Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

Ad Hoc Networks 3 (2005) 403–450

Ad Hoc Networks

www.elsevier.com/locate/adhoc

Bluetooth scatternet formation: A survey

Roger M. Whitaker [a,*], Leigh Hodge [a,1], Imrich Chlamtac [b]

[a] *Centre for Mobile Communications, School of Computer Science, Cardiff University, Queen's Buildings, 5 The Parade, P.O. Box 916, Cardiff CF24 3XF, UK*

[b] *Erik Jonsson School of Engineering and Computer Science, The University of Texas at Dallas, P.O. Box 830688, Richardson, TX 75083-0688, USA*

Received 1 October 2003; accepted 1 February 2004
Available online 3 March 2004

Abstract

This paper describes the issue of piconet interconnection for Bluetooth technology. These larger networks, known as scatternets, have the potential to increase networking flexibility and facilitate new applications. While the Bluetooth specification permits piconet interconnection, the creation, operation and maintenance of scatternets remains open. In this paper, the research contributions in this arena are brought together, to give an overview of the *state-of-the-art*. First, operation of the Bluetooth system is explained, followed by the mechanism for link formation. Then, the issue of piconet interconnection is considered in detail. Processes for network formation, routing and intra- and inter-piconet scheduling, are explained and classified. Finally, the research issues arising are outlined.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Bluetooth; Scatternets; MAC; Routing; Scheduling; Connectivity

1. Introduction

The Bluetooth (BT) technology, as specified in the Bluetooth System Version 1.1 [29], is the first flexible, mass market protocol for wireless ad hoc operation. This means that global control of the network is relinquished, permitting spontaneous deployment without dependence on fixed infrastructure. However, decentralised network organisation is required to enable self-planning and management, which is a key challenge in developing fourth generation technology.

The BT specification has been converted into IEEE standard 802.15 [16,88]. From the outset, the specification has been mass-market oriented, driven by a special interest group (SIG) of major manufacturers, keen to develop a robust, flexible,

[*] Corresponding author. Tel.: +44 29 2087 4812; fax: +44 29 2087 4598.
E-mail addresses: r.m.whitaker@cs.cf.ac.uk (R.M. Whitaker), l.e.hodge@cs.cf.ac.uk (L. Hodge), chlamtac@utdallas.edu (I. Chlamtac).
[1] Supported by the EPSRC (GR/S23155/01).

1570-8705/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2004.02.002

and economically viable technology that can be easily incorporated into a multitude of devices to enhance their functionality via short connectivity, using low power radio links (10–100 m) in the unlicensed 2.4 GHz industrial, scientific and medical (ISM) band. Most current Bluetooth devices offer the modest range of 10 m using the (low) power class 2. This is a deliberate action, as Bluetooth is currently primarily a technology for personal area networks (PAN), where device battery power is limited. However, in the long term, higher power and extended range could lead to a range of further applications, beyond personal area networks. Additionally the technology may well prove a useful starting point for the development of other applications, and systems, such as those in the military arena.

In its current form, Bluetooth is well on the way to reaching impressive levels of penetration, helped by its royalty free status. It is estimated that 35 million chip-sets were produced in 2002, with an estimated rise to 510 million by 2006 [35]. The characteristics of the Bluetooth chip [96] have facilitated this, based on an occupancy of 90 mm^2, power consumption of 25 μA when idle with a peak of 25 mA, and a cost of less than 4 USA dollars per terminal for high volume production.

Since the conception of the Special Interest Group (SIG) in 1998 and the release of specification (v1.1 in February 2001), Bluetooth has received a considerable amount of attention, being vigorously marketed by the SIG who promise a technology to "seamlessly connect all your mobile devices". Commercial interest frequently centres around its application (potential future application is given in [20] for example). An important limiting factor in developing future applications are the networking possibilities that the technology can facilitate. As currently specified, Bluetooth self-organises networks of up to eight active devices in *piconets*, but the specification permits much more. Piconet interconnection is possible, permitting multi-hop links between out-of-range devices. However, mechanisms for establishing and maintaining such scatternets remain open, challenged by the need to provide totally decentralised, high performance solutions for potentially dynamic environments.

Consequently, the extent to which the Bluetooth networking capabilities can be advanced are receiving particular attention, and this is the focus of our survey on the *state-of-the-art*. The rapid pace of research in this area has led to a large, dispersed and fragmented literature, which broadly seeks to further investigate and extend the operation of Bluetooth at the equivalent of the network layer in the traditional OSI model. This constitutes development for network formation, maintenance, scheduling and routing in the context of Bluetooth.

We begin by introducing the Bluetooth specification and the operation of the system for a single piconet. To support scatternets, we then consider the extension of functionality in the BT specification. The broad issues involved in defining a protocol to support scatternet operation are described in Section 2. In Section 3, we describe the general issues which influence the design of protocols in the context of Bluetooth. In Section 4, we consider the different possible topologies that have been proposed for scatternet applications. In Section 5 we consider the problem of forming and maintaining a scatternet, specifically the process of establishing device role. In Section 6, we describe the different techniques that have been applied to assess network performance. In Section 7, we consider the issue of routing. Finally, in Section 8 we address the problem of scheduling inter- and intra-piconet communication.

1.1. Moving from piconets to scatternets

The original and first use of Bluetooth technology has been for small clusters of devices, which operate using the concept of a *piconet*. This is a collection of devices sharing the same communication channel. Devices such as laptops, mobile phones, PDAs are currently the main proponents of the technology, and it is likely to remain this way in the short term. However, there are a number of arguments which support the development of the technology for inter-connecting piconets to form *scatternets*. Like many technological developments, scatternets may precede concrete applications. However, high levels of market penetration will increase the density of Bluetooth enabled devices, thereby increasing the potential for scatternet based applications.

Scenarios requiring a greater amount of connectivity have received limited attention. In [38], the authors argue that scatternet functionality is important to allow flexible formation of Bluetooth personal area networks. Additionally, it is argued that scatternet functionality may also be used to improve the performance of a group of nodes that are either already part of a scatternet, or part of separate piconets. The roles of devices in such nodes may be rearranged to adapt to a new traffic distribution.

Bluetooth also can be used in a cellular fashion via access points for wireless LAN applications, which creates further opportunities for scatternet applications. Bluetooth WLANs are already in service for large scale commercial IP applications, such as conference scenarios like "Cebit 2003", where 150 Bluetooth access points were provided for piconet formation [47]. Critics may be quick to dismiss the use of Bluetooth in this context, due to superior performance of the dominant wireless LAN technology, WiFi (IEEE 802.11b). This offers a higher data rate over a much greater distance (50 m versus 10 m for Class 2 Bluetooth devices). In [20], it is pointed out that while these comparisons are technically correct, Bluetooth has a much more powerful business model associated with it, based on two key points. Firstly, Bluetooth is designed for a myriad of wireless PAN applications. Secondly, the cost and size of the technology means that it is being included in equipment by default. These points mean that opportunities will exist to further network devices, including access points, for Bluetooth applications.

Finally, the low cost, power efficient specification of the technology means that there are potential Bluetooth applications in other ad hoc scenarios such as sensor networks [1]. These networks involve a large number of densely deployed wireless sensors, which need to be low power and low cost devices. The current literature acknowledges Bluetooth as being too expensive and too power consuming for sensor network application. However, there are a number of additional points which need to also be considered. The cost of a sensor network node should be less than 1 dollar to make the network feasible [74]. Clearly, the current Bluetooth production cost of 4 dollars [96] exceeds this, but it is feasible that this cost will fall further. In [27], it is argued that Bluetooth is not currently efficient for sensor networks because turning them on and off adds to the energy consumption. However, this has to be contrasted against the additional features that a Bluetooth sensor network could provide, including higher data rates than many sensor network solutions [1], off-the-shelf specification and compatibility with any other Bluetooth enabled device. This would enable direct access to other wired and wireless services. There may be scenarios where these advantages out weigh the increase in cost and additional power consumption. If Bluetooth were adopted in this context, scatternet formation for large networks would be essential.

2. The Bluetooth specification

Although it is not the purpose of our paper to describe the BT specification in detail, we do provide an overview to set the context of research activities. The Bluetooth radio system is defined to resolve the following fundamental issues in an ad hoc communication scenario, concerning application of the radio spectrum, discovery of devices, connection establishment, channel allocation, medium access control, interference and power consumption. These aspects are the focus of our description of BT. In Fig. 1, we present the organisation of the Bluetooth stack. There are now many detailed expositions of various aspects of the BT system including books [17,62,70] and general articles [11,31,32,38], from which we provide a brief summary.

2.1. Radio layer

At the lowest level, the *radio layer* of the core specification defines the wireless interface. Spread spectrum frequency-hopping occurs in the 2.4 GHz ISM band using either 79 or 23 radio frequencies in countries with restrictions in the ISM band. A fast hopping rate of 1600 hops per second occurs, using pseudo-random hopping sequences, which

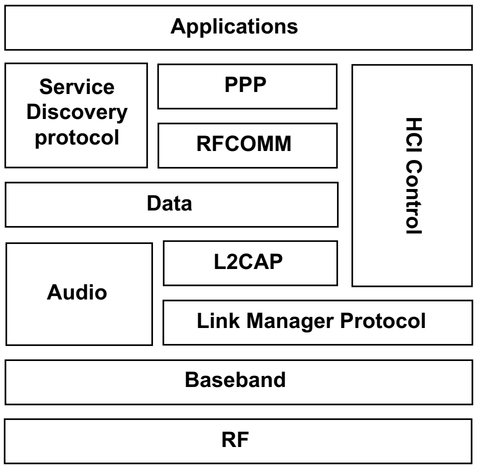

Fig. 1. Organisation of the BT stack.

provide an automatic method for controlling interference from other sources in the unregulated ISM band. These frequencies are located at $(2402 + \alpha)$ MHz, for $\alpha = 0,1,\ldots,78$. The modulation technique is Gaussian Frequency Shift Keying (GFSK) which enables a simple low cost radio chip design and a transmission speed of up to 1 Mb/s. The baud rate is 1 Msymbols per second. Three power classes are defined for transmission, 20 dBm (100 mW), 4 dBm (2.5 mW) and 0 dBm (1 mW). A service threshold of -70 dBm or better, is required. For further details of the radio layer, the reader is referred to [32], which provides a description of the system including link budget characteristics and signal-to-interference ratio information.

2.1.1. Hopping sequences

Frequency hopping provides interference diversity gain and frequency diversity gain, which are required for multiple channels to co-exist independently. The frequency hopping mechanism used by each device operates in a "black box" fashion, with a device identity and clock as input, and a frequency as output. More specifically, the sequence is selected by the unit identity, and the phase by the clock. A huge number of sequences are available to each unit, with each covering about 23 h. This prevents repetition of an interference pattern when hopping sequences support different communication channels in the same proximity. Thirty two hops are designed to span about 64 MHz of spectrum. This is designed to maximise interference immunity by spreading as much as possible over a short time interval. Over a larger time interval, each frequency in the sequence has equal chance of occurrence, and perturbation of the clock and/or identity leads to an instantaneous change in the selected hop. This permits devices to switch between different hop sequences and permits devices to time share between two or more communication channels. This requirement means that storage of sequences in memory is infeasible. Sequences must be generated on the fly using logical circuitry.

2.2. Baseband

The *baseband* specifies how the radio layer should be employed to facilitate communication between Bluetooth devices. This layer defines the concept of a piconet, which is BT's fundamental logical topology for organising group-wise communication, under decentralisation. At this point, homogeneous devices are distinguished by their *Bluetooth device address* which is a unique 48-bit address, hard-coded into the Bluetooth chip. Additionally, each device holds a free-running *clock* which ticks every half time-slot for a hopping rate of 1600 hops per second. Exchange of clocks and BT addresses is fundamental to the formation of a piconet. This is defined as a group of devices that share the same communication channel. The key innovation is that piconets are formed in decentralised ad hoc manner, without intervention or assistance. Homogeneous devices take on one of two different roles: *master* or *slave*. Each piconet consists of exactly one device whose role is the master, and at most seven other active devices whose roles are slaves. The role of master and slave are relative to one piconet, at one point in time. It is the master who defines the communication channel used by all members of the piconet. The first device to initiate the formation of a piconet becomes the master. Every other device in range is assigned a locally unique *active member address*. These take up the role of the *slave* within the master's piconet. At most seven *active* slaves participate in each piconet, but additional slaves can be registered with the master and sustain the *parked* mode. Devices outside of any piconet sustain the *stand-by* mode.

The master functions to co-ordinate the intra-piconet communication. The master may communicate with any member of the piconet, but slaves in the piconet may only communicate with the master. This means that the communication between slaves may only occur via the master. The principle role of the master is maintenance of synchronisation. This is organised by referencing slots in piconets as odd or even, according to the second least significant bit of the clock of the master. To determine the frequency hopping sequence in a piconet, slaves maintain the offset time between their clock and that of the master, using a slot dwell time of 625 μs, and apply pseudo-random sequencing of frequencies. The master uses each time slot in the sequence to communicate with a slave, before moving onto the next slave in the sequence. However, multi-slot packets requiring three or five slots are permitted. When these are transmitted, the transmit frequency remains constant. The master then resumes transmission with the slaves whose turn it would have been under single slot transmission.

Uplink and downlink between master and slave occurs using time division duplex, with the master only communicating in even numbered slots, and slaves in odd numbered slots. Furthermore, slaves may only transmit to the master if the master has just transmitted to the slave, giving the master control over the slave for purposes of controlling multiple slot transmissions. Note that the piconets transmission channel is fundamentally dependent on the clock of the master. Consequently a device cannot be a master in two piconets simultaneously, since this would result in two piconets having the same communication channel, resulting in significant co-channel interference between piconets.

2.2.1. Device admission

Prior to commencing communication in a piconet, the slaves need to identify the address and clock of the master, and similarly the master needs to know the identity of slaves. This is organised by *inquiry* and *paging* phases defined at the baseband layer. A *scan* procedure is carried out by listening devices who are idle.

The device carrying out the inquiry phase becomes the master in a future piconet. The inquiry process is used to discover other devices and paging is used to establish connection with them, via invitation. These are uni-directional processes which require the participation of both the inquired and inquirer. Each needs to commence from a different initial state to ensure discovery. The principle problem that inquiry involves is collision avoidance between receiver and sender. Two devices cannot exchange messages until they synchronise the channel (i.e., frequency hopping sequencing) for communication. Bluetooth uses the simple approach of defining a globally adopted hopping sequence known to all devices. However the inquirer hops at a greater speed than the inquired, and the inquirer listens in between transmissions for a response.

A further complication exists concerning multiple listeners. To avoid collision between responses from different inquired devices, a randomised back-off time is set by each device to stagger reply times. Eventually the sender device collects the basic information (such as device address and clock) from the listeners, which is then used in the paging process. This procedure is performed by the inquirer, and is an invitation to join the piconet. With the information sent by the paging device, the enquired device can join the piconet as a slave. After joining a piconet, the slave may negotiate role reversal, where the original master becomes the new slave and vice versa.

The scan process is used by idle devices which periodically become active and listen for devices trying to make a connection. Every time a device wakes up, a different hop carrier is scanned for a period of approximately 10 ms. The hop carrier scanned is defined by the *wake-up sequence*. This is a cyclic sequence of 32 hops, in a pseudo-random order defined uniquely for each device. Here there is a trade-off between power consumption and response time.

More complex steps exist for admitting a *new* slave into an existing piconet. There are two processes by which this may occur, either *active* or *passive*. The master may inquire on new nodes in its transmission range and invite them to join the piconet. Alternatively the master may wait to be discovered, by performing the scan phase. For both options, communication in the piconet must be suspended. Therefore a trade-off exists between piconet capacity and latency of device admission.

2.3. Medium access control

The BT specification has been developed to permit co-existence of a large number of unco-ordinated communications. This is enabled by a large number of independent channels, each of which is used by a restricted number of devices. Each channel is defined by a master device. For the modulation technique adopted, a single frequency hopping channel in the ISM supports 1 Mb/s, and theoretically, the channel with 79 carriers can support 79 Mb/s. However, the non-orthogonality of the hop sequences means that this limit is unlikely to be reached. Channel capacity must be shared between members of the piconet, and so the number of active slaves is deliberately limited to seven. When the number of piconets is large relative to the number of devices, channel capacity will be exploited.

The master/slave role within a piconet only exists for the duration of the piconet. Once disbanded, each device could potentially resume as a master or slave. As well as defining a piconet, the device with master status controls all traffic across the piconet and admission control. Slots are used alternatively for master–slave and slave–master communication. To avoid collision of multiple slave transmissions, the master adopts a polling technique, which involves the master defining which slave is to transmit in each slave-to-master slot. Only the slave addressed in a master-to-slave slot may communicate in proceeding slot. If the master has information to send a specific slave, the slave is polled implicitly and can return information. If the master has no information to send, it polls the slave with a short polling packet. As the master schedules bi-directional traffic, scheduling algorithms need to be applied to efficiently use channel capacity. Although masters operating in the same area use different channels, there is the probability of loosing a packet due to another transmission using the same carrier hop. Information is transmitted without listening for a clear carrier. If data is received incorrectly, it is resent at the next opportunity. This does not occur for voice calls. The simplicity of BT and the fast hopping speed makes collision avoidance schemes inappropriate.

As portable low power devices are envisaged for BT operation, control of power consumption is important. A number of modes of operation have been defined to minimise power consumption. In idle mode, the device performs the scan operation for less than 1% of the time. The *park* mode reduces device activity further, but can only be applied to slave devices after a piconet has been formed. In park mode, devices remain synchronised with the master but do not return a packet with a payload. Park mode permits more than seven slaves to participate in a piconet. A further low power mode is the *sniff* mode, where the slave does not scan in every master–slave time slot, but has a low duty cycle. Effectively the device only wakes up periodically to communicate with the master. Finally, the *hold* mode is used to put devices to sleep. This is used to suspend intra-piconet communication while the master searches for new devices to admit to the piconet.

2.3.1. Link types and packet format

Two types of link can be supported. A single *asynchronous connectionless link* (ACL) is supported between a master and a slave, for services such as bursty data traffic. Additionally, up to three *synchronous connection-oriented* (SCO) links may be supported in a piconet, for services such as voice traffic. The SCO link is a point-to-point link supported by reservation of duplex slots (odd and even). The ACL link is a point-to-multi-point link from the master to all slaves. An ACL link can use all of the remaining slots on the channel not used for SCO links. The slotted structure of the piconet channel means that ACL and SCO links can be mixed.

BT streams the information into packets, with a single packet being sent in each slot. Each packet consists of an access code (72 bits), packet header (54 bits) and payload (0–2745 bits). The access code is used to ensure to identify the piconet to which the packet belongs. Only if the access code matches that of the piconet master will the packet be accepted by the recipient. This prevents packets from one piconet being falsely accepted in another piconet. The packet header contains link control information, a 3 bit slave address to distinguish slaves in the piconet, a 1 bit acknowledgement for repeat request, a 4 bit packet type code to define 16 different payload types and an 8 bit header error check code used to detect errors during transmission. The header has maximum size of 18 information bits which are protected by further coding.

Four control packets are defined. These are the *identification packet*, consisting of only the access code and used for signalling; the *null packet* which contains the access code and a packet header, being sent only to convey link control information; the *poll packet*, sent by the master to force a response from the slaves; and the *synchronisation packet*, used to exchange real-time clock and identity information between devices.

Both forward error correcting and packet retransmission schemes are included in BT. For FEC, a 1/3-rate code and a 2/3-rate code are used. The 1/3-rate code is a simplistic approach where a 3-bit repeat is used with a majority decision at the recipient. This is used on the packet header and can additionally be used on the payload for SCO links. The 2/3 rate FEC code consists of a shortened Hamming code which can be applied to the payloads of either SCO or ACL links.

2.3.2. Link manager and host controller interface

The baseband state machine is principally controlled by the BT *link manager*. This firmware is provided with the link control hardware, and handles link setup, security and control. Its remit includes control of paging, changing slave modes, and handling required changes in master/slave roles. It also supervises the link and controls handling of multi-slot packets. Link managers communicate with each other using the *link management protocol* (LMP). This is organised by LMP packets which are sent in the payload of packets on asynchronous connectionless links and are flagged by a bit in the ACL header.

Some link controller hardware may include a host controller interface (HCI) layer above the link manager. This is a firmware layer which is used to separate the BT baseband and link manager from a transport protocol. The HCI defines a standard interface independent method of communicating with the firmware. Three standard transport mechanisms are supported in BT: USB, RS-232 and UART. The HCI allows a Bluetooth application to access BT hardware in the absence of the transport layer or other hardware implementation details. The HCI layer forms a component in the BT stack, but it does not constitute a peer-to-peer communication layer since the HCI command and response messages do not flow across the physical link.

2.3.3. Link layer and L2CAP

All BT protocols above the link manager and HCI are software based. The lowest level is the logical link control and adaptation protocol (L2CAP) specification which is effectively BTs link layer. The L2CAP delivers packets received at higher layers to the other end of the link. It is required because baseband packet size is too small for transporting higher layer packets. The L2CAP resolves this, and operates over an ACL link provided by the baseband. It performs segmentation, re-assembly and multiplexing of high level applications above the HCI. The L2CAP supports the multiplexing of several logical channels over the devices ACL links. A single ACL link is always available between the master and any active slave. This provides a point-to-multi-point link supporting data transfer. A detailed description of the channels, channel state machine, connection, configuration, disconnection and L2CAP packets is given in [78].

2.3.4. The service discovery protocol and profile specification

The service discovery protocol (SDP) is used to determine which BT services are available on a particular device. Each Bluetooth device can act as a client or server in discovering services with SDP. The SDP only provides information about the different services which are available. Further protocols (either from BT or elsewhere) must be used in order to use a service. In order to preserve the efficiency of the process and minimise the amount of information which needs to be carried across the BT link between devices, universally unique identifiers (UUIDs) are used. The SDP operates with each server cataloguing the all available services provided by the device, and each server operates search and browse facilities.

To support interoperability for a range of applications, profile specifications define how a fundamental range of operations should be organised. These define usage for generic access, service delivery, mobile telephony interconnection, intercom, serial ports, headsets, dial-up networking, fax services, LAN access, and usage models for generic object exchange, object push, file transfer and synchronisation. Of particular interest are the arrangements for serial port and LAN access. The former case is facilitated by adopting the RFCOMM protocol. This emulates the signals on an RS-232 interconnection cable and is based on the ESTI 07.10 standard. This permits the emulation and multiplexing of several serial ports over a single channel. The key benefit from adopting the RFCOMM is that it enables legacy applications that have been written to operate using serial cables, to run over a BT link.

2.4. Inter-piconet communication

As well as permitting piconets to co-exist independently, the BT specification makes provision for connectivity between piconets. A collection of connected piconets is called a *scatternet*. Inter-piconet communication is facilitated due to packet based communication over slotted links. This means that particular Bluetooth devices can time slice communication between the channels of multiple piconets. Recall that a device can instantaneously change hopping scheme with knowledge of master identity and master clock. In order to allow jumps to be feasible, a guard time occurs in the traffic scheduling to allow for slot misalignment between different piconets. Accordingly a *hold* mode is used to allow a unit to temporarily leave one piconet and visit another piconet. The hold mode may also be used as an additional power control mechanism. The role of the device in each piconet is flexible, with the constraint that a device cannot be a master in more than one piconet, by definition. However, it is acceptable for a device to be a master in at most one piconet and a slave in multiple other piconets. Beyond the functionality for devices to time-slice between multiple piconets, support for the creation, operation and maintenance of scatternets remains open.

3. Factors influencing scatternet protocol design

Bluetooth networks are distinguished from other networks currently associated with ubiquitous and pervasive computing in a range of ways: spontaneous network formation, isolation from infrastructure, simple low cost devices with power constraints and links with states that permit low power. The decentralised formation, maintenance and operation of fully connected networks is specified for small numbers of devices, organised via the concept of a piconet. The pre-requisite in establishing a piconet is that all devices are in range at least one device, which is required to operate as the master. This precludes obtaining fully connected networks in scenarios where

- a larger number of devices (i.e., greater than 8) require connectivity;
- not all devices requiring connectivity are within range of at least one member.

In both these scenarios, scatternets are required: that is multiple interconnected piconets. An important paper fully exposing the possibilities of inter-piconect communication is [38]. Here it is stressed that in the context of personal area networks (PANs), although a single piconet may be sufficient, the possibility for a device in the PAN to be present in multiple piconets is essential to allow the combination of various Bluetooth usage cases, examples of which are provided.

Although scatternets are supported in the BT specification, protocols are not explicitly defined for creation, maintenance and operation. The complexity of these tasks significantly increases, when moving from single piconets to multiple connected piconets. The fundamental issues to be addressed concern:

- *Device status*
  Each device needs to determine its role with respect to (possibly) multiple piconets, specifically is role(s) as master and/or slave. This

defines the links between devices and therefore the topology of the network, which influences the performance of the network. There is a large degree of freedom in the number of feasible alternative scatternets (see [12]), which defines a significant combinatorial optimisation problem. This is made more difficult by the decentralised nature of the problem, characterised by a lack of a centralised entity with global knowledge.

- *Routing*
  In a single piconet, any pair of devices are at most two hops apart. The master has total knowledge of all slaves, and controls their ability to send and receive messages. Therefore in a single piconet, packet delivery between any pair of devices becomes trivial. However, complexity is significantly increased in scatternets, since a packet may have to transverse multiple piconets to reach its destination. Additionally there may be a number of alternative routes, the knowledge of which must be organised and distributed.
- *Intra- and inter-piconet communication schedules*
  The master in a piconet is required to share time between slaves. Some slaves will become significantly more important than others, particularly if they perform a relay function between distinct piconets. Consequently schemes for scheduling become increasingly important. Additionally, devices involved in multiple piconets can only participate in one at a time. Consequently resource sharing schemes need to minimise potential conflicts.

Mechanisms to determine device status, scheduling and routing are influenced by many factors. These include *distribution of devices*, *scalability*, *device differentiation*, *environmental dynamism*, *integration between co-ordination issues* and *level of centralisation*. These factors are important because they serve as a guide in protocol design.

3.1. Distribution of devices

A commonly applied assumption in the design of protocols is that all devices are in mutual range (e.g., [4,49–51,53,56,76,80]). Under this assumption, the primary aim is to increase the number of devices which can connect and form a network. This assumption simplifies scatternet formation, because local (i.e., 1 hop) neighbourhood knowledge at each device constitutes global network knowledge. This assumption is applied, based on the observation that the current expectations for Bluetooth are focussed on personal area networking, where a maximum network diameter of 10–15 hops is envisaged.

3.2. Scalability

In the role of WPAN, network scalability is a limited concern. However, network protocols which scale will be of use if applications are developed involving a large number of nodes, such as in sensor networks, as surveyed in [1]. The scenarios which are considered range from personal area networks (such as in [80] with up to 36 devices) to the largest scenarios involving 2000 devices (e.g., [102]). The issue of scalability is best addressed by protocols which do not assume that all devices are in range. For some protocols, analytical results can be derived on time complexity and messaging complexity, relative to number of devices. In these approaches, some properties of scaling are effectively guaranteed (e.g., [49]). As noted in [30], simple protocols may have good scaling properties but at the expense of overall network quality.

3.3. Device differentiation

It is likely that the most appropriate scatternet for a given collection of devices will depend on characteristics of the device such as battery life, computational abilities and likely traffic load. In [22] for example, these factors have been incorporated into the protocol to create the network. However the predominant stance involves assuming devices are homogeneous. This follows from the specification, and the fact that any Bluetooth device can take a role as master or slave. However a number of authors make further logical hierarchies between devices, particularly for the creation of scatternets. These devices take leading roles, frequently through acquiring and using greater knowledge of other devices (e.g., [80,102]).

3.4. Environmental dynamism

As each device is autonomous in an ad hoc scenario, synchronisation between devices, in terms of device start-up and activity time, cannot be assumed. This means that scatternet membership is likely to change and the network will need to repair or heal under conditions of failure. The cases under which failure occurs, for example, are addressed in [56]. Mechanisms will need to rapidly react in particularly dynamic environments, such as those considered in [95]. On-demand techniques are one solution to environmental change. This approach is used in [57,58], for example, but operates at the cost of increased overhead. Additionally, creation of networks with high levels of fault tolerance are desirable, as sought in [95].

3.5. Integration between co-ordination issues

Each of the co-ordination issues can be addressed in isolation, or with integration. However, intra- and inter-piconet scheduling is dependent on routing, which is dependent on the network topology (i.e., collective device status), and this could potentially be exploited in protocol design and operation. Consequently, some authors have argued for significant integration via cross layer optimisation. A range of solutions and approaches can be undertaken. For example, [2] seeks to determine network topologies which maximise the number of device pairs which can operate simultaneously. The integration between scatternet structure and inter-piconet communication is addressed in [43]. The development of protocols which form particular topologies can aid routing. For example, [24] adopts ring structures only, while in [90], a tree structure is adopted with Bluetooth device addresses carefully used to aid packet navigation. In [57,58], on demand network formation is developed where routing is implicit.

3.6. Level of centralisation

A key issue in protocol design is the reliance on centralisation, where one entity (e.g., single device or subset of devices) gains total network knowledge. The benefit is that such knowledge makes higher performance solutions achievable. However, this induces potentially greater overhead, against which increases in network quality must be assessed. Greater overhead is more likely to be permissible in environments with little dynamism. In fully decentralised protocols (e.g., [98]) no device has greater knowledge than any other. In partially decentralised protocols, hierarchies exist based on the level of knowledge devices attain (e.g., [80]). The extreme case is when one node gains total total knowledge to create the network. Independent of use as a protocol, entirely centralised problem solutions have been considered (e.g., [60]) and are challenging computational problems in their own right.

4. Link formation and network topology

The fundamental step in setting up a scatternet is local device discovery and the formation of point-to-point links, where pairs of devices learn of each other's identities and synchronise hopping sequences to form piconets. In scatternet scenarios, a number of devices are required to participate in more than one piconet to facilitate network-wide connectivity. We describe the basic link establishment process provided in the Bluetooth specification.

4.1. Bluetooth link establishment

Link establishment in Bluetooth is controlled by four processes:

- *Inquiry*
  This process consists of broadcasting inquiry packets, which do not contain the senders identity or other any information.
- *Inquiry scan*
  In this state, devices listen for inquiry packets, and upon detection of an inquiry packet, the device broadcasts an inquiry response packet. This contains the devices identity and clock of the device in inquiry scan mode.
- *Page*
  Under this process, a device tries to establish a connection with a device whose identity and

clock are known. Page packets are sent, which contain the sending devices address and clock for synchronisation. The packets sent can only be received by those devices with particular identities.

- *Page scan*
  In this state, a device listens for a page packet. Receipt is acknowledged and synchronisation between the page and page scan devices is established.

These four processes need to be co-ordinated and controlled in order to establish and maintain piconets and scatternets. From the baseband specification (see Section 2.2), Bluetooth defines a simple process for link formation between a pair of devices. This process is asymmetric since the protocol distinguished the devices as a sender and receiver. The sender starts in the inquiry state while the receiver is in the inquiry scan state. Initially there is a frequency synchronisation delay until the sender transmits on the frequency on which the receiver is currently listening (recall that the senders and receivers are hopping at different rates). When the inquiry packet is received, the receiver backs off for a random time period (uniformly distributed in the range 0–639.375 ms). Hopping then resumes at the hop it was listening to, immediately prior to back-off. A second frequency synchronisation delay then occurs, and a second inquiry packet is received from the sender.

The receiver replies with a frequency hopping synchronisation (FHS) packet containing the receivers address and clock value and the receiver enters the page-scan state. Upon receipt of the FHS packet, the sender enters the page state. In the page state, the sender transmits a Device Access Code (DAC) packet, that can be uniquely heard by the receiver. The senders address contained in the FHS packet, is used to estimate the phase of the receiver, in order to eliminate the frequency synchronisation delay. The receiver responds with a DAC packet, and the sender replys with a FHS packet. The receiver then uses this information to determine the channel hopping sequence and the phase of the sender. At this point the receiver becomes the slave in the communication channel and replys with another DAC packet. As soon as the sender receives this DAC packet, the sender becomes the master in the connection, and both devices are now synchronised across the same hopping sequence.

The delay components of link formation are random back-off delay and the frequency synchronisation delay, which occurs twice. In [80], this is shown to be at most 659.375 ms when 32-hops are available for the universal frequency hopping set used in the inquiry process, and 649.375 ms when 16-hops are used. This connection establishment delay, however, depends on the pre-assignment of roles for the devices. In an ad hoc scenario, no pre-assignment of roles will be available, making this link establishment process problematic.

4.1.1. Symmetric link formation protocol

In [79,80], a protocol for symmetric link formation is developed, which is analysed in [81]. The approach is symmetric in the sense that no sender or receiver allocation is required. This is achieved by ensuring that the two nodes (nominally the sender and receiver) alternate node status between inquiry and inquiry scan states, in an unco-ordinated manner. When the devices meet for a sufficiently long time interval in opposite states, they try to connect according to the asymmetric link formation protocol.

Clearly the schedules for state alternation are crucial in determining the link formation delay. In [81], it is shown that a random schedule with state residence times following a common random distribution are preferable to deterministic solutions, following intuitive logic. The mean and variance of link delay formation for random state alternation are derived in [81].

Salonidis et al. [81] propose the scheme as a tool which can be generally used by protocols for scatternet formation "on the fly", which are built above this. Such an example is [102], where the protocols are developed on the assumption that devices know the identifiers of their one-hop neighbours. The approach introduced in [80] is well used, with this and similar mechanisms being adopted by a range of authors including [7,9,10,53,72,76,83,94]. The symmetric protocol for link formation is a suitable mechanism for device discovery of single-hop neighbours, when a temporary piconet is formed specifically for this purpose. The general issue of device discovery in a multi-hop rather than single-hop scenario has been addressed in [6].

4.2. Network topology

Scatternet formation protocols are mainly required to control the states of inquiry, inquiry scan, page and page scan at each device, so that groups of devices synchronise on common hopping sequences, to form interconnected piconets. This determines the device status as a master or slave in possibly multiple piconets. Additionally, the protocol is required to account for the dynamic nature of the ad hoc environment, where scatternet membership is variable, and device residence in the scatternet is unpredictable.

The role that devices take in the scatternet determines the network topology and consequently the performance of the network. For example, the number of piconets a device belongs to, particularly in situations where are many potential master nodes, has significant implications on characteristics of the network. Of crucial importance are *bridges*, which are devices that reside in more than one piconet. There are two types of bridge. If the bridge is a master in one piconet, the bridge is of a *master–slave* type. Otherwise the bridge is a slave in all piconets to which it belongs and is a *slave–slave* type.

Bridge location and distribution is important since participating multiple piconets leads to increases in processing and communication overhead. Note that bridges can only be active is one piconet at a time. Switching between piconets can reduce throughput, and increases the demands on inter-piconet scheduling. However, traded against this, bridge devices improve connectivity and thereby reduce path length between devices which can improve throughput. An example of a possible scatternet topology with a variety of possible bridges is displayed in Fig. 2. The protocols introduced for scatternet formation frequently use heuristics to control possible configurations which can be formed, including:

- Bridge devices must never be masters. This reduces the scheduling burden on masters, who then only need to consider intra-piconet communication.
- The number of bridges per piconet is restricted. This restricts the number of potential inter-piconet conflicts for bridging devices, but at the expense of limited potential alternative routes.
- The number of piconets must be kept as small as possible. This reduces the number of communication channels used, and the potential for interference.

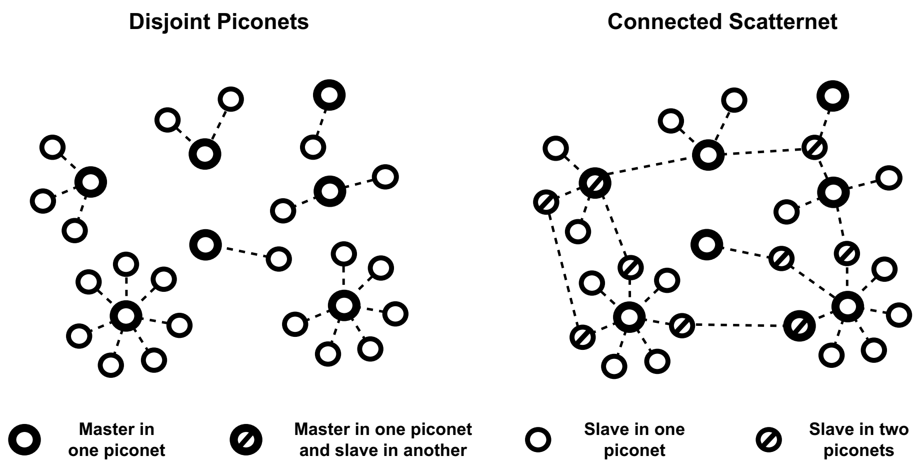

Fig. 2. Examples of piconet and scatternet topologies.

Table 1
A summary of topological features imposed in scatternet formation

Paper	Bridge devices must never be masters	Number of bridges per piconet restricted	Number of piconets must be kept as small as possible	Piconets must not be linked by more than one bridge	A device must participate in a limited number of piconets
Barriere et al. [5]	•				•
Law et al. [51]			•		•
Lin et al. [56]		•	•	•	•
Petrioli et al. [71]					•
Ramachandran et al. [76]	•				
Sato et al. [83]				•	•
Salonidis [80]				•	•
Stojmenovic et al. [89]					•
Wange et al. [98]				•	
Zaruba et al. [102]					•
Zhang et al. [103]	•	•			•
Zhen et al. [104]					•

- Piconets must not be linked by more than one bridge. This minimises the co-ordination issues which need to be addressed by scheduling.
- A device must participate in a limited number of piconets. This controls the amount of inter-piconet scheduling at the device level.

The way in which these assumptions have been applied are summarised in Table 1.

4.3. Modeling

In order to represent the Bluetooth system for scatternet formation, a model is required. This abstracts the problem to some extent, and frequently involves graph theory, where devices represent the vertices and links between devices represent edges. A range of graph theoretic structures can be used in the modeling process so that the resultant scatternets have topologies with particular characteristics.

Nodes may be *labelled* according to their particular role in the scatternet. Indeed, this is a central task in formation of a scatternet, and is a pre-requisite for routing and scheduling. In addition to being labelled as a master or slave with respect to a single piconet, a node may be labelled as a bridge (master–slave or slave–slave). For slave nodes, the degree of a vertex determines the number of piconets to which it belongs. The degree of a master determines the size of the piconet it defines. Control of degree has been a central issue in many scatternet formation models and algorithms, particularly in [7,25,30,53,56,60,71,80,89].

Protocols for scatternet formation frequently use the concept of a *visibility graph*. This represents the graph of all possible links which *could* be formed between devices in range. In this context, the protocols for scatternet formation must label nodes and select edges from the visibility graph.

4.4. Classes of topology

Beyond applying particular characteristics in Table 2, protocols can be designed to create particular network topologies, based on a graph theoretic structure. In this section we review the frequently used structures.

4.4.1. Trees

A tree is a connected graph with no cycles. This means that a Bluetooth network which forms a tree has a minimal number of edges for connectivity. However this also means that there is no choice of route for traffic between nodes: only a single path exists. Additionally this topology is susceptible to partitions when links and devices fail.

In [43], a two level hierarchical tree structure is formed, composed of a *root*, followed by *leaves*. The root is the initiating piconet, and the slaves

Table 2
A summary of scatternet formation and maintenance protocols

Proposer	Centralised/ global coordination	Distributed/ local coordination	Radio range	Fixed positions	Two hop/ gateway	Multi-hop/ intermediate gateway	Geometric approach	Graph-based approach	Ring structure	Tree structure	Hierarchical structure	Mesh structure	Limited number of nodes	Minimise/ limit number of piconet	Power management	Weighting of nodes	Optimise capacity	Optimise throughput	Dynamic/ maintenance	Max–min optimisation
Barriere et al. [5]		•	•		•														•	
Baatz et al. [2]		•	•			•		•									•			•
Basagni and Petrioli [9,72]		•				•										•				
Chiasserini et al. [22]	•	•			•										•		•	•	•	•
Chun et al. [24]		•	•						•											
Cuomo et al. [25]		•			•												•	•		
Law et al. [51]		•	•		•									•					•	
Li et al. [54]	•	•				•	•	•							•					
Li et al. [53]	•	•					•	•												
Lilakiatsakun et al. [55]				•	•						•									
Liu et al. [57]		•				•														
Lin et al. [56]		•							•										•	
Marsan et al. [60]	•				•										•		•	•		•
Petrioli et al. [71]		•			•	•						•				•				
Sato et al. [83]		•																	•	
Salonidis et al. [80]		•	•		•	•							•							
Stojmenovic et al. [89]		•						•											•	
Sun et al. [90]		•								•	•									
Tan et al. [92,94,95]		•			•					•									•	
Wang et al. [98]		•			•															
Zaruba et al. [102]	•	•			•	•				•										
Zhen et al. [104]		•				•									•	•				

in this piconet then form bridges (and masters) in the new piconet. The authors find that this approach has lower throughput than a non-tree alternative topology where bridges are only master nodes, but performs better in terms of average system delays.

Routing considerations for tree based structures have been considered in [90]. Routing in trees is simplified because there is a unique path between source and destination devices. This means that the routing considerations need only address dissemination of this particular route. Sun et al. [90] propose a protocol to create and maintain routing table information, through extending the idea of a binary tree search to sub-trees in the network. Protocols are also introduced to create such tree structures.

In [94] the use of a tree topology is justified by simplified routing, and the authors explicitly mention the tension between connectivity and latency. It is argued that minimising the total number of links in the topology reduces the contention for transmission slots in the time division duplex scheme. The protocol proposed achieves a minimum number of average piconets per bridge by ensuring that bridges participate in exactly two nodes. In [95], the tree formation algorithm is evaluated in dynamic conditions.

A protocol to form so-called Bluetrees is proposed in [102]. This operates using two phases, in the first of which sub-trees are formed using particular nodes. The second phase concerns spanning these sub-trees. The paper makes a number of interesting observations concerning tree structures and Bluetooth. In particular, in an open, interference and obstacle-free environment, if a node n has more than five neighbours then there are at least two nodes among the neighbours that are themselves neighbours. Additionally, it is noted that most distributed algorithms (and consequently protocols) for finding a spanning tree in a network operate by creating subtrees and then expanding them to form a single spanning tree.

The advantages of the tree structure are based in simplicity. A minimal number of links leads to simple routing issues. However, this structure can only be applied in dynamic environments if the protocol can quickly respond to link failures, which in the case of a tree, are certain to partition the network.

4.4.2. Gabriel and Yao graphs

These structures are closely related to the concept of a *unit disk graph* (UDG), in which nodes in the Euclidean plane are considered to define a disk of unit radius. In a UDG, an edge is defined between a pair of vertices if their Euclidean distance is at most one. A *constrained Gabriel graph* over a graph G contains an edge uv if and only if:

- $uv \in G$;
- the open disk when using uv as diameter does not contain any node w from G such that both uw and wv are in G.

Consequently a Gabriel graph produced from a graph G has vertices with a sparse geographical dispersion. The *Yao graph* from a graph G selects edges based on the proximity of associated nodes. A Yao graph with an positive integer parameter k is defined as follows. At each node u, any k equally separated rays originate and define k separate cones. In each cone, a shortest directed edge is selected.

The Yao and Gabriel graphs are used in [54] to construct and analyse general network topologies for wireless communication. One useful point is that these graphs can easily be constructed in a decentralised manner. The investigation considers the power efficiency of the shortest paths between source and destination, and the tolerance of node movement using a power stretch factor. In [53], the approach is extended specifically for scatternet formation, and the authors propose a new sparse geometric structure for this purpose.

In [89], Gabriel and Yao graph structures also feature in a dominating set based approach for formation and maintenance of scatternets. A dominating set in a graph G is a subset of nodes who, collectively, are adjacent with all other nodes in the graph. Consequently these sets are useful for purposes of connectivity.

4.4.3. 1-Factors

In [2], the authors seek to construct scatternet topologies to ease the problem of inter-piconet scheduling. A matching in a graph G is a subset of edges such that no two are incident. A perfect matching has $n/2$ edges, where n is the number of vertices. When n is odd, a near perfect matching describes a matching with $(l-1)/2$ edges. Note that (near-) perfect matchings are best possible. A graph which can be partitioned into (near-) perfect matchings is *1-factorizable*, as these matchings are also known as (near-) 1-factors.

It is pointed out that a matching determines a possible communication pattern in the scatternet. This is because each edge represents a communicating master–slave pair, so the size of a matching (i.e., number of edges) determines the number of simultaneously active piconets. Consequently (near-) 1-factors are desirable. Baatz et al. [2] briefly describe how to build scatternets from 1-factors of complete graphs, which occur when all devices are in transmission range.

4.4.4. Ring structures

In contrast to purposely avoiding topologies containing cycles (i.e., Section 4.4.1) some scatternet topologies advocate their use, principally due to increasing reliability. In [24], ring structure scatternets called *Bluerings* are proposed, consisting of n devices where each device belongs to two piconets and has two links in total. Each device is both a master and a slave, and a BlueRing supports a maximum of $\lfloor \frac{n}{2} \rfloor$ active links. This leads to a structure where there are exactly two mutually disjoint paths between source and destination for all pairs of devices. Since there is a minimal number of devices for a master to switch between, the authors point out that switching overheads will be limited. Additionally, routing is very simple since incoming packets are merely forwarded. However, the authors also point out two disadvantages of the ring structure. Firstly large diameter rings could have high packet delay, and secondly, not all visibility graphs will have the necessary 2-connectivity to create a BlueRing.

In [56], the use of more general ring structures is explored, via the BlueRing protocol (unrelated to that in [24]). In this approach, relatively small numbers of piconets are sought, which are interconnected to form a ring. A set of principles guide the protocol: specifically, nodes participate in no more than two piconets and two piconets are inter-connected by at most one bridge. Additionally, bridges are never masters, and piconets are required to contain at least two slaves. This is because the protocol requires that distinct slaves are used to communicate with the adjacent piconets in the ring.

5. Scatternet formation and maintenance protocols

Protocols for scatternet formation are regarded by many authors as finite state machines, controlling states including inquiry, inquiry scan, page and page scan. The protocol ideally needs to function independently at each device, without any synchronisation between devices (e.g., [30]). But this makes it difficult to control the network produced, or guarantee particular properties of the network, such as end-to-end connectivity. A compromise to this approach are staged or phased protocols, where all devices perform activities with a loose degree of synchronisation. For example, three common phases are device discovery, piconet formation, and subsequent interconnection of piconets. Time intervals are defined in which each device must perform each phase (such as neighbourhood discovery), but the devices are unsynchronised within phases. This loose degree of synchronisation makes it possible to prove properties of the resultant network (e.g., [72]). Phased protocols for formation frequently assume *en-masse* device start-up to establish these properties, since it is difficult to establish global results on network characteristics otherwise.

In this section, we consider the principle contributions concerning protocol development for scatternet formation. We describe how the protocols operate, and the rationale behind them.

5.1. BlueStars and BlueMesh

In [72] the concept of *BlueStars* is introduced. This is a three-phase protocol which operates by discovering topology, creating piconets and then configuring these to form a scatternet. Piconets are referred to as BlueStars and the connected scatternet is termed a *BlueConstellation*. Topology discovery is dependent on the symmetric knowledge of neighbouring devices. This is explicitly engineered using temporary piconets. Weights of neighbours are built up when device identities are exchanged.

In the second phase, individual piconets are formed using a dynamically computed weight which expresses the suitability of a node for the master role. This leads to the selection of the master and slaves in disjoint piconets. Nodes are selected when they have the biggest weight in their own neighbourhood. Once a node has decided on the role of master, it pages neighbours to become slaves. The receiving node will join its first biggest neighbour who provides an invitation.

Finally, in the last phase, gateway devices (i.e., bridges) are selected for interconnection of piconets. This uses information gathered during the formation of piconets. Each master determines its neighbouring (i.e., 2-hop or 3-hop) master devices. A master is designated as an *init* master if it has the highest weight among all its neighbouring masters. The *init* masters control the scatternet formation, selecting bridges with the largest weight among alternatives. In the case of three hop neighbours, pairs of intermediate bridges with the highest weight sum are chosen. Proof that the properties of the protocol are fully specified, and analysis of this approach is given in [7].

The approach is novel in its criteria for device self-selection: a device selects itself given knowledge of its own weight, and that of its neighbours. Weights are built up at the device discovery stage and represent the degree of a vertex in the visibility graph. The resulting scatternet has a mesh structure in which there are multiple paths between source and destination pairs. Additionally the protocol does not require devices to be in each others range. It is assumed that the location of devices remains static throughout, and maintenance of the scatternet is not considered. An important point is made concerning the considerable device discovery time, which is caused by three factors: (1) the need to adopt stochastic mechanisms so that neighbouring pairs of devices can each discover each other using uni-directional inquiry functions; (2) the impossibility of identifying the inquirer leads to the construction of temporary piconets between neighbours which have already discovered each other; (3) lengthy back-off times as stipulated in the BT specification. Simulation results in [7] show that significant improvements are possible after reducing the back-off time.

In BlueStars, the number of slaves is not bounded and the consequence of this is that some slaves may need to be parked, resulting in delays. This is resolved by a further protocol [71] called *BlueMesh*, which produces scatternets with a number of interesting properties. The protocol provides multiple paths, the lengths of which are reported to perform well in comparison with that best possible in the visibility graph. Additionally masters have at most seven slaves, and on average, each node does not assume more than 2.3 roles.

BlueMesh operates in two phases, the first of which involves discovery of one and two hop neighbours. The second phase is an iterative procedure which consists of role selection followed by gateway selection. As with BlueStars, weights are applied to determine which nodes are to be masters, and the slaves in a piconet are selected as a dominating set over the masters two hop neighbours. Numerous properties of the BlueMesh protocol are proven in [71].

5.2. Scatternets via node insertion and removal

In [22], formation of scatternets is considered by defining procedures to handle topology changes in a Bluetooth network. These are based on the insertion and removal of nodes, and the cases when these operations need to be performed. The aim of these procedures is to provide feasible scatternets which have desirable properties such as full connectivity, high throughput, and in particular, reduced overheads due to control messages. The method can also be used both to form and maintain scatternets and has a high level of flexibility in this regard.

For node insertion, a node wishing to join starts the process by broadcasting identity packets. Nodes in proximity, which are listening, respond with a FHS (frequency hop synchronisation) packet if they are willing to accept the new neighbour. Possibly more than one neighbour may reply, in which case a decision may be required for connection choice. The authors of [22] propose criteria to make this decision, using classification of devices and their status. A generic node class is defined by a triple (x,y,z) where x is an identifier for computational capabilities and battery capacity, y records the devices role (e.g., unspecified, slave, master etc.) and z indicates traffic load of the node. These triples are passed between devices using short encoding. The inquiring device then selects its neighbour(s) for connection. The inquiring device is given an ordered range of alternative nodes to page, choosing the highest ranking alternative which is permissible. The criteria for permission are defined using the triple (x,y,z).

For node removal, the changes in network topology caused by the nodes absence depend on the role it played. Four different possibilities are considered based on the role of the device. The most challenging scenarios are when bridges and/or masters stop participating. The removal procedures in these cases are structured so that alternatives are sought using variations on insertion procedures. The rules provided by the protocol ensures that no negotiation or collective decision making between subsets of nodes, is required.

5.3. BlueRings

The BlueRing protocol designed in [24] is designed to create a circular chain of devices, each with bridge status. Two alternative algorithms, NODE-ID and HEAD-SEEK-SCAN, are introduced to build up the required topology, principally controlled by identifying and updating the head and tail of chains of nodes, as they are formed. The principal difficulty in forming the required ring structure is premature formation of a ring consisting of only some of the nodes in the network. The algorithms proposed overcome this by first forming a linear structure consisting of all nodes in the network, and then closing the ring by forming the last link only after a time out period.

Both of the proposed algorithms operate iteratively for a fixed time in each round, using two procedures, SEEK and SCAN. SEEK is the authors notation for the inquiry procedure. The former process denotes a device actively attempting to discover other neighbouring devices. A seeker sends out Bluetooth inquiry messages in the hope of getting a response from other devices. SCAN is the process of listening, and replies with an inquiry message when it detects a seeker. The SCAN and SEEK mechanisms emulate the processes presented in [49].

The NODE-ID algorithm operates with isolated nodes having equal chance of performing SEEK or SCAN. At the same time, the head of any line of nodes performs SCAN and the tail performs SEEK. All intermediate nodes in the chain remain idle. Criteria are defined such that the BT identity of the head of any chain is always the largest identifier in that component. This is enforced by ensuring a seeker component connects to a scanner component if and only if the identity of the seeker or head of the seeker component is larger than the identifier of the scanner. The algorithm terminates when the seeker tail of a connected component gets the first response from the scanner head for a consecutive number of iterations.

In the HEAD-SEEK-SCAN algorithm, only the head of a line and isolated nodes perform SEEK or SCAN operations. This means that only one device in each connected component performs scatternet formation in each round. This prevents premature cycle formation in each iteration of the algorithm. Unlike the NODE-ID algorithm, device identifiers do not play a role in operation of the algorithm. At each iteration, the isolated nodes and the head of chains perform SEEK or SCAN operations, with equal probability. These devices maintain two variables, *componentLength* and *tailInfo* which respectively detail the number of devices in the component and the identifier of the tail of the component. When a seeker obtains its first response from a scanner, the seeker pages the scanner to form a temporary connection, to enable the seeker and the scanner to exchange *componentLength* and *tailInfo* variables. The temporary connection becomes permanent if one or more of the seeker or scanner are isolated nodes. Otherwise the tail of the shorter component is broken and the tail of the shorter component forms a link with the head of the other component. The algorithm terminates when the head of a line receives no response after a set number of iterations.

For both algorithms, it is assumed that all devices are in mutual range, and that the processes are synchronised. It is shown that the HEAD-SEEK-SCAN algorithm outperforms the NODE-ID algorithm, and is comparable to the approach proposed in [49]. The main advantage of the BlueRing approach is that it provides a degree of routing reliability (i.e., an alternative path) if a node or link fails. Routing is simplified by packets being forwarded around the ring. However there are disadvantages. Firstly, although it may be possible to have a single operational piconet, the location of devices can prohibit the formation of a ring. Only in cases where the visibility graph has 2-connectivity can BlueRing formation take place. Secondly, network diameter will be necessarily high, leading potential packet delays. A further observation concerns the number of piconets used. Contradictory to many other approaches which seek to minimise the number of piconets (and consequently the number of communication channels and bridges), the ring approach uses a maximal number of piconets, assuming devices are restricted to piconets only.

5.4. Distributed scatternet formation procedure

In [25], a distributed approach is defined in the *distributed scatternet formation procedure* (DSFP), which forms a scatternet incrementally and sequentially. Devices are 'inspected' by the DSFP, in sequence. Each device takes the decision on whether to join the current network, based on the decisions made by devices which have already joined. For a node to make a decision, at least one other node in its neighbourhood must already have joined the DSFP, and is therefore part of the scatternet. The main steps in the procedure are: (1) neighbourhood discovery; (2) organisation of devices into sequential order; (3) inclusion of devices in the scatternet, based on this order, deciding which connections to establish with those already active, and which role to assume, with the aim of optimising a target metric.

The approach uses adjacency matrices for checking the performance of network configurations, and a range of different performance measures are considered. The approach is inspired by centralised approaches for network configuration, and specific details of the distributed algorithm are not fully described due to the short paper length of [25].

5.5. Simple and lightweight scatternet formation

In [30], a process is introduced based on randomisation. The approach is intentionally simple and lightweight, with each device using only neighbourhood information for decision making. This minimises overhead, increasing the protocols ability to rapidly adapt to changes. The approach is used to gain a better understanding of when more sophisticated approaches are required.

The protocol operates using a range of cases. Synchronisation is defined as a cycle of inquiry and scan between a pair of nodes. All nodes have one of four states (unassigned, master, slave, bridge) and initially all are unassigned.

When two nodes synchronise for the first time, and both are unassigned, the one with the highest degree becomes master, and the other becomes a slave in the piconet defined by the highest degree node. When two nodes synchronise and one is unassigned and one is a master, the unassigned node joins the piconet of the master if it has less than seven slaves. When two nodes synchronise and one is unassigned and the other is a slave, the unassigned node becomes the master of a new piconet, and the other node joins the piconet as a slave unless it is already a bridge in b piconets. Finally, if two nodes discover each other and neither is unassigned, then a range of cases are considered. If the both are masters then neither changes state. If one is a master and the other is a slave in a different piconet, then the slave becomes a bridge unless it is already a bridge in b piconets. Optionally the master may refuse the new slave if it already has a bridge to the slave's piconet.

This approach represents a simple way of creating and interconnecting piconets. No phased approach is required and all devices need not be in range. The protocol can also be easily extended to permit device removal. However properties of the scatternet (such as connectivity) cannot be guaranteed and are explored empirically in [30].

5.6. Scatternets via merging, moving and migration

In [49–51] various aspects of a new scatternet formation protocol are introduced. The protocol is designed to operate on isolated but in range devices, and operates under synchronisation. The algorithm operates with devices being partitioned into components, with a component constituting a set of interconnected devices. A component can be a single device, a piconet or scatternet. In each component there is one device which is *leader*. There is no freedom in choice of leader for the single device case. For a piconet, a master is leader. For a scatternet, one master is leader. Leaders perform a centralised role over the partition, and may come in and out of retirement as required.

All leaders iteratively perform a MAIN procedure in synchronisation, and initially all devices are leaders. In the MAIN procedure, a leader calls SEEK (i.e., inquiry) on a probabilistic basis ($1/3 < p < 2/3$). Otherwise the leader activates SCAN. This means that each leader has more chance of running SCAN than SEEK. Also during each round, the leader structure means that only one device in each component is running SEEK or SCAN.

When a leader executes SEEK, it tries to acquire a new slave which is running SCAN. However, the probabilistic nature of the protocol means that this is not always successful. Therefore, if a leader is unable to contact a slave after a certain time, the leader gives up and tries again with the MAIN procedure in the next round. In each round, a graph theoretic matching may be found between the SEEK devices and SCAN devices. The time required in SEEK and SCAN modes is investigated in [49].

When a leader u running SEEK connects to a slave v running SCAN, a procedure CONNECTED is invoked. The new link leads to a larger connected component, and reorganisation of the piconets and leader occurs via three operations: MOVE, MERGE and MIGRATE. These operations involve ensuring that new, larger connected components have only one leader, and involve the redistribution of slaves to masters without leader status, as far as possible. This requires the assumption that every pair of devices are within range. However in [50], extensions are proposed which do not require this assumption, and operations are defined which permit the protocol to be applied in dynamic environments. The protocol is shown to have $O(\log n)$ time complexity and $O(n)$ message complexity.

5.7. Partial Delaunay triangulation scatternet formation

In [53], a decentralised geometric algorithm is introduced which creates a degree-limited scatternet. The approach seeks to consider the geographical dispersion of devices and select links so that the resultant topology has a planar property. Unusually, the protocol requires knowledge of the geographic location of devices, and GPS is posed as a solution to this problem. Other authors (e.g., [7]) have criticised this assumption as being unrealistic. The protocol is synchronous in its operation and ensures that no piconet has more than seven slaves. The approach is conceptually complex, involving a series of phases in which the topology is built up by selection of links subject to graph topology. We briefly describe these.

Initially the neighbourhood discovery phase renders each node with knowledge of potential neighbours within transmission range. The next optional phase involves partial Delaunay triangulation. This is a new class of graph topology, which is planar and more dense than other common subgraphs. In the next phase, the degree of each node is limited to seven, by applying the Yao structure, and master–slave relationships are formed in subgraphs. Subsequently a clustering based approach follows, consisting of several iterations. In each iteration, nodes with unallocated role having higher keys than any of their unallocated-role neighbours apply the Yao structure to bound the degree, decide master–slave relations and inform all neighbours about either deleting an edge or their master–slave decision. Two ways of defining the master–slave relation are considered: node with initially higher degree becomes master or a cluster based approach. In the cluster-based approach, a dominating set of masters in the degree limited subgraph is implicitly constructed, and some gateway piconets are added to preserve connectivity. It is mentioned that creation and maintenance require small overhead.

In [89], further geometric approaches are proposed for formation and maintenance based on the concept of the dominating set. Geometric solutions to scatternet formation can potentially lead to higher quality scatternets because they use more information on the distribution of the devices. However, gaining location information is potentially expensive and out of context with the mission of Bluetooth as a low cost technology. Realistically, the cost of location information establishment will need to fall substantially before geometric approaches could be applied to scatternet formation.

5.8. BlueRing of trees

Although of the same name as the protocol introduced in [24], the BlueRing protocol introduced in [56] has an entirely different format. In [24], all devices in the scatternet were contained in the ring topology. In [56], the protocol involves a ring of masters and slaves, and can be viewed as a ring of piconets. The structure of this topology makes possible easy schemes for uni-cast and broadcast communication. These schemes are stateless in the sense that no routing information needs to be defined and recorded. Single point and multi-point failures are tolerated by the protocol. It is assumed that all devices are in range of each other for start up of the protocol.

A centralised formation mechanism, similar to that in [80], is proposed, which involves the identification of leaders, which control the setup of the scatternet. A binary parameter, RING-MEM, is maintained by each device, to indicate whether it has become a member of the BlueRing. Construction involves two phases. In stage one, each device chooses to operate inquiry with a probability p and perform inquiry scan with probability $1 - p$. When inquiry meets inquiry scan, the two devices set up a temporary piconet to exchange RING-MEM and device address information. A so-called *winning* device is established at this stage. The device with RING-MEM = 1 wins when the other device has RING-MEM = 0. In the case of a tie, the device which holds the most information on other device identities wins. The loser also provides the winner with all its information on other BT device identities. Subsequently the temporary piconet is removed. The winner becomes a *leader* if no further inquiry or inquiry scan message is received within an inquiry timeout period. The leaders enter a page stage, trying to collect other non-leaders, which forms stage two of the procedure.

In stage two, the leader designates several devices as masters by paging them and setting up a temporary piconet. The process for achieving this is not fully detailed in [56]. For each designated master, the leader provides its information of its slaves, including downstream and upstream bridges. The mechanism for selecting the bridges is not described. Criteria for maintaining a formed scatternet are detailed, based on two scenarios: single-point failure and multi-point failure. The latter case is more demanding as it could require reformation of the ring structure if bridges are missing or masters are missing. Case-by-case analysis is given on how to resolve all possible failures. This protocol is specifically designed to integrate the issues of formation, maintenance and routing, but may be limited by the distribution of the devices.

5.9. Scatternet formation via clustering

In [76], scatternet formation is approached from the problem of clustering in ad hoc networks. Two approaches to clustering are proposed: a randomised distributed linear complexity algorithm which constructs a minimal set of star-shaped clusters and a deterministic algorithm. The algorithms are developed generally and then applied to Bluetooth. The assumption is that all devices are within mutual transmission range. The goal of the algorithms is to maximise the number of nodes in each cluster so that the number of piconets is minimised.

The randomised algorithm operates in two phases, the first of which results in devices being provisionally designated as master or slave. The second phase corrects the effect of randomness introduced previously, by using a deterministic algorithm to decide on the final set of masters and slaves. A super-master is elected which collects information about all the nodes. The super-master can then run a centralised algorithm to form a network of desired topology between piconets resulting from the clusters. Each node conducts t rounds of Bernoulli trials with a defined probability of success p. A node which is successful at least once becomes a master designate, otherwise the node is a slave designate. In the second phase, the designated masters select designated slaves which participate in electing the super-master node. A large amount of information is passed between devices during the election process which implicitly builds up the knowledge of the network.

In the deterministic algorithm, the initial phase uses the idea that nodes discovering each other form a tree of responses, the root of the tree being a master, who collects information about other devices which connect to the component. Criteria are formulated which ensure that the component necessarily has a tree structure. Subsequently the second phase of the randomised algorithm is repeated to elect the super-master, who again is given this label due to having centralised knowledge of the topology. The authors do not propose particular methods for inter-connection of the piconets given the knowledge of the master. Throughout, the algorithms operate using careful control of the inquiry, inquiry-scan, page and page-scan modes. Both algorithms are synchronous in the sense that device failure is not permitted, and new devices will not be connected if they become active after the initial phase. The authors conclude that the randomised algorithm outperforms the deterministic approach.

5.10. Scatternet formation and extension for maintenance

A protocol is introduced in [83] which performs three separate processes: neighbour discovery, formation and adjustment for changing location of a fixed number of devices. Neighbourhood discovery is performed by setting up temporary links which are removed once identities are exchanged. This process continues for a fixed amount of time and then all nodes start the formation phase. Each device sets a random timer to start this phase, to prevent competition by all devices trying to construct piconets simultaneously. A device remains in page scan state until the timer is expired. When expired, the device enters the page state, and tries to construct a piconet as master. The device pages the devices previously discovered, and contacts them one-by-one. If the paging is successful, the master–slave relationship is established, and the slave cancels its formation timer and does not try to construct a piconet. A master continues paging neighbours until it has seven slaves, or there are no neighbours to page. In the process of piconet formation, it is ensured that no piconets share more than one device and no device belongs to more than two piconets.

In subsequent phases, the aim is to increase the connectivity by making connections between disconnected but local piconets, using neighbourhood information held by slaves from the neighbourhood discovery phase. Again, a random back-off is used to stagger the each piconets activities. When activated, the master identifies slaves that belong to an other piconet, and the master obtains the addresses of devices in the adjoining piconet(s). Then the master identifies all neighbours of slaves which are not connected to the adjoining piconet(s). The master selects the minimum number of slaves covering the disconnected devices, and instructs those devices to go into piconet formation phase. This results in a bridge between disconnected components.

Finally, a scatternet adjustment protocol is defined to deal with device movement within the scatternet. This involves the slaves belonging to exactly one piconet periodically going into inquiry-scan state, and disconnected nodes which were formally masters maintaining inquiry state to form a piconet, and former slaves perform the inquiry scan state. This means that the network will maintain a fixed number of masters. It is not assumed that all devices are in mutual range of each other. The scatternet adjustment protocol is shown to maintain a degree of network connectivity.

5.11. Bluetooth topology construction protocol

In [80], a symmetric protocol is defined, without the need for sender or receiver role to be predefined. This is achieved by forcing the two nodes to alternate independently between the inquiry state and the inquiry-scan state, and when there is sufficient time overlap between opposing states, the connection will be established between devices. The schedules for alternation are explored in [81], where analytical calculation of the mean and variance for link formation delay are presented when state residence times follow a common random distribution.

This symmetric link establishment procedure is extended to a distributed scatternet formation protocol. The protocol seeks to collect information about all nodes in the network prior to forming the scatternet. The aim is to ensure that scatternets are formed which are appropriate to the devices which participate. The *Bluetooth topology construction protocol* (BTCP) consists of three phases: initially a leadership election process which is chosen to permit asynchronous operation of devices in the network; role determination is then performed and finally actual connection establishment is performed.

In the co-ordinator election, an asynchronous, distributed election of a co-ordinator node occurs. This node will ultimately hold centralised information on the count, identities and clocks of all network devices. The process requires all nodes to have a variable recording the number of votes, initialised as 1. Each node then alternates between inquiry and inquiry-scan states. When a node pair discover each other, they compare votes. The device with the largest number of votes in the pair is classed as the *winner*. In the case of a tie the device with the largest address is the winner. The loser device sends to the winner, all the device FHS packets of the nodes it has won so far. The loser then terminates the connection to the winner in the pair and enters the page-scan state. This means that it will not be able to hear inquiry messages and will only page messages from nodes that page it in the future, thereby removing the loosing device from the election process, and preparing it for subsequent protocol phases. At this point the winning device has increased its votes by the number of votes carried by the looser, and it continues the election process by resuming the alternation between inquiry and inquiry scan. The process continues until one device remains as winner, and all others are in page-scan state, awaiting to be paged by a node with information about them.

The second phase concerns role determination. The unique remaining winner acts as co-ordinator and has the FHS packets of all nodes. Initially the co-ordinator checks if more than one piconet is required. If not the case, the co-ordinator becomes master and connections with all other nodes are made, who become slaves. If more than one piconet is required, the co-ordinator must decide on the role that each device must perform in the final scatternet. In [81], for n devices, it is shown that for the scatternet to be fully connected, the minimum number of masters p satisfies

$$p = \left\lceil \frac{17 - \sqrt{289 - 8n}}{2} \right\rceil, \quad 1 \leqslant n \leqslant 36.$$

Note that number of devices is fixed at a maximum of 36. The co-ordinator selects itself and $p - 1$ other devices as masters. $[p(p-1)]/2$ other nodes are defined as the scatternet bridges, and the remaining devices are distributed equally among the masters for slave-only designation. A temporary piconet is created between all masters and the co-ordinator to distribute the allocation of slaves per piconet. Finally in the third phase, actual connection establishment occurs.

The election mechanism is a novel approach to creating and identifying devices with superior network topology knowledge. Throughout, the underlying assumption is that all devices are within transmission range of each other. Also, setting up the temporary piconet between masters in the second phase caps the number of piconets which are permissible in the final scatternet. The formation algorithm assumes en-masse device start up.

5.12. Self-routing Bluetree scatternet formation

In [90], a tree-shaped scatternet formation protocol is introduced which seeks to minimise the number of piconets involved, and facilitate simple routing. The protocol operates using only the ordering of device addresses and a number of operations are defined based on this information. An important concept in the protocol is the definition of *range*. For a given device *i* in a tree, this is a pair containing the smallest and largest device addresses rooted by *i*. The tree is structured so that a device *i* has range (x_i,y_i), then any device with identity greater than x_i and less than y_i must be routed at *i*. This permits easy navigation around the tree.

Under this protocol, all devices, independent of their status or connectivity, infinitely cycle through a range of modes which perform inquiry, inquiry scan, page, page scan and connection. These are incorporated with a range of bespoke operations to join, update range and prune connected components as they grow. However in order to maintain the range property, a lock mechanism is used which permits only one connected component at a time, to be admitted to any particular tree. A feature of the protocol is that when two connected components join, disconnection (i.e., pruning) of nodes may be required to ensure that the resultant tree has the required range property. It is the role of the root in the inquired tree to find the correct position to incorporate another connected component and update the range values in the tree. The protocol does not specify the point at which device role (master/slave) is established in creating the piconets. The protocol operates in an asynchronous manner and does not require all devices to be in mutual range. The operation of the protocol is not always guaranteed under changing membership (node failure), and the particular circumstances under which problems occur are documented in [90]. The protocol seeks to integrate network formation with routing issues, so that routing induces minimal overhead.

5.13. Tree scatternet formation

In [94,95], a protocol is presented for forming scatternets with a tree structure. The authors focus on designing the protocol to deal with frequent changes in scatternet membership. In particular, the protocol seeks to converge to a steady state, in which the scatternet is fully connected. The protocol handles both nodes arriving incrementally and en-masse, and similarly for departure. Randomisation is used to ensure a balance between each nodes data communication and communication to maintain the scatternet. At any point in time, the *tree scatternet formation* (TSF)-generated scatternet is a forest of connected tree components. A connected component with only a single device is called a free-node. Each node operates autonomously with only local communication, and alternates between two states: FORM and COMM. In the COMM state, the node is involved in data communication with other nodes in its connected component.

The FORM state has two sub-states, composed of inquiry and inquiry scan, as advocated in [80] for symmetric link formation. The protocol permits either free nodes to connect to free nodes, free nodes to connect with non-root nodes, root nodes to connect only to root nodes, and non-root nodes to connect only to free nodes. These properties ensures that TSF produces loop-free topologies. The joining node takes its role (master/slave) to preserve the current structure of device roles in the inquired component. Time division between FORM and COMM states is important. If the connected component large, more time is spent in the COMM state, and conversely so. The amount of time in the COMM state is set proportionally to the degree of the node, dependent on the nodes age, which is checked via a threshold. Consideration is given to the optimal value for the random interval in which devices should stay in the FORM state.

The protocol also gives consideration to healing properties and link loss. When a master losses connection to a slave, it only need check whether it is a free node. When a slave loses connection to a master, it updates its node type and sets its age to zero. A leaf node in this situation becomes a free node and an internal node becomes a root node. The authors note that any root is only permitted to have seven slaves and consequently this restricts the total size of the scatternet, but it is reported that this does not impede the performance of the protocol for tests involving up to 100 nodes.

5.14. Bluenet scatternet formation scheme

The scatternet formation scheme proposed in [98] is designed to produce a flat network structure without any hierarchy. In comparison with a tree structure, it is shown that although the protocol spends more resources on maintaining scatternet links, more communications traffic can be carried. The protocol does not require all devices to be in range, and is designed for the setup of the network.

The phased procedure begins by each node locally building up the visibility graph. Then the nodes follow by randomly entering the page state, to try and invite a fixed number of neighbours to joint its future piconet. The invited nodes become slaves and stop paging. Devices which remain isolated follow in the next phase by paging all neighbours, to try and gain connection in at least one piconet. If more than one piconet is found in which it can participate, then the device becomes a bridge. In the final phase, the formed piconets, which at this stage are largely disjoint, seek interconnection. The master of each piconet instructs slaves to set up outgoing links. The mechanism for achieving this is not provided. The resultant scatternet is not guaranteed in the sense that not all BlueNets are connected even when the initial topologies are. However the probability of connectivity occurring increases with device density.

5.15. Blueroot and distributed Bluetrees

In [102], two scatternet formation protocols are introduced for the creation of tree topologies. The first protocol, Blueroot, creates a tree structure using a designated node, called the *Blueroot*. It is assumed that each node knows whether or not it is the Blueroot, the identifiers of the one-hop neighbours, and whether they are currently part of a piconet. The Blueroot node pages its neighbours, one by one, implying that the Blueroot will be master. If a node is paged and it is not currently part of any piconet, it accepts the page, thus becoming a slave to the paging node. Once a node has been assigned the role of slave, it initiates paging all of its neighbours one by one, and so on.

When building a Bluetree as described, it is possible that a master is assigned too many slaves. A geometric observation is used to reconfigure the Bluetree, to ensure that no master has more than five slaves. It is observed that in an interference- and obstacle-free environment, if a node has more than five neighbours, then there are at least two nodes among the neighbours which are themselves neighbours. Based on this observation, an algorithm is executed at each master node. If a master m has more than five slaves, the master polls the slaves to find the identifiers of neighbours, and to find out how many slaves they handle themselves. Using this information, the master can find two slaves, say s_1 and s_2, which could possibly connect. Node s_1 establishes a connection with s_2, where s_2 becomes a master, while s_2 is instructed to disconnect from m. This operation retains the tree property and ensures than piconets have a limited, feasible number of active slaves.

In an extension to Blueroot, a protocol called distributed Bluetrees, is introduced. This is a tree formation protocol with further distribution, saving on setup time. As with the protocol described above, connectivity is established while maintaining a limited number of roles per device. Initially a collection of nodes, called *init nodes*, are selected in a distributed manner. These are selected as nodes with the highest identifier in its neighbourhood. These nodes initiate the Blueroot protocol in parallel, with two modifications. Firstly, when a piconet connection is established, the slave will be informed about the identifier of the root of the tree. Secondly, when paging neighbourhood nodes which are already part of a Bluetree, information on respective roots is exchanged. The information collected via these modifications is used in the second phase of the protocol. This connects sub-tree scatternets into one spanning the entire visibility graph. This is performed using a virtual graph, whose virtual nodes are the Bluetrees formed in the first phase of the protocol. An edge exists between virtual nodes in the virtual graph if the respective Bluetrees can be connected via an edge from the visibility graph. On dedicating one of these virtual nodes the Blueroot, the Blueroot algorithm can be run on the virtual graph.

The protocols are principally designed for enmasse formation of the scatternet, but they are unusual in being performed on a large number (up to 2000) of devices.

5.16. Blue-star island scatternets

The Blue-star island protocol [104] operates with each device continuously cycling through inquiry, inquiry-scan states, on a randomised basis. When a handshake takes place between totally isolated nodes, a link is established and the master/slave status is allocated. After joining a piconet, slave nodes periodically enter the inquiry-scan state to facilitate discovery by other masters. Bridge status is permitted for slaves up to a predefined limit on the number of interconnecting piconets. Nodes with master status periodically enter the inquiry-scan state, until a limit on the number of slaves per piconet is reached. If a master node scans another master node, it breaks all current links and joins the piconet as a slave. All slaves in the broken piconet become totally isolated devices without either master or slave status, and start the cycle of inquiry, inquiry scan to find new links.

This simple protocol does not guarantee connectivity, but is highly flexible, with no device synchronisation required, and is fully decentralised. It is also sufficient to deal with changes in scatternet membership and dynamic scenarios. The approach gives some consideration to scheduling (via minimising the number of piconets) and also mentions the issue of route discovery in this context.

5.17. Centralised scatternet formation

A number of authors have dealt with the problem of scatternet formation with centralised knowledge. This is of limited use for the application of a protocol, but is useful for two purposes:

1. finding the best possible performance for a given visibility graph;
2. gaining knowledge on scatternet configurations and performance.

The considerable size of the search space (see [12]), in terms of number of feasible scatternets, precludes analytical or exhaustive treatment of this problem. An alternative is to take a distribution of devices, and consider the best possible performance which can be achieved, using the global knowledge (i.e., visibility graph). On application of optimisation techniques, the resultant scatternet can be directly compared with the scatternet formed by the decentralised protocol. This approach was first advocated in [61], using simple heuristic algorithms to generate feasible scatternets, which were then analysed in detail. Subsequently, more sophisticated optimisation approaches have been introduced.

In [22,60], a mathematical programming formulation is used. This involves minimisation of the traffic load at the most congested node, subject to constraints concerning full network connectivity, piconet size and number of piconets, specified traffic requirements (desired source–destination flows), and the number of roles devices perform. The approach assumes that routes are specified between the source and destinations, and the amount of traffic is also specified. This input reduces the search space (i.e., number of possible solutions) because edges along the routes must necessarily be included in the scatternets produced. Therefore the approach focuses on specifying the roles of devices.

In other papers considering centralised scatternet formation [53,54,102], the thrust has been to adapt centralised approaches for decentralised application.

5.18. Summary

In Table 2, we present a classification of approaches for scatternet formation and maintenance. Characteristics of each approach are characterised in terms of control mechanisms, topology and optimisation objectives. Below we define the terms used in Table 2.

- Control mechanism
  - *Global control*—control is centralised at a single device. This node gains full knowledge of all other devices in the area and performs scatternet formation, informing each node of its role and the links between them.
  - *Local control*—scatternet formation is decentralised. Devices determine their roles in the absence of a fully centralised co-ordinator.
- Topology
  - *Radio range*—all devices are in radio range of each other.
  - *Fixed positions*—the positions of devices are fixed.

  - *Gateway*—allows piconets to be interconnected only by common slaves.
  - *Intermediate gateway*—bridges are allowed to be masters in one piconet.
  - *Geometric approach*—uses location information (theoretically via GPS) to determine topology.
  - *Graph based approach*—uses a specific graph to determine the topology, other than ring or tree.
  - *Ring structure*—devices are formed into a ring graph based structure.
  - *Tree structure*—devices are formed into a tree graph based structure.
  - *Hierarchical structure*—nodes form a logical hierarchy (other than slave/master).
  - *Mesh structure*—nodes form a graph with the explicit requirement of at least two paths between device pairs.
  - *Limited number of nodes*—the approach is applicable to a limited numbers of devices only.
- Optimisation and control of the topology
  - *Minimise/limit the number of piconets*—to control interference and inter-piconet communication.
  - *Power management*—to minimise consumption in the network formation process.
  - *Weighting of nodes*—to reflect the suitability for use as masters.
  - *Optimise capacity*—the approach directly considers this aspect.
  - *Optimise throughput*—the approach directly considers this aspect.
  - *Dynamic/maintenance*—a repair response to changes in scatternet membership.
  - *Max–min optimisation*—explicit optimisation of quantifiable multiple objectives.

6. Topology performance measurement

Evaluating the communication performance of an ad hoc wireless network is a challenging problem in its own right [97]. A single, overall best protocol for scatternet formation and maintenance does not exist. Firstly, different scenarios require different performance properties. Secondly, there are various different ways to assess the performance characteristics of both the protocol and the topology it creates and maintains. We consider the alternative approaches for assessing scatternet performance.

6.1. Properties of the protocol

Properties of the protocol relate to its ability to create and maintain scatternets. Low time complexity is particularly important in situations of en-masse start-up. Low message complexity is particularly important in highly dynamic environments were scatternet membership is frequently changing, and generally important for minimising power consumption.

- *Time complexity*
  Time complexity describes the protocols ability to scale in terms of time taken to create a fully connected scatternet. This is particularly relevant for en-masse device start up, where big 'O' notation can be used to describe the worse-case effects of doubling the number of devices on the proportion of time taken. In many simple protocols, it is not possible to derive this analytically (e.g., [30]), since network connectivity is not guaranteed. In these situations, empirical observation is necessary. However, some authors have developed formation protocols from which time complexity can be analytically derived. For example, [49] determines network formation in $O(\log n)$, (assuming n nodes). In [76], $O(n)$ time complexity is established. A range of authors [7,8,14,49,80,83,95] make empirical measurements on this aspect.
- *Message complexity*
  Messaging complexity describes the protocols ability to scale in terms of number of messages required for purposes of creating a fully connected scatternet. As messaging for network formation precludes use of the network for communication purposes, it is important to minimise messaging, particularly in dynamic environments where only a small time window exists for exchanging set up messages. In [50],

messaging is considered explicitly, and it is established that for n devices, the protocol has messaging complexity of $O(n)$. The same result is established for the protocol introduced in [76]. For other protocols, empirical observation of the total number of messages is not considered, because this is better suited to analytical treatment. In preference, time complexity is normally empically observed.

- *Environmental dynamism*
  The performance of scatternet formation and maintenance protocols in dynamic environments is important for ad hoc operation. Protocol efficiency in dynamic scenarios has received little attention other than in [95], where a range of critical cases are considered. These focus on healing partitions and dealing with en-masse and incremental departures and arrivals.

6.2. Properties of the scatternet

The performance of a scatternet topology can be evaluated using measures which are either *traffic dependent* or *traffic independent*. Traffic dependent measures require the specification of a traffic profile, defining source and destination of packets, and packet flows are considered. This needs specification of routing techniques (see Section 7) and scheduling techniques (see Section 8). If studying Bluetooth topology in isolation, as with many scatternet formation protocols, this makes evaluation difficult. Consequently, *traffic independent* performance measures are frequently used. We consider the range of traffic independent measures used in the literature.

- *Average shortest path length*
  In the absence of a pre-defined routing protocol and without definition of traffic flow, the potential for high quality routes can be assessed by shortest path length between source and destination devices. This can be benchmarked against the best-possible shortest path between corresponding devices in the visibility graph. The disparity between shortest path length in the scatternet and that in the visibility graph can be taken as a performance measure. A range of authors (e.g., [7,8,14,25,56,60,71,72,95,98,102,104]) use this approach. However, the measure has to be considered in a wider context, by noting that shortest paths may not necessarily lead to high levels of throughput, particularly if multiple paths use common devices which effectively form a bottleneck. The authors [14,49,104] contrast the shortest path approach by considering maximum network diameter. Shortest path assessment is likely to be a realistic measure in the case of light traffic loads.
- *Traffic flow across a bottleneck*
  A couple of authors [22,60] have directly addressed the potential drawback of shortest path length by considering the performance of traffic flow at bottlenecks identified in the network. This is potentially useful but in practice, the identification of such bottlenecks is likely to depend on the flow of traffic across the network. Consequently assumptions must be introduced concerning routing across the network.
- *Average number of slaves per piconet*
  This is an important metric because the number of slaves per piconet dictates the quantity of piconets required across the network. In a connected scatternet of n devices, with each piconet containing k devices, there must be at least $\lceil (n-1)/k \rceil$ piconets (see [51]). As each piconet has its own channel, the number of piconets also dictates the number of channels per scatternet. When the number of piconets is high, the potential for collisions and channel degradation is increased. However, within a piconet, a potentially higher link capacity is experienced. Consequently protocols generally seek to maximise piconet size (up to seven devices), with the exception of [24], which maximises the number of piconets by forming a ring structure in which every piconet contains exactly two devices. A range of authors [7,8,22,49,53,71,72,104] monitor the performance of protocols taking into account the average number of devices.
- *Average number of roles per device*
  A range of authors [7,8,49,53,71,72,102] note that this factor impacts on the performance of the network by the switching overhead that

it induces. Consequently a number of authors constrain this, while others choose to measure this factor as a minimisation objective.

- *Capacity and throughput*
Scatternet capacity broadly relates to the networks ability to carry traffic, and this is interpreted in various ways by different authors. In [24], this is approximated by measuring the maximum number of successful simultaneous device-pair communications possible. In [25], the effect of piconet polling and number of slaves per piconet are used to estimate per-link capacity. Traffic dependent metrics based on the residual capacity are introduced which are dependent on assumptions for routing. The paper [84] derives an approximate function for capacity based on a range of characteristics that relate to the network. Throughput describes the maximum flow which can be achieved between source and destination devices and has been adopted for analysing scatternet performance in [56,57,60,73,94,98]. Well known algorithms (such as the Ford–Fulkerson approach as described in [98]) can be readily applied. The paper [94] is unusual in deriving approximate functions for assessing throughput in the context of a scatternet.
- *Average path latency*
This metric seeks to approximate the communication latency between device pairs in the scatternet. Tan et al. [94] attribute this to three factors: hop count, intra-piconet scheduling delay and inter-piconet bridging delay. These factors are dependent on the policy adopted for scheduling and routing, which poses problems given the current lack of consensus on these issues, and lack of realistic scatternet traffic patterns. Consequently in [94], a metric is introduced which is independent of scheduling and traffic, and approximates the average path latency between all pairs of source and destinations for a given scatternet, by observing the latency contributions from inter- and intra-piconet scheduling. This approach is also adopted in [24]. Alternative to this is direct simulation of delay (e.g., [43]). However this requires the assumptions on traffic, scheduling and routing. These are present in some approaches to scatternet formation, particularly when there is a well-defined underlying topology structure such as a tree or ring. In such cases (e.g., [57]) fewer approximate metrics need to be imposed.
- *Interference*
It is widely acknowledged that the fast frequency hopping scheme adopted by Bluetooth is resilient to interference. Consequently most authors choose to neglect interference when assessing scatternet performance. The only authors to include this factor for neighbouring piconets are Lin et al. [56]. As 79 frequencies are used for hopping, each of which is equally likely, the probability that a time slot suffers interference is approximated as $(78/79)^{R-1}$ where R is the number of piconets in transmission range of each other. Similar analysis is applied in [32].
- *Network lifetime*
As mobile devices are likely to be power restricted, energy efficiency is important to enhance network lifetime. Few authors directly address this issue. The only paper to consider this directly for BT is [73], which defines the network lifetime as the time until at least one device exhausts all battery power. Two power saving techniques are introduced (one using battery power based master/slave switch and the other using distance based power control) to manage traffic flow across the scatternet.
- *Reliability of the network*
If a protocol can rapidly respond to link and device failure, then the protocol will be able to ensure reliability, via the provision of potential routes across the network. However, the extent to which outage causes critical failures will depend on the provision of alternative routes in the scatternet. Flat network structures (e.g., [72]) have been advocated for increased reliability, and the provision of alternative paths has been considered in [24,56]. Specific network topologies, or networks with certain characteristics, are frequency sought since properties such as reliability may be guaranteed to a certain extent, such as in a mesh configuration.

6.3. Performance simulation

To assess performance in the presence of traffic, simulation is required. The IBM extension to the network-simulator 'ns' [69] is termed BlueHoc [15,48], and is open-source, written in C++. This emulates the Bluetooth stack and the operations which are available in the Bluetooth specification, and has been adopted in [50,57,94,95,104]. BlueHoc is a useful starting point for protocol performance measurement and has been subsequently enhanced in [7] for example, by additional mechanisms to handle collisions and enable devices to alternate between inquiry and inquiry scan. In [34], further development is reported, to consider node movement. A Java alternative to BlueHoc is Simjava [85], which is adopted in [68] for purposes of simulating service discovery.

7. Routing

To facilitate multi-hop communication, a packet will need to be relayed via a number of masters and bridges before it reaches its intended destination. A number of authors (e.g., [13,99]) point out that routing in the context of Bluetooth differs to that for general ad hoc network scenarios. Consequently, a different set of design compromises are required for routing techniques, as compared to those being developed for general ad hoc applications (see [77] for an overview) by the Internet Engineering Task Force (IETF) mobile ad hoc network (MANET) working group. For example, in [13], it is pointed out that due to packet size limitation at the baseband layer, MANET style solutions will require fragmentation and packets at each relay device. Consequently, there will be an increased buffering requirement at each device leading to a higher delay at each hop. This means that it is advantageous to support forwarding across the network at the slot level, as it will reduce the buffering that is required.

It is likely that the Internet protocol (IP) will be commonplace in the context of scatternets, and therefore it is possible that this layer could be used for routing. However, a number of arguments are proposed in [38] against sole reliance on this layer, for scatternet routing. In particular, a routing function on the IP layer would need to be adapted, so that routing function can be integrated into the scatternet formation function. This violates the principle of keeping the IP layer independent from the link layer. Additionally, the IP routing approach would preclude other non-IP based applications. Bluetooth devices are not compelled to host an IP layer to be part of an operational piconet. It is concluded in [38] that the best way to manifest routing functionality is at the layer beneath IP.

The context in which Bluetooth scatternets are likely to operate, also affects the design requirements. For personal area network applications, traffic characteristics, mobility patterns and scaling requirements will also differ from classic ad hoc networks. It is likely that in many instances, scatternets will be quasi-static, short lived and small. Therefore, scalability and adaptivity features may not always be essential. Instead, protocol simplicity, power and bandwidth conservation are particularly important. Additionally, for efficiency purposes, integration with scatternet formation processes is desirable.

As for other scenarios, choices for routing strategies fall into the broad categories of *pro-active* and *re-active* approaches. Traditional routing protocols, used for Internet routing for example (and also in ad hoc networks), are pro-active in that they maintain routes to all nodes. This type of approach requires storage for routing tables, and procedures are required to generate the routing tables and ensure that they are up to date, as the ad hoc nature of scatternets means that links will be temporary. In a reactive protocol, routes are determined dynamically. We describe the ways in which these approaches have been proposed in the context of Bluetooth scatternets. It is notable that very limited development has occurred in this area.

7.1. A reactive protocol

The protocol proposed in [13] is a source-oriented approach, which creates routes only when desired by the source device. The commonly used method of representing routes in source routing headers is to include a list of the node identities. This would be a wasteful approach in Bluetooth scatternets. To avoid this, Bhagwat et al. [13] seek to reduce the overhead by introducing an efficient route representation for uni-cast and multi-cast traffic. This approach is called the *routing vector method* (RVM), and relies on representing piconets by a local identification number (*LocID*), selected locally by bridges. Within a piconet, a 3-bit MAC address (*MacAddr*) is used to identify the slaves. The sequence of *LocID* and *MacAddr* values is used to identify routes, as opposed to using a sequence of Bluetooth identities. This leads to a substantial reduction in the number of bits per hop, induced as an overhead.

When a packet is sent, the sequence of addresses corresponding to the route to be followed is written to the route vector field of the header. A source routing algorithm is employed to determine the path between the source and destination nodes. Since scatternets are mobile units, a protocol is proposed where two or more maximally disjoint routes are determined, and each packet is sent via all routes. Discovery of the first route occurs using a flooding approach across the entire scatternet, using search packets. Each bridge that propagates a search packet appends the piconet identities from which it has received the packet and also appends the packets next destination. When the final destination first receives a search packet, it returns a reply packet along the reverse path. The second route is built similarly to the first. Once routes have been defined, a packet is sent from the source to the destination via the RVM route. When a relay receives a packet, it sends the packet to the master of the piconet corresponding to the first *LocID*, to be forwarded to the unit whose MAC address is given by the first *MacAddr*. Before sending, the first pair (*LocID, MacAddr*) are removed from the list.

On demand route discovery is recommended by the authors on the assumption that only a small set of devices will need to communicate at once. The route is kept in the packets, not routing tables, to minimise storage overheads. The authors indicate there will be a limit beyond which such networks cannot be scaled, through they expect this approach to suffice in the majority of applications.

7.2. Hybrid protocols

Hybrid protocols combine the characteristics of reactive and pro-active protocols. In [23], an approach is presented, derived from a Dynamic Source Routing (DSR) approach [40] designed for ad hoc networks, which has been adjusted to fit Bluetooth characteristics. When forming a route, this approach considers the delay that may be expected at masters due to connections to many slaves. A Blueroute layer manages a cache at each Bluetooth unit, which may contain routes from this unit to other units. When performing route discovery, the source checks its cache to see if it has a route to the destination. A process is repeated where a unit floods the packet to its neighbours when it is not the packets destination and does not have a cached route to the destination. If the unit has a cache of the destination route, this is used to determine the route. No detail is given regarding the size of cache at each Bluetooth node.

In [44], a hybrid reactive-pro-active Zone Routing Protocol approach that employs a combination of pro-active and reactive schemes is proposed. Since Bluetooth nodes are expected to have low resources, schemes that require storage of large routing tables, may not be viable. Using a hybrid approach allows Bluetooth masters to store smaller routing tables that provide routing information for nodes within a maximum number of hops (denoted *MAXHOPS*). Similarly to [23], if a node does not have a full path to the destination, reactive path discovery is employed by flooding the scatternet with a route request. The first node that has a path to the requested node, sends a route reply packet back to the source. This packet follows the route taken to reach this node and this information along with the nodes path knowledge is used to define the route.

The parameter *MAXHOPS* determines the performance of the scheme. Simulations show that the pro-active part of the scheme was able to communicate large amounts of routing information at low overheads. A high value of *MAXHOPS* leads to small route acquisition latencies, but at the expense of higher routing overhead and higher information storage costs. It is also shown that in networks with large numbers of idle nodes, the pro-active part may cause unnecessary overhead, particularly for large values of *MAXHOPS*.

7.3. Power conserving routing

Many wireless network scenarios require energy efficient network protocols. A general survey of this area is given in [41]. Recently, several Bluetooth specific routing methods have been proposed that aim to conserve power.

7.3.1. Power dependent routing

In [73], a reactive routing approach is proposed which is aimed at providing energy efficient techniques by utilising knowledge of the available battery power of the Bluetooth devices as a cost metric in choosing the routes. It is assumed that during scatternet formation, each piconet is given a unique piconet ID (PID). When a device wants to discover a route to a destination, a flooding approach is employed in which it sends a route request packet to its master. The master appends its PID and its available battery power to the request packet and forwards it to all associated bridge points in the piconet. Each bridge appends its Bluetooth address and adds its available power level to the cost field and forwards the packet to all other piconets it is associated with. This process continues until the request packet reaches the destination device. The destination may get multiple copies of the route request packet through different routing paths each having different costs. The destination waits for a specified amount of time to get multiple copies of the route request packet through different paths. It then selects the path that has the maximum cost field (i.e., maximum cumulative battery power in the path). The destination then sends a route reply packet to the source device. The route reply packet contains the selected route vector.

7.3.2. On demand routing

In [58], an on demand approach to the building, routing and scheduling of a Bluetooth scatternet is presented. The most common approach for scatternet formation protocols is to interconnect all Bluetooth devices at the initial network startup stage and maintain all Bluetooth links thereafter. In reality, the physical and data link only need to be created and supported when two Bluetooth devices wish to communicate. If no communication is required between Bluetooth devices, they can go into a low power standby state. This power saving principle has been ignored by most scatternet formation algorithms, which tend to interconnect all Bluetooth devices as a complete scatternet at the initial startup stage, and maintain the full connectivity of the network at the data link level. Instead of forming a complete scatternet, the authors propose a scatternet-route structure that enables the dynamic establishment of Bluetooth links only along the traffic routes. The differences between this approach and the complete scatternet approach are similar to those between table driven routing protocols and on-demand routing protocols. The former maintains network-wide route information for fast setup, but incurs substantial signalling traffic and power consumption. The latter suffers long route setup delays, but is more power efficient in a mobile ad hoc environment. Where power is a concern, as in a Bluetooth scenario, the on demand approach may be favourable.

As with reactive routing approaches, a route discovery packet (RDP) is flooded to the whole network to find a route to the destination. Upon receiving the first RDP packet, the destination node sends a route reply packet (RRP) back to the source along the discovered route. When transferring the RRP, point-to-point Bluetooth links are created to connect the members of the new route and at the same time, the routing tables of these devices are filled with the new route discovery information. When the RRP arrives at the source device, the scatternet route is ready for the first data packet to be sent from the source. Analysis and simulation show that whilst this approach has non-negligible setup delay, it achieves high link utility, stable network throughput, and fast packet transmission.

7.4. Self-routing structures

One approach to simplifying the potentially complex issue of routing is to create a topology in which there are unique paths between source and destination. Such structures necessarily have to be trees. In [90], this approach has been extended further by structuring the hierarchy in the tree so that messages are self-routing in the sense that they need carry no routing information, only simple rules to guide their way through the topology, which can be regarded as a search tree. Further details of this approach are given in Section 5.12. Slightly more complex than the tree topology is the cycle or ring structure, which also leads to very simple routing. This has been tackled in [24,56], as described in Sections 5.3 and 5.8.

8. Scheduling

Scheduling is required to transfer packets between devices which may reside in the same or different piconets. The Bluetooth specification contains only limited information on how scheduling is to be performed. The nature of Bluetooth scatternets means that whilst existing scheduling approaches used in wireless networks are applicable for intra-piconet communications, more consideration is required to effectively apply them in topologies involving interconnected piconets.

Within a piconet, scheduling is undertaken by master devices, and controls the devices choice for communication with (and therefore from) a slave in the time division duplex. The master decides which slave is the one to next access the channel. A slave is authorised to deliver a packet to the master only if it has received a polling packet from the master. The *polling cycle* determines the order in which the slave are polled, and the number of slots for data transmission. To manage message forwarding between piconets, consideration must be given to the effects of scheduling on bridge devices, which reside in multiple piconets. An amicable relationship must be established between schedules in overlapped piconets to avoid conflicts: a master should only poll a bridge when it is not engaged in communication in another piconet. Furthermore, inter-piconet scheduling must satisfy inter-piconet traffic demand, so that bottlenecks are minimised. This requires consideration of how best to divide a bridges time between multiple piconets. The queue at a device refers to the number of packets awaiting transmission. A schedule is *exhaustive* if the queue at each device is cleared before the next device is polled. We consider the advances which have been made concerning intra- and inter-piconet scheduling for Bluetooth.

8.1. Intra-piconet scheduling

Intra-piconet scheduling only considers the scheduling problem for a single piconet. Three issues need to be addressed in the design of an intra-piconet scheduler:

1. *slave activity:* avoidance of polling slaves which have no data to transmit;
2. *fairness:* give preference to slaves based on their relative importance;
3. *efficiency:* minimise the delay for slaves who are waiting to transmit data.

This section begins with an examination of intra-piconet schedulers that build upon an existing scheduling approaches.

8.1.1. Round Robin based scheduling

Round Robin (RR) is one of the simplest and most widely used scheduling algorithms, most commonly used to schedule CPU time. Applying this to intra-piconet scheduling means that slaves are polled in a cyclic fashion. A variety of RR based approaches have been proposed and evaluated. These are distinguished by the criteria for defining the cycle, and the length of time (slots) each master–slave pair is permitted for communication. In [21], Capone et al. consider the problem of designing an efficient and simple polling and scheduling scheme for Bluetooth. Three RR scheduling schemes are proposed:

- *Pure Round Robin (PRR):* A fixed cyclic order is defined and a single chance to transmit is given to each master–slave queue pair according to the cyclic order. This scheme is not exhaustive and each slave is granted a fixed number of slots.
- *Exhaustive Round Robin (ERR):* As with PRR a fixed order is defined, but the scheme is exhaustive and it does not switch to the next slave until both the master and the slave queues are empty.

- *Exhaustive Pseudo-cyclic Master queue length (EPM):* A dynamic cycle order is defined at the beginning of each cycle (each slave is visited exactly once per cycle) according to a decreasing master to slave queue length order.

The difference in performance between ERR and EPM is shown to be negligible, and this suggests that the use of a pseudo-cyclic dynamic scheme based on partial information of the queue status is not necessary since the fixed cycle scheme gives good performance. The PRR scheme performs the poorest, in general. While exhaustive schemes such as ERR are shown to be effective in symmetric scenarios, they may be less appropriate where traffic demand is asymmetric. An additional drawback of exhaustive schemes is that slaves with large amounts of data to transmit may capture the channel, leading to unfair relative distribution of bandwidth. Whilst it clearly makes sense to offer more bandwidth to slaves with high loads, some attempt must be made to do this fairly. The simplest way is to modify the ERR scheme, limiting the number of slots that can be performed each pair cycle. This functionality is employed in the Limited Round Robin (LRR) approach. At low loads, the delays of the LRR scheme are higher than that of ERR. At higher loads, LRR can achieve lower delay than ERR due to the wastage of slots by the ERR scheme.

A further proposed enhancement to the LRR scheme, defined in [21], extends it to consider the activeness of slaves. A performance gain is made by reducing the rate of visit to queues which have been empty in the last visits and hence, should have a lower probability of being the longer queues. This should also reduce the time wasted polling idle slaves. This leads to extension of the LRR scheme to the Limited Weight Round Robin (LWRR) scheme [21]. LWRR adopts a weighted round robin algorithm with weights dynamically changed according to the observed queue status. At the beginning, each slave is assigned a weight equal to maximum priority (MP), which constitutes a variable which is set within the scheduler.

The weight is used to calculate how many cycles a slave must wait before it is next polled, where the number of cycles to wait is given by the MP slave weight. Each time a slave is polled and no data is exchanged between master and slave, the weight of the slave is reduced by 1, until a minimum weight of 1 is reached. In this case, the slave has to wait a maximum of MP − 1 cycles before it gets a chance to send data. When an exchange occurs between a slave and its master, the weight of the slave is increased to the MP value. Evaluation of LWRR shows that it always performs better than the other schemes, and is usually very close to that of the ERR.

8.1.2. Extended Round Robin scheduling

The work of Lee et al. [52] extends that in [21]. A new polling scheme is proposed, called Pseudo-Random Cyclic Limited slot-Weighted Round Robin (PLsWRR). This has two important properties. Firstly, as in LWRR, it attempts to distinguish between slaves on the basis of their "activeness", which constitutes their traffic history. Secondly, the polling order for each cycle is determined in a pseudo-random manner. LWRR has two disadvantages due to the number of slots in a polling cycle being variable. Firstly, an inactive slave needs to wait for a long time to get a chance to exchange data packets, if the previous polling cycles have large numbers of slots. This can lead to high delay for an idle slave. Secondly, an idle slave is polled frequently if the previous polling cycles have a small number of slots. This may reduce the efficiency of the system. The PLsWRR scheme extends the LWRR scheme by employing a RR algorithm in which the weights of slots are changed according to the activeness of the slave in the previous cycle.

Initially, each slave is assigned a slot-weight equal to Max-Slot Priority (MSP). This value is a variable which can be set in the system, taken to be 160 slots in the simulation, which equates to a maximum waiting time of 100 ms. When a slave is polled and no data is exchanged, the slot weight of the slave is reduced by the number of slots in the previous cycle. As with LWRR, the lowest weight value is 1 and if data is exchanged between the slave and the master, the slot-weight of the slave is increased to the MSP. PLsWRR uses the number of slots as opposed to number of cycles (as in LWRR) to reduce the polling to less active slaves. PLsWRR guarantees that slaves wait a maximum of MSP slots to get a chance to be polled. This makes the behaviour more reliable than LWRR, in which the slave waits for a bounded number of cycles, but the length of these cycles may be variable. As a result, unlike LWRR, PLsWRR works effectively regardless of the length of the previous polling cycles, avoiding the two disadvantages of LWRR.

Evaluations show that PLsWRR outperforms LWRR. It is also shown that the pseudo-random cyclic order polling in PLsWRR provides fairness to the flows, with all flows in the piconet having equal shares of the total capacity.

8.1.3. Efficient double cycle scheduling

The efficient double cycle (EDC) scheduling approach introduced in [18,19] seeks to preserve the fairness of a typical round robin technique but increase efficiency by avoiding wasted polling of devices with no data to send. Polling only the active devices leads to a *polling sub-cycle*, which was initially addressed in [37]. The EDC approach extends the application of a polling sub-cycle to a separation of up-link and down-link scheduling. In the downlink direction, the master has knowledge of packet queues for slaves. In the other direction, it can be assumed that the master has a probabilistic knowledge based on the feedback the master gets when polling the slaves. This means that the master can only estimate the probability of an inactive slave by exploiting knowledge of slave transmissions in previous cycles.

A partial decoupling in the scheduling of downlink and uplink transmissions is achieved by defining a double polling cycle, which consists of an uplink polling sub-cycle (cycle_{up}) and a downlink polling cycle (cycle_{dw}). During both these cycles the master updates a subset of slaves that are eligible for polling. For the downlink cycle, these are defined as $E(\text{DW})$, and $E(\text{UP})$ denotes the members eligible for the uplink cycle. Different rules are applied for the selection of slaves in $E(\text{UP})$ and $E(\text{DW})$. These rules unsure that there is fairness separation for the different directions: specifically cycle_{up} ensures fairness in the uplink direction, and similarly for cycle_{dw} in the downlink direction. The sets $E(\text{UP})$ and $E(\text{DW})$ are simultaneously updated at the beginning of each cycle, with knowledge of traffic loads gained from the previous cycle. The set $E(\text{DW})$ is determined by the master, using knowledge of queue lengths. For the uplink polling cycle, if a slave has sent a packet with a null payload in the previous cycle, then its polling window is doubled (until a maximum value is reached). Otherwise a polling window of 1 is used, and therefore the device is polled in the next cycle. Simulation results presented in [18,19] show that EDC significantly improves the throughput of TCP connections when compared to a round robin scheduling approach.

8.1.4. Adapting to flow and utilising multiple slots

In addition to reducing the polling frequency to slaves with no data to transmit, improvements can be gained from reducing the polling frequency for slaves with low volumes of data to transmit. Das et al. [26] describe three algorithms that each adjust the polling rate of slaves, based on the amount of data they have to transmit.

In the first approach, called *Adaptive Flow based Polling* (AFP), the polling rate is increased for slaves that have more than a threshold value (*buf-thresh*) of data in the buffer, and decrease the polling rate for those slaves that have no data to send. This is controlled by updating the polling interval for each device. Initially this is a uniform (default) value, but subsequently changes according to the number of packets in the slaves buffer. If there are more than *buf-thresh* packets in the buffer, the packet is sent and the polling interval is set to the default value. In this case, there is a high flow rate for this slave, and hence its polling interval is reduced so that it can be served more frequently. If there are less than *buf-thresh* packets in the buffer, the packet is transmitted and the polling interval is unchanged. If a poll packet is transmitted and a null packet is received, the current polling interval at the device is doubled (up to a defined limit).

In the second algorithm, the *Sticky* algorithm, the rate of polling is not affected, but slaves with high levels of data to transmit are allowed to transmit multiple packets. Each slave is serviced in a cyclic fashion and if the slave has more than *buf-thresh* packets in its buffer, a maximum level of *num-sticky* packets are transmitted. If the number of packets in the buffer is less than *buf-thresh*, a single packet is transmitted as in RR scheduling.

The final algorithm, *Sticky Adaptive Flow based Polling* (Sticky AFP) combines aspects of AFP and *Sticky*, such that when the slave has more than *buf-thresh* packets to send, a maximum of *num-sticky* packets are transmitted.

A simulation is used in [26] to assess the the TCP throughput for different values of *num-sticky*, and a comparison is performed between the three algorithms. Both AFP and *Sticky* algorithms give significantly better performance than RR. The *Sticky* algorithm is found to have the lowest end-to-end delay, whilst *Sticky AFP* has the highest. *Sticky* reduces queue occupancy by transmitting multiple packets from queues with high backlog, thereby preventing queue overflow and reducing end-to-end delay. On the other hand, *Sticky AFP* causes a marked increase in end-to-end delay of intermittent constant bit rate traffic, because flow is set infrequently for such bursty sources. It is concluded in [26] that either *AFP* or *Sticky* (with a high value of *num-sticky*) result in the best overall performance.

8.1.5. Predictive fair polling

The predictive fair polling (PFP) approach is based on consideration of best effort traffic [100]. This means that the slaves get the same fraction of their fair share resources, where a fair share is equal to the share that the slaves would have been given when a generalised processor sharing system was used. This is opposed to the case where fairness determines that slaves get the fraction of their negotiated quality of service requirements.

The scheduling operates without information about the offered load. This means that neither packet size nor arrival times are known for scheduling purposes. Consequently a range of estimators are used, the output from which is fed into a decision making process. The estimators used approximate traffic demand and data availability. The traffic demand estimator calculates a moving average of inter-arrival times of packets. The data availability predictor calculates the probability P_i of each slave i having a baseband packet waiting for transmission to the master. This leads to a fair share measure fs_i for each slave i, defined as

$$\mathrm{fs}_i = \frac{P_i}{\sum_{k=1}^{n} P_k},$$

where n is the number of slaves. The fair share measure is compared with the number of polls since the last poll to slave i (denoted s_i). For each slave i, this is called the *fraction of fair share* denoted Ffs_i. If $s_i \geqslant \mathrm{fs}_i$, this is defined as 1. Otherwise

$$\mathrm{Ffs}_i = \frac{s_i}{\mathrm{fs}_i}.$$

The decision on whom to poll next is determined using fraction of fair share and P_i. Clearly, the polling of a slave i with $P_i = 1$ and $\mathrm{Ffs}_i = 0$ is urgent, while polling a slave with $P_i = 0$ and $\mathrm{Ffs}_i = 1$ isn't urgent. In between these extremes, the polling decision is made using a measure of *urgency*, denoted U_i, for each slave i. This is defined as

$$U_i = \alpha P_i + (1-\alpha)(1-\mathrm{Ffs}_i), \quad 0 \leqslant \alpha \leqslant 1.$$

Tuning the setting of parameter α affects the relative impact of the fraction of share compared to P_i, in the measure of urgency. The decision on whom to poll next is determined by the slave with the highest U_i value. The authors find that tuning the value of α to 0.1 leads to the low response times (sum of waiting times and service times of packets) when such packets are generated under Poisson processes. The authors also propose extensions to deal with quality of service traffic handling and duplex traffic handling.

8.1.6. Prioritising links where master and slave have data to transmit

In [42], the proposed approach to polling is motivated by three issues: (1) state of the queues at the master and slaves; (2) the traffic arrival process at these queues; and (3) the packet length distribution at the master and slave. Two scheduling policies are introduced, *Priority* and *K Fairness*, which are master–slave queue state dependent. In these approaches, master–slave pairs are distinguished based on the state of the queues at the master and slaves, the master having a separate queue for each slave. If a node (master or slave) has data to send, the node is in mode 1, otherwise mode is 0. This leads to four distinct states for a master–slave pair. It is assumed that binary information regarding the status of the queue at a slave is known at the master, with transfer using free bits in the payload packet header.

In the Priority Policy (PP), higher priority is given to master–slave connections in the 1–1 state over master–slave pairs in the 0–1 or 0–1 states (which have equal priority), with exclusion of the 0–0 state. The PP policy achieves higher throughput than a pure round robin policy since connections in the 1–1 state are given a proportionally higher number of time slots.

The *K Fairness* policy imposes a fairness bound, where waiting is limited by the factor K slots. Round robin scheduling is performed among all master–slave connection pairs that are in 1–1, 0–1 or 1–0 states. When the scheduler is at a 0–1 or 1–0 pair, its service is sacrificed to a 1–1 connection provided that the difference between the service provided by any two back-logged connections does not exceed K slots. Counters are maintained to track the excess or deficit service received for the queue across each master–slave pair. The master–slave pair that has received the maximum excess service (service sacrificed to it) from other pairs is denoted $q_{\max}$ and the master–slave pair that has sacrificed the maximum service to other master–slave pairs is denoted $q_{\min}$. Both $q_{\min}$ and $q_{\max}$ are defined with respect to each of the back-logged queue pairs (i.e., 1–0, 0–1, or 1–1). When scheduling a 0–1 or 1–0 type connection, if the difference received by $q_{\max}$ and $q_{\min}$ exceeds K, no further service is sacrificed by the pair being scheduled. This ensures that the max–min fairness bound [46] is maintained. This policy achieves higher throughput than RR and also maintains fairness. KFP gives better throughput than PP with more fairness. In PP the unfairness rises quadratically with an increase in p. In KFP, the unfairness is bounded and increases linearly. The fairness bound is potentially useful in defining a QoS guarantee for KFP.

8.1.7. Queue management based on capacity and multi-slot framing

The performance of a scheduling approach is reflected in the profile of queues, since this is indicative of speed and flow. It is essential to control the length of the queue and also the amount of time any packet remains delayed in the queue. Ideally the scheduler should smooth traffic fluctuation and avoid long delays. The use of multi-slot framing can be advantageous for a number of reasons. Firstly, multi-slot framing benefits effective payload transmission as a 1-slot frame has associated overheads besides the payload. The subsequent slots in a multi-slot frame can avoid overheads such as the frame header. Secondly, different speeds can be used for different link directions.

Luo et al. [59] present a scheduling algorithm which seeks to address these issues. They show that in a scenario with exact knowledge of queue status for each slave at the master, high performance can be achieved. However, this is not practical in an applied scenario, due to the overheads required to obtain this information at the master. Consequently, the authors introduce the notion of *predictive capacity assignment* as a replacement to the real time queue analysis. The result is *Adaptive Capacity Ratio with Intelligent Frame Scheduling* (ACRIFS). In ACRIFS, the master is assumed to have a queue for each slave. The capacity threshold of these queues is calculated periodically based on equations that consider the traffic and the propagation delay of the link. The scheduler uses the maximum queue length at the master to select the queue for transmission, and in addition, sees if the transmitted slot is within the capacity threshold. A predictive capacity ratio is used to encourage the master to determine multi-slot framing without requiring knowledge about queue status from the slaves, making ACRIFS a practical solution. Simulation and analysis of this approach shows that it significantly outperforms the benchmark RR approach, when considering packet delay.

8.1.8. Packet-type priority approach

The scheduling approach in [82] is unique in considering a mix of voice and data packet types, each of which may have different levels of priority. A mixed data and voice traffic profile is simulated, modelled by an interrupted Bernoulli process. Performance is considered based on mean packet delay and probability of packet loss for data traffic. A Bluetooth frame is used in which eight SCO slots are reserved for voice traffic and 16 ACL slots are reserved for data traffic. For the ACL slots, scheduling of packets of two possible different data types is considered (denoted D1 and D2).

Three schemes are considered. New schemes called *priority* (PR), *alternating priority* (AP) are compared against a round robin scheme. In the PR scheme, voice has the highest priority, D1 has the second priority and D2 has the lowest priority. In the AP scheme, slots are reserved for both data traffic components in a round robin manner, but the unused slots can be used by other data traffic. The round robin approach uses 3-slot cycles.

The effectiveness of these scheduling methods are studied by varying the traffic load and burstiness of the data streams and varying queue capacity. The results show, that with an infinite queue, the length of the queue grows dramatically when offered load and burstiness are high. In all three scheduling schemes, queue capacity has an essential impact on loss and mean delay for all studied traffic cases. Empirical results show that PR and AP have approximately equal probability of packet loss, significantly lower than that of the RR approach. The AP scheme also has the advantage of fairness and efficiency in comparison to the other schemes.

8.1.9. Bin packing

Yang et al. [101] apply the concept of bin packing to the scheduling problem. The aim of the general bin packing problem is to pack items of various sizes into a set of bins of fixed capacity such that the number of bins required is minimised. In off-line bin packing, the size of all items is known and (near) optimum solutions can be found through the use of exhaustive or meta heuristic approaches. In online bin packing, the size of items is not known and they must be packed as they arrive (items already packed cannot be removed or rearranged). Scheduling in Bluetooth can be viewed as an online bin packing problem with a limited amount of look ahead, since the scheduler can potentially determine the size of queued packets at the head of the line (HOL) for each slave, and decide which to pack (i.e., include in the current frame).

Two scheduling strategies are considered—*Look Ahead* (LA) and *Look Ahead Limited Round Robin* (LARR). In LA, a *Next Fit* (NF) approach is used in which bins are filled in sequence (analogous to filling frames in sequence). The NFD extension to NF involves arranging HOL packets in non-increasing order of size before they are packed in frames. Problems with this approach are two-fold. Firstly, every time a packet is scheduled, the new largest HOL packet must be determined. Secondly, since the algorithm always attempts to schedule the largest packet, it may be some time before packets from some slaves are scheduled (starvation).

Look Ahead Round Robin attempts to overcome these problems. It combines the simplicity of RR with the LA approach to avoid starvation and reduce the computational complexity by no longer attempting to find the largest HOL packet. Instead, slaves are serviced in a RR fashion, and if the packet fits the frame, it is scheduled. This approach differs from round robin because when the packet does not fit, instead of waiting for the next frame, the algorithm looks ahead and attempts to schedule a packet from the next slave in the RR sequence. Analysis and simulation show that LA and LARR perform significantly better than RR. Since LARR is less complex and avoids starvation whilst maintaining a similar level of performance, the authors advocate this scheduling approach for Bluetooth.

8.1.10. Interference aware scheduling

The only approach to scheduling which tries to minimise the effects of interference is that in [28]. In this paper, an IEEE 802.11 system operating in direct spread spectrum mode is considered as the source of interference, though the approach may be adapted to any source of interference. The approach exploits the fact that devices in the same piconet will not be subject to the same level of interference on all channels. The approach distributes channels to devices to maximise throughput whilst maintaining fairness of access. The algorithm, called *Bluetooth Interference Aware Scheduling* (BIAS), has three components, namely, a channel estimation procedure, a procedure that weights devices based on channel access priority, and a credit function that controls fair access to bandwidth.

The channel estimation procedure is used to detect the presence of interference in the frequency band. Each slave maintains a table that indicates if a frequency is used or clear, depending on whether the bit error rate exceeds a threshold. Since the master controls all transmissions, the slaves send their frequency usage table to the master which can then make use of this data in the channel estimation phase to optimise the frequency allocation on each time slot, to avoid packet transmission on a channel with high interference. A credit system is used to control the bandwidth allocated to each device, in order to ensure that no device gets more than its fair share of the available bandwidth. Priority is given to devices with few good channels over those with higher channel availability. Simulation results show that BIAS can reduce the probability of packet loss where interference exists, but with a slight increase in the mean access delay for the worst-case scenario.

8.2. Inter-piconet scheduling

Inter-piconet scheduling is required to co-ordinate the activities of bridge devices which reside in multiple piconets. Such bridge devices must switch between piconets to enable inter-piconet communication. The switching process induces an overhead due to guard time: each switch may cost up to two slots, which are not available for communication. This occurs as a consequence of the clock, the speed of which may differ slightly between masters. Bluetooth permits a clock drift of 20 parts per million against the ideal timing during activity. In low power modes this is extended to 250 parts per million.

Within a piconet, the master always expects a slave to be available. This means that conflicts can arise, in which a master polls a bridge when the bridge is engaged in another piconet, causing slot wastage. Therefore the scheduling algorithm must ensure that slaves are available to a master when it wishes to communicate with them. A simple solution to this problem uses slot reservation, in which specific slots used in the piconet are reserved for particular master–slave pairs. Two masters sharing a common slave cannot reserve the same slot to poll the same bridge. These slots are called *rendezvous points*. As with intra-piconet scheduling, inter-piconet scheduling must be responsive to the network capacity considerations. In particular, it is essential that bridge device are able to focus capacity on the piconet with the greatest load at any moment in time.

All approaches to inter-piconet scheduling are required to tackle common issues. The rendezvous points must be determined, and the rendezvous window must be controlled. Also, this needs to be integrated with intra-piconet scheduling, the dynamic nature of traffic, quality of service requirements, and interference considerations (if applied). There are many different ways of approaching these issues, from the creation of deterministic processes based on simple heuristics, to controlled, randomised processes.

The Bluetooth *sniff* state is commonly employed to determine rendezvous points. This is a power saving mode that allows the slave device to reduce the duty cycle. Instead of listening to the master in every downlink slot, the sniff mode allows a slave to listen to a master in a reduced number of time slots. Thus the *sniff* mode relieves a slave from its full duty cycle, permitting the bridge to switch between piconets. Additionally, *hold* mode approaches have been proposed, along with a call for a new Bluetooth mode called *jump*. We begin by looking at approaches which are non-load adaptive. Then we consider how these have been extended to adaptive scheduling which supports QoS based approaches. Finally we conclude with an overview of an interference aware approach.

8.2.1. Non-load-adaptive scheduling

Johansson et al. [39] propose a periodic rendezvous approach, called the *Maximum Distance Rendezvous Point* (MDRP) algorithm. The basis of the MDRP algorithm is that redenzvous points (RPs) are as far away from each other as possible. The MDRP algorithm uses a periodic super frame, that is common to all nodes in the scatternet, and denotes the time period between two RPs for any master–bridge pair. For a pair of masters connected via a bridge, the distance between the RPs indicates the amount of time the bridge spends in each piconet, referred to as the *rendezvous window*.

The MDRP algorithm attempts to distribute the time spent in each piconet by a bridge equally, by maximising the distance between each RP of a bridge. A master node assigns RPs to bridges as follows. Suppose the bridge belongs to i other piconets, and has RPs $r_1, r_2, \ldots, r_i$ with these piconets. The master node has j other bridges to which it has assigned RPs $r_{i+1}, r_{i+2}, \ldots, r_{i+j}$. These RPs define slot numbers in the super-frame and are in every super frame period. Based on the list of $i + j$ RPs already defined, new RPs are defined at the middle slot in the largest interval between successive existing RPs. This is a simple mechanism for allocation of time between piconets. However, the authors point out that such simplicity and robustness comes at a price, since the algorithm is not load adaptive to temporary changes in network traffic: simply the RV allocation is dependent on the number of piconets a bridge is associated with. Other authors have tried to define adaptive methods capable of changing with the dynamic nature of the traffic. We consider these in detail.

8.2.2. Adaptive scheduling using a RV max–min optimisation approach

In [45], Kazantzidis et al. seek to extend the rendezvous point approach introduced in [39] by not only selecting non-conflicting rendezvous points, but seeking optimal choices of RV point. This is addressed in the context of mobility, where new piconets join existing piconets, at which point RV point selection needs to be considered. The aim is to select RV points so that the minimum forwarding throughput between piconets is close to optimal. The underlying rationale is to avoid establishing RV points that take away forwarding throughput from other piconet pairs. In this way, bottleneck links are avoided and overall forwarding throughput across the scatternet is maintained.

Note that the establishment of a new RV point for a visiting piconet not only affects the throughput between the home-visiting piconet pair, but it also affects the throughput of all connections of the visiting piconet. The proposed approach involves optimising forward throughput equations at bridge devices. This has a potentially expensive overhead, involving the optimisation of local forwarding throughput equations. However the overall aim in this instance is determination of high quality solutions.

8.2.3. Adapting to wasted polls and traffic load

Zhang et al. [103] propose a *Flexible Scatternet-wide Scheduling* (FSS) scheme that can adapt scheduling based on wasted polls and traffic load. The scheme is based on a switch table concept, which is formed when the scatternet is created. This table directs bridge devices to switch between the multiple piconets to which it belongs. Conflicts are avoided by the master polling slaves only at slots when the bridge device is synchronised with the master.

The switch table is maintained and updated, to adjust for load and improve system performance. When the scatternet is initiated, a switch table is generated for each bridge, where master nodes are considered in round robin order and a switch occurs every frame. The FSS scheme acts upon the scheme using two algorithms: a flexible (intra-piconet) scheduling algorithm and a switch table modification algorithm.

In the scheduling algorithm, the master polls the slaves in a weighted round robin manner. Initially, the weight of each slave is computed based on the estimated traffic load in each master–slave link. This means that in a cycle, only a subset of its slaves may be polled. In order to decide the frequency of polling of slaves for a master, each slave has a polling weight tuple (P,R), where P indicates that the slave should be polled every P cycles, and R represents the maximum frequency that a slave can be polled in a cycle. A master can dynamically change the polling weights based on estimation of traffic load, based on previously wasted polls. If a poll is wasted (both slots allocated for polling are not used), the offending slave has P increased (up to a certain threshold), otherwise P is decreased until it reaches 1. For slaves where $P = 1$, if a poll is wasted, R is decreased (until it reaches 1) and P is increased. If a poll is not wasted, R is increased until it reaches an upper bound.

While the scheduling algorithm is able to flexibly schedule slots used to poll slaves and bridge nodes, it cannot change the quantity or positions of slots that a bridge node can use. To address this problem, the switch table modification algorithm permits a bridge device to dynamically update its switch table, based on the traffic load. This allows a device to change its switching pattern between piconets. The switch table update can be initiated by the master or slave. For the slave, the bridge monitors the outgoing queue length and incoming queue length for each of its master–slave links. The master with the longer queue length should get more time slots compared to a master with a relatively low queue length. Based on this idea, the master with the larger queue length borrows slots from the master with the relatively low queue length. The authors define detailed data structures and processes by which the borrowing process is implemented.

8.2.4. Adaptive scheduling using a credit scheme and adaptive presence point density

Baatz et al. [3,4] present an adaptive scheduling which approach utilises the concept *presence points*. These are defined points where communication between a master and slave *may* commence. Presence points are used as an alternative to defining local schedules, and enable each device to quickly determine whether its peer is in the same piconet. If so, communication may begin between the devices, otherwise another presence point may be tried, without loosing a significant amount of bandwidth. Additionally, the length of a particular communication period is not pre-determined, but depends on current link utilisation and the amount of data ready to exchange. As the communication schedule is determined online rather than a priori, simple calculations are required to determine how long to stay in specific piconets. Additionally, presence points must be relatively dense so that there are still some available, even in the presence of interference.

The authors integrate presence points with slots used by the *sniff* mode of operation. Sniff slots are regarded as possible presence points at which peers may start communicating. In order to decide when to abort an ongoing sniff event in order to use an upcoming sniff slot, a priority scheme is introduced, where each device associates a particular priority for each of its links. These priorities are device centric, and different devices may give different priorities to the same link. A link has higher priority relative to another link if it has previously been untreated unfairly, relative to other links supported by the device. This is quantified by tracking the number of slots used by each device.

A potential problem that the authors address with this approach is the frequent switching, and therefore bandwidth wastage, that it might induce. Consequently in [4], the density of sniff slots is reduced exponentially over time, by deliberate skipping of presence points. This scheme is called the *Adaptive Presence Point Density* (APPD) scheme, where each device manages an internal parameter, which determines the time between use of sniff slots. Each device may increase this parameter if, for example, sniff slots are not taken or no data transmission occurs.

8.2.5. Inter- and intra-piconet scheduling using pseudo-random sequences of RV points

Racz et al. [75] propose a lightweight approach to scheduling known as the *Pseudo-Random Coordinated Scatternet Scheduling* (PCSS) algorithm. The approach involves bridge devices assigning meeting points with their peers such that the sequence of meeting points follows a pseudo-random sequence which is different for each pair of devices. Consequently uniqueness of the pseudo-random sequence guarantees that meeting points with different peers of the same node will collide only occasionally. The key advantage is that this removes the need for explicit information exchange between peer devices, which significantly reduces complexity.

The algorithm uses checkpoints, equivalent to RV points. Checkpoints are assigned for a pair of devices, based on the Bluetooth clock of the master and the MAC address of the slave. This scheme guarantees that the checkpoint sequence generated by the master and the slave is the same (without need for communicating the sequence), while also ensuring the sequences belonging to different node pairs will be different. The probability of a collision, where a device can attend only one of the checkpoints, is dependent on the frequency of checkpoints. This is chosen dependent on the free capacities of the node or on the amount of data to transmit. Dynamic adjustment is permitted, with an increase or decrease the intensity of checkpoints depending on the amount of user data to be transmitted and on the capacity of the device. In [75], PCSS was verified in having higher throughput than the equivalent approach without the protection using pseudo-random sequencing.

8.2.6. *Dynamically scheduled meetings*

The advantage of PCSS as proposed in [75] is that it achieves scheduling with little overhead. However, because it is based on a randomised scheme, as the density of nodes grows, scheduling conflicts will arise where one node is actively waiting for another node that is busy communication with some other node. Tan et al. [93] present a *Locally Coordinated Scheduling* (LCS) approach that is able to co-ordinate nodes in a manner that eliminates all scheduling conflicts, but with associated additional overhead.

The LCS approach is designed to dynamically adjust the schedule based on workload conditions. It is based on the concept of scheduled meetings, at which devices meet to exchange data. At the end of each meeting, the nodes negotiate the start time and minimum duration of the next meeting. At each occurrence, a *parent* device sends a list of possible future meeting start and finish times to a *child* device, and the child device replies with desired start and finished times of one future meeting, which fall within one of the meeting periods suggested by the parent.

The role of LCS is to monitor traffic characteristics associated with each link and arrive at an efficient scatternet-wide schedule based on them. Computation of start time is based on whether the data rate is increasing, decreasing or stable. This enables LCS to respond to varying traffic conditions quickly, without wasting resources. The duration of the next meeting is based on queue size, and the past history of transmissions, in order to set a duration that is just large enough to exchange all back-logged data. Similarly, the meeting recess interval is adjusted according to the data rate and the nature of traffic flows.

LCS is comprehensive in the sense that it utilises a range of further techniques to achieve efficient scheduling. For example, meetings with similar characteristics are grouped to reduce bandwidth wastage and end-to-end latency. Additionally, meetings at various parts of the scatternet are scheduled in a hierarchical fashion to exploit the use of parallel communications, to improve throughput. LCS also tries to tolerate disruption in connectivity, by providing a fall back communication mechanism where nodes are not able to communicate during agreed meeting periods.

8.2.7. *QoS*

Son et al. [86,87] develop two inter-piconet scheduling approaches, based on satisfying measures for quality of service (QoS). In [86], the aim of inter-piconet scheduling is to provide just enough capacity for quality of service, via the bridge devices time sharing activities. Quality of service in this context is defined generally, and could be included as measurements on buffer size, packet delays, and numbers of packets queueing at devices in the network. Such traffic information is periodically sent via the link manager to masters which take part in inter-piconet communication. The offered traffic is estimated and predicted from a record of the collected information. This is compared with the allocated capacity, and the inter-piconet scheduler decides whether the QoS can be satisfied. If not, the QoS is changed accordingly. This process is performed at regular intervals. The QoS calculation is quite complex and consequently look-up tables are proposed for real time application.

In the second approach [87], resource allocation for inter-piconet scheduling is formulated as a convex optimisation problem, with multiple objectives that characterise the maximisation of total network flows and the minimisation of the total cost of flows. A distributed iterative capacity allocation scheme is proposed to perform the optimisation. This seeks to divide network capacity to links in such a way that maximises the network flows and satisfies the QoS requirements (such as time delays, queueing sizes, etc.) as formulated in the governing convex optimisation equations. Once allocation of capacity has occurred at a device, allocations of link capacities are iterated along each link for routes across the network. Capacity to support the longest route paths is allocated as a priority. The authors indicate that there are some open issues that need to be addressed, such as the amount of messaging required by this approach, and issues that arise due to mobility of nodes in scatternets.

8.2.8. Load adaptive and hold mode inter-piconet scheduling for small scatternets

Har-Shai et al. [33] restrict their focus to small scatternets using a *load adaptive algorithm* (LAA). Under LAA, the algorithm determines the duration of the bridge activity in the different piconets such that the delay incurred by packets requiring inter-piconet connection is reduced. The algorithm adapts based on a range of factors related to the network performance, including device activity and queue size. Progression above defined thresholds for any of the factors prompts a switch between piconets (it is assumed that only a pair of piconets are operating). The hold mode utilized is at the bridge to enable one Bluetooth device to leave a piconet. The main difference between commonly used sniff and hold modes is that the duration of the hold period must be set every time the slave is placed into hold mode, whereas the parameters of sniff mode are set once and can be used repeatedly. Consequently the parameter setting for the hold mode determines the performance of the algorithm.

8.2.9. A new Bluetooth mode—jump mode

Jonasson et al. [36] propose an additional Bluetooth mode called *jump*. This mode includes a set of communication rules that enable efficient scatternet operation by offering flexibility for a device to adapt its activity in different piconets, under different traffic conditions. The jump mode operates in a distributed manner, allowing each device to jump in and out of the jump mode link. By default, a jumping node is absent from a link. When a jumping node wants to be present in a link, it has to signal its presence to the peer node of the link.

The basic premise is that each jumping node divides its time into time windows of pseudo-random length, denoted *rendezvous (RV) windows*, spending one or more RV windows in a piconet before jumping to another piconet, and signals its decision to all concerned nodes. When a jumping node is present on a jump mode link during an RV window, capacity is distributed by the intra-piconet scheduling mechanism. A jumping master polls its slaves as usual, and jumping slaves that have signalled their presence are incorporated into the masters polling scheme.

The essence of this scheme is that a jumping node uses the same sequence of RV windows in all of its piconets. Employing such sequences of RV windows in conjunction with the simple signalling protocol makes it possible for all the jumping nodes peers to know whether it will be present on their link during each RV window. One advantage is that under this approach, the dependencies between the inter- and intra-piconet schedulers are be kept to a minimum, enabling more or less independent evolution in both fields. The paper describes only the mechanisms of the jump mode, and consequently algorithms can be further developed to implement the mode in scheduling.

8.2.10. Interference aware scheduling

Sun et al. [91] point out that the in situations where inter-piconet scheduling is required, piconets are within range of each other, and therefore interference considerations are important. Although the effects of interference are minimised by fast frequency hopping, it cannot be completely avoided. The simulation reported by Sun et al. [91] shows that interference can degrade system performance by 30%.

However, if the master of a piconet knows the hopping sequence of an adjacent piconet, it can avoid interference by not sending packets when both piconets hop to the same frequency. Any bridge has the information on the hopping sequence of each master, and can distribute this between adjacent piconets. Interference can then be avoided by careful scheduling as described in [91]. Assuming that a slaves reply is a single slot, when a master intends to send an n-slot packet, it can first check if any of the adjacent piconets will use the frequency in the next $n+1$ slots. If some adjacent piconets will use the frequency, the master checks to see if its device address is greater than those of masters in the other adjoining piconets. If it is (or no other adjacent piconets hop to the frequency) then the master can safely transmit. Otherwise, the master suppresses transmission, waits for the next time slot and starts the whole process again.

This scheme avoids interference between adjacent piconets, and can potentially be implemented with a range of scheduling approaches. A small communication overhead is required when the gateway node sends the masters information to the adjacent piconets masters. Additionally, some computational overhead is required where the master pre-calculates the hopping sequence of the adjacent piconets along with its own. Also, when traffic load is not heavy, this scheme may reduce bandwidth utilisation since the master may suppress its transmission, although other piconets on the same frequency, do not transmit. However, when traffic load is medium to high, this scheme should help alleviate the interference, thus increasing bandwidth utilisation. This scheme does not consider interference from piconets that are physically close, but topologically far away (i.e., those that are physically close but do not share a bridge with the piconet).

8.2.11. Queue theoretic analysis

The performance of a range of bridging strategies have been assessed analytically in [63–67]. Queueing theory has been applied to a range of difference scenarios, each scenario focusing on a different topology configuration at the point of bridging. In [63,67], the scenario where bridges have a master and slave role has been considered. In [65,67], the scenario where bridges have only roles as a slave are considered. These contributions focus on modelling access delay, and consider the associated probability distribution of end-to-end delay times for both local (intra-piconet) and non-local traffic. Sensitivity of various parameters, on piconet performance, is also considered. Analysis suggests that the main criteria in minimising end-to-end packet delay should be in minimising the end-to-end delay for inter-piconet rather than intra-piconet traffic. This is suggested as a practical alternative to minimising a weighted average of local and non-local delays. Analysis also indicates that the optimum value for the time interval between bridge exchanges is dependent on current values of traffic parameters, such as burst arrival rate and mean burst size.

9. Conclusions

Bluetooth is an interesting development in pervasive and ubiquitous communication because it represents the first mass market, low cost technology, with opportunities for high levels of penetration. Developing protocols for scatternet formation and maintenance is important, as this opens up increased possibilities for flexible networking and new applications. The state-of-the-art in this area has developed rapidly, and a number of significant contributions have already been made. However, a number of challenging key issues remain, and the following observations can be made.

Firstly, simplifying assumptions, particularly in scatternet formation (e.g., en-masse device start-up) limit the applicability of numerous proposed protocols. Further development is this area would be beneficial. Secondly, it is largely the case that issues of network formation, scheduling and routing have been studied in isolation. Given the dependencies between these issues, further integration of the issues may lead to higher performance solutions. Thirdly, there has been limited systematic benchmarking applied to the proposed techniques. This makes it difficult to draw conclusions on the relative effectiveness of alternative approaches. Finally, the operation of the technology under mobility has received little attention. The extent to which the operation of interconnected piconets can be sustained under mobility, is important given that potential applications may occur for geographically dynamic devices. Consideration of limits of device mobility on network operation, including the limits of mobility speed that could be sustained from the networking (not just transmission) point of view, is an interesting open question. The ongoing development of methodologies for scatternet formation is being documented at www.scatternet.org.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, Computer Networks 38 (2002) 393–422.

[2] S. Baatz, C. Bieschke, M. Frank, P. Martini, C. Scholz, C. Khul, 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.

[3] S. Baatz, M. Frank, C Kuhl, P. Martini, C. Scholz, Adaptive scatternet support for bluetooth using sniff mode, in: Proceedings of the 26th Annual IEEE Conference on Local Computer Networks, LCN 2001, pp. 112–120.

[4] S. Baatz, M. Frank, C. Kuhl, P. Martini, C. Scholz, Bluetooth scatternets: An enhanced adaptive scheduling scheme, in: Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2002, pp. 782–790.

[5] L. Barriere, P. Fraigniaud, L. Narayanan, J. Opatrny, Dynamic construction of blue-tooth scatternets of foxed degree and low diameter, in: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2003.

[6] S. Basagni, R. Bruno, C. Petrioli, Device discovery in bluethooth networks: a scatternet perspective, Lecture Notes in Computer Science, Vol. 2345, Springer, Berlin, 2002, pp. 1087–1092.

[7] S. Basagni, R. Bruno, C. Petrioli, Performance evaluation of a new scatternet formation protocol for multi-hop bluetooth networks, in: The 5th International Symposium Wireless Personal Multimedia Communications, 2002, pp. 208–212.

[8] S. Basagni, R. Bruno, C. Petrioli, A performance comparison of scatternet formation protocols for networks of bluetooth devices, in: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003 (PerCom 2003), pp. 341–350.

[9] S. Basagni, C. Petrioli, Multihop scatternet formation for bluetooth networks, in: Proceedings of the 55th IEEE Vehicular Technology Conference, VTC Spring 2002, pp. 424–428.

[10] S. Basagni, C. Petrioli, A scatternet formation protocol for ad hoc networks of bluetooth devices, in: Proceedings of the 55th IEEE Vehicular Technology Conference, VTC Spring 2002.

[11] P. Bhagwat, Bluetooth: technology for short-range wireless apps, IEEE Internet Computing 5 (3) (2001) 96–103.

[12] P. Bhagwat, S.P. Rao, On the characterization of bluetooth scatternet topologies. Available from <http://www.winlab.rutgers.edu/~pravin/publications/publist.htm>.

[13] P. Bhagwat, A. Segall, A routing vector method (RVM) for routing in bluetooth scatternets, in: Proceedings of the 1999 IEEE International Workshop on Mobile Multimedia Communications (MoMuC '99), pp. 375–379.

[14] V. Bhatnagar, G. Kesidis, Bluetooth scatternet formation using proximity information of an election protocol, in: Proceedings of the Joint 2nd IEEE International Conference on Net Working and IEEE International Conference on Wireless LANs and Home Networks (ICN'02 and ICWLHN'02), Atlanta, August 26–29, 2002.

[15] BlueHoc. Available from <http://www.oss.software.ibm.com/developerworks/opensource/bluehoc/>.

[16] R.C. Braley, I.C. Gifford, R.F. Heile, Wireless personal area networks: an overview of the ieee p802.15 group, Mobile Computing and Communications Review 4 (1) (2000) 26–33.

[17] J. Bray, C.F. Sturman, Bluetooth: Connect Without Cables, Prentice-Hall, Englewood Cliffs, NJ, 2001.

[18] R. Bruno, M. Conti, E. Gregori, Wireless access to Internet via bluetooth: performance evaluation of the edc scheduling algorithm, in: Proceedings of the 1st Workshop on Wireless Mobile Internet, 2001, pp. 43–49.

[19] R. Bruno, M. Conti, E. Gregori, Bluetooth: architecture, protocols and scheduling algorithms, Cluster Computing 5 (2002) 117–131.

[20] S. Buttery, A. Sago, Future applications of bluetooth, BT Technology Journal 21 (3) (2003) 48–55.

[21] A. Capone, M. Gerla, R. Kapoor, Efficient polling schemes for bluetooth picocells, Proceedings of the IEEE International Conference on Communications (ICC 2001), Vol. 7, 2001, pp. 1990–1994.

[22] C.F. Chisasserini, M.A. Marsan, E. Baralis, P. Garza, Towards feasible topology formation algorithms for bluetooth-based wpans, in: Proceedings of the 36th Annual Hawaii International Conference on Systems Sciences, 2003, pp. 313–322.

[23] C.-S. Choi, H.-W. Choi, Dsr based bluetooth scatternet, in: ITC-CSCC, 2002.

[24] F. Chun-Choong, C. Kee-Chaing. Bluerings—bluetooth scatternets with ring structures, in: IASTED International Conference on Wireless and Optical Communication, 2002.

[25] F. Cuomo, T. Melodia, A general methodology and key metrics for scatternet formation in bluetooth, in: Proceedings of the IEEE Global Telecommunications Conference, 2002, GLOBECOM '02, pp. 941–945.

[26] A. Das, A. Ghose, A. Razdan, H. Sarab, R. Shorey, Enhancing performance of asynchrounous data traffic over the bluetooth wireless ad hoc network, in: Proceedings of IEEE INFOCOM 2001.

[27] S. Cho, A. Chandrakasan, E. Shih, B.H. Calboun, Energy efficient link layer for wireless microsensor networks, in: Proceedings of the IEEE Computer Society Annual Workshop on VSLI (WVLSI'OO), Orlando, FL, 2001, pp. 16–21.

[28] N. Golmie, R.E. Van Dyck, A. Soltanian, Interference of bluetooth and IEEE 802.11: simulation modeling and performance evaluation, in: Proceedings of the 4th ACM international workshop on modeling, analysis and simulation of wireless and mobile systems, 2001, pp. 11–18.

[29] Bluetooth Special Interest Group. Available from <http://www.bluetooth.com/>.

[30] R. Guerin, E. Kim, S. Sakar, Bluetooth technology: key challenges and initial research, in: Proceedings of the Conference on Network and Distributed Simulations.

[31] J.C. Haartsen, The bluetooth radio system, IEEE Personal Communications 7 (1) (2000) 28–36.

[32] J.C. Haartsen, S. Mattisson, Bluetooth—a new low-power radio interference providing short-range connectivity, Proceedings of the IEEE 88 (10) (2000) 1651–1661.

[33] L. Har-Shai, R. Kofman, A. Segall, Inter-piconet scheduling in bluetooth scatternets, in: Proceedings of the OPNETWORK Conference, 2002.

[34] C. Hsu, Y. Joung, An ns-based bluetooth toplogy construction simulation environment, in: Proceedings of the 36th Annual Simulation Symposium, 2003.

[35] In-Stat/MDR. Available from <http://standards.instat.com/>.

[36] N. Johansson, F. Alriksson, U. Jonsson, Jump mode—a dynamic window-based scheduling framework for bluetooth scatternets, in: Proceedings of MOBIHOC 2001, pp. 204–211.

[37] N. Johansson, U. Korner, P. Johansson, Evaluation of scheduling algorithm for bluetooth, in: Proceedings of IFIP Broadband Communications, Hong Kong, 1999.

[38] P. Johansson, M. Kazantzidis, R. Kapoor, M. Gerla, Bluetooth and enabler for personal area networking, IEEE Network 15 (5) (2001) 28–37.

[39] R. Johansson, R. Kapoor, M. Kazantzidis, M. Gerla, Rendezvous sheduling in bluetooth scatternets, in: Proceedings of the IEEE International Conference on Communications (ICC 2002), pp. 318–324.

[40] D.B. Johnson, D.A. Maltz, Dynamic source routing in ad hoc networks. Available from <http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-05.txt>.

[41] C.E. Jones, K.M. Sivalingam, P. Agrawal, J.C. Chen, A survey of energy efficient network protocols for wireless networks, Wireless Networks 7 (4) (2001) 343–358.

[42] M. Kalia, D. Bansal, R. Shorey, Data scheduling and sar for bluetooth MAC, Proceedings of the 51st IEEE Conference on Vehicular Technology, Vol. 2, 2000, pp. 716–720.

[43] M. Kalia, S. Garg, R. Shorey, Scatternet structure and inter-piconet communication in the bluetooth system, in: Proceedings of IEEE National Conference on Communications, 2000.

[44] R. Kapoor, M. Gerla, A zone routing protocol for bluetooth scatternets, in: IEEE Wireless Communications and Networking Conference, 2003, pp. 1459–1464.

[45] M. Kazantzidis, M. Gerla, On the impact of inter-piconet scheduling in bluetooth scatternets, in: Proceedings of the International Conference on Internet Computing, IC 2002, pp. 37–43.

[46] S. Keshav, An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network, Addison-Wesley, Reading, MA, 1999.

[47] R. Kraemer, personal communication with ceo of lesswire. Available from <http://www.lesswire.com>, 2003.

[48] A. Kumar, BlueHoc manual. Available from <http://oss.software.ibm.com/bluehoc/tutorial/docpage.html>.

[49] 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 2001, Long Beach, CA, USA, October 2001.

[50] C. Law, A.K. Mehta, K. Siu, A new bluetooth scatternet formation protocol, Journal of ACM Mobile Networks and Applications 8 (5) (2003) 485–498.

[51] C. Law, K. Siu, A bluetooth scatternet formation algorithm, Proceedings of the Global Telecommunications Conference, IEEE GLOBECOM '01, Vol. 5, 2001, pp. 2864–2869.

[52] Y.-Z. Lee, An efficient and fair polling scheme for bluetooth, Proceedings of MILCOM, Vol. 2, 2002, pp. 1062–1068.

[53] X. Li, I. Stojmenovic, Y. Wang, Partial Delaunay triangulation and degree limited localized bluetooth multihop scatternet formation, Draft, 2003.

[54] X. Li, P. Wan, Y. Wang, O. Frieder, Sparse power efficient topology for wireless networks, in: Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS '02), pp. 3871–3880.

[55] W. Lilakiatsakun, A. Seneviratne, Wireless home networks based on a hierarchical bluetooth scattenet architecture, in: Proceedings of the Ninth IEEE International Conference on Networks, pp. 481–485, 2001.

[56] T. Lin, Y. Tseng, K. Chang, C. Tu, Formation, routing and maintenance protocols for the bluering scatternet of bluetooths, in: Proceedings of the Hawaii International Conference on System Science (HICSS-36), Big Island, Hawaii, January 6, 2003.

[57] Y. Liu, M. Lee, T. Saadawi, On demand formation of bluetooth scatternet, Proceedings of MILCOM, Vol. 2, 2002, pp. 1069–1074.

[58] Y. Liu, M.J. Lee, T.N. Saadawi, A bluetooth scatternet-route structure for multihop ad hoc networks, Proceedings of the IEEE Journal on Selected Areas in Communications 21 (2) (2003) 229–239.

[59] R. Luo, R.M. Edwards, G.A. Manson, Enhancing bluetooth scheduler with predictive link capacity assignment plus multi-slot framing, Proceedings of the 10th International Conference on Telecommunications (ICT 2003), Vol. 2, 2003, pp. 959–964.

[60] M.A. Marsan, C.F. Chiasserini, A. Nucci, G. Carello, L. De Giovani, Optimizing the topology of bluetooth personal area networks, Proceedings of the IEEE Infocom, 2002.

[61] G. Miklos, A. Racz, Z. Turanyi, A. Valko, P. Johansson, Performance aspects of bluetooth scatternet formation, in: Proceedings of the First Annual Workshop on Mobile and Ad Hoc Net working and Computing, MobiHOC 2000, pp. 147–148.

[62] B. Miller, C. Bisdikian, Bluetooth Revealed: The Insider's Guide to an Open Specification for Global Wireless Communications, Prentice-Hall, Englewood Cliffs, NJ, 2000.

[63] V.B. Misic, J. Misic, Bluetooth scatternet with a master/slave bridge: a queueing theoretic analysis, Proceedings of the IEEE Global Telecommunications Conference, GLOBECOM '02, Vol. 1, 2002, pp. 207–211.

[64] V.B. Misic, J. Misic, Bridges of bluetooth county: topologies, scheduling and performance, Proceedings of the IEEE Global Telecommunications Conference, GLOBECOM '02, Vol. 1, 2002, pp. 207–211.
[65] V.B. Misic, J. Misic, Minimzing end-to-end delays in bluetooth scatternet with a slave/slave bridge, in: Proceedings of the Eleventh International Conference on Computer Communications and Networks, 2002, pp. 634–639.
[66] V.B. Misic, J. Misic, Modelling bluetooth piconet performance, IEEE Communications Letters 7 (1) (2003) 18–20.
[67] V.B. Misic, J. Misic, Performance of bluetooth bridges in scatternets with exhastive service scheduling, in: Proceedings of the 36th Annual Hawaii International Conference on System Sciences, January 6–9, 2003, pp. 311–320.
[68] N.A. Nordbotten, T. Sekie, N.D. Aavaag, Service discovery in bluetooth scatternets.
[69] NS: the network simulator. Available from <http://www.isi.edu/nsnam/ns/>.
[70] C.E. Perkins, Ad Hoc Networking, Addison-Wesley, Reading, MA, 2001.
[71] C. Petrioli, S. Basagni, Degree-constrained multihop scatternet formation for bluetooth networks, in: Proceedings of the IEEE Globecom 2002, Taipei, Taiwan, November 17–21, 2002.
[72] C. Petrioli, S. Basagni, I. Chlamtac, Bluemesh: Degree-constrained multihop scatternet formation for bluetooth networks, ACM/Kluwer/SPIE Mobile Networks and Applications 9 (1) (2004) 33–47 (Special Issue on Advances in Research of Wireless Personal Area Networking and Bluetooth Enabled Networks).
[73] B.J. Prabhu, A. Chocklingham, A routing protocol and energy efficient techniques in bluetooth scatternets, in: Proceedings of the IEEE International Conference on Communications, ICC 2002, Vol. 5, 2002, pp. 3336–3340.
[74] J. Rabaey, J. Ammer, J.L. da Silva Jr, D. Patel, Picoradio: ad hoc wireless sensor networking of ubiquitous low-energy sensor/monitor nodes, in: Proceedings of the IEEE Computer Society Annual Workshop on VSLI (WVLSI'00), Orlando, FL, 2000, pp. 9–12.
[75] A. Racz, G. Miklos, F. Bubinszky, A. Valko, A pseudo random coordinated scheduling algorithm for bluetooth scatternets, in: Proceedings of the 2001 ACM International Symposium on Mobile Ad Hoc Networking & Computing, 2001, pp. 193–203.
[76] L. Ramachandran, M. Kapoor, A. Sarkar, A. Arggwal, Clustering algorithms for ad hoc wireless networks, in: Proceedings of the DIAL-M Workshop, 2000, pp. 54–63.
[77] E.M. Royer, A review of current routing protocols for wireless networks, IEEE Personal Communications 6 (2) (1999) 46–55.
[78] K.V.S.S.S.S. Sairam, N. Gunasekaran, S.R. Redd, Bluetooth in wireless communication, IEEE Communications Magazine 40 (6) (2002) 90–96.
[79] T. Salonidis, P. Bhagwat, L. Tassiulas, Proximity awareness and fast connection establishment in bluetooth, in: Proceedings of the First Annual Workshop on Mobile and Ad Hoc Networking and Computing, MobiHOC 2000, pp. 141–142.
[80] T. Salonidis, P. Bhagwat, L. Tassiulas, R. LaMaire, Distributed topology construction of bluetooth personal area networks, in: Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM, Vol. 3, 2001, pp. 1577–1586.
[81] T. Salonidis, L. Tassiulas, Performance issues of bluetooth scatternets and other asynchronous tdma ad hoc networks, Technical Research Report CSHCN TR 2002-23, Centre for Satellite and Hybrid Communication Networks, University of Maryland, USA, 2002.
[82] V. Sangvornvetphan, T. Erke, Traffic scheduling in bluetooth network, in: Proceedings of the Ninth IEEE International Conference on Networks, 2001, pp. 355–359.
[83] T. Sato, K. Mase, A scatternet operation protocol for bluetooth ad hoc networks, Proceedings of the 5th International Symposium on Wireless Personal Multimedia Communications, Vol. 1, 2002, pp. 223–227.
[84] A. Seth, A. Kashyap, Capacity of bluetooth scatternets, 2002.
[85] Simjava. Available from <http://www.dcs.ed.ac.uk/home/hase/simjava>.
[86] L.T. Son, H. Schioler, O.B. Madsen, Predictive scheduling approach in inter-piconet communications, in: Proceedings of the 4th International Symposium on Wireless Personal Multimedia Communications, 2001.
[87] L.T. Son, H. Schioler, O.B. Madsen, Hybrid distributed iterative capacity allocation over bluetooth network, in: Proceedings of the 8th International Conference on Communication Systems (ICCS 2002), pp. 583–588.
[88] IEEE standards. Available from <http://standards.ieee.org/catalog/>.
[89] I. Stojmenovic, Dominating set based bluetooth scatternet formation with localized maintenance, in: Proceedings of the International Parallel and Distributed Processing Symposium, IPDPS 2002, pp. 148–155.
[90] M. Sun, C. Chang, T. Lai, A self-routing topology for bluetooth scatternets, in: Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks, I-SPAN '02, pp. 13–18.
[91] M.-T. Sun, S. Wang, C.-K. Chang, T.-H. Lai, H. Sawatari, H. Okada, Interference aware mac scheduling and sar policies for bluetooth scatternets, Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '02), Vol. 1, 2002, pp. 11–15.
[92] G. Tan, Interconnecting bluetooth-like personal area networks, in: Proceedings of the 1st Annual Oxygen Workshop, 2001.
[93] G. Tan, J. Guttag, A locally coordinated scatternet scheduling algorithm, in: Proceedings of the 27th Annual IEEE Conference on Local Computer Networks, LCN '02, 2002.
[94] G. Tan, A. Miu, J. Guttah, H. Balakrishnan, Forming scatternets for bleutooth personal area networks, MIT Technical Report, MIT-LCS-TR-826, 2001.

[95] G. Tan, A. Miu, J. Guttah, H. Balakrishnan, An efficient scatternet formation algorithm for dynamic environments, in: Proceedings of the IASTED Communications and Computer Networks (CNN), 2002.
[96] Texas Instruments. Available from <http://www.ti.com/rd/brf6100>.
[97] C.-K. Toh, M. Delwar, D. Allen, Evaluating the communication performance of an ad hoc wireless network, IEEE Transactions on Wireless Communications 1 (3) (2002) 402–414.
[98] Z. Wang, R.J. Thomas, Z. Haas, Bluenet—a new scatternet formation scheme, in: Proceedings of the 35th Annual Hawaii International System Sciences, HICSS '02, pp. 779–787.
[99] J.P.F. Willekens, Ad hoc routing in bluetooth, Lecture Notes in Computer Science, Vol. 2213, Springer, Berlin, 2001, pp. 130–144.
[100] R. Ait Yaid, G. Heijenk, Polling best effort traffic in bluetooth, Wireless Personal Communications 23 (2002) 195–206.
[101] D. Yang, G. Nair, B. Sivaramakrishnan, H. Jayakumar, A. Sen, Round robin with look ahead: a new scheduling algorithm for bluetooth, in: Proceedings of the International Conference on Parallel Processing Workshops, 2002, pp. 45–50.
[102] G.V. Zaruba, S. Basagni, I. Chlamtac, Bluetrees—scatternet formation to enable bluetooth-based ad hoc networks, Proceedings of the IEEE International Conference on Communications, ICC 2001, Vol. 1, 2001, pp. 273–277.
[103] W. Zhang, G. Cao, A flexible scatternet-wide scheduling algorithm for bluetooth networks, in: Proceedings of the 21st IEEE International Performance, Computing, and Communications Conference, 2002, pp. 291–298.
[104] B. Zhen, J. Parj, Y. Kim, Scatternet formation of bluetooth ad hoc networks, in: Proceedings of the 36th Annual Hawaii International Conference on System Sciences, HICSS '03, 2003, pp. 312–319.

Roger M. Whitaker holds a Ph.D. degree in Discrete Mathematics (1999) and a B.Sc. degree in Mathematics and Management Science. He is a lecturer and a co-director of the Centre for Mobile Communications, School of Computer Science, Cardiff University, UK. Prior to this position, he carried out research for the UK Radiocommunications Agency into spectrum efficiency. His research addresses the application of Computer Science to the design, co-ordination and optimisation of wireless networks and systems. He is currently leading a number of externally supported research projects in this area.

Leigh Hodge holds a Ph.D. in Computer Science (2002) and a B.Sc. degree in Computer Science (1995), both from the School of Computer Science, Cardiff University, Wales, UK. He is currently a research associate at the Centre for Mobile Communications at Cardiff University. His current research addresses the application of meta-heuristic and evolutionary approaches to the optimisation of wireless network design for Bluetooth and ad hoc networks.

Imrich Chlamtac holds a Ph.D. degree in computer science from the University of Minnesota. Since 1997 he has been the Distinguished Chair in Telecommunications at the University of Texas at Dallas and holds the titles of Sackler Professor at Tel Aviv University, Israel, The Bruno Kessler Honorary Professor at the University of Trento, Italy, and University Professor at the Technical University of Budapest, Hungary. He is a Fellow of the IEEE and ACM societies, a Fulbright Scholar and an IEEE Distinguished Lecturer. He is the winner of the 2001 ACM Sigmobile annual award and the IEEE ComSoc TCPC 2002 award for contributions to wireless and mobile networks, and of multiple best paper awards in wireless and optical networks. He has published over 300 papers in refereed journals and conferences, and is the co-author of the first textbook on Local Area Networks (Lexington Books, 1981, 1982, 1984) and of Mobile and Wireless Networks Protocols and Services (John Wiley, 2000). He serves as the founding Editor-in-Chief of the ACM/URSI/Kluwer Wireless Networks (WINET), the ACM/Kluwer Mobile Networks and Applications (MONET) journals and the SPIE/Kluwer Optical Networks (ONM) Magazine.