Available online at www.sciencedirect.com
SCIENCE DIRECT®
Ad Hoc Networks 3 (2005) 777–794
Ad Hoc Networks
www.elsevier.com/locate/adhoc

Bluetooth scatternets: criteria, models and classification

K.E. Persson, D. Manivannan *, M. Singhal

Laboratory for Advanced Networking, Department of Computer Science, University of Kentucky, Lexington, KY 40506, USA

Received 1 November 2003; accepted 1 March 2004
Available online 28 April 2004

Abstract

Bluetooth ad hoc networks are constrained by a master/slave configuration, in which one device is the master and controls the communication with the slave devices. The master and up to seven active slave devices can form a small Bluetooth network called a piconet. In order to build larger network topologies, called scatternets, the piconets must be interconnected. Scatternets are formed by allowing certain piconet members to participate in several piconets by periodically switching between them. Due to the fact that there is no scatternet formation procedure in the Bluetooth specification, numerous different approaches have been proposed. We discuss criteria for different types of scatternets and establish general models of scatternet topologies. Then we review the state-of-the-art approaches with respect to Bluetooth scatternet formation and compare and contrast them.

Keywords: Bluetooth; Scatternet formation; Piconet; Device discovery; Ad hoc; Personal area network

1. Introduction

Bluetooth is a networking technology aimed at low-powered, short range applications. It was initially developed by Ericsson, but is governed as an open specification by the Bluetooth Special Interest Group [1]. In the last few years, Bluetooth technology has been extensively covered in the literature [1–7]. Although some see Bluetooth as a competing technology to 802.11 for small Wireless Local Area Networks (WLANs), the authors among many others contend that it a complementary wireless solution aimed at Wireless Personal Area Networks (WPANs) [2]. Standards for WPANs are currently being developed by the IEEE 802.15 Working Group for Wireless Personal Area Networks [8].[1] A WPAN is generally considered to be a small, short range ad hoc

* Corresponding author. Tel.: +1 859 257 9234; fax: +1 859 323 3740.
E-mail addresses: karl@cs.uky.edu (K.E. Persson), mani@cs.uky.edu (D. Manivannan), singhal@cs.uky.edu (M. Singhal).

[1] The IEEE 802.15.1 task group is developing a Bluetooth standard based on version 1.1 of the Bluetooth specification.

1570-8705/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2004.03.014

network of low power devices that surround a person or an object, e.g. a desktop computer or a vehicle. Based on version 1.1 of the Bluetooth specification, piconet networks are restricted to 8 devices with no direct interconnections between slaves. The formation of scatternets is essential to break this boundary by allowing interconnections between piconets.

Bluetooth operates in the 2.4 GHz Industrial, Scientific, Medical (ISM) frequency band. ISM is unlicensed and available worldwide, which makes it suitable for technologies such as Bluetooth and 802.11b. The main difference between Bluetooth and 802.11b (both operate in the 2.4 GHz band) is that 802.11b uses DS-CDMA while Bluetooth uses FH-CDMA [4]. The Frequency Hopping Spread Spectrum (FHSS) modulation allows Bluetooth piconets to communicate independently, with minimal interference from adjacent piconets and other ISM devices. There are 79 hop carriers defined for Bluetooth, with a 1 MHz spacing, in the ISM band.

A piconet consists of a single master device and up to seven active slave devices. Each piconet master determines a frequency hopping spreading sequence that the slaves must follow in order to stay synchronized to the piconet channel. The Time Division Duplex (TDD) frequency hopping (FH) channel is divided into 625 μs time slots. It is driven by the master, which eliminates contention problems within a piconet. The master transmits during even time slots, while slaves are restricted to sending packets back to the master during the directly following odd time slots. The master addresses each slave either individually by its AM_ADDR, or by a piconet-wide broadcast using AM_ADDR 000. In the Bluetooth specification [1], a round-robin intra-piconet scheduling scheme (IRPS) is defined to determine the communication schedule. Additional IRPS schemes are described in Section 2.5.

A scatternet topology is formed by interconnecting piconets. Since each piconet has a unique frequency hopping sequence, piconet interconnections are done by allowing some nodes to participate in more than one piconet. These, so called *bridge nodes*, divide their time between piconets by switching between FH channels and synchronizing to the piconet's master. In general, a node can only be the master in one piconet but is allowed to participate in other piconets as a slave.

In the Bluetooth specification [1], the functionality and membership properties of piconets are described in detail while scatternet formation is only mentioned briefly. Numerous approaches for scatternet formation have been proposed since the specification was published. The main idea behind scatternet formation is to interconnect adjacent piconets. We can interconnect piconets by scheduling disjoint slots, for each piconet, during which the bridge nodes can communicate with the piconet master.

The rest of the paper is organized as follows. In Section 2, we cover the fundamentals. We briefly overview how Bluetooth piconets are formed in Section 2.1. In Section 2.2 we discuss different formation metrics and constraints for scatternet formation. Thereafter, in Section 2.3 we describe the scatternet models and their differing characteristics. These models are then used to classify the proposed scatternet formation solutions reviewed in Section 3. Devices have to discover each other and form links before a scatternet can be formed, since Bluetooth utilizes FH channels. Link formation is covered in Section 2.4. Section 2.5 describes some proposed solutions for intra-piconet scheduling. Section 2.6 covers inter-piconet scheduling and discusses how efficient solutions improve performance in a scatternet. In Section 3 we review the state-of-the-art approaches with respect to scatternet formation. We begin with single-hop protocols in Section 3.1, in which all devices must be within transmission range of each other to form a scatternet. Thereafter, we cover multi-hop protocols in Section 3.2. These approaches offer more flexibility since all devices are not required to be within radio proximity of each other. Section 3.3 describes optimized solutions that provide theoretical results on how scatternet topologies could be constructed efficiently. Section 4 discusses routing in scatternets and reviews some proposed approaches. Finally, in Section 5 we summarize the paper and present concluding remarks.

2. Bluetooth technology

2.1. Piconets

Bluetooth devices communicate with each other in a master/slave configuration. This means that devices do not communicate as equal peers on a collision prone medium, as in 802.11 [9]. Fig. 1(a) shows an example of a two-node piconet with one master and one slave device. The frequency hopping channel is divided into time slots. Each time slot is 625 μs and contains a baseband transmission. A baseband transmission can last for one, three or five time slots. After each transmission, the piconet devices hop to another one of the 79 frequency hopping carriers. In order to maintain synchronization, Bluetooth uses loosely synchronized clocks, which means that slaves use an offset from the master's clock to stay synchronized. The master's BD_ADDR provides the identity of the piconet and its native clock determines the time slot boundaries.

The communication pattern in a piconet is controlled by the master. The master polls each slave in a round-robin fashion during even numbered time slots. The slaves are only allowed to transmit during odd time slots directly following a poll from the master. Fig. 1(b) illustrates an example of communication on the TDD slotted channel between the master and the slave. The master transmits during even time slots and the slave responds in the following odd time slot. Based on the specification, in larger piconets the master addresses each slave using a round-robin intra-piconet scheduling (IRPS) algorithm. Section 2.5 covers other proposed IRPS algorithms. For more details on the Bluetooth physical and link layers, please refer to [1,4,5].

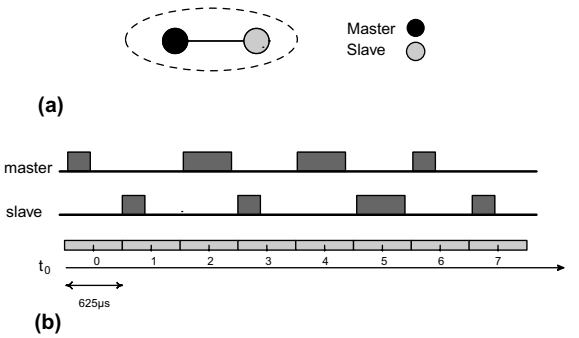

Fig. 1. Piconet. (a) Two-node piconet with a master and a slave device. (b) Packet transmission between master and slave on FH TDD channel.

Multiple piconets can coexist without significant interference from each other, since all slaves maintain synchronization to a uniquely identified FH channel. In [10], Zurbes concludes that inter-piconet interference is only significant in dense scenarios of 50–100 overlapping piconets. Although piconets can coexist, it is also desirable to be able to interconnect them. As previously stated, the 8 device upper bound on piconet capacity and the use of FH channels necessitate piconet switching. In order to interconnect two or more piconets, at least one device must participate in more than one piconet. By allowing certain devices to function as bridges and switch between participating piconets, these *bridge nodes* can relay packets between piconets. The bridge node must also keep track of all its connected piconet masters' identities and clock offsets, since clocks are just loosely synchronized. Upon switching from a piconet, the bridge node changes its operational mode from ACTIVE to HOLD. In HOLD mode, the device maintains an AM_ADDR but is not active for a fixed time interval. Consequently, the bridge node cannot respond to master polls while it is absent from the piconet. Upon switching back, the bridge must re-synchronize to the master in order to follow the frequency hopping sequence again. Miklos et al. conclude in [11] that piconet switching poses a significant overhead and has a major impact on system performance. It is therefore important for overall scatternet performance not only that the scatternet topology is carefully constructed, but also that piconet switching is scheduled as efficiently as possible. Section 2.6 describes some of the proposed approaches for inter-piconet scheduling (IPS).

2.2. Formation metrics and constraints

In the Bluetooth specification [1], scatternets are entirely conceptual. There are no existing guidelines for how a scatternet topology should be

formed. Two scatternet topologies that are formed from separate approaches can appear radically different and retain divergent characteristics. The resulting scatternet topology is therefore based on how the piconets are interconnected. What constraints are used and the desired scatternet criteria have a significant impact on the resulting topology.

Examples of some criteria used are listed below:

- Complete scatternet connectivity.
- Maximized aggregate bandwidth.
- Minimized average routing path length.
- Maximized average node availability.
- Minimized bridge switching overhead.
- Communication group clustering.
- Self-healing.
- Multi-hop node participation.
- On-demand scatternet formation.

One of the most important decisions to make before forming a scatternet is to determine the constraints placed on bridge nodes. A node is only allowed to participate as a master in a single piconet, but can join any number of other piconets as a slave on a time share basis. This type of bridge is called a master/slave (MS) bridge. The other, more restrictive, type of bridge is the slave/slave (SS) bridge. An SS bridge is only allowed to operate as a slave in the piconets in which it participates. Misic et al. point out that the mean access delay in scatternets with SS bridges is lower than when MS bridges are used [12]. The main reason is that intra-piconet communication ceases while the MS bridge is participating in another piconet. In contrast, disallowing MS bridges eliminates some tree, ring and mesh topologies that produce shorter average path lengths and, in some cases, avoid routing loops and provide simpler scatternet routing [13–18].

Another formation metric that directly affects scatternet performance is the bridge degree. Bridge degree can be defined as the number of piconets in which a node is allowed to participate. In [11,19], the authors conclude that there is a direct correlation between bridge switching overhead and scatternet performance. By placing constraints on the maximum bridge degree, as done in [19–22], scatternet performance is increased. Restricting bridge overlap by allowing only one bridge between any two piconets is another possible constraint [20].

A performance metric that is difficult to determine a priori is the traffic pattern [23]. Traffic-flows between nodes have a significant impact on throughput and overall performance [24]. More specifically, groups of devices that frequently communicate with each other make up Communicating Groups (CGs) and should be clustered together for better performance [25]. Generally this is only a metric that is used in optimized strategies, since it requires knowledge of traffic patterns and node relationships. This will be addressed further in Section 3.3.

2.3. Scatternet models

Based on the metrics and constraints discussed in the previous section, a number of general scatternet models can be derived. This section categorizes different types of scatternet topologies and compares their characteristics.

The scatternet formation approaches in the literature, which we review, can be classified into the following topology models:

- Single Piconet Model (SPM).
- Slave/Slave Mesh (SSM).
- Master/Slave Mesh (MSM).
- Tree Hierarchy (TH).
- Master/Slave Ring (MSR).
- Slave/Slave Ring (SSR).

The simplest type of scatternet is the Single Piconet Model (SPM), illustrated in Fig. 2(a) [25]. An SPM is the only scatternet type that is natively supported in the Bluetooth specification. In an SPM, as in a traditional piconet, a single master and up to seven active slaves communicate. The rest of the participating slaves are placed in PARK mode and can be substituted in when the master needs to communicate with any of them. This model, although simple and natively supported in the specification, is very inefficient and requires active slaves to be placed in PARK mode in order to accommodate a parked slave.

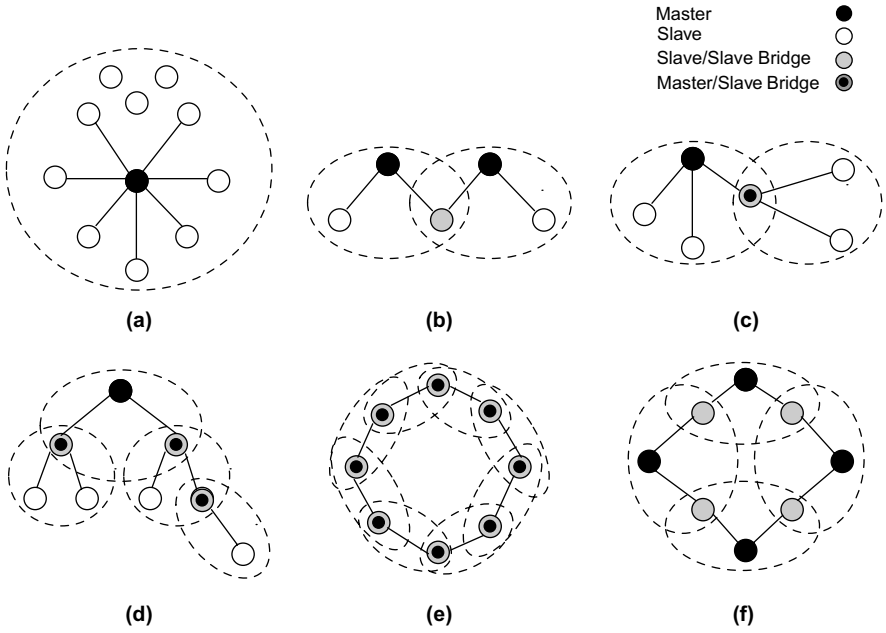

Fig. 2. Scatternet topology models. (a) Single Piconet Model (SPM), (b) Slave/Slave Mesh (SSM), (c) Master/Slave Mesh (MSM), (d) Tree Hierarchy (TH), (e) Master/Slave Ring (MSR), (f) Slave/Slave Ring (SSR).

Two variations of mesh topologies are shown in Fig. 2(b) and (c). In Fig. 2(c), a Master/Slave Mesh (MSM) that interconnects piconets using MS or SS bridges is shown. An MSM is the most non-restrictive scatternet model, since it allows any type of piconet interconnection to be formed. Degree constrained MSMs place an upper-bound on the number of piconets in which a bridge node participates. For example, the BTCP protocol by Salonidis et al. is a 2-MSM degree constrained scatternet, where 2 is the upper-bound on the number of piconets in which a bridge node participates [20]. Slave/Slave Mesh (SSM) scatternets, illustrated in Fig. 2(b), are similar to MSMs, except that they only allow SS bridges. Generally, this increases scatternet performance at the cost of additional protocol complexity.

Fig. 2(d) shows a Tree Hierarchy (TH) topology. TH scatternet topologies differ from the preceding models in that they have a single root node and descendant tree nodes. The Tree Scatternet Formation (TSF) protocol by Tan et al. is an example of a TH scatternet [13]. TSF will be discussed in detail in Section 3.1.1. The main advantages of TH topologies over the preceding models is that they have logarithmic average path length and simplify scatternet routing. On the other hand, the root node is a bottleneck point and node disconnections close to the root node can partition the scatternet and substantially reduce node availability.

Master/Slave Ring (MSR) and Slave/Slave Ring (SSR) models, in Fig. 2(e) and (f), are ring structured scatternets. The main reason to form a ring topology is to alleviate the bottleneck problems in TH topologies [16] while maintaining simple scatternet routing [26]. However, ring models suffer from partitioning problems and significantly longer average path lengths and mean access delays.

Depending on the purpose of the scatternet and what metrics are emphasized, any of the aforementioned models can be appropriate. For instance, in applications where routing and access delay are of great importance, tree hierarchy solutions might be the most suitable. In other scenarios where connectivity and availability are the predominant metrics, the mesh models might work better.

2.4. Link formation

After determining the purpose of the scatternet, the participating nodes must be discovered before a topology can be formed. In Bluetooth, this is called device discovery. Due to the FH channel MAC scheme employed in Bluetooth, devices cannot simply detect all other nodes within radio proximity, as done in 802.11 [9]. Instead, Bluetooth devices must enter the INQUIRY or INQUIRY_SCAN states and transmit inquiry packets or listen for inquiries respectively. Two trains of 16 predetermined frequency hop carriers are used for device discovery. There is no a priori clock synchronization and each Bluetooth device merely has a native clock. Therefore, there is no way for inquiry-scanning devices to know exactly at which FH channel an inquiring device is transmitting. Thus, the inquiring device broadcasts Inquiry Access Code (IAC) packets and changes hop carriers at twice the rate of the inquiry scanning device. This allows devices to discover each other independent of clock synchronization. If the inquiry scanning device receives an IAC packet, it backs off for a Random Backoff (RB) delay period and responds with a Frequency Hopping Sequence (FHS) packet, which contains its own BD_ADDR and clock value. Thereafter, paging procedures can commence and the previously inquiring device can page the respondent using a Device Access Code (DAC) packet. If the paging procedure is successfully completed, a link is established with the initially inquiring device as the master. If the roles need to be reversed, a master/slave switch can also be performed. After the link has been established, the slave stays synchronized to the piconet using the frequency hopping sequence (FHS) and clock value it received from the master during paging.

As described in the Bluetooth specification [1], device discovery is not a gratuitous process. In order for device discovery to begin, device roles must have been preassigned. Devices must subsequently enter the INQUIRY or INQUIRY_SCAN state. For scatternet formation, this link establishment process should happen gratuitously. Devices that wish to participate in the scatternet should automatically engage in device discovery and form a link either as a master or as a slave. Gratuitous link establishment can either be done symmetrically or asymmetrically.

Salonidis et al. propose a symmetric link protocol in [20,27]. In their approach, devices alternate independently between INQUIRY and INQUIRY_SCAN states with randomized state residence intervals. Two devices successfully discover each other if they remain in complementary states for longer than a lower-bound interval.

An asymmetric approach to link establishment is proposed by Ching et al. in [21]. Their randomized approach determines inquiry and inquiry scan roles probabilistically. Devices repeat this procedure until a link is established.

In [28], Ramachandran et al. evaluate a deterministic and a randomized algorithm, variations of the two preceding solutions respectively. Their results suggest that the randomized approach produces significantly lower link establishment delay over the deterministic (or symmetric) algorithm.

2.5. Intra-piconet scheduling

The Bluetooth specification describes a round-robin intra-piconet scheduling algorithm, in which the master polls each slave in a cyclic manner without considering queue lengths or other traffic dependent metrics [1]. This naive polling scheme is generally called Pure Round Robin (PRR). PRR provides fairness between the slaves, but wastes unnecessary time slots and does not provide any Quality of Service (QoS) bounds.

Kalia et al. propose more efficient polling schemes in [29]. Their polling algorithms utilize the queue state of the masters and slaves to determine the polling order. Their Priority Polling (PP) scheme prioritizes links where both the master and the slave have non-empty queues. To prevent link starvation and provide fairness among the links, they also propose the K-Fairness scheduling policy (KFP). KFP does round-robin scheduling among links on which one or both queues are non-empty. When one of the queues is empty, the algorithm sacrifices the current link polling to a link where both queues are full. An absolute fairness bound of K slots is guaranteed, by not sacrificing additional link polls, when the difference between the

maximum and minimum provided service exceeds K slots. Both algorithms significantly increase slot utilization over the PRR scheme. However, both PP and KFP assume that the master has complete knowledge of the queue lengths at the slaves.

Capone et al. point out that the algorithms in [29] are idealistic, since the master only has partial knowledge of the queue lengths [30]. Due to the master-driven polling structure, slaves cannot inform the master of their queue lengths until after being polled by the master. The authors highlight that the PRR scheme provides fairness but is not exhaustive. On the other hand, the Exhaustive Round-Robin (ERR) scheme can potentially introduce link starvation. Instead, they propose the Limited and Weighted Round Robin (LWRR) intra-piconet scheduling scheme. LWRR polls the slaves in a weighted round-robin manner. The weights are adaptively decreased when an empty queue is encountered and reset to its original value when the queue is non-empty again. In this manner, queues that are likely to be empty are polled less frequently.

Golmie et al. present the Bluetooth Interference Aware Scheduling (BIAS) algorithm, which reduces the impact of interference in the channel [31]. The idea is to detect excessive interference on frequency carriers and then avoid packet transmissions. Thereafter, the bandwidth leftover by the constrained (by interference) devices are reallocated fairly among the rest using a credit system. The algorithm provides short term fairness between unconstrained (interference-free) devices, while maintaining their guaranteed service rates.

A precursor to efficient scatternet communication is the existence of an inter-piconet scheduling protocol. This is described further in the next section.

2.6. Inter-piconet scheduling

Inter-piconet scheduling (IPS) is a prerequisite to scatternet formation. Efficient inter-piconet scheduling minimizes wasted time slots when bridge nodes that are participating in other piconets are polled. Therefore, it is important that time slots are scheduled efficiently. The performance analysis by Miklos et al. in [11] suggests that decreasing the bridging overhead and number of links is fundamental for good performance in a scatternet. The introduction of an inter-piconet scheduling (IPS) algorithm is therefore necessary.

In [2], Johansson et al. describe an IPS approach based on Rendezvous Points. A Rendezvous Point (RP) is a scheduled time slot at which the master will poll the bridge node. Thus, the bridge node should be present in the piconet at the RP. Johansson et al. present an RP algorithm in [32]. They emphasize that for IPS algorithms to be efficient, there should be some coordination with an IRPS algorithm (described in Section 2.5). The authors also assume that master devices are *never* selected as bridges. This assumption is well supported by the experimental analysis by Misic et al. in [12]. They conclude that the mean access delay is lower when slave/slave bridges are used. If master/slave bridges are used slaves cannot communicate at all when the master is away, since all communication is master initiated. The Rendezvous Point IPS algorithm by Johansson et al. in [32] is called Maximum Distance Rendezvous Point (MDRP). The basic idea is to maximize the distance between RPs within a periodic super frame. Based on the analysis by Miklos et al. in [11], maximizing the time between the polling of bridge nodes increases performance for static traffic models. However, the MDRP algorithm is not adaptive and therefore does not react to bursty traffic. It also requires complex coordination of the RV points.

Racz et al. propose the Pseudo-Random Coordinated Scatternet Scheduling (PCSS) IPS algorithm in [33]. In their approach, they use a pseudo random sequence of checkpoints generated from the master clock and BD_ADDR of the bridge slave, instead of the RPs used in [32]. In the event of checkpoint collisions (when a bridge has multiple checkpoints scheduled in different piconets or when traffic patterns change), the checkpoint intensity can be increased or decreased adaptively.

In [34], Johansson et al. propose a scheduling algorithm which they call JUMP mode. JUMP mode is a proposed new operational mode. Hence, it requires modification to the current specification. The JUMP mode approach is very similar

to [32,33]. Bridge nodes signal their presence on a link at pseudo random rendezvous windows. Both slave/slave and master/slave bridge nodes are allowed, and are called jumping slaves and jumping masters respectively. By signalling its presence on a link, the jumping node notifies the link peer that it will be present during the entire RV window. To reduce wasted polling of jumping slaves, Johansson et al. allow jumping slaves to establish a long term signaling scheme. Jumping masters can also signal to its slaves that it will be absent during a RV window, so that the slaves won't have to listen for polls during the master's absence.

Another approach to mitigate the problem of masters polling absent bridge nodes is addressed in [35]. Zhang et al. call this the *bridge conflict problem*. Their Flexible Scatternet-wide Scheduling (FSS) scheme polls slaves in a weighted round-robin fashion. The polling weight is represented by a tuple (P,R) where P indicates the scheduled cycle period of the node and R indicates the maximum polls in a scheduled cycle. The polling weight can be adaptively adjusted by the master, based on estimated traffic on the link. Each bridge node has an associated switch-table that indicates when the bridge will be present and in which piconet. This bridge table is replicated at the connected master, but modified by each bridge node independently (based on its queue length). The algorithm provides an adaptive scheduling scheme. However, it requires a multi-phased single-hop scatternet formation protocol, such as BTCP [20], since the switch tables are initially created by a coordinator.

Har-Shai et al. present an approach applicable to smaller scatternets that do not require complex coordination in [36]. Their Load Adaptive Algorithm (LAA) is a completely dynamic approach, in which the next rendezvous point is scheduled when the bridge switches piconets. This approach avoids establishing inflexible periodic schedules. The slave/slave bridge node is also assumed to be connected to exactly *two* piconets. LAA determines when a bridge node should switch piconets based on several factors. The algorithm uses queue length to determine how soon it should switch piconets. An upper-bounded time commitment interval is also used to notify the master of how long the bridge will be absent, in order to prevent wasted polls. The algorithm is adaptive and requires minimal coordination.

3. Scatternet formation

Scatternet formation can be accomplished in several different ways. We distinguish between single-hop, multi-hop and optimized solutions. Single-hop solutions require that all devices are within radio transmission range of each other, while multi-hop solutions allow devices to join the scatternet if they are within proximity of at least one participating device. Optimized solutions are often not of practical use, but offer theoretical insight into how scatternet topologies can be efficiently constructed.

Multi-hop solutions are distributed by nature. Within the category of single-hop solutions we also distinguish between coordinated and distributed approaches. For coordinated solutions, some entity has complete knowledge of the network and assigns roles and connections to all participating devices. This can be done by electing a coordinator using a leader election process [37].

3.1. Single-hop topologies

The scatternet formation protocols described in this section are classified as single-hop topologies. This means that they rely on the assumption that all devices involved in the formation are within radio transmission range of each other. The Bluetooth specification specifies three power control classes and allow transmit power control between 0 and 20 dB m [1]. Although power control schemes have been proposed, e.g [38], we only consider approaches that use static TX power. We further classify single-hop topologies based on whether the formation uses a coordinator or if it is completely distributed.

3.1.1. Coordinated solutions

Coordinated single-hop topologies are formed by an omnipotent device that has been elected to coordinate the formation.

Salonidis et al. present one of the first protocols for scatternet formation [20,27]. Their Bluetooth Topology Construction Protocol (BTCP) aims to solve the problem of asymmetric link formation, and forms a Master/Slave Mesh (MSM) topology. The authors emphasize that spontaneous link formation requires devices to automatically engage in device discovery. Whereas device discovery in the specification is designed to deterministically select an inquiry state, they suggest that devices should voluntarily enter either the INQUIRY or INQUIRY_SCAN state. In their symmetric link formation protocol, devices alternate between INQUIRY and INQUIRY_SCAN after a random state residence time. If two devices meet in complementary states, for longer than the required formation delay, a link is formed. Although the authors claim that the protocol is distributed, since devices spontaneously engage in scatternet formation, it requires a centralized leader with global knowledge and is therefore not considered a distributed solution. The formation protocol creates a 2-MSM connected scatternet using single bridge links between exactly two piconets. BTCP is divided into three phases. In the first phase, a coordinator is chosen using a leader election process. Thereafter, the coordinator determines the master and bridge roles based on a formula that imposes a 36 node upper bound on the scatternet. Finally, the links are formed. BTCP requires en masse node arrival and does not take in to consideration mobility and node failures.

Ramachandran et al. present a deterministic and a randomized algorithm, applicable to Bluetooth scatternets, to form ad hoc clusters in [28]. Both approaches involve a leader election of a super-master, which subsequently forms the actual topology in a centralized manner. The deterministic approach is similar to the symmetric link formation in BTCP, in which nodes alternate between INQUIRY and INQUIRY_SCAN states. Nodes that discover each other form a virtual inquiry response tree with the root node as a master. The master then forms a cluster, or a piconet, from the devices in the virtual response tree. Thereafter, a super-master is elected among the masters and the clusters are interconnected. In the randomized approach, devices determine their inquiry role probabilistically using several rounds of Bernoulli trials. This determines the master and slave designates and ultimately the makeup of the clusters. The MSM scatternet topology is then formed by the super-master in the same manner as in the deterministic approach. Both algorithms require en masse node arrival and do not specify how bridges interconnect the clusters.

Zaruba et al. present Bluetrees in [14]. They present two variations of their algorithm; Blueroot Grown Bluetrees and Distributed Bluetrees. The Distributed Bluetrees approach is a multi-hop solution and is described in Section 3.2. A Blueroot Grown Bluetree is a TH topology that is formed from an arbitrarily selected coordinator node, called the *Blueroot*. A rooted spanning tree is built from the Blueroot using the neighborhood topology graph. The root node is a master and every one-hop neighbor is a slave. The children are then assigned an additional role as a master in another piconet, and the tree is recursively formed. Moreover, each internal tree node is a master/slave bridge node. In order to prevent exceeding the seven slave limitation in piconets and introduce excessive overhead, the authors limit the number of slaves to 5. Blueroot Grown Bluetrees requires radio vicinity for all nodes, while its distributed counterpart does not.

In [39], Sun et al. present a self-routing Bluetree TH scatternet topology. Their approach is directly based on Blueroot Grown Bluetrees (described above) and binary search trees. After the tree is formed, routing is trivial. The node insertion position in the Bluetree is determined by the root based on the BD_ADDR of the new node. QUERY messages are passed up the tree to inform internal nodes and the root of the current range of BD_ADDRs of each node's children. Therefore, when the root node receives a JOIN request from a new node it can easily determine in which of its children's ranges the node should be placed. Once the root finds the correct insertion position, the new node is inserted as a leaf node. If the newly inserted node is the root of a subtree, some of its children might violate the Bluetree range constraints. This is solved by the root propagating allowable range messages down the tree. Once a node detects a child that is outside the allowable range, that

child is disconnected and subsequently rejoined by the root at the correct position. The protocol supports incremental node arrival and is self-healing, even when multiple nodes fail. It makes scatternet routing trivial with the tradeoff of continuous tree maintenance. As with other TH topologies, it has the problem of bottlenecks close to the root and possible scatternet partitioning. It is also possible that unbalanced trees are formed, which eradicates the benefit of the logarithmic average path length.

A Bluerings protocol is presented by Lin et al. in [26]. Their approach forms a 2-SSR topology, in which only slave/slave bridges are used. In contrast to the Bluerings approach in [16] (described in Section 3.1.2) this protocol allows complete piconets to join the ring structure. The protocol relies on a leader election process to form the piconets, and assigns bridge connections in accordance with the ring topology constraints. Packet routing is done unidirectionally, except during maintenance operations when the ring is broken. The packets can then be routed in the reverse direction. Exactly two bridge nodes are present in each piconet; one upstream and one downstream. The bridges are further connected between exactly two piconets to ensure the ring structure. The protocol is self-healing and reconstructs the ring upon node failures using Dedicated Inquiry Access Codes (DIACs) to reconnect missing bridge links. The protocol simplifies routing with the tradeoff of longer average path lengths.

3.1.2. Distributed solutions

In contrast to the previous section, distributed single-hop approaches do *not* depend on a single device to form the scatternet.

In [21], Law et al. present a randomized protocol that forms a Slave/Slave Mesh (SSM). Devices are partitioned into components, which consist of a single device or a piconet in which the master is the leader of the component. Similar to the randomized link formation approach in [28], the authors use asymmetric link formation. Devices probabilistically determine whether to enter the INQUIRY or INQUIRY_SCAN state. Each leader of a component, disjoint device or master, attempts to either add additional slaves to its piconet or, if it currently has no slaves, tries to join another piconet. Each leader of a component tries to find other leaders and relinquish leadership. Finally, only one component leader is still active to connect additional slaves. To optimize the scatternet topology, the protocol allows merging of piconets and migration of slaves between them. The protocol incrementally forms a 2-SSM with bridge nodes participating in exactly two piconets. It is optimized to minimize time and message complexity while allowing incremental joins. However, it does not handle device failures.

Chun-Choong et al. propose an approach to form ring structured scatternets in [16]. Their Bluerings approach produces a 2-MSR topology, in which all bridge nodes are master/slave bridges and each bridge connects to exactly two piconets. In order for a complete Bluering to be formed, en masse node arrival and radio vicinity is assumed. The algorithm is categorized as a distributed solution since each node is assumed to be engaged in formation independently–although simultaneously. There are two variations of the algorithm. In the first, the head of the semi-connected ring performs INQUIRY while the tail performs INQUIRY_SCAN. Disjoint nodes probabilistically choose between the two inquiry states. In order to prevent loops from forming before all nodes are included, the head of the semi-connected ring must have the largest identifier of the connected nodes. By allowing inquiry scanning nodes to only connect to smaller identifiers, loops can be avoided. In the second variation, loops are prevented by disallowing the tail from engaging in inquiry and inquiry scanning completely. Further, only disjoint nodes and the head are allowed to probabilistically select an inquiry state. Both variations enclose the ring after a timeout period during which no additional nodes are connected. The ring topology has the advantage of two paths to any node (as long as the ring is maintained) and constant path length. Routing is also made trivial. The disadvantages of the approach include excessive packet latency and much longer average path length ($N/2$) than in TH topologies. The algorithms do not provide any ring maintenance for incremental arrivals and node failures either.

A tree hierarchy (TH) scatternet formation protocol is presented by Tan et al. in [13,40]. The Tree

Scatternet Formation protocol (TSF) forms a TH scatternet in a distributed way, allows incremental arrivals, and handles node failures. As in [21], TSF partitions devices into components. Each component in TSF is either a single free node or a subtree that seeks to join another tree in the forest. Similar to BTCP [20], TSF uses symmetric state transitions to establish links. When links are formed, the master becomes the root and the slave becomes a leaf node. TSF restricts free nodes to only connect to other non-root nodes and other free nodes, while root nodes can only connect to other root nodes. When two root nodes connect, one becomes the master while the other becomes a slave. These restrictions prevent loops from forming. Self-healing of the tree is native to the protocol. Internal nodes that lose connectivity to their parent become roots and attempt to connect to another root node, while roots that lose all their child nodes become free nodes. TSF isolates communication between root nodes using the Limited Inquiry Access Code (LIAC), which is native to the specification. TSF is distributed, produces a connected scatternet, is self-healing, and simplifies scatternet routing. However, due the nature of TH topologies the root node is a bottleneck in the network and node failures close to the root partitions large portions of the scatternet. The authors mention in [13] that TSF is not guaranteed to heal network partitions when all devices are not within radio vicinity. The protocol is therefore classified as a single-hop topology. Based on its decentralized nature and the interconnection of rooted subtrees it could, however, function in a multi-hop scenario. Although it does not necessarily maintain its self-healing and connected properties in such case. A multi-hop optimization of TSF called SHAPER [18] is described further in Section 3.2.

3.2. Multi-hop topologies

In this section we cover protocols that do not require all devices to be within radio vicinity of each other.

In Section 3.1.1, we described the coordinated single-hop Bluetrees approach by Zaruba et al. [14]. The multi-hop version, Distributed Bluetrees, works analogous to Blueroot Grown Bluetrees except that multiple *init nodes* are used. Using an election process, the node with the highest ID in the local neighborhood is selected as an *init node*. The second phase of the protocol merges the subtrees into a single TH scatternet topology. Whereas Blueroot Grown Bluetrees required radio vicinity for all nodes, Distributed Bluetrees does not. It works in a distributed manner to form a multi-hop scatternet, but cannot always guarantee connectivity of all subtrees. However, since the topology is a tree it also simplifies scatternet routing.

Wang et al. present Bluenets in [15]. In their approach, the scatternet topology is a 2-SSM. They emphasize that, in comparison to Bluetrees, average path length in Bluenets is shorter and a Bluenet can sustain higher traffic flows. The authors explain both these observations by stating that in the heavily connected mesh topology (2-SSM) paths are not required to go along the congested and non-optimal path through the root node. Of course, the tradeoff is that routing in the Bluenet scatternet is much more complex than in Bluetrees. Analogous to Bluetrees, Bluenets are not able to guarantee scatternet-wide connectivity.

Basagni et al. propose a multi-hop solution in [22]. In their three-phased approach, devices first engage in discovery using a symmetric link formation protocol. In addition to the traditional symmetric link formation behavior seen in [20], devices also exchange a weight parameter. After the discovery and information exchange, devices have knowledge of the local neighborhood graph. Based on the exchanged weights, devices in the local neighborhood with the highest weight, called *init* devices, become masters. These devices initiate the formation phase and start to form Bluestars (piconets). Disjoint devices that receive a page from a device with a larger weight join as slaves. After the *init* devices have successfully paged all its neighbors with smaller weight, the second phase is done. Devices that only have neighbors with larger weight become masters and page other devices with smaller weight. In the final phase, a BlueConstellation scatternet is formed. The BlueStar masters determine the *init* masters by comparing weights with neighboring masters that are either two or three hops away. The *init* masters then

instruct their slaves to page specific neighbors, in order to form gateways to neighboring BlueStars. Thereafter, they engage in a master/slave switch and the paging slave becomes a bridge node between the two BlueStar piconets. However, only neighboring masters that are two hops away can receive the page from the gateway slave. In the event that the neighboring master is three hops away, a gateway piconet is created to join the two BlueStars. The protocol guarantees scatternet connectivity in a multi-hop environment. However, it neither provides self-healing nor does it allow incremental node arrival. This is due to the fact that the protocol is divided into multiple phases. The BlueMesh protocol, by the same authors, is presented in [19]. This protocol is very similar to the BlueStars protocol described above [22]. The resulting topology is an SSM and the authors claim that it is degree constrained. They conclude that on average bridge nodes participate in no more than 2.3 piconets [19].

In [18], Cuomo et al. present the SHAPER algorithm. It is directly based on TSF [13], but ensures connectivity and self-healing in a multi-hop scenario. The main difference between TSF and SHAPER is that while TSF uses a set of coordinators (that are roots of subtrees), SHAPER allows both root and non-root nodes (of partitioned subtrees) to form links and initiate tree reconfiguration. The main tradeoff from allowing all nodes to form links to other subtrees is the fact that they must all periodically engage in either inquiry or inquiry scanning. In TSF this is limited to coordinator nodes, which prioritizes scatternet communication over costly formation for non-root nodes. However, as the authors of SHAPER also state, this potentially prevents healing of the TH topology when coordinators are not within radio transmission range of each other. SHAPER avoids this and ensures connectivity. Initial tree formation is conducted in the same fashion as in TSF. The important differences lie in the way SHAPER handles link establishments between nodes in different tree partitions. There are several cases. In the first case, in which both connected nodes are roots, the root of the larger tree becomes the new root and the topology is reconfigured accordingly. If the slave on the link has the larger tree, a master/slave switch must be performed first. This case is similar to TSF. In the second case, either the slave device is a root and the master a non-root or the slave device is a free node. Either way, the tree is simply attached and the parameters are updated. The third case occurs when the master on the link is either a root or a free node and the slave is part of a tree. This is analogous to the first case, except tree sizes do not have to be compared. A master/slave switch must be performed first and the topology is then reconfigured. In the last case, both connected nodes are non-root nodes. The smaller of the two trees is reconfigured and the parent-child relationships are inverted so that all nodes in the reconfigured tree are descendants of the connected node. To prevent multiple reconfigurations from initiating simultaneously, SHAPER implements a locking mechanism that forces non-root nodes to obtain permission from the root before reconfiguring the tree. The authors claim that SHAPER has a lower average formation delay than TSF. As previously mentioned, the tradeoff is that non-root nodes have less time to engage in communication, which reduces communication efficiency.

A radically different approach, from the solutions described so far, is presented by Liu et al. in [41]. They point out that approaches that attempt to ensure scatternet connectivity must be periodically maintained, regardless of whether there is any traffic across the links. Instead, they propose a solution that establishes a scatternet on-demand along multi-hop routing paths. The scatternet is built as part of the route discovery and only devices along the route from the source to the destination are included. In order to form a scatternet along the route, master and slave roles must be assigned to the intermediate nodes. Ideally, every other device is a master and the rest are slaves. All bridge nodes are then slave/slave bridges. However, due to dynamic route changes and uneven route lengths they also allow master/slave bridges when necessary. As in general ad hoc on-demand routing protocols, a route request packet is flooded in the network from the source node [42,43]. Once the destination receives a route request packet, it sends a route reply packet along the reverse path to the source. The route flooding

is somewhat more difficult in Bluetooth than in traditional DS-CDMA ad hoc networks, since a link has to be established before information can be exchanged between nodes. The authors note that establishing point-to-point links along all potential paths, in order to broadcast the route request, imposes an excessive amount of overhead. Instead, they incorporate the route request packets in the pre-existing inquiry broadcast mechanism. Two types of Extended ID (EID) packets, which contain the additional route request information, are needed since enough information cannot be accommodated in a single EID packet. This is accomplished by sending the type 2 packet during the response time slot (after the inquiry broadcast), since the type 1 EID packet does not require instant acknowledgement. In fact, the RB delay mechanism from the BT specification discourages immediate replies in order to thwart collisions. After the destination receives the first route request, the scatternet is formed backwards along the reverse route by the transmission of the route reply packet. Data packet routing is thereafter accomplished by next-hop entries at each intermediate node, similar to [42]. To reduce the path latency due to bridge switching overhead, the authors propose to align the time slots along a route for efficient path traversal. However, when multiple routes are in effect this becomes increasingly difficult. The scheme also requires modification to the specification in order to work.

3.3. Optimized topologies

Scatternet formation approaches based on theoretical foundations offer important insight into how optimized topologies can be constructed. This section covers some of the theoretical approaches proposed.

Yun et al. present an approach that forms an MSM topology in [17]. Their Bluestars approach models the discovery neighborhood as an inquiry graph I. From the inquiry graph I, $2^{|I|}$ topology subsets are available and one is selected. A combination of asymmetric and symmetric link formation is used. Symmetric link formation is used for en masse arrival, while asymmetric is used for incremental arrivals. After determining a topology graph T, neighboring nodes are grouped into Bluestars (piconets) and links are established between each neighboring Bluestar. Bluestars are further pruned into Bluestars* by removing redundant links. This solution is similar to [22], but requires a costly determination of the optimal topology subset before the scatternet is formed.

In [23], Cuomo et al. define a methodology for scatternet formation. They describe the scatternet as a bipartite graph, and model it using an adjacency matrix. After evaluating all possible matrices with respect to the chosen optimization metrics, an optimized topology is found. The SSM topology can then be formed using a centralized strategy. The evaluation metrics used are classified as traffic dependent and traffic independent. The traffic dependent metrics include residual scatternet capacity and average node load, while the traffic independent metrics include maximized overall capacity and average path capacity.

Chiasserini et al. present an optimized approach for scatternet formation that attempts to minimize the traffic load in [24]. The authors assume that traffic patterns and routes are known a priori and formalize the topology formation as a min–max problem. First, they locate the bottleneck node in the network, and thereafter determine the topology in order to minimize the traffic load at that node. Although the approach is not applicable in practice, the authors note that it provides insight into how topologies that fulfill the capacity constraints on Bluetooth scatternets should be formed. They also discuss a distributed approach that incrementally forms a scatternet and utilizes Dedicated Inquiry Access Codes (DIACs) to classify nodes based on their individual characteristics, such as load, battery power, and computational capacity.

Baatz et al. present an approach to optimize scatternet capacity in [44]. They note that due to the FH-CDMA structure and communications scheduling employed in Bluetooth, only a subset of nodes can communicate on disjoint links at any time. Therefore, they model the scatternet as a directed graph and determine matchings between nodes so that no two links (edges) communicate with the same node (vertex) at the same time. Thereafter, they define a set of these matchings as

a *scatternet schedule*. Based on the schedule, a feasible rate vector is defined that conforms to the scatternet capacity constraint. In order to maximize capacity while ensuring that all links are served in a fair way, a max–min fair rate vector is obtained that equally distributes the scheduling across all the maximal matchings. A scatternet topology is then determined by partitioning the links into disjoint sets so that (near) perfect matchings, called (near) 1-factors, are obtained. Depending on how these 1-factor matchings are chosen, the topology will differ. The authors state that the actual topology formation can be done using a centralized single-hop approach such as BTCP [20].

Scatternet formation based on dominating sets is of interest due to the similarity between the connected dominating set (CDS) and the subsets of masters and bridges in scatternets. Wan et al. present a distributed approximation algorithm to construct a connected dominating set in ad hoc networks [45]. Their algorithm consists of two phases, during which a minimum independent set (MIS) and a dominating tree is formed respectively. The MIS consist of a subset of nodes that are separated by exactly two hops. The dominating tree T^* is constructed during the second phase, and its internal nodes form a CDS. The algorithm is not directly applicable to scatternets, but presents a constant factor approximation to the NP-hard Minimum Connected Dominating Set (MCDS) problem. Stojmenovic present an algorithm for scatternet formation that is based on the concept of dominating sets in [46]. The approach is based on the CDS algorithm from [47], in which a CDS is constructed by including all nodes and then removing locally redundant nodes. Stojmenovic eliminates redundant neighbors by applying a Yao subgraph construct on the Relative Neighborhood Graph (RNG) or the Gabriel Graph (GG), in order to produce the dominating set. Thereafter, device roles are determined so that the scatternet can be interconnected.

Zussman et al. study capacity assignment in scatternets in [48]. They develop some algorithms to minimize the average delay in the scatternet. They observe that scatternets using only slave/slave bridges are necessarily bipartite, and state the problem of optimal capacity assignment for both bipartite and non-bipartite graphs. By assuming an existing topology and traffic flow information, they develop an optimized algorithm that finds the optimal capacity vector for the capacity assignment problem. They also develop some heuristic algorithms that find approximate solutions to the same problem. They conclude that bipartite scatternet topologies (using only slave/slave bridges) provide the best performance.

4. Scatternet routing

After a scatternet topology has been constructed, it should be capable of efficiently accommodate data traffic. In most topologies, this requires a scatternet routing protocol. As we have seen, there are numerous different approaches to form a scatternet. This is due to the fact that scatternets are not part of the Bluetooth specification. Therefore, the resulting topologies could have very different characteristics. Thus, one general scatternet routing protocol that works in all topologies seems highly unlikely. General wireless ad hoc network routing protocols, such as AODV [42] or DSR [43], have the potential to work in Bluetooth scatternets if they are modified. Depending on the formation protocol used, different modifications are necessary. In some strictly structured topologies, such as tree (TH) or ring (MSR and SSR) topologies, routing is trivial. For instance, in the protocol by Sun et al., routing reduces to traversing a binary search tree [39]. The ring topology in [26], where packets are unidirectionally forwarded around the ring, is another example. However, most scatternet mesh topologies require an explicit routing protocol.

Bhagwat et al. present the Routing Vector Method (RVM) for routing in scatternets in [49]. They assume the existence of a mesh topology scatternet. In RVM, each bridge node assigns local identifiers to each connected piconet. Thereby, a source route of alternating local identifiers and AM_ADDR's (of the outgoing bridge node in the next hop piconet) can be constructed. Routes are created using a route discovery procedure, in which a route request is broadcasted. Once the destination receives a request, it unicasts a reply

Table 1
Protocol comparison chart

Protocol name	Coordinated	Distributed	Multi-hop	Incremental arrival	Topology model	Link formation	Self-healing	Ensures connectivity	Comments
Salonidis et al. [20,27]	Yes	No	No	No	2-MSM	Symmetric	No	Yes	Max 36 devices
Ramachandran and co-workers [28]	Yes	No	No	No	MSM	Asymmetric/ symmetric	No	Yes	Bridge connections are not defined
Zaruba et al. [14]	Yes	No	No	No	TH	Asymmetric	No	Yes	Blueroot Grown Bluetrees version
Sun et al. [39]	Yes	No	No	Yes	TH	Asymmetric	Yes	Yes	Self-routing
Lin et al. [26]	Yes	No	No	Yes	2-SSR	Asymmetric	Yes	Yes	Stateless routing
Law et al. [21]	No	Yes	No	Yes	2-SSM	Asymmetric	No	Yes	Optimized for time and message complexity
Chun-Choong and Kee-Chaing [16]	No	Yes	No	No	2-MSR	Asymmetric	No	Yes	Stateless routing
Tan et al. [13,40]	No	Yes	Yes, if root nodes are within vicinity	Yes	TH	Symmetric	Yes	No	Can function in multi-hop environment
Zaruba et al. [14]	No	Yes	Yes, if *init* nodes are within vicinity	No	TH	Asymmetric	No	No	Distributed Bluetrees version
Wang et al. [15]	No	Yes	Yes	No	2-SSM	Asymmetric	No	No	
Basagni and co-workers [22,19]	No	Yes	Yes	No	Degree constrained SSM	Symmetric	No	Yes	Ensures connectivity in multi-hop environment
Cuomo et al. [18]	No	Yes	Yes	Yes	TH	Symmetric	Yes	Yes	Multi-hop version of TSF
Liu et al. [41]	No	Yes	Yes	Yes	MSM	Asymmetric	Yes	No	Scatternet is formed along on-demand route

along the reverse route. RVM discovers two distinct routes for increased robustness. The protocol greatly simplifies the problem by ignoring mobility and node failures. However, the RVM protocol describes a modification to a general source routing protocol that can potentially be applicable to Bluetooth scatternets.

A similar solution is presented by Prabhu et al. in [38]. The novel idea in this approach is that during the route discovery the intermediate nodes append their battery power level to the request. Thereby, when the destination receives multiple requests it does not automatically choose the shortest route but selects the path which has the maximum cumulative battery power.

An approach that integrates routing with scatternet formation [41] was described in Section 3.2. Liu et al. present a protocol that constructs scatternets on-demand along a route. Their approach, which is also based on source routing, uses an intuitive idea to broadcast the route request information along with the inquiry packets during device discovery. A scatternet is then formed along the reverse route for the duration of the communication session.

In general, routing in wireless ad hoc networks is a difficult issue. The uses of FH-CDMA in Bluetooth and the multitude of proposed scatternet formation protocol just makes it more difficult. Scatternet routing is an area that still requires a lot more research.

5. Summary

In this paper we discussed different criteria for scatternet formation and presented topology models that conform to varying constraints. We reviewed the state-of-the-art approaches with respect to scatternet formation, and categorized the approaches into single-hop, multi-hop and optimized solutions. We further classified the resulting topologies by the general topology models. Single-hop solutions required that all devices were within radio vicinity of each other, whereas multi-hop protocols did not impose such a constraint. We also subdivided the single-hop solutions into coordinated and distributed approaches. Coordinated algorithms required a leader to control the formation, while distributed solutions did not. The reviewed scatternet formation approaches are compared and contrasted in Table 1. As previously mentioned, depending on the criteria for the scatternet and the constraints imposed, any of the solutions from Table 1 can be suitable. We also reviewed some optimized solutions that, for the most part, are not directly usable in practice. We chose not to include theses solutions, from Section 3.3, in Table 1. However, they do provide testimony on how scatternet topologies could be constructed in theory.

Acknowledgements

This research was supported in part by the National Science Foundation, CAREER Award # CCR-9983584 and NSF Grant # 0324836. A preliminary version of this paper appeared in the Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC 2004).

References

[1] Bluetooth Special Interest Group. Available from: <http://www.bluetooth.com>, BluetoothSpecification Version 1.1, February 2001.
[2] P. Johansson, M. Kazantzidis, M. Gerla, Bluetooth: an enabler for personal area networking, IEEE Network 15 (5) (2001) 28–37.
[3] B. Chatschik, An overview of the bluetooth wireless technology, IEEE Communications Magazine (2001) 86–94.
[4] J.C. Haartsen, The bluetooth radio system, IEEE Personal Communications Magazine 7 (1) (2000) 28–36.
[5] J. Bray, C.F. Sturman, Bluetooth: Connect Without Cables, Prentice Hall, Englewood Cliffs, NJ, 2001.
[6] N.J. Muller, Bluetooth Demystified, McGraw-Hill, New York, 2001.
[7] B.A. Miller, B. Chatschik, Bluetooth Revealed, Prentice Hall, Englewood Cliffs, NJ, 2001.
[8] IEEE. Available from: <http://grouper.ieee.org/groups/802/15/>, IEEE 802.15 Working Group for WPANs.
[9] IEEE. Available from: <http://standards.ieee.org/getieee802/>, IEEE 802.11 Standard, 1999.
[10] S. Zurbes, Considerations on link and system throughput of bluetooth networks, in: 11th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2000), vol. 2, 2000, pp. 1315–1319.

[11] Gy. Miklos, A. Racz, A. Valko, P. Johansson, Performance aspects of bluetooth scatternet formation, in: Proceedings of the First Annual Workshop on Mobile Ad Hoc Networking and Computing, MobiHoc, 2000, pp. 147–148.
[12] J. Misic, V.B. Misic, Bridges of bluetooth county: topologies, scheduling and performance, IEEE Journal of Selected Areas in Communications 21 (2) (2003) 240–258 (Special issue on Wireless LANs and Home Networks).
[13] G. Tan, A. Miu, J. Guttag, H. Balakrishnan, An efficient scatternet formation algorithm for dynamic environments, in: IASTED International Conference on Communications and Computer Networks, Boston, MA, 2002.
[14] G.V. Zaruba, S. Basagni, I. Chlamtac, Bluetrees—scatternet formation to enable bluetooth-based ad hoc networks, in: IEEE International Conference on Communications (ICC2001), 2001, pp. 273–277.
[15] Z. Wang, R.J. Thomas, Z. Haas, Bluenet—a new scatternet formation scheme, in: 35th Hawaii International Conference on System Science (HICSS-35), Big Island, HI, 2002.
[16] F. Chun-Choong, C. Kee-Chaing, Bluerings—bluetooth scatternets with ring structures, in: IASTED International Conference on Wireless and Optical Communication (WOC 2002), Banff, Canada, 2002.
[17] J. Yun, J. Kim, Y. Kim, J.S. Ma, A three-phased hoc network formation protocol for bluetooth systems, in: Proceedings of the 5th International Symposium on Wireless Personal Multimedia Communications (WPMC '2002), 2002.
[18] F. Cuomo, G. Di Bacco, T. Melodia, Shaper: a self-healing algorithm producing multi-hop bluetooth scatternets, in: Proceedings of the IEEE Globecom 2003, San Francisco USA, 2003.
[19] C. Petrioli, S. Basagni, Degree-constrained multihop scatternet formation for bluetooth networks, in: Proceedings of the IEEE Globecom 2002, Taipei, Taiwan, 2002.
[20] T. Salonidis, P. Bhagwat, L. Tassiulas, R. LaMaire, Distributed topology construction of bluetooth personal area networks, in: IEEE INFOCOM 2001, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, 2001, pp. 1577–1586.
[21] C. Law, A.K. Mehta, K. Siu, Performance of a new bluetooth scatternet formation protocol, in: Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc2001), Long Beach, CA, USA, 2001.
[22] S. Basagni, C. Petrioli, Multiphop scatternet formation for bluetooth networks, in: IEEE 55th, Vehicular Technology Conference, vol. 1, 2002, pp. 424–428.
[23] F. Cuomo, T. Melodia, A general methodology and key metrics for scatternet formation in bluetooth, in: Proceedings of IEEE GLOBECOM 2002, vol. 1, Taipei, Taiwan, 2002, pp. 941–945.
[24] C.F. Chiasserini, M.A. Marsan, E. Baralis, P. Garza, Towards feasible topology formation algorithms for bluetooth-based wpans, in: 36th Hawaii International Conference on System Science (HICSS-36), Big Island, HI, 2003.
[25] M. Kalia, S. Garg, R. Shorey, Scatternet structure and inter-piconet communication in the bluetooth system, in: IEEE National Conference on Communications New Delhi, India, 2000.
[26] T.Y. Lin, Y. Tseng, K. Chang, C. Tu, Formation, routing, and maintenance protocols for the bluering scatternet of bluetooths, in: Proceedings of the 36th Hawaii International Conference of System Sciences, Big Island, HI, 2003.
[27] T. Salonidis, P. Bhagwat, L. Tassiulas, Proximity awareness and fast connection establishment in bluetooth, in: First Annual Workshop on Mobile and Ad Hoc Networking and Computing, MobiHOC 2000, 2000, pp. 141–142.
[28] A. Aggarwal, M. Kapoor, L. Ramachandran, A. Sarkar, Clustering algorithms for wireless ad hoc networks, in: Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, Boston, MA, USA, 2000, pp. 54–63.
[29] M. Kalia, D. Bansal, R. Shorey, Data scheduling ans sar for bluetooth mac, in: IEEE 51st Vehicular Technology Conference Proceedings, VTC 2000-Spring, Tokyo, vol. 2, 2000, pp. 716–720.
[30] A. Capone, M. Gerla, R. Kapoor, Efficient polling schemes for bluetooth picocells, in: IEEE ICC 01, vol. 7, Helsinki, Finland, 2001, pp. 1990–1994.
[31] N. Golmie, N. Chevrollier, I. ElBakkouri, Interference aware bluetooth packet scheduling, in: Proceedings of IEEE GLOBECOM, 2001, pp. 2857–2863.
[32] P. Johansson, R. Kapoor, M. Kazantzidis, M. Gerla, Rendezvous scheduling in bluetooth scatternets, in: Proceedings of ICC 2002, New York, 2002, pp. 318–324.
[33] A. Racz, G. Miklos, F. Kubinszky, A. Valko, A pseudo random coordinated scheduling algorithm for bluetooth scatternets, in: Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Long Beach, CA, USA, 2001, pp. 193–203.
[34] N. Johansson, F. Alriksson, U. Jonsson, Jump mode—a dynamic window-based scheduling framework for bluetooth scatternets, in: Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Long Beach, CA, USA, 2001.
[35] W. Zhang, G. Cao, A flexible scatternet-wide scheduling algorithm for bluetooth networks, in: IEEE International Performance, Computing, and Communications Conference (IPCCC), 2002.
[36] L. Har-Shai, R. Kofman, G. Zussman, A. Segall, Interpiconet scheduling in bluetooth scatternets, in: Proceedings of the OPNETWORK 2002 Conference, 2002.
[37] H. Garcia-Molina, Elections in distributed computing system, IEEE Transactions on Computers 31 (1) (1982) 48–59.
[38] B.J. Prabhu, A. Chockalingam, A routing protocol and energy efficient techniques in bluetooth scatternets, in: IEEE ICC'2002, New York, 2002, pp. 3336–3340.
[39] M. Sun, C. Chang, T. Lai, A self-routing topology for bluetooth scatternets, in: Proceedings of I-SPAN 2002, Manila, Philippines, 2002.

[40] G. Tan, A. Miu, J. Guttag, H. Balakrishnan, Forming scatternets from bluetooth personal area networks, Tech. Rep. MIT-LCS-TR-826, Massachusetts Institute of Technology. Available from: <http://lcs.mit.edu/>, October 2001.
[41] Y. Liu, M.J. Lee, T.N. Saadawi, A bluetooth scatternet-route structure for multihop ad hoc networks, IEEE Journal on Selected Areas in Communications 21 (2) (2003) 229–239.
[42] C.E. Perkins, E.M. Belding-Royer, S. Das, Ad hoc on demand distance vector (AODV) routing, IETF Internet Draft. Available from: <http://www.cs.ucsb.edu/ebelding/txt/aodvid.txt>, November 2002.
[43] D.B. Johnson, D.A. Maltz, Y. Hu, J.G. Jetcheva, The dynamic source routing protocol for mobile ad hoc networks (DSR), IETF Internet Draft. Available from: <http://www.monarch.cs.rice.edu/internet-drafts/draft-ietf-manet-dsr-07.txt>, February 2002.
[44] S. Baatz, C. Bieschke, M. Frank, C. Kuhl, P. Martini, C. Scholz, Building efficient bluetooth scatternet topologies from 1-factors, in: Proceedings of the IASTED International Conference on Wireless and Optical Communications, WOC 2002, Banff, Alberta, Canada, 2002.
[45] P. Wang, K.M. Alzoubi, O. Frieder, Distributed construction of connected dominating set in wireless ad hoc networks, in: Proceedings of IEEE INFOCOM 2002, New York, 2002, pp. 1597–1604.
[46] I. Stojmenovic, Dominating set based bluetooth scatternet formation with localized maintenance, in: Proceedings of the International Parallel and Distributed Processing Symposium, IEEE Computer Society, 2002, pp. 148–155.
[47] J. Wu, H. Li, On calculating connected dominating set for efficient routing in ad hoc wireless networks, in: Proceedings of the 3rd ACM International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, 1999, pp. 7–14.
[48] G. Zussman, A. Segall, Capacity assignment in bluetooth scatternets—optimal and heuristic algorithms, Mobile Networks and Applications 9 (1) (2004) 49–61 (Special Issue on Advances in Research of WPAN and Bluetooth Enabled Networks).
[49] P. Bhagwat, A. Segall, A routing vector method (rvm) for routing in bluetooth scatternets, in: IEEE International Workshop on Mobile Multimedia Communications (MoMuC '99), 1999, pp. 375–379.

K.E. Persson received a BS degree in Computer Science from the University of Kentucky in 2001. He is currently a Ph.D. student in the Laboratory for Advanced Networking at University of Kentucky, Lexington. He is a member of ACM, IEEE, and IEEE Computer Society. His research interests include wireless and mobile computing systems, ad hoc networks, sensor networks, cellular networks, and distributed systems.

D. Manivannan received the Ph.D. degree in Computer Science from The Ohio State University, Columbus, Ohio, in 1997. He is an assistant professor in the Department of Computer Science at University of Kentucky, Lexington. He has published his research papers in the areas of fault-tolerance and synchronization in distributed systems, routing in wormhole networks, channel allocation in cellular networks, security in wireless personal area networks and security of mobile agents. He has served as program committee member for several International Conferences and also served as reviewer for several International Journals published by ACM, IEEE, Elsevier, Springer Verlag and others. He is a recipient of the CAREER Award from the US National Science Foundation. He is a member of ACM, IEEE, and IEEE Computer Society.

M. Singhal is a Full Professor and Gartener Group Endowed Chair in Network Engineering in the Department of Computer Science at University of Kentucky, Lexington. From 1986 to 2001, he was a faculty in Computer and Information Science at The Ohio State University. He received a Bachelor of Engineering degree in Electronics and Communication Engineering with high distinction from Indian Institute of Technology, Roorkee, India, in 1980 and a Ph.D. degree in Computer Science from University of Maryland, College Park, in May 1986. His current research interests include distributed systems, mobile computing, computer networks, computer security, and performance evaluation. He has published over 160 refereed articles in these areas. He has coauthored three books titled "Data and Computer Communications: Networking and Internetworking", CRC Press, 2001, "Advanced Concepts in Operating Systems", McGraw-Hill, New York, 1994 and "Readings in Distributed Computing Systems", IEEE Computer Society Press, 1993. He is a Fellow of IEEE. He is currently serving in the editorial board of IEEE Trans. on Knowledge and Data Engineering and IEEE Trans. on Computers. From 1998 to 2001, he served as the Program Director of Operating Systems and Compilers program at National Science Foundation.