Wireless Netw
DOI 10.1007/s11276-007-0036-7

Performance comparison of Bluetooth scatternet formation protocols for multi-hop networks

Zhifang Wang · Robert J. Thomas · Zygmunt J. Haas

Abstract The interest in Bluetooth technology has stimulated much research in algorithms for topology creation and control of networks comprised of large numbers of Bluetooth devices. In particular, the issue of scatternet formation has been addressed by researchers in a number of papers in the technical literature. This paper is an extension of the work presented in [14, 15]. In this paper we present a complete description of what we believe to be a promising scatternet formation protocol – BlueNet, which was first proposed in [15]. Some modifications and enhancements are made to improve the connectivity of resulting scatternets. The metrics are chosen to evaluate the performance of resulting scatternets, such as the reliability, the routing efficiency, the piconet density, and the information carrying capacity. Based on the chosen metrics, performance is then compared among the scatternet samples generated by BlueNet and other two representative multi-hop scatternet formation protocols, i.e., BlueTrees [16] and LSBS [1]. Finally in the conclusion a discussion is presented on the compared scatternet formation protocols.

Keywords Bluetooth · Scatternet formation protocol · Performance evaluation

Z. Wang (✉) · R. J. Thomas · Z. J. Haas
ECE Cornell University,
Ithaca, NY 14853, USA
e-mail: zw18@cornell.edu

R. J. Thomas
e-mail: rjt1@cornell.edu

Z. J. Haas
e-mail: haas@ece.cornell.edu

1 Introduction

Bluetooth was proposed as a low cost universal wireless communication technology designed to enable various devices to communicate seamlessly without wires. In July 1997, the Bluetooth Special Interest Group (SIG) published an open specification for Bluetooth wireless communication. It was believed that Bluetooth would become one of the major technologies for short-range wireless networks and wireless personal area networks [2, 9, 17]. Though not explicitly addressed in the specification, Bluetooth wireless communication provides an enabling technology for multi-hop ad hoc networks. Taking into account the low cost of Bluetooth chips, large and inexpensive networks can be constructed and be used for automated information gathering, distributed micro sensing, and remote controls in many applications such as on-line field measurements for system controls and monitoring the health of an electrical power system. In our research we seek ways to satisfy the communication needs in substation automation. The power engineers need to place hundreds of sensors or devices in locations that are critical for power system monitoring and operating, to monitor voltages, currents, temperature, humidity, equipment usage, etc. Many of these applications could benefit from use of a wireless network as opposed to wired ones, because wireless networks can be more cost and time effective and are also easier to deploy. In this paper we address the problem of designing a Bluetooth wireless network composed of fixed nodes for deployment in existing power substations.

1.1 Bluetooth basics

A Bluetooth radio operates in the unlicensed Industrial-Science-Medical (ISM) band at 2.45 GHz and adopts frequency-hop transceivers to combat interference and fading. The ISM band contains either 79 or 23 separate RF channels, depending on the country where the technology is to be used. Each channel has a 1-MHz bandwidth. The nominal radio range of a Bluetooth device is 10 meters with a transmit power of 0 dBm. The range can be extended up to 100 meters by adding an amplifier with a transmit power of 20 dBm.

The basic structure for communication in a Bluetooth network is the *piconet*. A piconet contains one *master* node and up to 7 active *slave* nodes. All active Bluetooth nodes inside a piconet share the same 1-Mbps FHSS channel in a TDM scheme. All transmissions among Bluetooth devices in the same piconet are supervised by the master node operating over a channel-hopping sequence generated from the master's Bluetooth device address at a rate of 1,600 hops per second [2, 9].

If multiple piconets coexist in one area, some Bluetooth nodes are chosen to serve as *bridges* between the overlapping piconets, by allowing them to participate in more than one piconet (but a node can be a master in only one piconet) on a time-sharing basis. A group of connected piconets is referred to as a *scatternet*. Because of the need to re-synchronize its radio from one piconet to another and to perform the necessary signaling, the bridging nodes necessarily consume some of their capacity while switching between different piconets. This causes some performance limitations when building scatternets.

The Bluetooth specification defines the operations/ procedures for device discovery, called *Inquiry* and *Inquiry Scan*, and the operations/procedures for setting up master-slave links, called *Page* and *Page Scan*. In order to collect identification and clock information from its neighbors, a potential master node performs the Inquiry operation by transmitting a generic ID packet (called *inquiry ID packet*) over a generic inquiry frequency hopping sequence (called *inquiry FHS*), at a rate of 3200 hops per second. Both the inquiry ID packet and the inquiry FHS are known by all Bluetooth nodes. A node involved in an Inquiry Scan keeps listening for the generic inquiry ID packet (over a short window of 11.25 ms) by changing its listening frequency over the same inquiry FHS every 1.28 seconds. Once the node hears an inquiry ID packet, it will waits for a random back-off interval; then after hearing the inquiry ID packet for the second time, it responds with a specific signaling packet, which contains its own identification and clock.

After collecting the necessary information, the potential master node will enter the *page* state, during which it explicitly invites another device to join the piconet where it will be the master. The paging node repeatedly sends a generic slave ID packet over the page hopping sequence, which is generated based on the identification and clock speed of the paged device. A paged device must enter the *page scan* state to listen for, and subsequently respond to, the pages. The paging device changes it's transmit frequency at a rate of 3,200 hops per second and the listening paged device changes its listening frequency every 1.28 seconds. After receiving the page, the potential slave sends a confirmation to the master and the master sends back a FHS packet containing information that allows the slave to join and participate in communications in the master's piconet.

The Bluetooth device discovery mechanism, i.e., inquiry and inquiry scan, does not provide mutual neighbor knowledge, since only a node in inquiry scan can send its identification and clock to the inquiring node, but not vice versa. In [12] a scheme is proposed to achieve a symmetric neighbors' knowledge exchange. That is, the two neighbors, after a handshake in the inquiry and inquiry scan modes, proceed to page and to page scan mode, respectively. They then set up a temporary master-slave link and exchange any necessary information. When the exchange is complete the link is severed. In the simulations contained in this paper, all of the selected protocols use this mechanism as part of their device discovery procedure.

Given the distribution of a set of Bluetooth nodes, a *visibility graph* is created, which defines the network topology. In the visibility graph, a link exists between any two nodes that can hear each other. However, with the device discovery mechanisms provided by the Bluetooth Specification, it can be very time-consuming to discover all the links in the visibility graph. Therefore, one can make a tradeoff (as suggested in [11]) that includes getting only a partial but still connected topology graph, called a *discovered topology*, within a shorter period of time (e.g., 10 or 20 seconds).

1.2 Overview of scatternet formation protocols

Two Bluetooth nodes, even if they are in each other's radio range, cannot really communicate until a direct master-slave link has been established between them (in the case of one-hop communications), or between each pair of direct neighbors along the communication route (in the case of multi-hop communications). In order to avoid the delay of setting up all the necessary communication links for a large-scale Bluetooth network, a connected scatternet has to be built before communications can really take place. On the other hand, maintaining a scatternet inevitably costs some network's resources. Therefore, an efficient scatternet requires tradeoffs between maintaining a decent level of connectivity and reserving enough network resources for effective communications.

The problem of how to form an efficient scatternet in practical networking scenarios is still an open issue. In [8], G.

Miklos et al. generate random topologies for a set of Bluetooth nodes and investigate the possible correlation between topology parameters and scatternet performance through a number of simulation studies. In [16], G. Zaruba et al introduce "*Bluetrees*", which has two variations, namely, *Blueroot Grown BlueTrees* and *Distributed BlueTrees*. The former builds a scatternet starting from some arbitrarily specified node called *Blueroot*. The latter speeds up the scatternet formation process by selecting more than one root for tree formation and then merging the trees together to generate one "big" tree. In [6] and [13] the authors also propose decentralized algorithms for building tree-structured scatternets. The former requires a fully in-range topology; i.e., every Bluetooth node in the network can connect to any other node. The tree-topology scatternets select the smallest possible number of links to form a connected network in order to spend the least amount of network resources on maintaining the scatternet. However, the resulting scatternet has an inherent deficiency due to its treelike structure; it does not exhibit a high degree of reliability. (In this paper the reliability of a scatternet is evaluated by determining the average connectivity of the remaining network after some nodes fail.) If one parent node is lost, all the children and grandchildren nodes below it will be separated from the rest of the network and part of the tree (or even the whole tree) has to be rebuilt in order to retain connectivity. In a mobile network, this may happen quite frequently, making the scatternet very susceptible to disconnections. In addition parent nodes in the scatternet are likely to become communication "bottlenecks", making it difficult for the network to afford multiple communication flows. Finally, a tree structure lacks efficiency in routing multiple communications, because all the routing paths have to transverse the tree in upward and downward directions. This becomes slow in a system with a large number of nodes.

In [3], C. Foo and K. Chua suggest *BlueRings*, which results in ring-topology scatternets. The ring structure eases the task of routing and tries to reduce the possible contention between links' transmissions. But as the system grows large, the traffic delay increases linearly and multiple communications can't be well supported, because of severe competition between links for resources.

The scatternet formation protocols introduced in [5, 7, 11], and [10], namely, *BlueNet*, the *Yao* protocol, *BlueStars*, and *BlueMesh*, are the only ones that can produce a meshed scatternet for a multi-hop topology. The four protocols have two properties in common. First, they all operate in a distributed way and can be implemented by each node based only on the local knowledge of the node's neighbors. Second, all the protocols require that they have a *discovered topology* graph in hand before the protocols begin to build scatternets. Some of the protocols may have special requirements of the *discovered topology* in order for them to work. For example, *BlueTrees* and the *Yao* protocol require that immediate neighbors have identical knowledge about their common neighbors in order for their *degree reduction algorithm*[1] to work correctly. Therefore in the implementation of the protocol described in [1], the authors use a *replenish phase*[2] to achieve the needed symmetry in the *discovered topology*. On the other hand, *BlueMesh* needs two-hop neighbor information, which has to be achieved by two rounds of device discovery.

Our scatternet formation scheme, *BlueNet*, originally presented in [15], proposes a way to build a connected scatternet through phase transitions and operations. Five phase states are used, i.e., phase-0, phase-1, phase-2, phase-3, and "finish". During phase-0 and phase-1, separate original piconets are built. Any remaining isolated nodes that cannot form or join any original piconet during the phase-0 process will become connected to a neighboring piconet through operations defined in phase-2. In phase-3 the original piconets are interconnected with each other through cross-piconet links. Finally a mesh-topology scatternet is built, whose component piconets have a bounded number ($\leq N_{\max}$) of slaves. Though the scatternets generated by *BlueNet* tend to be denser than tree-structured scatternets and may require more communication overhead to maintain the scatternet links, simulations show that they have much better reliability and can carry much more communication traffic. As mentioned in [1] one shortcoming of *BlueNet*, due to the protocol limitation on the piconet size, is that the connectivity of the resulting scatternets cannot be 100% guaranteed, especially in a sparse environment. But in a medium-dense or dense connectivity environment (defined by a *density*[3] of one or greater), *BlueNet* works very well. This is based on the observation that no disconnections occurred during the 300 simulations described in [1]. In this paper we introduce some modification to the phase-3 operations, so that the connectivity of resulting *BlueNet* scatternets can be largely improved, without affecting other aspects of the performance. The details can be found in at the end of Section 2.

In [7] Li et al proposed a Bluetooth scatternet formation algorithm, referred to as the *Yao* protocol in [1]. The *Yao* protocol includes three phases. The first phase is for neighbor discovery and information exchange, during which each node learns about the node ID, the node degree, and

[1] A d*egree reduction algorithm* is a scheme used by some scatternet formation protocols, which is applied on the discovered topology before the protocol begins to build scatternets. It removes some links according to a specific rule, but still retains the connectivity of the resulting topology.

[2] The r*eplenish phase* is a specific phase used by some scatternet formation protocols, which executes *degree reduction algorithm*.

[3] In this paper d*ensity* is evaluated as the number of nodes per ten square meters. In all comparisons the Bluetooth radio range is set to 10 meters and the power level to 0 dBm.

the position of its one-hop or two-hop neighbors. The second phase, called planar sub-graph construction, is optional. During this phase, a sparser planar sub-graph can be created by eliminating some links from the *discovered topology* obtained in the first phase. In the third phase, the *Yao* structure is applied to the resulting graph from the first or the second phase to limit the number of wireless links at each node to at most k ($5 \leq k \leq 7$) while still retaining the connectivity of the resulting topology. Then the master-slave relations are assigned in the piconets and the scatternet is formed. This protocol is based on the assumption that each node knows the absolute or the relative position of itself and of each of its neighbors. Therefore each Bluetooth device is required to have extra hardware such as a GPS receiver, for determining geographic location.

The scatternet formation protocol presented in [11] by S. Basagni et al. is termed *BlueStars*. After the device discovery phase is completed, each node computes its *node weight*, which will be used and compared among neighbors to determine a node's master or slave role. The weight can be computed based on node degrees or on other parameters. Some nodes will decide to become master nodes if they have the largest weights in their neighborhood or if they learn that all of their larger-weighted neighbors have decided to become slaves. The other nodes will choose to become slave nodes waiting for the pages from their potential masters. Then the potential master nodes set up their own separate piconets, with their smaller-weights neighbors becoming their slaves. Finally, each master node selects appropriate gateway nodes to connect with all adjacent piconets. The resulting *BlueStars* scatternet may have more than seven slaves per piconet. This may degrade scatternet performance, as slaves need to be parked and unparked in order for them to communicate with their master. This problem can be solved by combining the *Yao* protocol from [7] and *BlueStars*. That is, before the *BlueStars* protocol proceeds to build its scatternet, the *Yao* structure is first applied to the *discovered topology* to reduce node degrees to no more than seven while still retaining the connectivity. The new combination, as suggested in [1] is called the LS-*BlueStars or LSBS* protocol. However, in order for the *Yao* structure to work, the added requirement of positioning hardware is needed.

One thing worth noting is that protocols such as *BlueTrees*, the *Yao* protocol, as well as *LSBS* all depend on some mechanism to limit their piconet size, which removes redundant links while still retaining network connectivity. Such a degree reduction mechanism is based on an observation in [16] that says that if a node contained in a unit disk graph has more than five neighbors, then there exist at least two of them, which are in each other's radio range. However, to make the observation valid and the degree reduction work correctly, two assumptions must be valid: 1. All nodes in the network are located in a planar space; and 2. Two nodes within each other's radio range can always hear each other. In the real world, locating all nodes in a large system in a planar space may not always be possible. Besides, considering the practical radio propagation environment, physically being within each other's radio range may not always imply connectivity, since the radio signal could be interfered with or shielded. In cases when these assumptions fail, the reduction protocols may remove a necessary link, resulting in a disconnected scatternet. Therefore, in a non-perfect environment the protocol should make sure that the remaining nodes are still reachable before removing a link, either through an Inquiry or a Page operation.

In [10] a new protocol named *BlueMesh* is proposed for building a connected scatternet, which has at most seven slaves per piconet, but without the extra requirement of position information. *BlueMesh* is similar to *BlueStars* except the way in which a master selects slaves among its neighbors. That is, if a master has more than seven neighbors, it only selects seven of them as its slaves, in such a way that all the others can be reached via the selected slave nodes. By this method, *BlueMesh* successfully limits the piconet size while eliminating the requirement for locating equipment. However, in order for *BlueMesh* to work, the device discovery phase has to discover two-hop neighborhoods for a node. This is achieved by implementing a two-round device discovery phase.

LSBS is in fact an improved or enhanced *BlueStars protocol* whose resulting scatternet would yield better performance than *BlueStars*. And *BlueMesh* is very similar to *LSBS* except in the way of how the degree reduction occurs during the building process. Consequently the resulting scatternets will have very similar, if not the same, performance. Therefore when we compare performance, only *LSBS* is chosen to compare with other two protocols *BlueTrees* and *BlueNet*. *BlueTrees* is selected because its resulting scatternet forms a connected topology with the least number of links for a set of Bluetooth nodes.

In [1] S. Basagni et al. implement and simulate four representative protocols, namely *BlueTrees*, *BlueNet*, *BlueStars*, and *LSBS*, based on *BlueHoc* (a Bluetooth simulation software based on NS2, developed by IBM). Performance metrics such as scatternet building time, the number of piconets, the number of slaves per piconet, the number of assumed node roles, and the scatternet route lengths are evaluated over different networks topologies. The performance evaluations in our paper are an extension of the work in [1]. We choose to compare the resulting scatternet performance of *BlueNet* with those generated by two other protocols, i.e., *BlueTrees* and *LSBS*. The metrics include the average connectivity percentage of the remaining network after some nodes break down, the average shortest path length, the piconet density, the link ratio, and the maximum traffic flows.

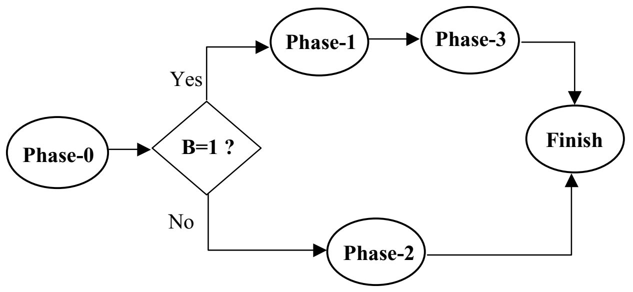


Fig. 1 Phase transitions for each bluetooth node in the *BlueNet* protocol

Among the performance metrics, the maximum traffic flow is most important, as it evaluates the information carrying capacity of the resulting scatternets.

In Section 2 a more detailed description of the *BlueNet* protocol is presented. At the end of Section 2, modifications of the *BlueNet* protocol are proposed for the purpose of speeding up the scatternet building process and for improving connectivity. In Section 3, performance metrics are introduced. These metrics are then used to compare scatternets generated by *BlueNet*, *BlueTrees*, and *LSBS* protocols. Finally, we conclude with a discussion on the compared scatternet formation protocols.

2 Description of the *BlueNet* protocol

The *BlueNet* protocol starts with a *discovered topology* obtained from the device discovery phase. Each node in the network then goes through the phase transition flow shown in Fig. 1 until it reaches the "finish" stage. The condition "B = 1?" in Fig. 1 asks whether the node can complete phase-0 by joining a phase-0 piconet or not. The *BlueNet* protocol is a distributed algorithm because the decisions about phase transitions and phase operations are made based only on local knowledge about each node's neighbors.

In the later part of this section, the *BlueNet* protocol will first be illustrated briefly by an example, and then the data records, phase transitions and operations will be described in more details in 2.1 and 2.2.

Consider the Bluetooth network example shown in Fig. 2, which represents a *discovered topology* for a 12-node system. A dashed line between two nodes indicates that they have discovered each other as neighbors during the device discovery phase. For example, node-1 found that nodes {6, 11, 12, 4} are its neighbors, while node-8 found nodes {3, 2, 4, 9} as its neighbors.

Initially, a Bluetooth node has no master or slave role. This initial mode is called phase-0. According to a pre-assigned probability, a phase-0 node performs the page or the page

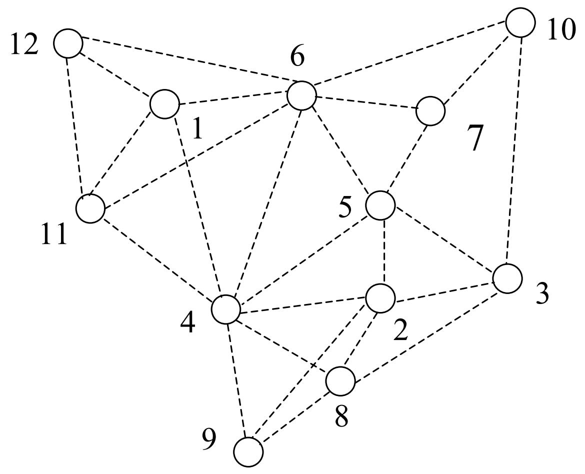


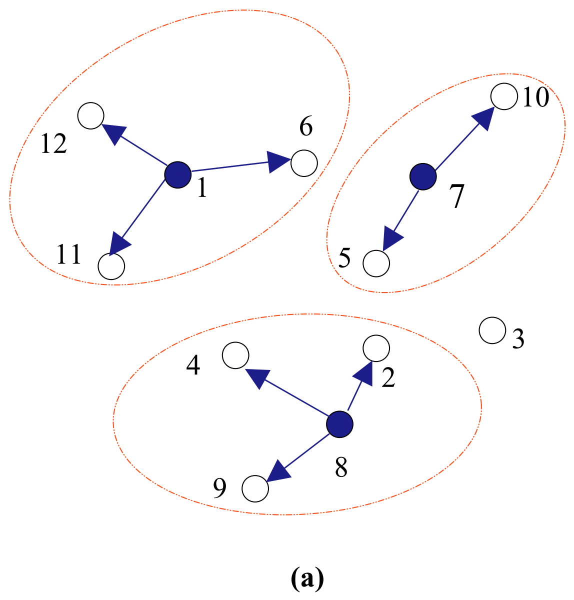
Fig. 2 Discovered topology for a 12-node system

scan procedures in order to set up potential master-slave links with its phase-0 neighbors. For example, nodes 1, 7, and 8 decide randomly to enter page state and start paging their neighbors, while all other nodes decide to do a page scan. Whenever a node is paging its neighbors, it records the total times it has paged each specific neighbor. If a neighbor can't be reached after a certain number of page procedures, the neighbor will be treated as "ceased".

When the phase-0 node-1 successfully pages the phase-0 node-12 and thus forms a master-slave link between them, both of the nodes enter phase-1. Then node-1 continues paging its other neighbors until it obtains a bounded number ($\leq N_{\max}$) of phase-0 neighbors as its slaves: {12, 11, 6} (Here $N_{\max}$ is set to 3 for the example). A piconet formed in this way like that of node-1 is called an *original piconet*, to be distinguished from the piconets formed later during phase-2 or phase-3. The master node is called an *original master*. Similarly node-7 and node-8 also page their neighbors and set up their own original piconets. Node-7 only gets two slave nodes, 5 and 10, because node-6 has decided to join node-1's piconet.

As shown in Fig. 3(a), the original piconets are depicted as dashed-line ovals, the master node is marked as a solid circle, the slave node is marked as a hollow circle, and a master-slave link is represented as an arrowed link pointing toward the slave node. Note that the links are two-way links and the arrow is used only to indicate the master and the slave nodes.

Continuing the example, node 3 first waits for pages from its neighbors. When it does not receive a page query after a random period of time, it then decides to enter the page state, and finally finds that all of its neighbors have already joined some original piconets. The isolated node-3 then decides to proceed to phase-2 and begins to page all of its neighbors {2, 5, 10}. Node-3 tries to interconnect all of its neighboring original piconets by acquiring one node as a slave from



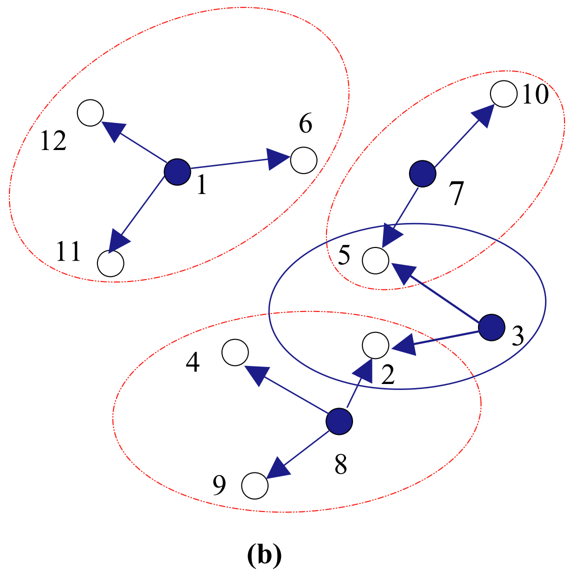


Fig. 3 (a) Original piconets in *BlueNet* (b) Phase-2 piconet in *BlueNet*

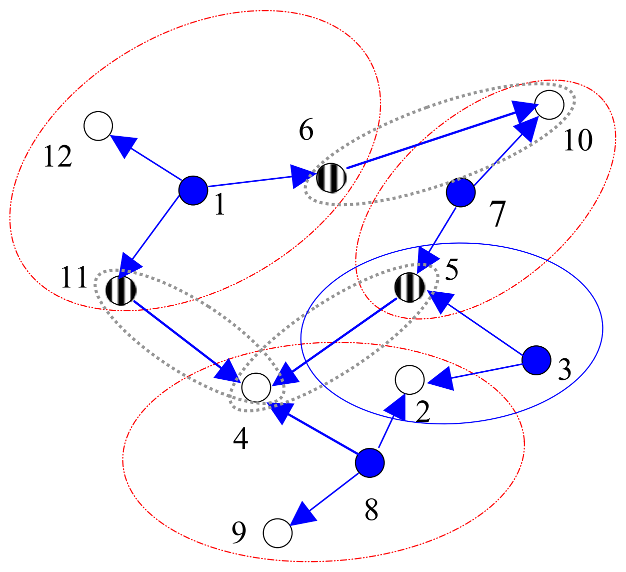


Fig. 4 Final *BlueNet*

each of such piconets and sets up its own piconet (with at most $N_{\max}$ slaves). It chooses not to page node-8, because it learns that node-8 is the original master of node-2. Finally node-3 chooses nodes 2 and 5 as its slaves, because node-10 belongs to the same original piconet as node-5. Then node-3 enters the "finish" state. The result is shown in Fig. 3(b), where the phase-2 piconet is depicted as a solid-line oval.

After learning the status of all their neighbors, the original masters 1, 7, and 8 separately lead their own original piconet members to enter phase-3 and instruct their slave nodes to set up cross-piconet links. The master node selects, by turns or by some specific rules, one of its slaves to do page while instructing all the other slaves to do page scan. When cross-piconet link are set up, the original master node keeps a record of which piconet it has been connected to. This avoids setting up more than one connection with the same neighboring piconet. The whole piconet finally enters "finish" when it has set up all the possible interconnections with adjacent piconets (with the restriction that each slave node has at most $N_{\max}$ inter-piconet links).

Slave node-11, instructed by its master node-1, then chooses to page slave node-4 and sets up a master-slave link with node-4. Then slave node-6 pages its neighbors $\{4, 10, 7, 5\}$, but node-7 and node-4 are finally eliminated from the set, since node-7 is an original master node of node-5 and node-10 and node-4 belongs to piconet of master node-8, which has been connected to the piconet of master node-1. Node-6 then sets up a master-slave link with node-10. Similarly, for node-7's piconet, the slave node-5 sets up an inter-piconet master-slave link with node-4. By now all neighboring piconets are connected, as shown in Fig. 4. The new piconets formed during phase-3 are depicted as dotted-line ovals and the bridge nodes that assume both master and slave roles are depicted as circles filled with stripes.

2.1 Definition of data records

When a node succeeds in paging another node, a temporary master-slave link is first set up between them for exchanging information, and updating their own data records. Then a decision is made by the paging node about whether or not to keep this link finally. The data records, which may be needed by each node during the formation of scatternet, are defined as below. Note that the data records are defined per node.

- A_0, is the *initial neighbor list* and includes all the node's neighbors obtained from the *discovered topology* graph. For each neighbor, its record is expanded to include the neighbor's original master ID and the total number of times being paged by this node as well. Recording paging times in A_0allows setting a timeout in case some node leaves or breaks down and becomes no longer reachable. That is, if a neighbor cannot be reached after a long of time, it will be treated as "ceased" and removed from A_0.
- A_{pi}, is the *phase-i neighbor list* and includes all the node's pageable neighbors when this node is in the phase-*i* mode. A_{p0} and A_{p1} have the same definition, containing all those neighbors whose original master ID is still unknown by a phase-0 or phase-1 node. A_{p2} contains all the neighbors that belong to a different original piconet other than those already connected by this phase-2 node. A_{p3} contains all the neighbors that belong to a different original piconet, as slave nodes, other than those already connected by this phase-3 node's original piconet.
- M_{pi}, is the phase-*i master record* and includes the IDs of all or some of its slaves (depending on the node's current phase mode), only if the node has a master role. Specially, if the node is an original master, M_{p1} contains all the slaves' ID in its original piconet, and M_{P3} contains only the ID's of those slaves that have non-empty A_{p3} and less than $N_{\max}$ slaves.
- S, is the *slave record* which records the node ID's of all its masters, if this node is a slave to some other node(s).
- K, is the *cross-piconet connection list* which contains the master node ID's whose original piconet is connected to this one, only used by the original master nodes.
- T, is the *node state* which records the node's original master ID and its phase mode. The node state is the exchanged information when two nodes perform handshake and set up a temporary master-slave link, so that they can update their records and decide whether to keep this link or not.

During the formation of scatternets, participating Bluetooth nodes make decisions, select paged nodes, exchange and update information based on their local information. A page node selects its paged nodes randomly or according to some specific rules from its corresponding pageable list of neighbors.

2.2 Phase operations and transitions

Phase operations and transitions for phase-0 through phase-3 are described in this subsection. The flow charts of each phase are shown in the Appendix.

2.2.1 Phase-0

Initially, if the *initial neighbor list* A_{P0} is non-empty, a phase-0 node chooses to enter the page state according to a specified page-probability p_0. The selection of the probability p_0 helps to control the number of piconets in the resulting scatternet. A paging node then selects a random list of neighbors to page from A_{P0}, while a scanning node continues to listen and waits for other nodes to page it.

After a period of page or page scan operations, if a node successfully invites another phase-0 node to be its slave or if it joins an original piconet of some other node, the node then enters phase-1. The initialization includes forming the master or slave record M or S and copying A_{P0} to the *phase-1 neighbor list* A_{P1}.Otherwise, the node updates its pageable neighbor list A_{P0} according to the information that it has learned from the previous page or scan period. That is, if the node learns that one of its neighbors has joined some piconet, it will record this in A_0 and remove this neighbor from A_{P0}. Finally if the node is still in phase-0 when A_{P0} becomes empty, the node proceeds to phase-2. During the initialization of phase-2, the content of the *initial neighbor list* A_0 is copied into the *phase-2 neighbor list* A_{P2}. The phase operations and transitions of phase-0 are shown in Fig. A.1.

It is worth noting here that the following two statements are different: (a) the original master ID of a neighbor j is unknown to *node-k*; and (b) a neighbor j has not joined an original piconet yet. For example, in Fig. 3(a), node-6 has a neighboring node-5, which already joined the original piconet of node-7. But before paging node-5, node-6 still has a record of node-5 as (5,0,0) and it knows nothing about node-5's state yet. In the phase operations, the neighboring nodes as per definition (a) above are treated as pageable.

2.2.2 Phase-1

Once a paging node becomes a master to some slave node, both of them will enter and stay in phase-1 until the master issues a change. The phase-1 master node then takes the control and determines the action for all the members in its piconet. The phase operations and transitions of phase-1 can be referred to Fig. A.2.

First the original master will check its *initial neighbor list* A_0 to define a special subset of its neighbors, U_P, which the master has never paged yet. Taking node-1 for example, after entering phase-1 with its first slave node-12, it has its A_0 as {(4,8,3), (6,0,0), (12,1,1), (11,0,0)}, and its initial *phase-1 neighbor list* A_{P1} as {6,11}. Since node-1 never paged node-6 and node-11, it forms its U_P as {6,11}. If U_P is non-empty, and the master has less than $N_{\max}$ slaves, the master will keep performing the page operation and will page those neighbors randomly selected from U_P. Otherwise the master randomly alternates between page and page scan trying to acquire more slaves, exchange state information, and update its pageable neighbor list A_{P1} by removing those who already get associated with some original piconet. If this master already obtained $N_{\max}$ slaves, it will also remove from its A_{P1} those neighbors in phase-0 after paging them and informing them about its becoming an original master. Finally when A_{P1} becomes empty, which means that the phase-1 master node has contacted all of its neighbors' states, the whole piconet enters phase-3. The master instructs all of its slaves to get ready to set up inter-piconet links; i.e., to initialize its *phase-3 master record* M_{P3} for the master node and to initialize the *phase-3 neighbor list* A_{P3} for each slave. For example, initially the original master node-1 has its *phase-1 Master record* M_{P1} as $\{12\}$ and its A_{P1} as $\{6{,}11\}$, after successfully paging node-6 and node-11, and making them join its piconet as slaves, node-1 expands its master record as $M_{P1} = \{12, 6, 11\}$, and removes node-6 and node-11 from its $A_{P1,}$ so that its A_{P1} becomes empty. Then node-1 will instruct its own piconet to proceed to phase-3.

2.2.3 Phase-2

As mentioned above, a node proceeds to phase-2 when it finds that all of its neighbors have joined some other original piconets; i.e., its *initial neighbor list* A_{P0} becomes empty. In phase-2, the isolated node tries to get interconnected with its neighboring piconets. The phase-2 operations and transitions are shown in Fig. A.3.

Initially, the *phase-2 neighbor list* A_{P2} contains all of the node's neighbors and from which the phase-2 node selects a list of paged nodes, randomly or according to some specific rules (say, slave nodes first). If a selected neighbor belongs to an original piconet it is already connected with (recorded in K) it will be removed from A_{P2} and from the paged list as well. Whenever the phase-2 master node gets exchanged information from a neighbor node-k, it will update its A_{P2} by removing node-k along with some other neighbors, which belongs to the same original piconet as node-k. If the phase-2 master has not obtained $N_{\max}$ slaves, it will acquire the newly paged node as its new slave node. Otherwise it just tears down the temporary master-slave link. Finally when there is no pageable neighbor left in A_{P2}, the phase-2 node enters the "finish" state.

2.2.4 Phase-3

In phase-3, the original master node instructs its slaves to set up cross-piconet links with neighboring piconets so that the entire network becomes interconnected. The phase operations and transitions of phase-3 can be found in Fig. A.4.

After entering phase-3, initially each original slave node forms its *phase-3 neighbor list* A_{P3} by including those neighboring nodes that belong to a different original piconet as slave nodes. All the original slaves with nonempty A_{P3} will be included in its *phase-3 master record* M_{P3}. The master randomly (or according to some specific rule) selects one slave from M_{P3} and instructs it to perform page, while the other nodes are instructed to perform page scan. If only one slave is left in M_{P3}, this slave will randomly alternate between page and page scan. This is done in order to break the possible deadlocks: when two neighboring piconets, say, piconet of master node-u and piconet of master node-v, each having only one slave node left in their own M_{P3}, say, node-w and node-x, respectively, which are neighbors to each other and keep page each other trying to set up a link but they won't get a response. However, if both of them alternate between page and page scan randomly, node-w and node-x can easily get a handshake and set up a link. Whenever a new cross-piconet link is set up the master will update its cross piconet record K and also update A_{P3} for each slave node in M_{P3} by removing all of those neighbors, which belong to this newly interconnected piconet. If there are no pageable neighbors left in A_{P3} for one slave, the slave will be removed from M_{P3}. When there are no eligible slaves left in the M_{P3}, the whole piconet enters the "finish" state.

2.3 Modifications and enhancements

As mentioned in [1], in a very sparse connectivity environment, there possibly exist disconnections in the resulting *BlueNet* scatternets even if the *discovered topology* is connected. Also the building speed is affected by the transition condition from phase1 to phase-3, because the master has to inform all of its neighbors about its status before the transition. And before the transition the slaves are refrained from setting up cross-piconet links. In order to improve above drawbacks, two modifications are suggested here.

a. A phase-1 master proceeds to phase-3 when it has obtained $N_{\max}$ slaves in its piconet or when its *phase-1 neighbor list* A_{P1} becomes empty.
b. After entering phase-3, an original master, if with less than $N_{\max}$ slaves, will first page its neighbors that have joined some other piconets as slave nodes, and try to set up cross-piconet links through the shared slave nodes. However, the total number of slaves in the master's piconet is still limited to at most $N_{\max}$. After that, it will instruct its slaves to perform the phase-3 operations as defined in 2.2.4.

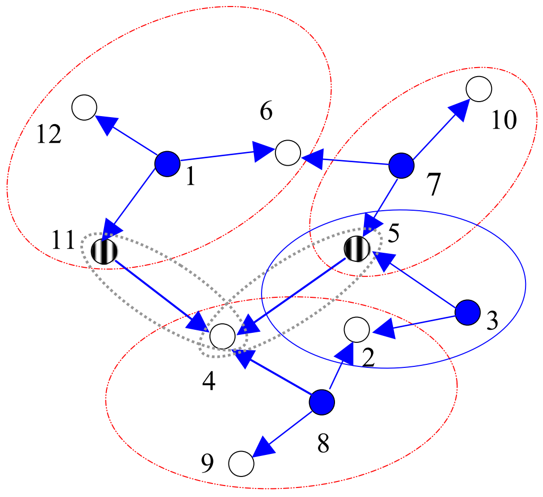


Fig. 5 "New" final *BlueNet*

Obviously, the modification of (a) allows a phase-1 master to enter phase-3 before its *phase-1 neighbor list* A_{P1} becomes empty if it has already obtained $N_{\max}$ slaves. In this way a node may start its phase-3 operations earlier and speed up the building process of a scatternet. For example, in our simulation to set up a scatternet for $30 \sim 110$ uniformly distributed class-3 Bluetooth nodes inside an area of 30 meter by 30 meter, the average protocol duration time (excluding discovery phase) can be shortened by $2 \sim 5\%$.

With the modification of (b), the *BlueNet* will proceed a little differently after entering phase-3. Take the system in Fig. 3 as an example. When the original master 7 enters phase-3, which has only 2 slaves, it will first check if it can get one more slave node from its neighbors and finally it may successfully invite node-6 to join its piconet as a shared slave. Thus the finally *BlueNet* for this network will be like that shown in Fig. 5. Beware that the shared slave node-6 cannot be treated as one member of node-7's original piconet though it joins node-7's piconet, because it already belongs to node-1's original piconet. With this modification, the connectivity of resulting *BlueNet* scatternets can be largely improved. For example, provided the *discovered topology* is connected, no disconnections occur in the resulting scatternets from 300 times of *BlueNet* formation simulation even for a very sparse connectivity network (as 30 nodes uniformly and randomly distributed in an area of 30 meters by 30 meters).

3 Performance analysis and comparison

In this section, the performance of resulting scatternets from *BlueNet* protocol is analyzed and compared with scatternets produced by other two protocols, *BlueTrees* and *LSBS*. First performance metrics are defined and then performance is evaluated through simulations.

3.1 Performance metrics

In order to evaluate the efficiency of the resulting scatternets, we adopt the following metrics.

a. Average Connectivity Percentage versus Breakdown Probability of a Node

With a specific breakdown probability P_{bk} of a node in the system, this metric indicates how many node-pairs, among all the possible pairs, remain connected (i.e., having a path between them), after N_{bk} nodes are lost. The connectivity percentage of the remaining network can be expressed as:

$$\eta = \frac{2N_{\text{conn}}}{(b_n - N_{bk})(b_n - N_{bk} - 1)} \tag{1}$$

where N_{conn} is the total number of connected node-pairs in the remaining network, N_{bk} is the number of broken-down node. In our simulation and comparison, for each scatternet sample, a number of broken-down nodes are randomly selected according to the breakdown probability P_{bk}. Obviously when $P_{bk} = 0$, $\eta = 1$ and when $P_{bk} = 1$, $\eta = 0$. When $N_{bk} \geq b_n - 1$; i.e., only one or no node is left in the network, we set $\eta = 0$. The connectivity percentage is calculated according to Eq. (1) and averaged among all the scatternet samples. The average percentage of connectivity in remaining network versus the breakdown probability represents the reliability of the scatternets.

b. Average Shortest Path Length –

$$\bar{d} = \tfrac{2}{b_n(b_n-1)} \sum_{(i \neq j)} d_{ij}, \tag{2}$$

d_{ij} is the short path length (hop count) between *node-i* and *node-j* in the resulting scatternet.

This index shows the routing efficiency of the resulting scatternet. It provides an estimate of the average routing delay in the resulting scatternet.

c. Piconet Density –

$$\xi = n_{mst} \big/ b_n, \tag{3}$$

where n_{mst} is the number of piconets formed and $b_n$is the number of Bluetooth nodes.

This index reflects the interference level among the piconets in a scatternet. Since the FHSS radios of each piconet operates over the shared 79 or 23 Bluetooth ISM channels, the more piconets that exist in the same neighborhood, the heavier the interference among them. Therefore, a too high piconet density should be avoided.

d. Link Coverage Ratio –

$$\rho = N_L/(b_n - 1), \tag{4}$$

which is defined as the ratio between the number of links N_L in a scatternet and the smallest number of links needed to form a connected network $(b_n - 1)$.

Obviously, a connected scatternet always has $\rho \geq 1$. This index represents the usage of potential links in a scatternet. In order to form an efficient communication network, the resulting scatternet should have a decent level of connectivity. On one hand, too large a link coverage ratio means wasted network resources since each active link costs node bandwidth to maintain it. On the other hand, if link coverage ratio is too small, it may cause routing inefficiency and congestion for multi-pair communications.

e. Maximum Traffic Flow – MTF_m is defined as the estimated average maximum traffic flow that can be carried by the resulting scatternets for all m-pairs of communication nodes. This index reflects the information carrying capacity of the resulting scatternets. In this paper MTF is evaluated through a heuristic greedy algorithm.

Define a network as $W = (N, A, c)$, where N is the set of nodes, A is the set of all links, and $c : A \to \Re^+$ or $c : A \cup N \to \Re^+$ is the link/node capacity function. Designating a particular node s as the source and another particular node t as the sink, we wish to know the maximum amount of information that can be transmitted per unit time from s through the network to t, without violating the link and node capacity limits, i.e.,

$$f^* = \max_{W=(N,A,c)} \{f : s \to t\} \tag{5}$$

where $f : s \to t$ denotes a traffic flow starting from s, going through the network W, and ending at t. This is called the *maximum flow* problem, which can be solved very conveniently and efficiently by the Ford-Fulkerson Algorithm [4, 5].

Now, assume that there exist m source-sink pairs in the network, $s_i \to t_i, i = 1, 2, \ldots, m$. Define the set $S = \{s_i | i = 1, 2, \ldots, m\}$ and the set $T = \{t_i | i = 1, 2, \ldots, m\}$. For simplicity, it is assumed that $S \cap T = \phi$ and $s_i \neq s_j, t_i \neq t_j$, for any $i \neq j$. The information flowing between a source-sink pair $s_i \to t_i$ is called the flow of a certain commodity, and the network has m different commodities flowing simultaneously. The problem of maximizing the sum of the multiple-commodity flows is very time-consuming and difficult to solve (see [4]). But a simple greedy algorithm can be used to approximate the optimal solution. The algorithm is discussed in the sequel.

Given the network $W = (N, A, c)$, and a list of m commodities $V = \{s_i \to t_i | i = 1, 2, \ldots, m\}$:

a. Using Ford-Fulkerson algorithm, solve the *maximum flow* problem for each commodity in V separately, as if only one commodity exists at one time. That is,

$$(f_i, c_i) = \max_{W=(N,A,c)} \{f : s_i \to t_i\} \tag{6}$$

where f_i is the max flow that can be achieved for ith commodity and c_i is the corresponding consumption of the used link and/or node capacities. Keep a record for each solution (f_i, c_i).

b. Select the commodity with the largest maximum flow $s_{i^*} \to t_{i^*}$, where $i^* = \max_i : \{f_i | i = 1, 2, \ldots, |V|\}$; then remove this commodity from the commodity list; and reduce the corresponding link capacities and node capacities $c_{i^*}$from c by the consumed amount. That is, $V \leftarrow V \backslash (s_{i^*} \to t_{i^*}), c \leftarrow c - c_{i^*}$.

c. Repeat step (a) and (b) until V becomes empty.

The Maximum Traffic Flow (MTF) is then calculated as the sum of the largest maximum flows obtained from each cycle, i.e.,

$$MTF = \sum_{k=1}^{m} f_{i_k^*} \tag{7}$$

where $f_{i_k^*}$ denotes the maximum flow obtained from the kth cycle.

In order to evaluate the MTF performance, we need to know the capacity function of a scatternet link. Since the real capacity-limitations in a scatternet are its node capacities, we simply set all the link capacities equal to $C_0 = 1000$ Kbps. In order to account for scheduling overhead, we follow the method recommended in [8]. That is, the intra piconet overhead is assumed to be $\Delta B_1 = 10$ Kbps for each slave that a master node holds; and the bridge overhead is $\Delta B_2 = 100$ Kbps for each additional piconet that a bridge node joins. Then the capacity of the i-th Bluetooth node can be written as:

$$C^i = C_0 - n_s^i \cdot \Delta B_1 - I_{\text{bridge}}^i \cdot \left(n_p^i - 1\right) \cdot \Delta B_2, \quad i = 1, 2, \ldots, b_n \tag{8}$$

where n_s^i is the number of its slaves if node-i is a master, otherwise n_s^i is equal to 0; $I_{\text{bridge}}^i = 1$ if node-i is a bridge node and 0 otherwise; and $n_p^i (\geq 1)$ is the total number of piconets that node-i participates in, obviously $n_p^i > 1$ for a bridge node.

In our simulation, the MTF performance for each scatternet is computed in the following way: for a fixed size of communication pairs, say, m, $(1 \leq m \leq \lfloor \frac{b_n}{2} \rfloor)$, k samples of *m-commodity* are selected randomly from all the possible node pairs in the network (i.e., each sample contains totally $2m$ different nodes, forming m source-sink pairs) and MTF is computed for each sample and averaged to represent the MTF performance of the scatternet. In this way the MTF performance is computed for each of the n sample scatternets generated from each protocol and the mean values are used to represent the average information carrying capacity for this set of scatternets. In our simulations, the total number of sample scatternets generated for each protocol, n, is set to be 300, and the total number of sample *m-communication pairs*, k, is set to be 4, due to the limitation of computation capacity.

3.2 Simulation and comparison

In this section, we compare the performance of the resulting scatternets from *BlueNet* and two other protocols, *BlueTree*s and *LSBS*. First a more detailed description of the protocols is provided. Then the simulation results are showed and the performance compared. In order for *BlueTrees* and *LSBS* to work correctly in the simulation, it is assumed that all the nodes are located in a planar space and that the radio propagation environment can guarantee that the two nodes within each other's radio range can hear each other perfectly.

3.2.1 BlueTrees and LSBS

Both the *BlueTree*s and the *LSBS* protocols require for their *discovered topology* that if a node u and a node v have discovered each other as immediate neighbors and have a common neighbor z (in the visibility graph), then either both discover z or neither of them does. Otherwise, the degree reduction process of the protocol may result in a disconnected topology. Therefore, implementation of the protocol in [1] adds a replenish phase to refine the *discovered topology* to satisfy the above property. During the replenish phase, the discovered neighbors exchanges their collected information about their neighbors, which is done by a second round of device discovery to replenish those un-discovered neighbors. In *LSBS* the replenish phase is more effective because its information exchange includes position information. Thus, a node can determine whether a new node is in its radio range by calculating the physical distance between them. Therefore the number of nodes to be paged can be largely reduced compared with that in *BlueTrees*.

Starting with a designated node, called blueroot, the *BlueTrees* protocol builds a tree-like scatternet [16]. First the blueroot node pages all of its one-hop neighbors and invites them to join its own piconet as slave nodes. These slave nodes, in turn, start paging their own neighbors and build their own piconets. This process continues until the tree is completed. In order to limit the number of slaves per piconet, it is observed that if a node contained in a unit disk graph has more than five neighbors, then there exist at least two of them, which are in each other transmission range. This observation is used to reconfigure the tree so that each master node has no more than seven slaves. If a master node v has more than seven slaves, it selects two of them, u and w, which are necessarily in each other's transmission range and instructs one of the two, say u, to be the master of the other node, w. The node w 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. Though the original paper does not discuss how to select or position the blueroot, the resulting scatternet depends on a selected node to start the formation procedure. When the *discovered topology* is a disconnected one, *BlueTrees* may fail because the building process may be interrupted at the disconnected area. In our simulation, the blueroot is either selected as the node that is closest to the center of coverage, or just selected randomly from among all nodes. The former is denoted as *BlueTree*(C) and the latter as *BlueTree*(R).

As mentioned before, the *LSBS* protocol described in [11, 13] is a combination of *Yao* construction and *BlueStars* protocol. First, the *Yao* construction is applied to the *discovered topology* to reduce the node degree in the network to no more than $k \leq 7$. At some given node u, the surrounding plane is equally separated into k cones by k rays originating at u. In each cone, choose only the closest node v, if there are any, and keep the edge between u and v. The other edges will then be deleted from the graph. Though some links are deleted, the *Yao* construction guarantees the connectivity of the resulting topology.

Based on the *Yao* topology, *BlueStars* then computes the weight for each node, as indication of its suitability to become a master. For example, the weight could be the topology degree of the node, the Bluetooth ID address, any other preassigned parameter, or a combination of these parameters. In this simulation, for the purpose of simplicity, the scatternet samples of *LSBS* use the node's ID number as its weight. The nodes with the largest weights in their neighborhood, called *init* devices, initiate the *BlueStars* formation. An init device assumes the role as a master and obtains all of its neighbors as its slave nodes. Then some other nodes, after learning that all of their neighbors with bigger weights (called "bigger" neighbors) have become slave nodes, decide to become masters and acquire their smaller neighbors as slaves. After this phase, the whole network is covered by disjoint piconets, each with less than k slaves. Two piconets are called neighbors if there exist a pair of member nodes, one from each piconet, which are neighbors in the *Yao* topology. The pair of neighboring member nodes are then selected as *gateway* nodes and the corresponding master nodes are called neighboring masters (denoted as *mNeighbors*). A master node is elected to be an *init* master (denoted as *iMaster*) if it has a larger weight than all of its neighboring masters. And the interconnection of disjoint piconets from previous phase is started by the *iMasters*. An *iMaster* node enters the page mode to page all the gateway slaves that belong to a different piconet, and instructs all of its own gateway slaves to page their gateway neighbors in a neighboring piconet. For a non-*iMaster* node, if there are *mNeighbors* with larger weights, it first enters page scan mode, and instructs all of its gateway slaves to bigger *mNeighbor*s to enter the page scan mode as well, until the links are set up. Then, if there are *mNeighbors* with smaller weights, it enters the page mode and instructs all of its gateway slaves to smaller *mNeighbors* to enter page as well, until the links have been set up. In this way, all the disjoint piconets from the previous phase are interconnected to form a scatternet.

3.2.2 Performance comparison

Simulations and computations were performed to analyze and compare the performance of the resulting scatternets from the *BlueNet*, the *BlueTrees* and the *LSBS* protocols. The number of Bluetooth nodes was chosen as $b_n = 30, 70, 90$, or 110 (with 10 meters of radio range). The nodes are randomly and uniformly distributed in a square 30 meters by 30 meters area. The discovered topologies are obtained from a 10-second device discovery with corresponding replenish phase as recommended in [1]. For each b_n, three hundred sample scatternets are generated by each protocol. The authors of [1] provided sample scatternets for *BlueTrees* with centrally specified blueroot (denoted as *BlueTree*(C)), *BlueTrees* with randomly specified blueroot (denoted as *BlueTree*(R)), and for *LSBS*. As mentioned before, both *BlueTrees* and *LSBS* need a replenish phase to refine their *discovered topology* to guarantee connectivity after degree reduction. With position information provided, the replenish phase of *LSBS* is more efficient than that of *BlueTrees*. Therefore, the final *discovered topology* for *LSBS* usually contains about 5–10% more links than that of *BlueTrees*. Although *BlueNet* does not need a replenish phase to work, in order for a fair comparison over the resulting scatternets, we used for *BlueNet* the same discovered topologies (with replenish phase) of *BlueTrees* and of *LSBS*. The corresponding samples are denoted as *BlueNet* and *BlueNet*' respectively in the plots. Later simulation results show that this mild difference in the *discovered topology* does not cause much difference in the performance of the resulting scatternets of *BlueNet*.

A set of MATLAB programs are developed to simulate the processes such as page(scan), inquiry(scan), hold/sniff, and etc to build up a scatternet from a discovered topology that was given by [1]. Corresponding time parameters are set same as or equivalent to those from [1]. In this implementation, the wireless physical layer are not simulated in details but represented by a uniform random process with trivial probability (0.001) of collision which results in lost of packets. The *BlueNet* samples are generated with the initial probability for nodes to enter the page state, p_0, set to 0.1 and the maximum number of slaves in a piconet, $N_{\max}$, set to 5. The parameters settings are chosen based on the studies in [15] and [14], resulting in scatternets with better average connectivity and throughput performance.

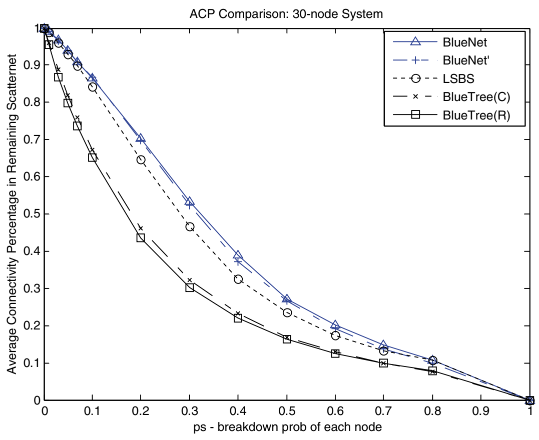


Fig. 6 Average connectivity percentage vs. P_{bk} 30-node system

Figures 6 to 8 present the average connectivity percentage versus node breakdown probability P_{bk} for 30, 90, and 110-node system, respectively. From the figures it can seen that *BlueTrees* has the worst reliability performance due to its tree-like topology. No matter how sparse or dense the node distribution is, the average connectivity percentage of the remaining network drops quickly as P_{bk} increases. Even a small number of nodes lost will cause the remaining network to have very poor connectivity. *LSBS* and *BlueNet* have much better robustness than *BlueTrees* due to their mesh-like topology. But *BlueNet* performs the best in all cases. In a very sparse connectivity environment such as a 30-node system, Fig. 6 shows that the average connectivity percentage of *LSBS* is very close to *BlueNet* because there are only few links available for the *BlueNet* or *LSBS* protocols to use. As the environment gets denser and more links become available, *BlueNet*s tends to utilize more links to form a connected scatternet, which results in much better re-

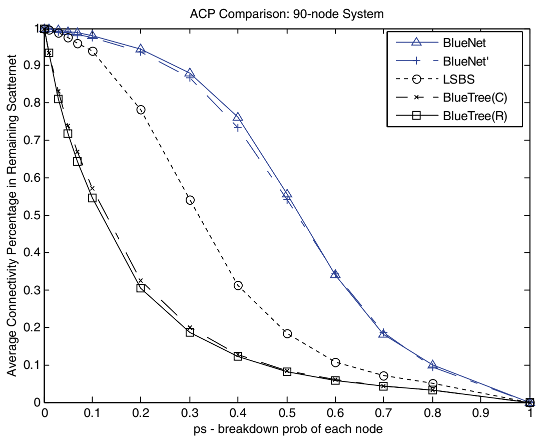


Fig. 7 Average connectivity percentage vs. P_{bk} 90-node system

Table 1 The average connectivity percentage of *BlueNet* and increments over *LSBS* and *BlueTrees* ($P_{bk} = 0.2$)

b_n	*BlueNet*	Increment over *LSBS*	Increment over *BlueTrees*
30	70%	6%	14–16%
50	85%	13%	46–48%
70	91%	18%	57–60%
90	94%	16%	61–64%
110	96%	17%	65–69%

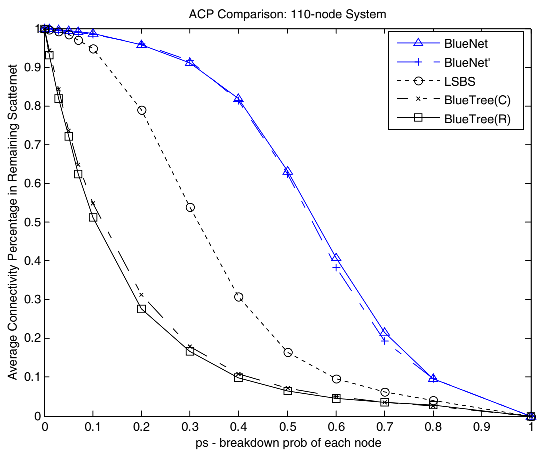


Fig. 8 Average connectivity percentage vs. P_{bk} 110-node system

liability than *LSBS* and *BlueTrees*. Specifically, using a node breakdown probability P_{bk} of 0.2 as an example, the average connectivity percentages of the remaining network in a 30, 50, 70, 90, and 110-node system are shown in Table 1 and compared with *BlueTrees* and *LSBS*.

Figure 9 depicts the average shortest path length the scatternets formed by the three protocols. BlueTree(R) has the longest average shortest path length, ranging from 5.4 to 7.8 as b_n increases. BlueTree(C), with a centrally assigned blueroot, improves this aspect of performance, varying from 5.0 to 7.0. But due to its tree-structure topology, its average shortest path length is still longer than the other two protocols. With a denser meshed scatternet structure, *LSBS* has a shorter average shortest path length, from 4.4 to 6.1. For *BlueNet*s, its scatternet has much more links in a meshed structure, so its average shortest path length is the shortest among all the protocols, from 3.9 to 3.6. As the network gets denser with $b_n$increasing, the average shortest path length even decreases slightly.

Figure 10 shows the average link ratios in the scatternets formed by the three protocols. BlueTree(R) and BlueTree(C) have link ratio equal to exactly 1, because a tree structure always uses only ($b_n - 1$) links to form a connected scatternet. The average link ratios of *LSBS* vary from 1.2 to 1.4, while those of *BlueNet* from 1.2 to 2.2. This shows that in a moderate dense to very dense network, *BlueNet*, without any usage of degree reduction technique, tends to build a scatternets with more links in the system. Though *BlueNet* seems to spend more network resources to maintain a scatternet, it results in shorter average shortest path length. The added reliability and the resulting maximum throughput in resulting scatternets also warrant these additional resources cost, as shown in the previous and following paragraphs. Piconet density vs. bn

Figure 11 shows the average piconet density in the scatternets formed by the three protocols. BlueTree(R) and BlueTree(C) have very close average piconet density, varying from 0.56 to 0.53 as the number of nodes b_n increases. The average piconet density of *LSBS* ranges from 0.51 to 0.56 as b_n increases. For *BlueNet*, its piconet density is affected more by the node density of the network, ranging from 0.45 to 0.71. It has a lower average piconet density than *BlueTrees* and *LSBS* in a sparse connectivity network, while it has a higher average piconet density in a moderate dense or dense network. The higher piconets density in *BlueNet* may create a bit more interference among neighboring piconets. The results from [8] suggest that a

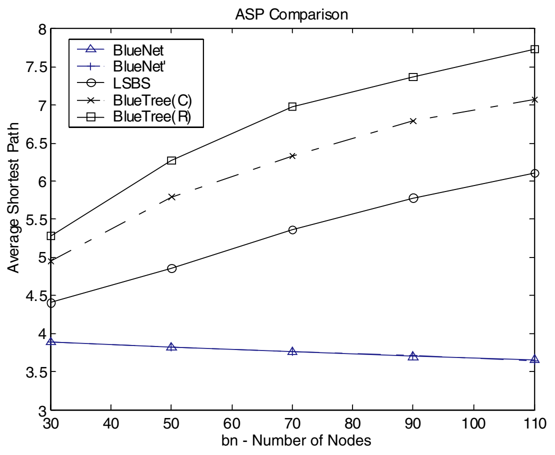


Fig. 9 Average shortest path length vs. bn

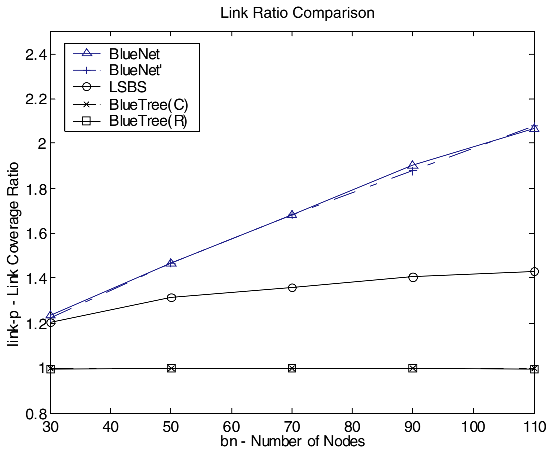
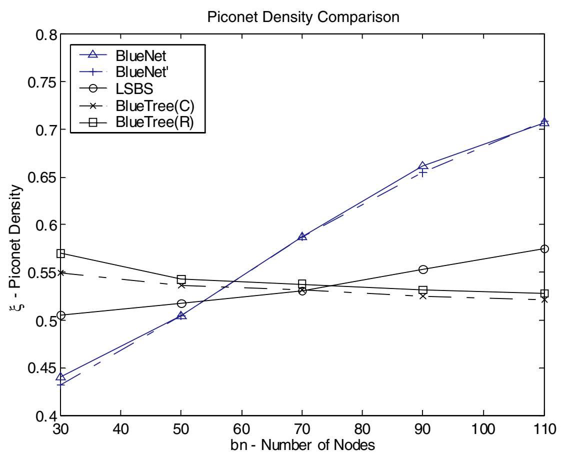


Fig. 10 Link-p vs. bn



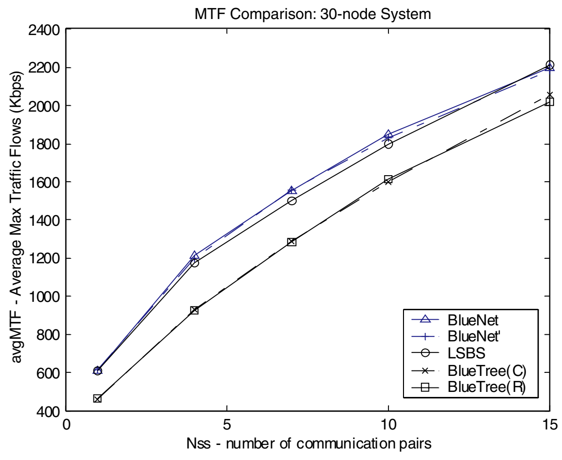


Fig. 12 MTF comparison: 30-node system

density of 0.71 in *BlueNet* should not cause significant deterioration of the network throughput compared with a piconet density of 0.5.

Figures 12 to 15 show the average MTF performance by the three protocols for 30, 50, 90, and 110-node systems. Figure 12 is for 30-node scatternets. In this sparse connectivity environment, BlueTree(R) and BlueTree(C) have almost the same MTF capacity. *BlueNet* has an average MTF capacity which is about 7%–31% higher than BlueTree. The average MTF capacity of *LSBS* is close to that of *BlueNet* except slightly different when N_{ss} ranges from 2 to 14.

Figure 13 depicts the average MTF performance for 50-node scatternets formed by the three protocols. In this moderate sparse environment, BlueTree(R) and BlueTree(C) have very close MTF capacity. *BlueNet* has an average MTF capacity which is about 12%–62% higher than *BlueTrees*. The average MTF capacity of *LSBS* is close to that of *BlueNet* when N_{ss} is small and becomes closer to and reaches that of *BlueTrees* as N_{ss} increases. (This same tendency of the average MTF of *LSBS* can also be seen in 70, 90 and 110-node systems.) The average MTF capacity of *BlueNet* is larger by

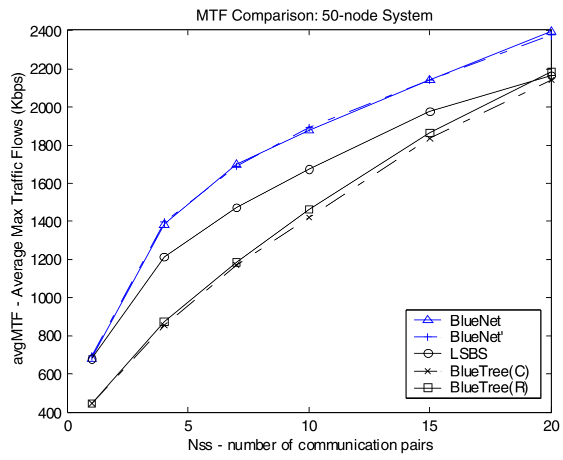


Fig. 13 MTF comparison: 50-node system

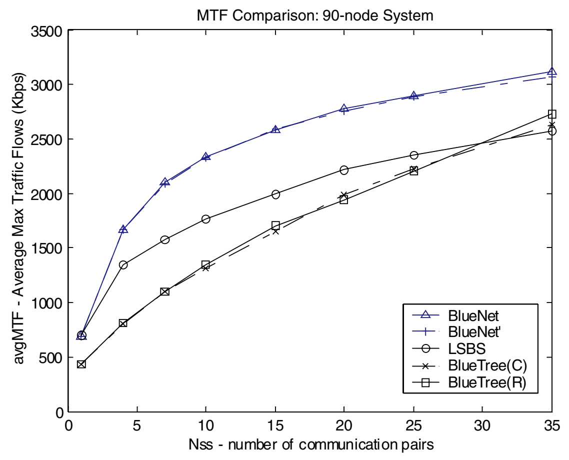


Fig. 14 MTF comparison: 90-node system

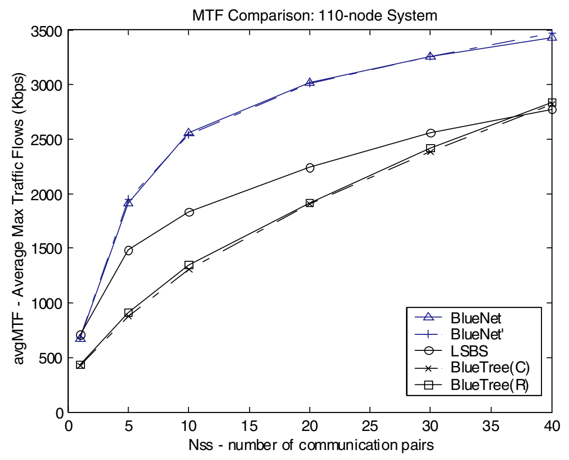


Fig. 15 MTF comparison: 110-node system

Table 2 The improvements of average MTF of *BlueNet* over *LSBS* and *BlueTrees* (for $N_{ss} = 2$–14)

b_n	Improvement of *BlueNet* over *LSBS*	Improvement of *BlueNet* over *BlueTrees*
30	0–3%	7–31%
50	0–15%	12–62%
70	0–23%	14–86%
90	0–33%	18–107%
110	0–39%	21–116%

0%–15% compared to that of *LSBS*. The comparison of MTF capacity of *BlueNet* with *LSBS* and *BlueTrees* are summarized in Table 2.

4 Conclusions

This paper provides detailed description of *BlueNet*, a distributed multi-hop Bluetooth scatternet formation protocol. Some modifications of the *BlueNet* protocol are also proposed on the phase transition from phase-1 to phase-3 and on the phase-3 operations, in order to speed up the process of scatternet formation and to improve the connectivity of the resulting scatternets. Concerning the connectivity, it is observed that even in our 300 simulations in a very sparse connectivity environment, no disconnections occur in the resulting BlueNet scatternets as long as the *discovered topology* is connected.

Metrics are chosen to evaluate the performance of resulting scatternets, such as the reliability, the routing efficiency, the network density and the information-carrying capacity. Among the metrics the reliability metric and the information-carrying capacity, defined by the authors, are most important ones. The former is defined as the average connectivity percentage versus node breakdown probability; and the latter is defined as the average maximum traffic flows that can be carried by the resulting scatternets for all *m*-pairs of communication nodes. A simple and efficient greedy algorithm based on Ford-Fulkerson algorithm is provided to estimate the maximum traffic flow in a resulting scatternet.

Then the performance of the scatternets resulting from the application of the *BlueNet*, *BlueTrees*, and *LSBS* protocols are compared based on the adopted metrics, From the comparison, it can be seen that the resulting *BlueNet* scatternet has much better reliability, a much lower average shortest path length, and higher information-carrying capacity than the other two protocols, although *BlueNet* tends to have a higher piconet density and use more links in the resulting scatternet especially when nodes are densely distributed.

In this paper we searched for a Bluetooth scatternet formation protocols suitable for power grid applications such as substation automation. In these applications, large and inexpensive communication networks, consisting of hundreds or thousands of fixed nodes, are needed to provide reliable communications for field measurements and real-time controls. Considering the low cost of Bluetooth chips and the low-cost installments/maintenance of wireless ad-hoc networks, Bluetooth provides a good alternative to those applications. Among the compared Bluetooth scatternet formation protocols, *BlueNet*, with its better performance on the reliability, the routing efficiency, and the information-carrying capacity, is more suitable for the power grid applications. Though a higher link usage in the *BlueNet* resulting scatternets (20%–120% higher than that of *BlueTrees* for different dense connectivity environments) may cause larger power consumption, the power consumption of Bluetooth chips (1–100 mw per chip) is not considered a problem in power grid applications.

Acknowledgments The authors wish to thank Stefano Basagni and Raffaele Bruno for the insightful comments and for providing the discovered topologies as well as the scatternet samples of *BlueTrees*, and *LSBS* for the performance comparison. This project was supported in part by the US Department of Energy through the Consortium for Electric Reliability Technology Solutions (CERTS) and in part by the National Science Foundation Power System Engineering Research Center (PSERC). The work of Zygmunt J. Haas on this project was supported in part by the U.S. National Science Foundation under the grants ANI-0329905 and CNS-0626751, and by the MURI Program administered by the Air Force Office of Scientific Research (AFOSR) under the contract F49620-02-1-0217.

Appendix

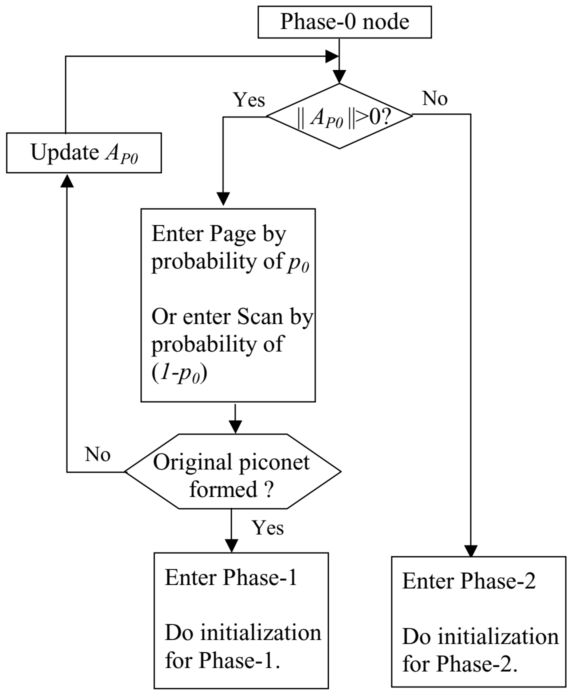
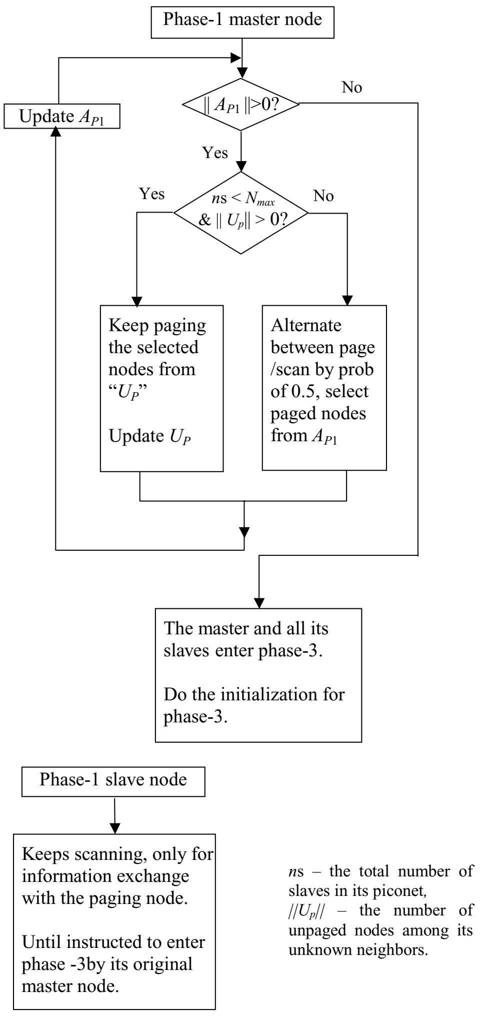
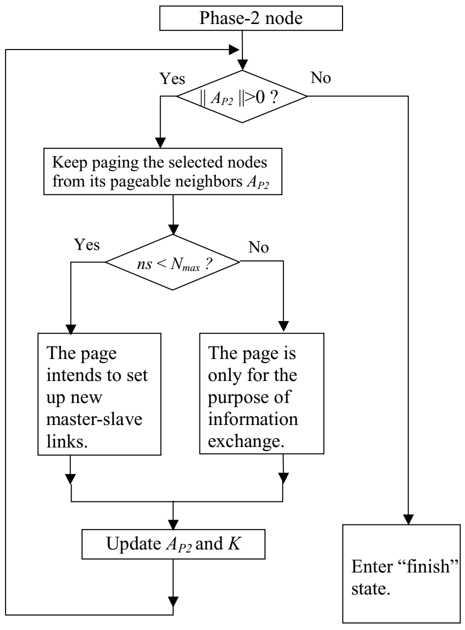

Phase-0 node
Yes
$\|A_{P0}\|>0$?
No
Update A_{P0}
Enter Page by probability of p_0
Or enter Scan by probability of $(1-p_0)$
No
Original piconet formed ?
Yes
Enter Phase-1
Do initialization for Phase-1.
Enter Phase-2
Do initialization for Phase-2.

Fig. A.1 Phase-0 operation and transition



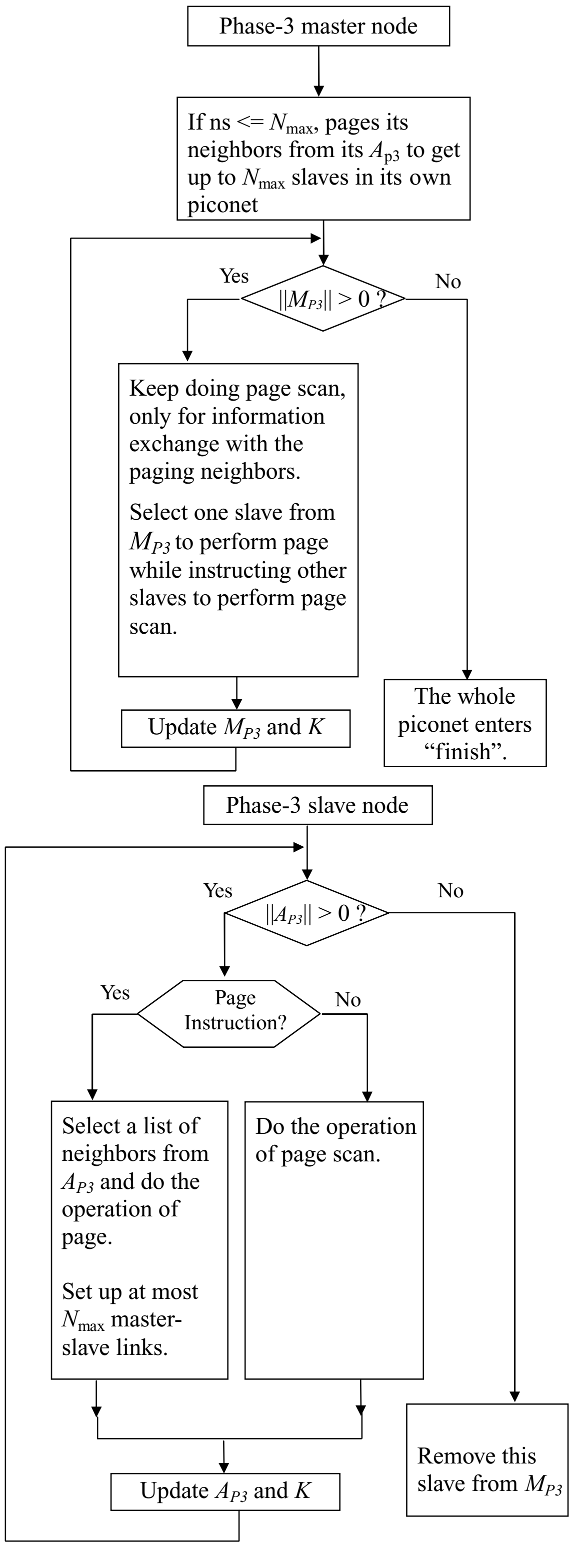
Phase-1 master node
$\| A_{P1} \| > 0$?
No
Yes
Update A_{P1}
Yes
$ns < N_{max}$ & $\| U_p \| > 0$?
No
Keep paging the selected nodes from "U_P" Update U_P
Alternate between page /scan by prob of 0.5, select paged nodes from A_{P1}
The master and all its slaves enter phase-3. Do the initialization for phase-3.
Phase-1 slave node
Keeps scanning, only for information exchange with the paging node. Until instructed to enter phase -3by its original master node.
ns – the total number of slaves in its piconet, $\|U_p\|$ – the number of unpaged nodes among its unknown neighbors.

Fig. A.2 Phase operations and transitions for phase-1 piconets



Phase-2 node
Yes
$\| A_{P2} \| > 0$?
No
Keep paging the selected nodes from its pageable neighbors A_{P2}
Yes
$ns < N_{max}$?
No
The page intends to set up new master-slave links.
The page is only for the purpose of information exchange.
Update A_{P2} and K
Enter "finish" state.

Fig. A.3 Page/Scan logic for phase-2 nodes



Phase-3 master node
If ns <= N_{max}, pages its neighbors from its A_{p3} to get up to N_{max} slaves in its own piconet
Yes
$\|M_{P3}\| > 0$?
No
Keep doing page scan, only for information exchange with the paging neighbors. Select one slave from M_{P3} to perform page while instructing other slaves to perform page scan.
Update M_{P3} and K
The whole piconet enters "finish".
Phase-3 slave node
Yes
$\|A_{P3}\| > 0$?
No
Yes
Page Instruction?
No
Select a list of neighbors from A_{P3} and do the operation of page. Set up at most N_{max} master-slave links.
Do the operation of page scan.
Update A_{P3} and K
Remove this slave from M_{P3}

Fig. A.4 Phase-3 operation and transition

References

1. S. Basagni, R. Bruno, G. Mambrini and C. Petrioli, Comparative performance evaluation of scatternet formation protocols for networks of bluetooth devices, Wirelss Networks 10 (2004) 197–213.
2. Bluetooth Special Interest Group, Specification of the Bluetooth System, version 1.0B, http://www.bluetooth.com/.
3. C.C. Foo and K.C. Chua, Blue-rings – Bluetooth scatternets with ring structures, in: *IASTED International Conference on Wireless and Optical Communication* (*WOC 2002*), Banff, Canada (July 2002).
4. T.C. Hu, *Integer programming and Network Flows* (Addison-Wesley Publishing Company, 1969).
5. R.H. Jones and N.C. Steele, Mathematics in communication theory, Ellis Horwood Limited, Market Cross Houre, Cooper Street, Chichester, West Sussex, PO 19 1EB, England (1989).
6. C. Law and K.Y. Siu, A bluetooth scatternet formation algorithm, in: *Proceedings of the IEEE Symposium on Ad Hoc Wireless Networks 2001*, San Antonio, Texas, USA (November 2001).
7. X. Li and I. Stojmenovic, Partical delaunay trianulation and degree-limited localized bluetooth scatternet formation, in: *Proceedings of Ad-hoc Networks and Wireless* (*ADHOC-NOW*), Fields Institute, Toronto, Canada (September 2002).
8. G. Miklos, A.R. Racz, Z. Turanyi, A. Valko and P. Johansson, Performance aspects of bluetooth scatternet formation, *MobiHoc 2000*, Boston (Aug. 2000) pp147–148.
9. B.A. Miller and C. Bisdikian, *Bluetooth Revealed* (Prentice Hall PTR, Upper Saddle River, NJ 07458, 2001).
10. C. Petrioli and S. Basagni, Degree-constrained multi-hop scatternet formation for bluetooth networks, in: *Proceeding sof the IEEE Globecom 2002*, Taipei, Taiwan, ROC (November 2002), vol. 1, pp. 222–226.
11. C. Petrioli, S. Basagni and I. Chlamtac, Configuring bluestars: Multi-hop scatternet formation for bluetooth networks, IEEE Transactions on Computers, Special Issue on Wireless Internet 52(6) (2003) 779–790.
12. T. Salonidis, P. Ghagwat, L. Tassiulas and R. LaMaire, Distributed topology construction of bluetooth personal area networks, in: *Proceedings of the IEEE Infocom 2001*, Anchorage, AK (April 2001), pp. 1577–1586.
13. G. Tan, A. Miu, J. Guttag and H. Balakrishnan, Forming scatternets from bluetooth personal area networks, MIT Technical Report, MIT-LCS-TR-826 (Oct 2001).
14. Z. Wang, Z.J. Haas and R.J. Thomas, BlueNet II – A detailed realization of the algorithm and performance analysis, in: *Hawaii International Conference on System Science* (*HICSS-36*), Big Island, Hawaii (January 2003).
15. Z. Wang, R.J. Thomas and Z.J. Haas, BlueNet – a new scatternet formation scheme, in: *Hawaii International Conference on System Science* (*HICSS-35*), Big Island, Hawaii (January 2002).
16. G.V. Zaruba, S. Basagni and I. Chlamtac, Bluetrees – scatternet foramtion to enable bluetooth-based ad hoc networks, ICC 2001, vol. 1 (June 2001), pp. 273–277.
17. G.V. Zaruba and P. Johansson, Guest editorial: Advances in research of wireless personal area networking and bluetooth enabled networks, Mobile Networks and Applications 9 (2004) 7–8.

Zhifang Wang (M'02) received her B.S. and M.S. degree in 1995 and 1998 respectively from EE Tsinghus University Beijing, China. After obtaining her Ph.D. from ECE Cornell University in 2005, she has been working as a postdoc in ECE Cornell under the supervision of Prof Robert J Thomas. Her research interests include system analysis, real-time excitation control and voltage control of electric power grids, wireless communications based on Bluetooth, and the reliable and secure communication architecture for wide-area monitoring and control of large-scale systems.

Robert J. Thomas (M'73, SM'83, F'93) currently holds the position of Professor of Electrical and Computer Engineering at Cornell University. He has been a member of the IEEE-USA Energy Policy Committee since 1991 and was the committee's Chair from 1997–1998. He has served as the IEEE-USA Vice President for Technology Policy. His technical background is broadly in the areas of systems analysis and control of large-scale electric power systems. He has published in the areas of transient control and voltage collapse problems as well as technical, economic and institutional impacts of restructuring.

Zygmunt J. Haas received his B.Sc. in EE in 1979 and M.Sc. in EE in 1985. In 1988, after earning his Ph.D. from Stanford University, he joined the AT&T Bell Laboratories in the Network Research Department. There he pursued research on wireless communications, mobility management, fast protocols, optical networks, and optical switching. In August 1995, he joined the faculty of the School of Electrical and Computer Engineering at Cornell University, where he is now a Professor. His *Wireless Network Laboratory* (*wnl.ece.cornell.edu*) is an internationally recognized group with extensive research contributions in the area of Ad Hoc and Sensor Networks. Dr. Haas is an author of numerous technical conference and journal papers and holds fifteen patents in the areas of high-speed networking, wireless networks, and optical switching. He has organized several workshops, delivered numerous tutorials at major IEEE and ACM conferences, and serves as editor of several journals and magazines, including the IEEE Transactions on Networking, the IEEE Transactions on Wireless Communications, the IEEE Communications Magazine, and the Springer Wireless Networks journal. He has been a guest editor of IEEE JSAC issues on "Gigabit Networks," "Mobile Computing Networks," and "Ad-Hoc Networks." Dr. Haas served as a Chair of the IEEE Technical Committee on Personal Communications and is currently serving as the Chair of the Steering Committee of the IEEE Pervasive Computing magazine. His interests include: mobile and wireless communication and networks, performance evaluation of large and complex systems, and biologically inspired networks.