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, 485–498, October 2003.
12. 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.
13. S. Basagni and C. Petrioli, "Multihop scatternet formation for Bluetooth networks," in *Proceedings of the 55th IEEE Semiannual Vehicular Technology Conference, VTC Spring 2002,* Birmingham, AL, May 6–9 2002, vol. 1, pp. 424–428.
14. 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.), *52,* 6, 779–790, June 2003.

15. 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, vol. 1, pp. 273–277.

16. 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.

17. 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.

18. 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.

19. C. Petrioli, S. Basagni, and I. Chlamtac, "BlueMesh: Degree-constrained multihop scatternet formation for Bluetooth networks," *ACM/Kluwer Journal on Special Topics in Mobile Networking and Applications (MONET),* Special Issue on Advances in Research of Wireless Personal Area Networking and Bluetooth Enabled Networks (G. Zaruba and P. Johansson, eds.), *9,* 1, 33–47, February 2004.

20. S. Basagni, "Distributed clustering for ad hoc networks," in *Proceedings of the 1999 International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN'99),* A. Y. Zomaya, D. F. Hsu, O. Ibarra, S. Origuchi, D. Nassimi, and M. Palis eds., Perth/Fremantle, Australia, June 23–25 1999, pp. 310–315, IEEE Computer Society.

21. C. Petrioli and S. Basagni, "Degree-constrained multihop scatternet formation for Bluetooth networks," in *Proceedings of the IEEE Globecom 2002,* Taipei, Taiwan, R.O.C., November 17–21 2002, vol. 1, pp. 222–226.

22. 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.

23. 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, vol. 1, pp. 208–212.

24. S. Basagni, R. Bruno, G. Mamerini, and C. Petrioli, "Comparative performance evaluation of scatternet formation protocols for networks of Bluetooth devices," *Wireless Networks, 10,* 12, 197–213, March 2004.

25. A. C.-C. Yao, "On constructing minimum spanning trees in *k*-dimensional spaces and related problems," *SIAM Journal on Computing, 11,* 4, 721–736, November 1982.