WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2003; **3**:517–537 (DOI: 10.1002/wcm.130)

A new BlueRing scatternet topology for Bluetooth with its formation, routing, and maintenance protocols

Ting-Yu Lin and Yu-Chee Tseng*,†

Department of Computer Science
and Information Engineering
National Chiao-Tung University
Hsin-Chu, 300
Taiwan

Keng-Ming Chang

Department of Computer Science and
Information Engineering
National Central University
Chung-Li, 320
Taiwan

Summary

The basic networking unit in Bluetooth is *piconet*, and a larger-area Bluetooth network can be formed by multiple piconets, called *scatternet*. However, the structure of scatternets is not defined in the Bluetooth specification and remains as an open issue at the designers' choice. It is desirable to have simple yet efficient scatternet topologies with good supports of routing protocols, considering that Bluetooths are to be used for *personal area networks* with design goals of simplicity and compactness. In the literature, although many routing protocols have been proposed for *mobile ad hoc networks*, directly applying them poses a problem due to Bluetooth's special baseband and MAC-layer features. In this work, we propose an attractive scatternet topology called *BlueRing*, which connects piconets as a ring interleaved by bridges between piconets, and address its formation, routing, and topology-maintenance protocols. The BlueRing architecture enjoys the following fine features. First, routing on BlueRing is stateless in the sense that no routing information needs to be kept by any host once the ring is formed. This would be favorable for environments such as Smart Homes where computing capability is limited. Second, the architecture is scalable to median-size scatternets easily (e.g. around $50 \sim 70$ Bluetooth units). In comparison, most star- or treelike scatternet topologies can easily form a communication bottleneck at the root of the tree as the network enlarges. Third, maintaining a BlueRing is an easy job even as some Bluetooth units join or leave the network. To tolerate single-point failure, we propose a protocol-level remedy mechanism. To tolerate multipoint failure, we propose a recovery mechanism to reconnect the BlueRing. Graceful failure is tolerable as long as no two or more critical points fail at the same time. As far as we know, the

*Correspondence to: Yu-Chee Tseng, Department of Computer Science and Information Engineering, National Chiao-Tung University, Hsin-Chu, 300, Taiwan.
†E-mail: yctseng@csie.nctu.edu.tw

fault-tolerant issue has not been properly addressed by existing scatternet protocols yet. In addition, we also evaluate the ideal network throughput at different BlueRing sizes and configurations by mathematical analysis. Simulation results are presented, which demonstrate that BlueRing outperforms other scatternet structures with higher network throughput and moderate packet delay.

KEY WORDS

ad hoc network
Bluetooth
mobile computing
personal area network (PAN)
piconet
routing
scatternet
wireless communication

1. Introduction

Wireless communication is perhaps the fastest growing industry in the coming decade. It is an enabling technology to make computing and communication anytime, anywhere possible. Depending on whether base stations are established or not, a wireless network could be classified as *infrastructure* or *ad hoc*. According to the radio coverage and communication distance, it can be classified as *wide area*, *local area*, *personal area*, or even *body area*.

This paper focuses on Bluetooth [1], which is an emerging PAN (Personal Area Network) technology, and is characterized by indoor, low-power, low-complexity, short-range radio wireless communications with a frequency-hopping (FH), time-division-duplex (TDD) channel model. Main applications of Bluetooths are targeted at wireless audio link, cable replacement, and ad hoc networking. The basic networking unit in Bluetooth is called *piconet*, which consists of one master and up to seven active slaves. For a larger widespread deployment, multiple piconets can be used to form a *scatternet*. A host may participate in two piconets to relay data, to which we refer as a *bridge* in this paper.

In the Bluetooth specification, the structure of scatternets is not defined, and it remains as an open issue at the designers' choice. In the literature, although many routing protocols have been proposed for *mobile ad hoc networks* based on wireless LAN cards [2], directly applying them poses a problem due to Bluetooth's special baseband and MAC-layer features [3]. It is desirable to have simple yet efficient scatternet topologies with good supports of routing protocols, considering that Bluetooths are to be used for PAN with design goals of simplicity and compactness. According to [4,5], Bluetooth-based mobile ad hoc networks need routing protocols closely integrated with underlying scatternet topologies. The reason stems from the physical and link-level constraints of Bluetooth technology, making legacy routing protocols for mobile ad hoc networks (e.g. [2]) unsuitable for scatternets.

Several previous papers [6–8] have addressed the performance issues, which motivate studies of the scatternet formation problem. References [9–12] propose various scatternet formation mechanisms (refer to the review in Section 2.4). However, all these works fail to provide clear and efficient routing protocols to run over the proposed scatternet topologies.

Until recently, Reference [13] proposed a routing protocol based on a two-level hierarchical scatternet. Two types of local networks are defined: *PAN* (*Personal Area Network*) and *RAN* (*Routing Area Network*). RAN is responsible for interconnecting PANs. All traffic from a PAN needs to go through the RAN to reach another PAN. However, this approach suffers from two drawbacks. First, the number of participating Bluetooth units is limited. Second, the RAN may become the bottleneck of the whole network, in terms of both communication delays and fault-tolerance capability.

In this work, we propose an attractive topology called *BlueRing* for scatternet structure, and address its formation, routing, and maintenance protocols. While similar to the IEEE 802.5 token ring in topology, our BlueRing differs from the token ring in several aspects owing to Bluetooth's special baseband features. First, the ring consists of multiple piconets with alternating masters and slaves and thus can *de facto* be regarded as a *ring of trees* since each master can connect to multiple active slaves. Second, no token actually runs on the ring. Third, since each piconet has its unique frequency-hopping sequence, multiple packets may be relayed on the ring simultaneously. Routing protocols to support unicast and broadcast on BlueRings are proposed. For bridges (slaves connecting two piconets), a bridging policy is clearly defined so as to relay packets efficiently.

The BlueRing architecture enjoys the following fine features. First, routing on BlueRing is stateless in the sense that no routing information needs to be kept or constructed by any host once the ring is formed. This would be favorable for environments such as Smart Homes where computing capability is limited. Second, the architecture is scalable to median-size scatternets (e.g. around $50 \sim 70$ Bluetooth units). In comparison, most star- or treelike scatternet topologies can easily form a communication bottleneck at the root of the tree as the network enlarges. Third, maintaining a BlueRing is an easy job even if some Bluetooth units join or leave the network. To tolerate single-point failure, we propose a protocol-level remedy mechanism. To tolerate multipoint failure, we propose a recovery mechanism to reconnect the BlueRing. Graceful failure is tolerable as long as no two or more critical points fail at the same time. As far as we know, the fault-tolerant issue has not been properly addressed by existing scatternet protocols yet.

The rest of this paper is organized as follows. Preliminaries are given in Section 2. The formation, routing, and maintenance protocols for BlueRing are proposed in Sections 3, 4, and 5, respectively. In Section 6, we present some analysis and simulation results. Finally, Section 7 summarizes the paper and points out our future work.

2. Preliminaries

2.1. Bluetooth Protocol Stack

Bluetooth is a master-driven, short-range radio wireless system. The smallest network unit is a *piconet*, which consists of one master and up to seven active slaves. Each piconet owns one frequency-hopping channel, which is controlled by its master in a TDD manner. A time slot in Bluetooth is 625 μs. The master always starts its transmission in an even-numbered slot, while a slave, on being polled, must reply in an odd-numbered slot.

Figure 1 shows the Bluetooth protocol stack. On top of RF is the Bluetooth Baseband, which controls the use of the radio. Four important operational modes are supported by the baseband: *active*, *sniff*, *hold*, and *park*. The active mode is the most energy consuming, where a Bluetooth unit is turned on for most of the time. The sniff mode allows a slave to go to sleep and only wake up periodically to check possible traffic. In the hold mode, a slave can temporarily suspend supporting data packets on the current channel; the capacity can be made free for other things, such as scanning, paging, inquiring, and even attending other piconets. Prior to entering the hold mode, an agreement should be reached between the master and the slave on the hold duration. When a slave does not want to actively participate in the piconet, but still wants to remain synchronized, it can enter the park mode (PM). The parked slave has to wake up regularly to listen to the beacon channel, for staying synchronized or checking broadcast packets.

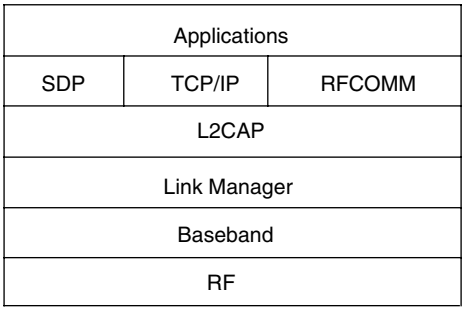


Fig. 1. Bluetooth protocol stack.

On top of Baseband is the Link Manager (LM), which is responsible for link configuration and control, security functions, and power management. The corresponding protocol is called Link Manager Protocol (LMP). The Logical Link Control and Adaptation Protocol (L2CAP) provides connection-oriented and connectionless datagram services to upper-layer protocols. Two major functionalities of L2CAP are protocol multiplexing and segmentation and reassembly (SAR).

The Service Discovery Protocol (SDP) defines the means for users to discover which services are available in their neighborhood and the characteristics of these services. The RFCOMM protocol provides emulation of serial ports over L2CAP so as to support many legacy applications based on serial ports over Bluetooth without any modification. Up to 60 serial ports can be emulated.

2.2. Operations of the Park Mode

This paper proposes a new topology called BlueRing for scatternet structure. Since a scatternet must involve multiple piconets, some devices must participate in more than one piconet. Such devices are called *bridges* in this paper, and a bridging policy is needed for them to efficiently relay packets from piconets to piconets. A bridge host has to frequently pause activities in one piconet and switch to another piconet. In this paper, we propose to adopt the PM for this purpose.

Below we give the reason why we choose park mode. The Bluetooth specification provides three options for a device to temporarily pause its current activity: sniff, hold, and park modes. The sniff mode has a periodical, prearranged wake-up pattern, and thus is more suitable for a device to switch from piconets to piconets with a regular pattern. It is not selected here because with a regular pattern the time slots may easily get wasted. Moreover, with our BlueRing, which chains a sequence of piconets, determining a good sniffing pattern is very difficult. The hold mode would be favorable if the amount of time that a bridge should stay in each piconet can be predetermined. Unfortunately, this assumption is unrealistic, especially in a dynamic environment. The PM is more favorable in our case since it allows a device to temporarily give up its current activity in one piconet for an arbitrary period of time until an unpark request is issued. The unpark request can be master-activated or slave-activated, but should be approved by the master. Hence, we adopt the PM in our bridging policy, considering its simplicity and flexibility.

In the following text, we review the PM operations in more detail. On entering the PM, a slave gives up its active member address AM_ADDR, but receives two new addresses:

- PM_ADDR: 8-bit Parked Member Address
- AR_ADDR: 8-bit Access Request Address

The PM_ADDR distinguishes a parked slave from the other parked slaves. This address is used in the master-initiated unpark procedure. In addition to the PM_ADDR, a parked slave can also be unparked by its 48-bit BD_ADDR (Bluetooth Device ADDRess). The all-zero PM_ADDR is a reserved address. If a parked unit has the all-zero PM_ADDR, it can only be unparked by the BD_ADDR. In that case, the PM_ADDR has no meaning.

The AR_ADDR is used by the slave in the slave-initiated unpark procedure. All messages sent to the parked slaves have to be carried by broadcast packets (the all-zero AM_ADDR) because they have no AM_ADDR. To support parked slaves, the master establishes a beacon channel when one or more slaves are parked. The beacon channel consists of one beacon slot or a train of equidistant beacon slots that is transmitted periodically with a constant time interval. The beacon channel is illustrated in Figure 2. In each period of T_B slots, a train of N_B ($N_B \geq 1$) beacon slots is transmitted. The start of the first beacon slot is referred to as the *beacon instant*, which serves as the timing reference. Parameters N_B and T_B are chosen such that there are sufficient beacon slots for

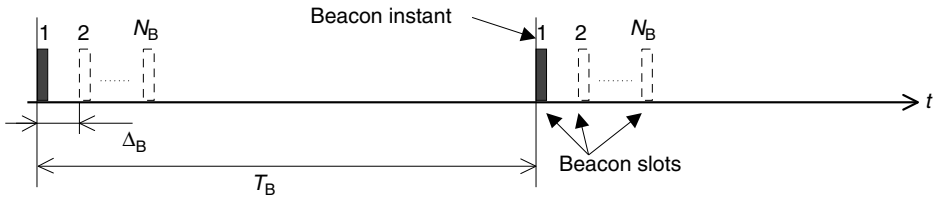


Fig. 2. General beacon channel format.

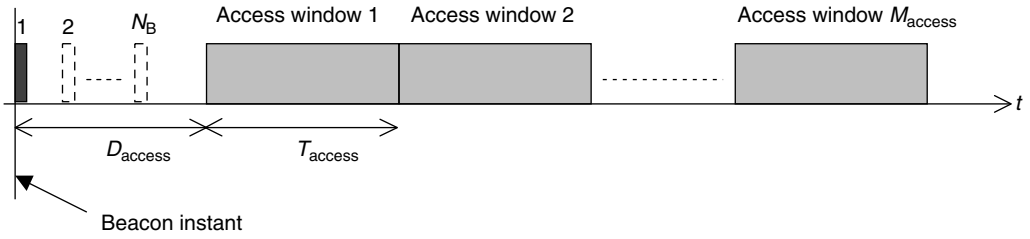


Fig. 3. Definition of access window.

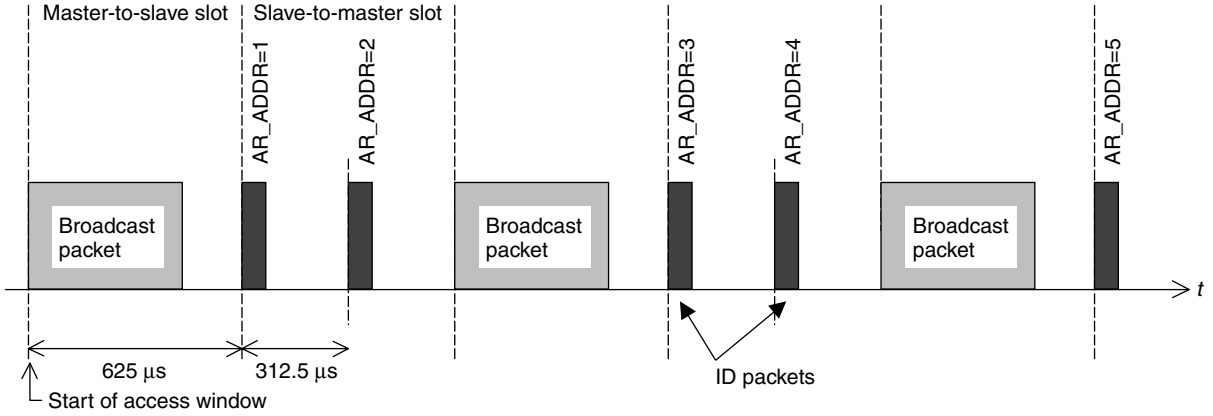


Fig. 4. Unpark request procedure in access window.

a parked slave to synchronize during a certain time window in an error-prone environment.

In addition to the beacon slots, an access window is defined where parked slaves can send unpark requests. To increase reliability, the access window can be repeated M_{access} times ($M_{\text{access}} \geq 1$), as shown in Figure 3. The first access window starts a fixed delay D_{access} after the beacon instant. The width of each access window is T_{access}. The same TDD structure is adopted by alternative transmission between the master and the slaves, as shown in Figure 4. However, the slave-to-master slot is divided into two half slots of 312.5 μs each. A parked slave should count the half slots and only in the proper half slot corresponding to its AR_ADDR may it respond. The parked slave is only permitted to send an unpark request if in the preceding master-to-slave slot a broadcast packet has been received. If the unpark request is approved, the master polls the parked slave.

2.3. Bluetooth Device Address and Access Codes

Following the IEEE 802 standard, each Bluetooth unit is assigned a unique 48-bit Bluetooth device address (BD_ADDR), which consists of three fields as follows:

- LAP: 24-bit lower address part,
- UAP: 8-bit upper address part, and
- NAP: 16-bit nonsignificant address part.

The clock and BD_ADDR of a master defines the FH sequence of its piconet. In addition, the LAP of a master's BD_ADDR determines the access code to be used within its piconet. In Bluetooth, each packet is preambled by a 68/72-bit access code. The access code is used mainly for synchronization and identification. Three types of access codes are available:

- Channel Access Code (CAC)
- Device Access Code (DAC)
- Inquiry Access Code (IAC)

The CAC is determined by the LAP of master's BD_ADDR and is used in connected mode, when piconets are already established. CAC, which precedes every packet, identifies all packets exchanged on the same piconet channel. Hence a Bluetooth unit can easily tell packets of its own piconet from others. The DAC is used for paging and responding to

paging. DAC is derived from the LAP of the paged device's BD_ADDR. The IAC is used in inquiry process. Two variants of IAC are supported:

- General Inquiry Access Code (GIAC) and
- Dedicated Inquiry Access Code (DIAC).

There is only one GIAC, which is for discovering all Bluetooth devices in range. However, there are 63 DIACs, each for discovering a specific class of Bluetooth units. Since IACs are derived from LAPs, a block of 64 contiguous LAPs is reserved for this purpose. One LAP is used for general inquiry, and the remaining 63 LAPs are reserved for dedicated inquiries. None of these LAPs can be part of a Bluetooth unit's BD_ADDR.

We will take advantage of the reserved DIACs in our BlueRing recovery procedures for fault tolerance.

2.4. Review of Scatternet Formation Algorithms

Below, we review some existing scatternet formation schemes. The work in [6] has studied one piconet with both active and park slaves. Slaves are switched between active and PMs based on the time stamps when they entered a state. A parked slave with the oldest time stamp is periodically unparked by parking the active slave with the oldest time stamp. This model suffers from low system throughput and long packet delays, and incurs many mode-switching overheads when the number of slaves is large. Also, with one piconet, the multichannel benefit of using scatternet is not exploited. According to [7], by grouping nodes into different piconets, significant performance improvement may be obtained. The reason is that simultaneous communications can occur among different piconets. However, since different FH channels of different piconets may still present radio interference, the Bluetooth Whitepaper [14] has suggested that up to eight piconets may coexist in the same physical environment.

Scatternet formation is explored in [9–12]. In [10], a two-stage distributed randomized algorithm is proposed to form a network of star-shaped clusters, where each cluster is a piconet with at most seven active slaves. The goal is to maximize the number of nodes into each piconet so that the number of clusters is minimized. However, how these piconets are interconnected is not addressed. A similar work is in [12], where a fast scatternet formation algorithm is proposed to connect piconets as a tree. How to form a treelike scatternet with bounded time and message complexities is presented in [9]; there is no limitation on the number of participant nodes. Clearly, the center/root host in a star/tree scatternet can become a communication bottleneck of the network. Furthermore, designing fault-tolerant routing protocols on a star/treelike network is a difficult job since any single fault will partition the network. In [11], assuming that all devices are within each other's radio coverage, a fully connected scatternet is constructed such that connectivity exists between each pair of piconets. At most 36 Bluetooth devices can participate in the scatternet.

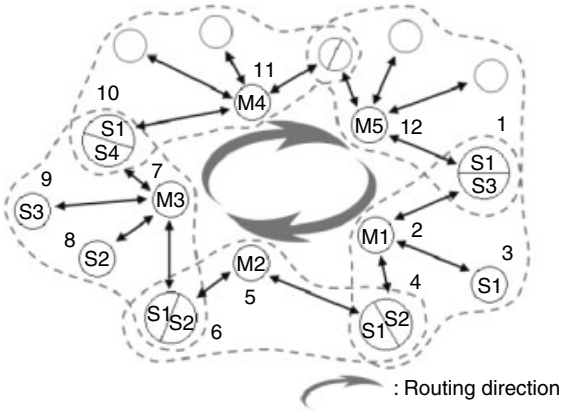
We note that all the above works [9–12] do not clearly address the corresponding routing and bridging protocols to be run over the proposed scatternets. While there are no standard criteria for good scatternet topologies, we conclude some guidelines for scatternet construction:

- The number of piconets should be kept as small as possible, so as to reduce interpiconet interference and communication complexity.
- A node should participate in at most two piconets, so as to reduce switching overheads.
- To reduce redundant interpiconet links, two piconets should not be connected by more than one bridge.
- Simple and efficient routing.
- Good mobility- or fault-tolerant capability.

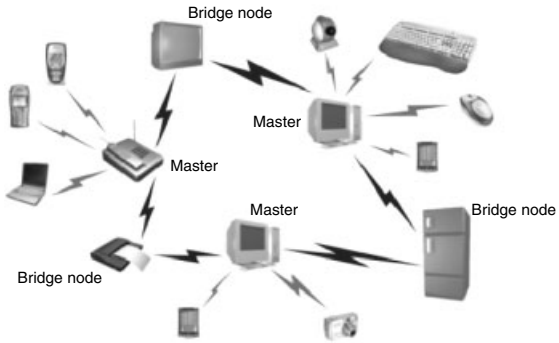
3. The BlueRing Formation Protocol

3.1. Network Architecture

In this subsection, we propose the BlueRing structure. A BlueRing is a scatternet consisting of a cycle of piconets that form a ring. Although physically the ring is undirected, logically we impose a direction on it (say, clockwise). So each piconet has a *downstream piconet* in the forward direction, and an *upstream piconet* in the backward direction. Packets will flow following the direction of the ring, until destinations are reached. In each piconet, two of the slaves are designated as *bridges*, one for connecting to the upstream piconet, called the *upstream bridge*, and one for connecting to the downstream piconet, called the *downstream bridge*. For instance, as shown in Figure 5, nodes 4 and 6 are upstream and downstream bridges of master M2, respectively. Similarly, each bridge also has an upstream and a downstream master. Therefore, each bridge host serves as an upstream bridge in one piconet and a downstream bridge in



(a)



(b)

Fig. 5. (a) The BlueRing architecture with nodes 2, 5, 7, 11, and 12 serving as masters and (b) a BlueRing example.

another piconet, and each piconet should have at least two slaves. Figure 5 illustrates the BlueRing architecture and a real example.

3.2. Initial Formation

In order to construct a BlueRing, we adopt a centralized formation mechanism similar to [11]. Assume that all Bluetooth devices are within the radio coverage of each other. We define a parameter RING_MEM for each Bluetooth to indicate whether it has become a member of the BlueRing or not (1 for 'yes', 0 for 'not yet'). Initially, RING_MEM equals 0. The construction has two stages.

- **Stage I**: Each Bluetooth chooses to perform inquiry (I) with probability p and inquiry scan (IS) with probability $1 - p$. When some I matches with some IS, the two Bluetooths establish a temporary piconet. Three parameters are exchanged between them: RING_MEM, number of acquired Bluetooths,
and BD_ADDR. First, their RING_MEMs are compared. The one with RING_MEM $= 1$ wins if the other's RING_MEM $= 0$. In case of a tie, the one that has gathered more Bluetooths' information wins. If the above cannot determine a winner, the tie is broken by their unique BD_ADDRs. The loser should provide the winner with all Bluetooths' information it has gathered. After the information exchange, the temporary piconet is torn down. The potential winner can claim itself as a leader if no further I/IS message is received within an *inquiry time-out (IT)*. Then the (potentially only) leader enters the page state, trying to collect other nonleaders, which must enter the page scan state, waiting to be paged. The details are in Stage II.
- **Stage II**: On the basis of the desired ring topology, the leader designates several Bluetooths as masters by paging them and setting up a temporary piconet. For each designated master, the leader provides it with the information of its slaves, including assigned downstream and upstream bridges. Upon receiving such information, each master pages its slaves and establishes its piconet. A unit serving as a bridge should make sure that both its downstream and upstream masters have connected to it properly. On becoming part of the ring, a Bluetooth sets its RING_MEM to 1.

The resultant BlueRing is quite fault-tolerable and scalable. We will discuss the maintenance protocol to handle Bluetooths leaving and joining in Section 5.

4. The BlueRing Routing Protocol

4.1. Unicast and Broadcast

In this subsection, we propose a routing protocol, which supports both unicasting and broadcasting on BlueRing. As mentioned earlier, data packets will be routed following the direction of the BlueRing. Since a packet flowing around the ring will eventually reach its destination piconet, no route discovery process is required. So routing on BlueRing is stateless since no routing table needs to be maintained (on the contrary, most routing protocols for ad hoc networks need to keep routing tables [2]).

To understand how packets are routed, we need to discuss the packet formats in more detail. The general Bluetooth baseband data packet format is shown in Figure 6. Each packet has a 72-bit access code, which can uniquely identify a piconet, followed

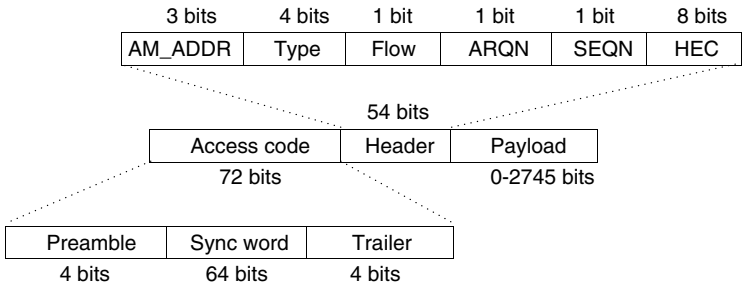


Fig. 6. General baseband packet format.

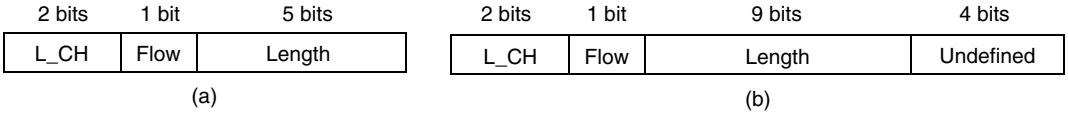


Fig. 7. Payload header formats: (a) single-slot packets and (b) multislot packets.

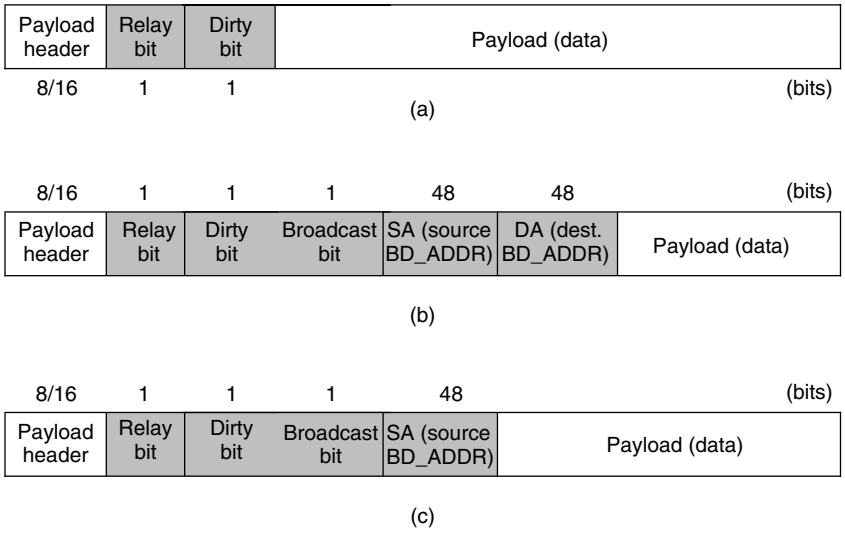


Fig. 8. Payload formats in BlueRing: (a) single-hop unicast communication, (b) multihop unicast communication, and (c) scatternet broadcast communication. The fields in gray are those added by BlueRing.

by a header and a payload. The header carries 18 bits of information, and is encoded by the 1/3 FEC (Forward Error Correction) code, resulting in a 54-bit header. The payload can range from 0 to 2745 bits.

Data packets supported by the ACL (Asynchronous ConnectionLess) link can occupy 1, 3, or 5 time slots. Type DM1/DH1 packets cover a single time slot, type DM3/DH3 packets cover three slots, and type DM5/DH5 packets cover five slots. On the payload field, there is also a payload header. Bluetooth adopts different payload headers for single-slot and multislot packets. Figure 7 details the formats of payload headers.

To route packets on our BlueRing, several control bits should be appended after the payload header. There are three formats for the payload field in BlueRing, depending on their communication types, as shown in Figure 8. These fields (in gray) are explained below.

- **Broadcast bit**: This bit distinguishes broadcast packets from unicast ones. This bit is set to TRUE if the packet needs to be disseminated throughout the whole BlueRing, and set to FALSE otherwise.
- **Relay bit**: For single-hop unicasting, the relay bit is set to FALSE, indicating that no relaying is needed to reach the destination. By observing a

packet with relay bit set to FALSE, a Bluetooth unit accepts the packet. For multihop unicasting packet, its relay bit is set to TRUE, indicating that the packet needs to be relayed to reach the destination. The packet continues to be relayed along the ring, until some master discovers that either the destination is itself or the destination belongs to its piconet. In the former case, the master accepts the packet. In the latter case, the master sets the relay bit to FALSE and forwards the packet to the destination slave. On seeing a packet with relay bit = FALSE, the slave realizes that this is a packet destined for itself, and thus accepts the packet. For broadcasting, if the source is a master, the relay bit is initialized to FALSE and the packet is broadcast to its piconet. All slaves within the piconet will accept the packet. On examining the packet content, the downstream bridge of the sending master will determine that it is a broadcast packet and needs to be further relayed. The downstream bridge will set the relay bit to TRUE and forwards the packet to its downstream master to continue broadcasting. The relay bit (= TRUE) is to inform the downstream master that the received packet should be relayed further. The downstream master should set the relay bit back to FALSE and broadcast the packet to its piconet. This procedure is repeated to disseminate the packet over the BlueRing. On the other hand, if the source is a slave, the relay bit is initialized to TRUE and the broadcast packet is sent to its master for broadcasting.

- **Dirty bit**: For single-hop unicasting, the dirty bit is set to FALSE. Along with the relay bit (= FALSE), the receiving side can easily tell that the packet is directly from the sending host. For multihop unicasting and broadcasting, the dirty bit is to detect the presence of orphan packets (with missing receiver) or duplicate broadcast to prevent packets from endlessly circulating on the BlueRing. Whenever a master touches a packet, the dirty bit is set to TRUE before relaying it to the next hop. This dirty bit, together with the source BD_ADDR, is to identify if a packet has traveled throughout the whole ring or not. If the packet has already traveled around the whole ring once, it is removed from the network to avoid unnecessary circulation. When first initiated, a packet sent by a slave has dirty bit = FALSE, and a packet sent by a master has dirty bit = TRUE. However, the former's dirty bit will be changed to TRUE once being relayed by the first master.
- **SA**: This field contains a 48-bit source Bluetooth Device ADDRess (BD_ADDR).
- **DA**: This field contains a 48-bit destination Bluetooth Device ADDRess (BD_ADDR).

The formal BlueRing routing protocol is provided in Figures 9 and 10. Figure 9 illustrates the operations to be taken by a slave upon receiving a packet. If the relay bit is 1, the packet is directly forwarded to the downstream master without further examining its content. If the relay bit is 0, the packet should be accepted and forwarded to the upper layer. The dirty bit can be used to determine the origin of the packet. If the dirty bit is 0, this packet is directly from the master; otherwise, it has been relayed and its origin can be found from the field SA. In addition, for those slaves acting as downstream bridges, they should check the broadcast bit. If the broadcast bit is 1, the bridge makes a copy of the packet (forwarding it to the upper layer), and sets relay bit to 1. Then the bridge forwards the packet to its downstream master.

Figure 10 shows the operations to be taken by a master upon receiving a packet. If the relay bit is 0, the packet is accepted. Otherwise, we check the dirty bit. For a packet with dirty bit = 1, we need to check if the SA (source host address) is in this piconet (including the master itself). If so, this is an orphan packet or duplicate broadcast and should be deleted from the network. Also, whenever the first master touches a packet with dirty bit = 0, the dirty bit is set to 1 (so as to detect future orphan packets/duplicate broadcasts). Then, depending on the broadcast bit, the master proceeds as follows. For a broadcast packet, the master sets the relay bit to 0, and broadcasts it to its piconet. For a unicast packet, the master needs to decide whether the packet should be forwarded or not. If the DA field is equal to the master itself, the packet is accepted. Otherwise, the DA field is compared with the list of BD_ADDRs belonging to this piconet. If so, the packet is forwarded to host DA in the local piconet; otherwise, the packet is forwarded to the downstream bridge.

Below, we demonstrate several routing examples based on the network in Figure 5(a). Figure 11(a) illustrates the packet contents for a single-hop unicast from node 8 to node 7 in piconet M3. Figures 11(b and c) are multihop unicasts from node 8 to node 10 and from node 3 to node 9, respectively; the former is an intrapiconet communication, and the latter an interpiconet communication. Finally, Figure 11 (d) illustrates the packet contents for a scatternet broadcast originated at node 5.

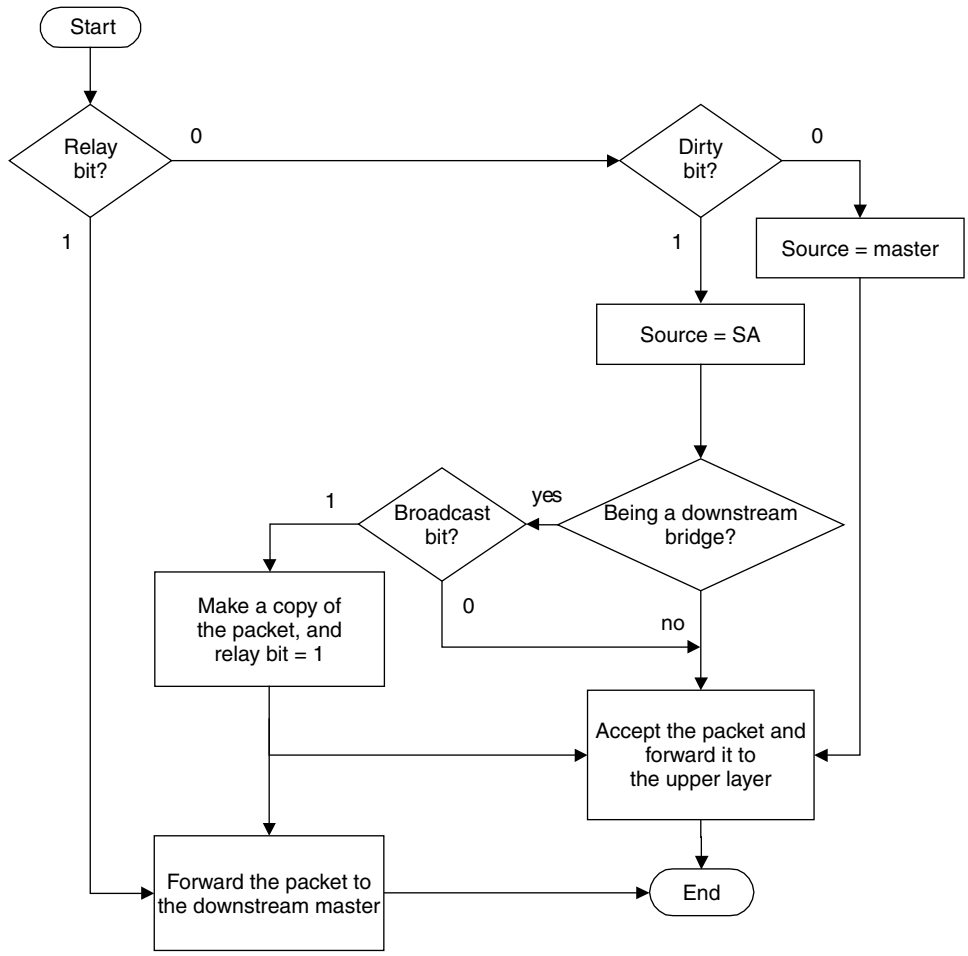


Fig. 9. The BlueRing routing protocol for slaves.

4.2. Bridging Policy

In BlueRing, a bridge host should participate in two piconets. Since Bluetooth is a TDD, FH radio system, a unit can only stay in one piconet at one time. In BlueRing, we propose to use parking and unparking for bridges to switch between piconets. We give the reason why we choose PM as follows. The Bluetooth specification provides three options for a device to temporarily pause its current activity: sniff, hold, and park modes. The sniff mode has a periodical, prearranged wake-up pattern, and thus is more suitable for a device to switch from piconets to piconets with a regular pattern. It is not selected here because with a regular pattern the time slots may easily get wasted. Moreover, with our BlueRing, which chains a sequence of piconets, determining a good sniffing pattern is very difficult. The hold mode would be favorable if the amount of time that a bridge should stay in each piconet can be predetermined. Unfortunately, this assumption is unrealistic, especially in a dynamic environment. The PM is more favorable in our case since it allows a device to temporarily give up its current activity in one piconet for an arbitrary period of time until an unpark request is issued. The unpark request can be master-activated or slave-activated, but should be approved by the master. Hence, we adopt the PM in our bridging policy, considering its simplicity and flexibility.

Below, we propose the bridging policy used in our BlueRing. A threshold-based strategy is adopted to initiate park/unpark requests. Three parameters are used in the bridging policy: (i) T_b: a threshold value to evaluate the queued packets in a bridge, (ii) T_m: a threshold value to evaluate the queued packets in a master, and (iii) T_{out}: a time-out value to evaluate when a bridge should switch piconets. Intuitively, under normal situations, a bridge will connect to its upstream piconet for most of the time. When some threshold conditions become true, it will switch

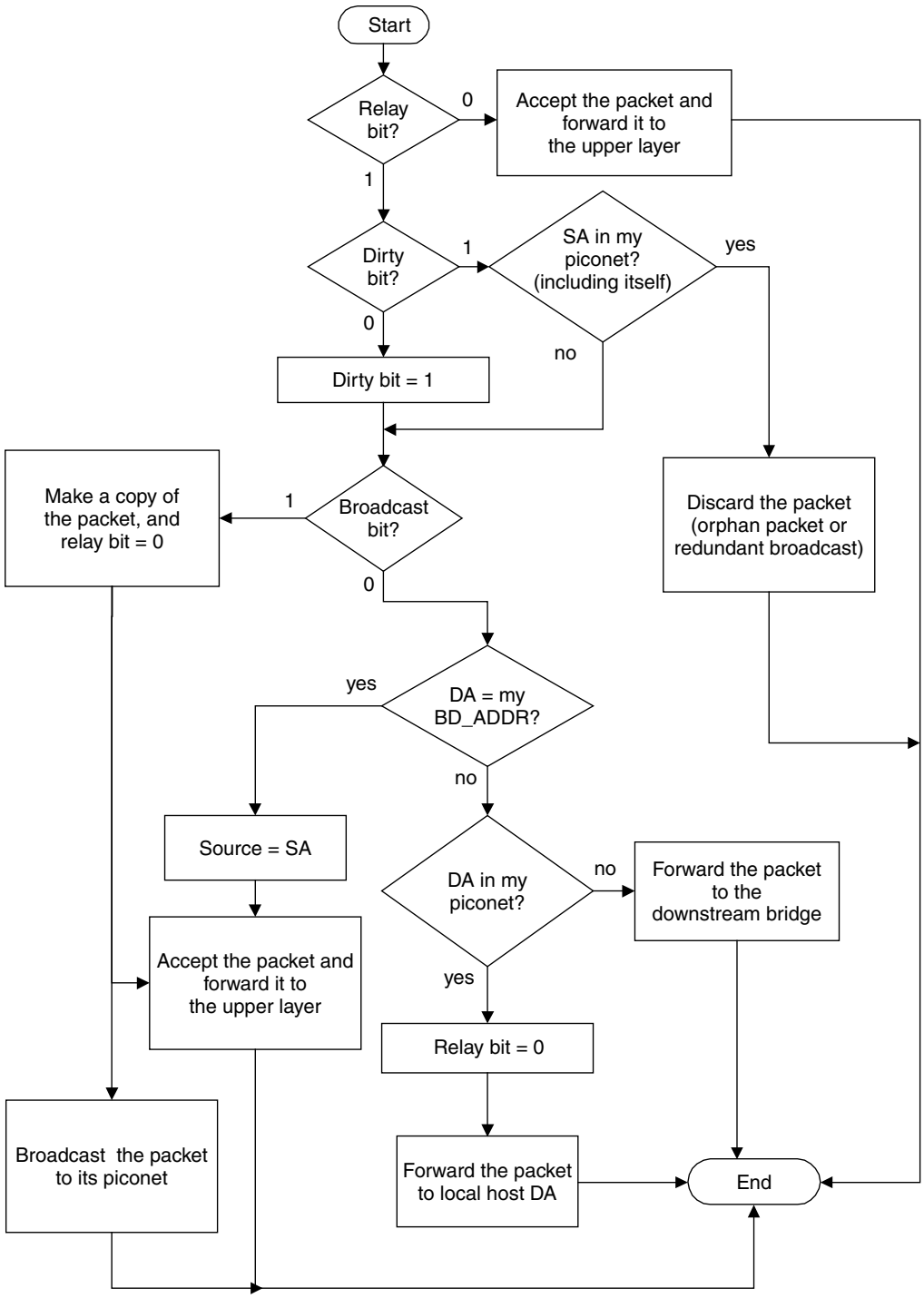


Fig. 10. The BlueRing routing protocol for masters.

to its downstream piconet. Once connected to its downstream piconet, the bridge will be treated with higher priority by its downstream master so as to drain the packets in its buffer. The detailed switching strategy is described below.

- **From upstream to downstream**: A bridge connecting to its upstream piconet should switch to its downstream piconet when (i) the number of queued packets to be relayed exceeds T_b or (ii) the clock T_{out} expires. In this case, a park request

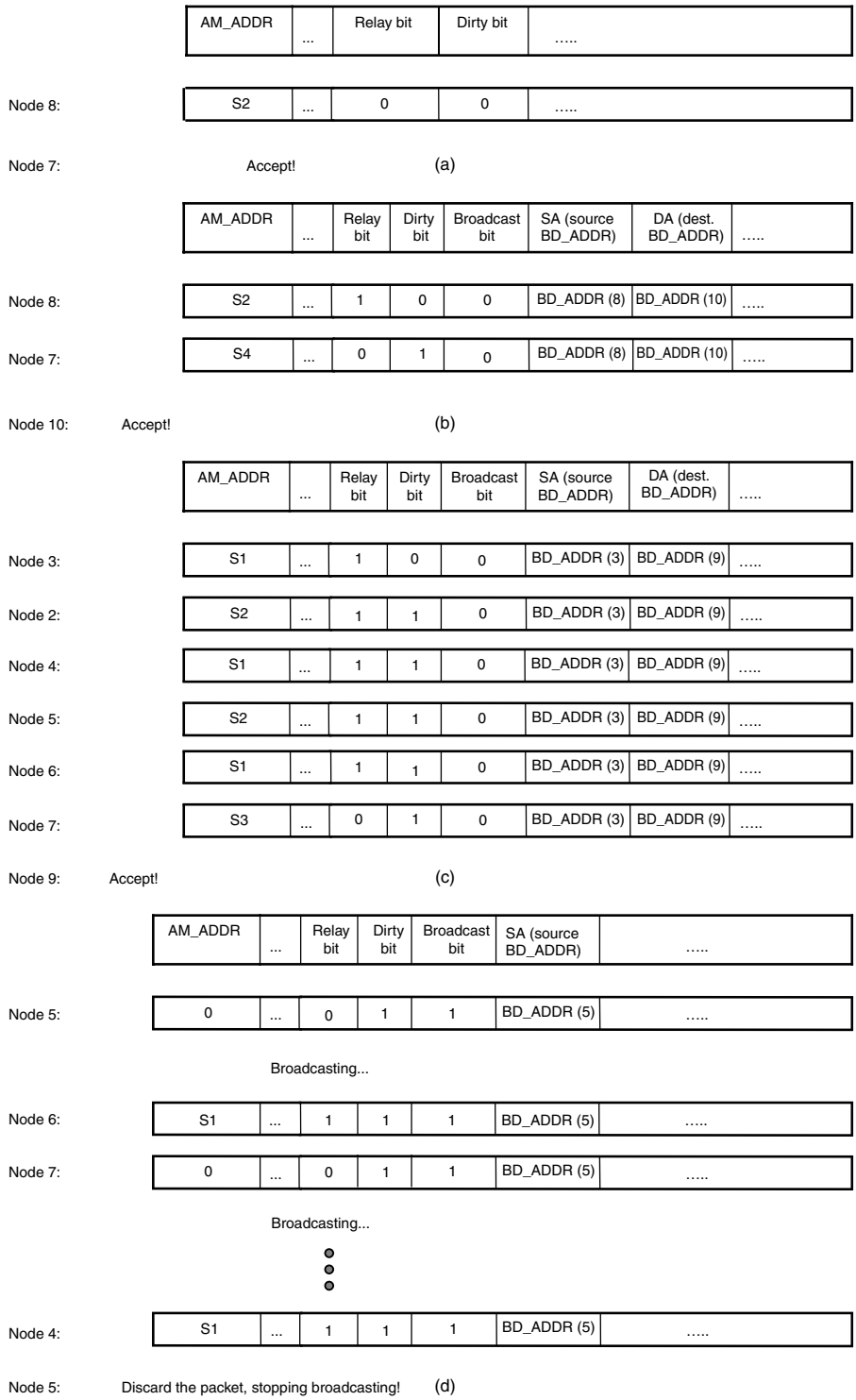


Fig. 11. BlueRing routing examples: (a) intrapiconet one-hop unicast (node 8 to node 7), (b) intrapiconet two-hop unicast (node 8 to node 10), (c) interpiconet multihop unicast (node 3 to node 9), and (d) scatternet broadcast originated at node 5.

should be sent to its upstream master and an unpark request should be sent in the next available access window in the downstream piconet. The downstream master should treat this bridge with higher priority to drain its buffered packets.

- **From downstream to upstream**: A bridge connecting to its downstream piconet should switch to its upstream piconet when (i) all its buffered packets have been drained by its downstream master or (ii) its upstream master has queued a number of packets exceeding the threshold T_m. In the former case, the unparking request is initiated by the bridge itself, while in the latter case, the unparking request is initiated by the upstream master to call the slave back. A bridge called by its upstream master should park its current piconet immediately, and switch to the calling piconet channel.

5. The BlueRing Maintenance Protocol

Fault tolerance is an essential issue in packet routing, especially under a mobile environment. In BlueRing, when any master or bridge leaves the network, the ring will become broken and reduce to a linear path. New Bluetooth units may join the network. In this section, we show how to maintain a BlueRing. To tolerate single-point failure, Section 5.1 proposes a protocol-level remedy mechanism. To tolerate multipoint failure, Section 5.2 proposes a recovery mechanism to reconnect the BlueRing. Note that the former one does not try to reestablish the BlueRing, so the network may become a linear path. The latter one will conduct local reconnection. Graceful failure is tolerable as long as no two or more critical points fail at the same time.

5.1. Single-point Failure

Suppose that one host serving as a master or a bridge fails. Since there is a default routing direction on BlueRing, a host is unable to reach another host in the backward direction if packets are always sent in the forward direction. The basic idea here is to add a new control bit called *Direction* after the payload header. This bit assists hosts to determine which routing direction (forward/backward) to be followed. Below, we summarize the necessary enhancements on our BlueRing protocol for the fault-tolerant routing.

- The default value for *Direction* is 0, which indicates the forward direction. When a master/bridge on the BlueRing detects that the next hop on the ring is not existing any more, it simply sets $Direction = 1$ and relays the packet backwards.
- Any master/bridge on receiving a packet with $Direction = 1$ should relay the packet in the backward direction.
- The condition for discarding orphan packets should be revised as follows. Observe that a packet with $Direction = 1$ may reach the source piconet more than once. It is erroneous to discard such a packet by the source piconet master on observing $Dirty = 1$. In this case, the packet should be allowed to continue traveling on the ring until the destination is reached or the other end of the BlueRing is reached. Therefore, the condition for determining a packet to be an orphan should be done by a master/bridge with no upstream node when observing a packet to be undeliverable with $Direction = 1$.
- (Optional) One optimization that can be done here is to have each master keep a list of destination addresses that are unreachable on the forward direction of the BlueRing. On seeing a packet destined to any host in the list, the packet can be sent directly on the backward direction to save communication bandwidth.

Note that if the failure point is a bridge, the whole network remains connected. If a master fails, the nonbridge slaves of the master will become orphans. The other masters should execute the inquiry process from time to time to collect such orphan slaves. The details are in the next subsection.

5.2. Multipoint Failure

The fault-tolerant routing protocol proposed above ensures that routing is unaffected, but leaving the broken point unfixed. In this subsection, we propose a recovery mechanism that can reconnect the network as a BlueRing. New hosts can join an existing BlueRing too. Only local reconnection is required. As long as no two critical points fail simultaneously, the protocol can work correctly.

Recall that Bluetooth provides 63 reserved DIACs for discovering certain dedicated units in range. Here, we propose to use two reserved DIACs, say DIAC1 and DIAC2, to facilitate BlueRing recovery. Also, the GIAC will be used to invite new hosts to join an existing BlueRing.

Below, we show how to manage cases of bridge leaving and master leaving. In some cases we will have to extend a BlueRing by creating more piconets.

- **Bridge missing**: When a bridge leaves, its downstream master performs DIAC1 inquiry, hoping to connect with another upstream bridge. On the other hand, the leaving bridge's upstream master checks if it has any other nonbridge slave that can serve as its new downstream bridge. If this is the case, the master notifies this bridge and commands it to enter DIAC1 inquiry scan. Otherwise, the upstream master should tear down its current piconet and wait to be discovered by other masters. This case will induce a missing master, which can be cured by the '**Master Missing**' procedure in the subsequent paragraph. Figure 12 illustrates the recovery procedures when the bridge node 4 leaves. Upstream master node 2 reassigns node 3 as its new downstream bridge and then node 3 enters DIAC1 inquiry scan. Meanwhile, the downstream master node 5 performs DIAC1 inquiry, and discovers node 3. A new connection is formed between nodes 5 and 3, healing the BlueRing.
- **Master missing**: When a master leaves, all of its slaves, except the downstream and upstream bridges, become orphans. The downstream bridge of the leaving master should change its state to a nonbridge and inform its downstream master to perform DIAC1 inquiry, in hope of finding a new upstream bridge. On the other hand, the upstream bridge of the leaving master enters DIAC1 inquiry scan, hoping to be discovered. Figure 13 shows the scenarios when master node 2 leaves, leaving node 3 isolated from the ring. The downstream bridge node 4 reduces a nonbridge slave and informs its downstream master node 5 to execute DIAC1 inquiry. Upstream bridge node 1 starts DIAC1

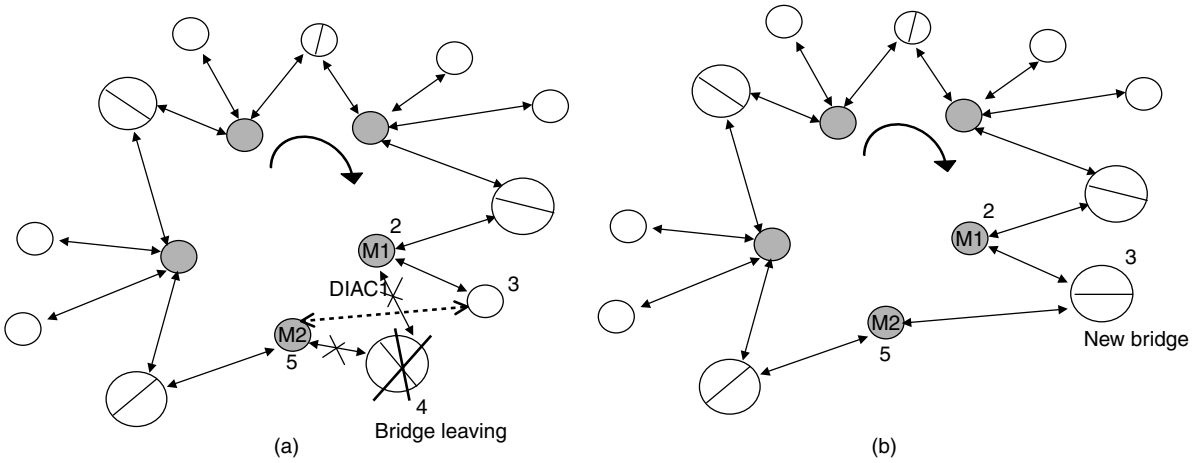


Fig. 12. An example to illustrate bridge-leaving recovery procedures: (a) DIAC1 discovering and (b) the reconnected BlueRing.

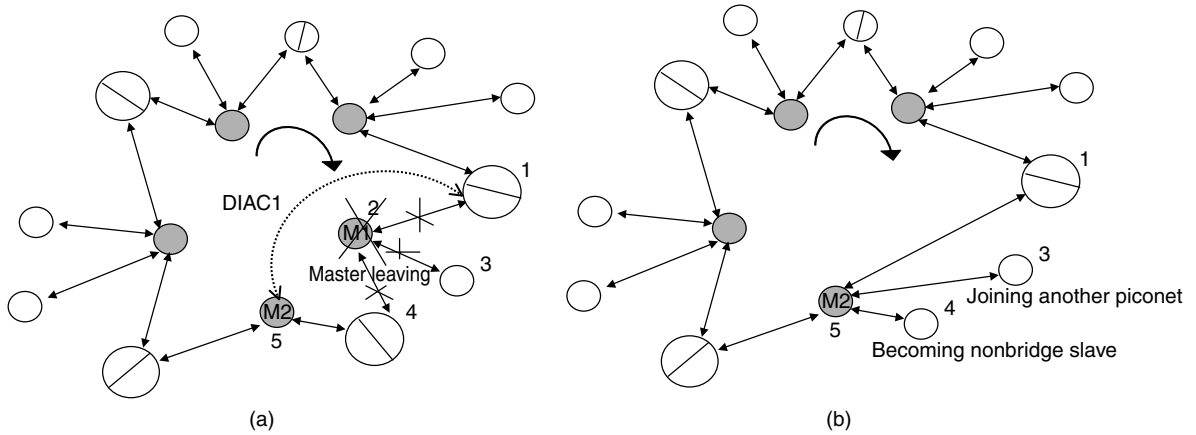


Fig. 13. An example to illustrate master-leaving recovery procedures: (a) DIAC1 discovering and (b) the reconnected BlueRing.

inquiry scan, and is discovered by node 5. A new connection is established between nodes 5 and 1. Moreover, by GIAC inquiry/inquiry scan, node 3 is discovered and invited to join node 5's piconet, thus becoming part of the ring again. So the BlueRing is reconnected.

- **Piconet splitting**: Recall that in order to include more Bluetooths into the BlueRing, each master should execute GIAC inquiry from time to time. This happens when new Bluetooth units join or orphan Bluetooth units appear. When the number of slaves belonging to a master exceeds a certain limit, we will split it into two piconets. The procedure is as follows. Assume that the desirable maximum number of slaves per piconet is α ($\alpha \geq 4$). Whenever the number of slaves a master possesses reaches α, the master sends out a *split_request* token to obtain split permission from all other masters. The *split_request* packet traverses the ring to ensure that concurrent splitting operations on the BlueRing do not exist. Once the split request is approved by all piconets on the ring, the master detaches its upstream bridge and two nonbridge slaves (this must succeed since $\alpha \geq 4$). Then the master starts DIAC2 inquiry. Upon discovering that the master is missing, the upstream bridge enters DIAC1 inquiry scan in search of new downstream master. On the other hand, one of the two detached nonbridge slaves is designated as new master and is provided with the information of the other detached slave, which is informed to enter page scan. The former will then page the latter, thus setting up a new piconet consisting of two members (one master and one slave). The new master designates its only slave as its downstream bridge, and starts DIAC1 inquiry to discover an upstream bridge. Meanwhile, the new master orders its downstream bridge to enter DIAC2 inquiry scan. Figure 14 shows a splitting example, assuming $\alpha = 4$. After obtaining the permission to split, master node 7

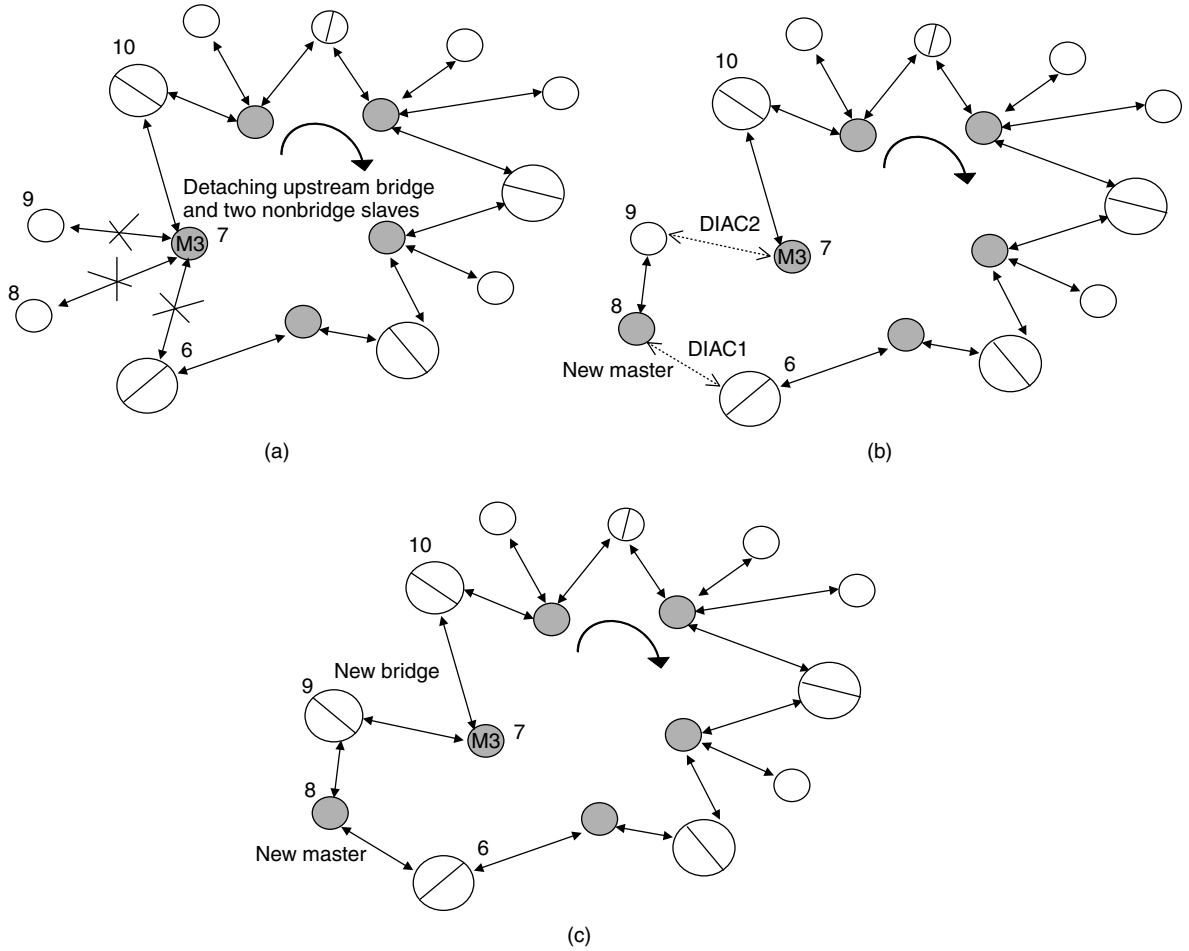


Fig. 14. A BlueRing extension example with $\alpha = 4$.

disconnects from nodes 6, 8, and 9, and designates node 8 as the new master. Immediately, node 8 sets up a new piconet with node 9, which serves as its downstream bridge. Then node 8 discovers node 6 by DIAC1 inquiry, while node 7 discovers node 9 via DIAC2 inquiry. The ring is now connected again with one more piconet. By creating more piconets, more new Bluetooths can join the ring, making BlueRing extensible.

We remark on two points. First, every split operation needs to detach two nonbridge slaves. In addition to downstream and upstream bridges, only piconets with no less than four slaves can request for splitting. This explains why we enforce $\alpha \geq 4$. Second, the *split_request* token should compete with other potential *split_request* tokens while traveling on the ring. This can be easily done by allowing a requesting master with a higher BD_ADDR to inhibit other requesting masters' tokens with lower BD_ADDRs.

6. Analysis and Simulation Results

In this section, we first evaluate the maximum achievable throughput of the BlueRing. Let the ring contain R piconets ($R \geq 2$) and each piconet contain S slaves (including both nonbridge slaves and bridges). So the total number of Bluetooths on the ring is $S \cdot R$. We shall derive the average number of hops that a data packet needs to travel before reaching its destination. This will depend on the role of the source, which can be master, bridge slave, and nonbridge slave. Table 1 summarizes these three cases. In the tables, destinations are classified into ring-body, intrapiconet-slave, and interpiconet-slave. Note that the ring-body contains all masters and bridges, and the rest of the slaves are classified into intra- and interpiconet cases.

On the basis of Table 1, we derive the average distances d_s, d_m, and d_b for packets originated at a nonbridge slave, master, and bridge, respectively, as follows:

$$d_s = \frac{c_{s1} \cdot d_{s1} + c_{s2} \cdot d_{s2} + c_{s3} \cdot d_{s3}}{c_{s1} + c_{s2} + c_{s3}} = \frac{SR^2 + SR - 2}{SR - 1} \quad (1)$$

$$d_m = \frac{c_{m1} \cdot d_{m1} + c_{m2} \cdot d_{m2} + c_{m3} \cdot d_{m3}}{c_{m1} + c_{m2} + c_{m3}} = \frac{SR^2 - R}{SR - 1} = R \quad (2)$$

Table I. Routing distances for packets originated at (a) a nonbridge slave, (b) a master, and (c) a bridge.

	Number of destinations	Distance
(a) Source is a nonbridge slave		
Intrapiconet-slave	$C_{s1} = S - 3$	$d_{s1} = 2$
Ring-body	$C_{s2} = 2R$	$d_{s2} = R + 1$
Interpiconet-slave	$C_{s3} = (S - 2)(R - 1)$	$d_{s3} = R + 2$
(b) Source is a master		
Intrapiconet-slave	$C_{m1} = S - 2$	$d_{m1} = 1$
Ring-body	$C_{m2} = 2R - 1$	$d_{m2} = R$
Interpiconet-slave	$C_{m3} = (S - 2)(R - 1)$	$d_{m3} = R + 1$
(c) Source is a bridge		
Intrapiconet-slave	$C_{b1} = S - 2$	$d_{b1} = 2$
Ring-body	$C_{b2} = 2R - 1$	$d_{b2} = R$
Interpiconet-slave	$C_{b3} = (S - 2)(R - 1)$	$d_{b3} = R + 1$

$$d_b = \frac{c_{b1} \cdot d_{b1} + c_{b2} \cdot d_{b2} + c_{b3} \cdot d_{b3}}{c_{b1} + c_{b2} + c_{b3}} = \frac{SR^2 + (S - 3)R}{SR - 1} \quad (3)$$

Since there are $(S - 2)R$ nonbridge slaves, R masters, and R bridges, the average traveling distance can be derived as

$$D_{\text{avg}} = \frac{(S - 2) \cdot R \cdot d_s + R \cdot d_m + R \cdot d_b}{SR} = \frac{S^2R^3 + (S^2 - S - 4)R^2 + (4 - 2S)R}{S^2R^2 - SR} \quad (4)$$

Taking $S = 7$ and $R = 3$, for example, the average traveling distance D_{avg} will be 3.89.

The following analysis further considers interference between piconets. Assume that the maximum throughput of a single piconet under an interference-free environment is T. By extending to a scatternet, different piconets that choose the same FH channel in the same time slot result in a collision. Given that 79 frequencies are available in Bluetooth, the probability that a time slot of a piconet suffers no interference, denoted by P_S, can be approximated by $(78/79)^{R-1}$, where R is the number of piconets in the transmission range of each other.

The available network bandwidth is $T \cdot R \cdot P_S$. Dividing this by the average traveling distance D_{avg}, we obtain the maximum achievable throughput $T_{\max}$ of BlueRing:

$$T_{\max} = \frac{T \cdot R \cdot P_S}{D_{\text{ave}}} \quad (5)$$

Using the Bluetooth nominal bandwidth $T = 1$ Mbps, $S = 7$, and $R = 3$, we can compute $T_{\max} =$ 751 kbps. Figure 15 shows the ideal throughput $T_{\max}$

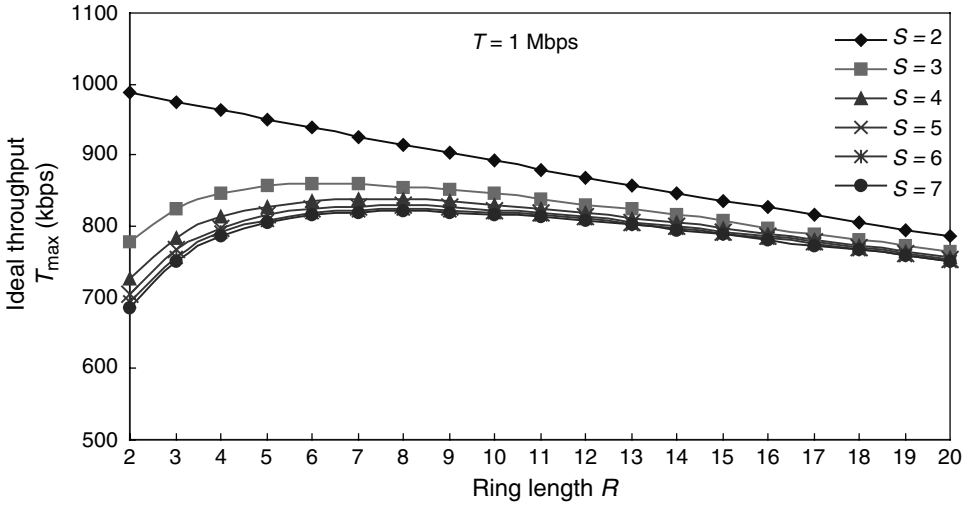


Fig. 15. Ideal BlueRing throughput for different R's and S's.

by varying R and S. From the curves, it can be seen that a BlueRing length of $R = 5 \sim 8$ would be quite cost-effective.

In the following text, we present our simulation results to verify the above theoretical analysis and to compare our BlueRing with other scatternet topologies. We simulate only DH1 data packets. For a single-slot DH1 packet, 336 bits (including the access code, header, payload header, payload, and CRC) are transmitted over a time slot with 625 μs duration, which contributes to a reduced $T = 538$ kbps throughput/piconet. For simplicity, we assume that collisions due to frequency overlapping do not happen, and thus $P_S = 1$. Taking a 21-node BlueRing ($S = 7$, $R = 3$) for instance, we obtain $T_{\max} = 415$ kbps, which predicts the saturation point in throughput. This can be used to verify the correctness of our analysis.

In our simulation, the number of Bluetooth devices that may participate in the BlueRing could be $N = 14$, 21, 28, 35 or 42. For each simulation instance, we initiate $N/3$ data-link connections, each with a randomly chosen source–destination pair. Each connection is an ACL link and can be an intra- or interpiconet communication. We also vary the ratio of the numbers of intra- to interpiconet connections. The numbers of intra- to interpiconet connections could be equal, more intraconnections, or more interconnections. We will evaluate the influence of this factor. For each connection, we assign it one of the three data arrival rates, 256 kbps, 128 kbps, and 16 kbps, with equal possibility. A master keeps a separate buffer queue for each of its slaves. No mobility is modeled. Besides, physical properties such as fading and interference are not considered. Each simulation run lasts for 75 s. Only DH1 data packets are simulated.

The bridging policy proposed in Section 4.2 is followed. Switching between piconets is realized using park/unpark procedures by following the proposed control message exchanges. We set the buffering threshold to 60% for bridges to switch between piconets. The maximum number of slaves per piconet is set to S. This factor will affect the ring length as well as packet delay. Two performance metrics are observed: *throughput* and *average packet delays*. In Section 6.1, we first study the impact of several BlueRing-related parameters. Then in Section 6.2 we present some performance comparison results with other scatternet structures.

6.1. Tuning BlueRing-related Parameters

In this subsection, we investigate three factors that may affect the performance of BlueRing: N (network size), S (maximum number of slaves per piconet), and bridge buffer threshold.

Figure 16 illustrates the throughput and average packet delays against S when $N = 14$, 21, 28, 35, and 42. For all values of N, we observe that the throughput increases and the packet delay decreases as S grows. When N is fixed, a larger S implies a shorter ring. So this indicates that a shorter ring length can result in higher network throughput and lower packet delay.

The bridge buffer threshold affects when a bridge node should switch to its downstream piconet. By setting the threshold to 20%, 40%, 60%, and 80% of the total buffer size, Figure 17(a) shows that the network

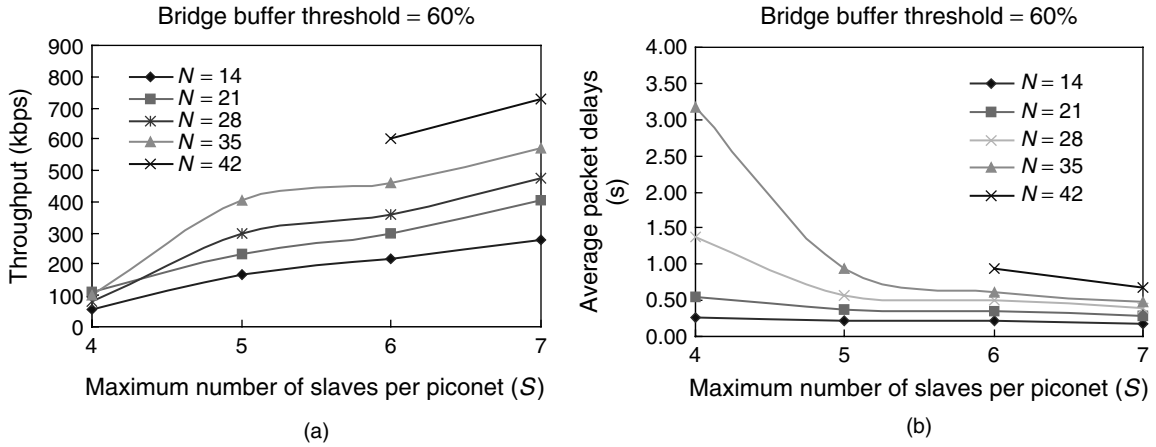


Fig. 16. Effect of ring length on BlueRing.

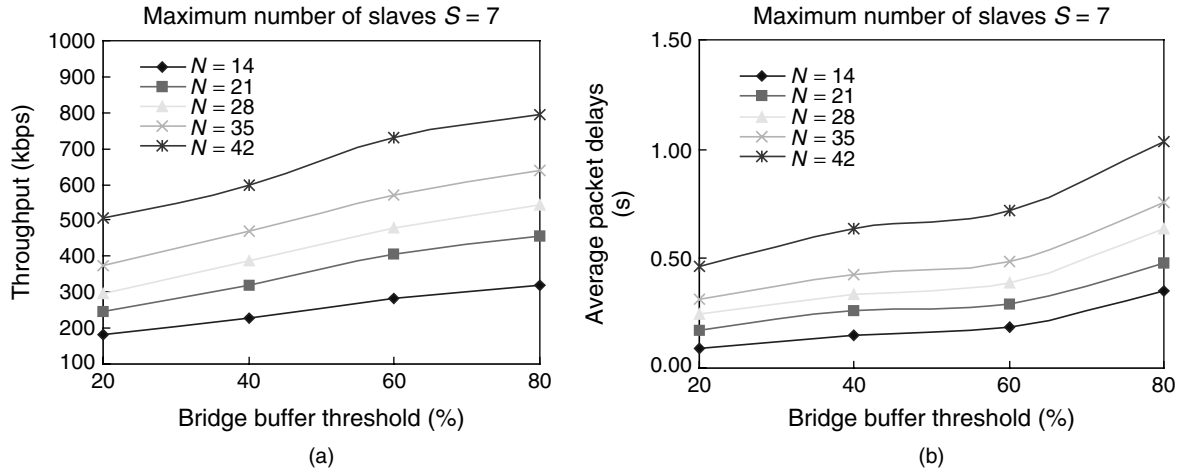


Fig. 17. Effect of bridge buffer threshold on BlueRing.

throughput will increase slightly as the threshold goes up. This is because a smaller threshold will incur more switches (and thus more switching overheads). On the other hand, Figure 17(b) also shows that the average packet delay will increase as the threshold grows, owing to longer queuing delays on bridges. Hence, the threshold value should be properly set to balance both network throughput and packet delay.

6.2. Performance Comparison with Other Scatternet Structures

We compare BlueRing with two other scatternet structures. The first one is a simple *single-piconet* structure. According to the Bluetooth specification, at most seven active slaves can be supported in a single piconet. To support more than seven slaves, the extra slaves must enter the *park* mode. In our implementation, we let the channel be shared by slaves in a round-robin manner. In other words, communicating entities are parked/unparked periodically, taking turns to access the channel. So, many extra control packet exchanges will take place. The second structure is the *star-shaped* scatternet proposed in [13]. One piconet is placed in the center. Each slave of the central piconet may be connected to another master, and if so, will act as a bridge of the two piconets. Noncentral piconets do not extend to more piconets. So this can be regarded as a two-level hierarchy. All interpiconet traffic must go through the central piconet, and thus this may present a traffic bottleneck, but the benefit is a less average number of hops that packets have to go through for interpiconet communications. Although the bridging policy is not specified in [13], here we adopt our threshold-based policy in Section 4.2 by regarding the

central piconet as upstream, and noncentral piconets as downstreams. When $N = 20$, Figure 18 compares our BlueRing with the star-shaped scatternet.

Figure 19 demonstrates the network throughput and packet delay under different traffic loads. Here, load is reflected by the number of connections initiated. Each connection has a data rate of 256 kbps and could be an intra- or interpiconet communication. Figure 19(a) shows that BlueRing saturates at the highest point compared to the other two structures. The saturated throughput is about 415 kbps, which is consistent with the prediction of our analysis. For the star-shaped structure, its throughput outperforms that of the single-piconet model when the number of simultaneous connections exceeds 6. This is because multipiconet has the advantage of using multiple FH channels at the same time. Figure 19(b) demonstrates that, when the number of simultaneous connections is below 10, both BlueRing and star scatternet suffer higher delays compared to the single-piconet case owing to bridging costs and larger network diameters. However, when traffic load becomes higher, the packet delay of single-piconet structure raises dramatically. The reason is that the throughput of single-piconet structure has reached a saturated point, which leads to significant increase of packet delays.

We also investigate the impact of different intra- to interpiconet connection ratios on the network performance. Note that with more interpiconet connections, the traffic load is higher. Figure 20 compares the throughput and average packet delay of the three scatternet structures under different traffic loads. Here the ratio of intrapiconet to interpiconet traffic is 3 : 1. The figure shows that BlueRing yields the highest throughput with moderate packet delay. For the single-piconet structure, the throughput saturates when the number of simultaneous connections reaches 6. This is because many time slots are wasted on exchanging control packets for park/unpark procedures. Furthermore, a single piconet can only

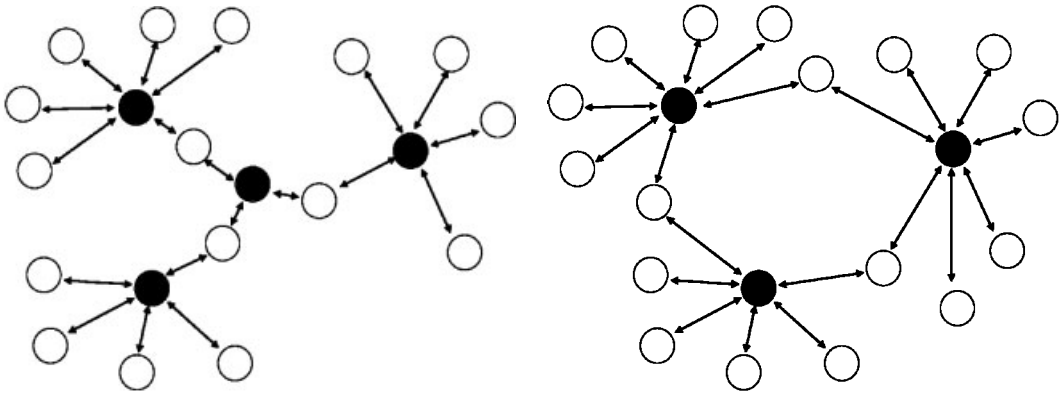


Fig. 18. Two scatternet topologies with $N = 20$ hosts: (a) star-shaped structure and (b) BlueRing (black nodes are masters).

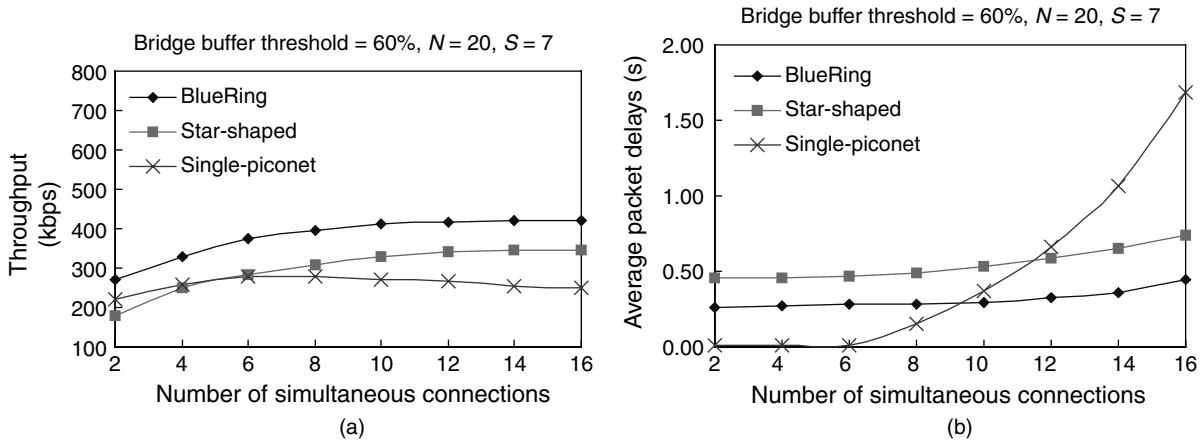


Fig. 19. Performance comparison by varying the number of connections: (a) throughput and (b) packet delay.

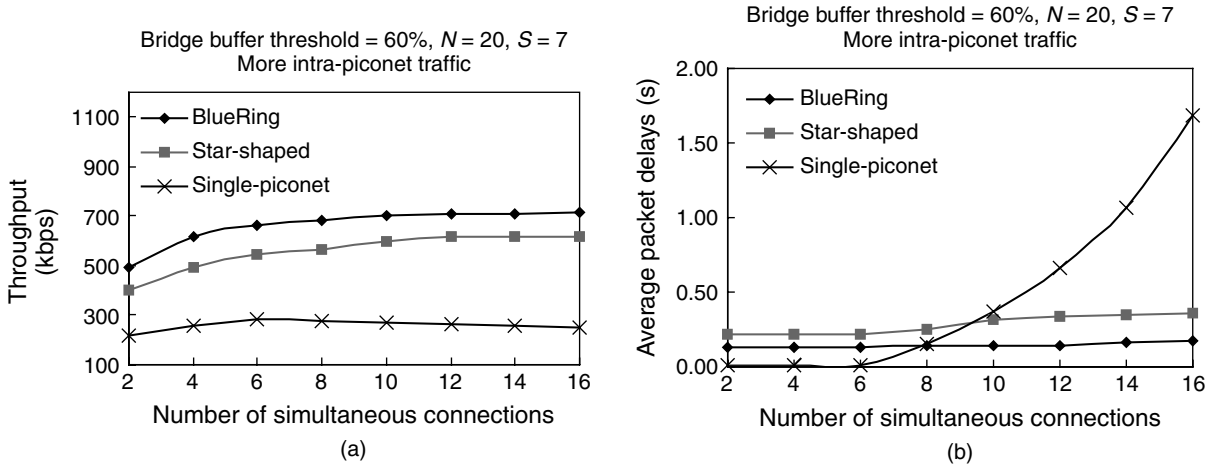


Fig. 20. Performance comparison by varying the number of connections (with more intrapiconet traffic): (a) throughput and (b) packet delay.

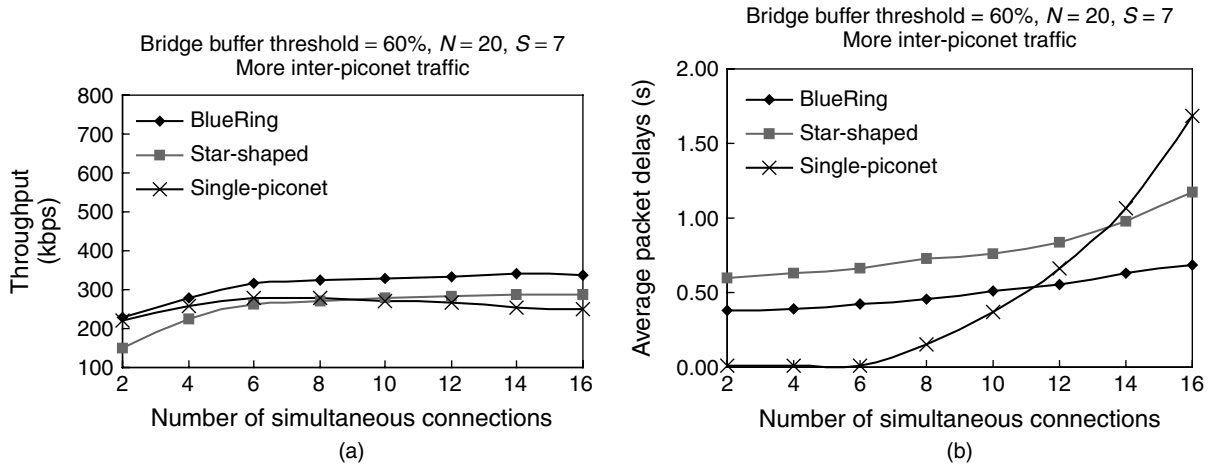


Fig. 21. Performance comparison by varying the number of connections (with more interpiconet traffic): (a) throughput and (b) packet delay.

utilize one FH sequence, and thus the maximum frequency utilization is only 1/79. On the other hand, BlueRing may utilize multiple FH sequences over the same space. This is what limits the capability of the single-piconet structure. With more intrapiconet connections, the BlueRing structure allows more simultaneous transmissions and shorter routing distances. For the star-shaped structure, the throughput is between those of BlueRing and single-piconet structures. With more intrapiconet connections, the bottleneck effect, incurred by the central piconet, of the star-shaped scatternet becomes less significant.

Figure 21 illustrates the case when there are more interpiconet connections (with intra- to interpiconet connection ratio equal to 1 : 3). The performance of the star-shaped structure drops significantly owing to more serious bottleneck effect. For BlueRing, the throughput also declines, but is still superior to the star-shaped structure. The single-piconet structure remains unaffected since all traffic is intrapiconet.

7. Conclusions

In this paper, we have designed the corresponding formation, routing, and topology-maintenance protocols for BlueRing. Owing to BlueRing's simplicity and regularity, routing on it is *stateless*, in the sense that no routing table needs to be kept by any host (and thus no route discovery procedure needs to be conducted prior to sending any packet). A protocol-level remedy is developed to keep the network alive when

there is a single-point failure on the ring. To tolerate multipoint failure, a recovery protocol is devised to reconnect the BlueRing. We believe that the important fault-tolerant issue has not been properly addressed by existing proposed scatternet protocols. To demonstrate the scalability of BlueRing with respect to network size, analyses and simulation experiments have been conducted. The results do indicate that BlueRing outperforms other network structures, such as single-piconet and star-shaped scatternet, with higher throughput and moderate packet delay. To conclude, we believe that the BlueRing is an efficient topology in terms of both network performance and fault-tolerant capability.

Our future works include analyzing the fault-tolerance capability of BlueRing and devising mechanisms to deal with more than one simultaneous failure on the ring. Moreover, a real implementation of BlueRing is also being planned in the National Chiao-Tung University and will be reported in our future work.

References

1. Bluetooth SIG Bluetooth Specification v1.1, http://www.bluetooth.com, February 2001.
2. Perkins CE. *Ad Hoc Networking*. Addison-Wesley: Reading, MA, 2001.
3. Bhagwat P, Segall A. A routing vector method (RVM) for routing in Bluetooth scatternets. In *IEEE Int'l Workshop on Mobile Multimedia Communications (MoMuC)*, 1999.
4. Groten D, Schmidt J. Bluetooth-based mobile ad hoc networks: opportunities and challenges for a telecommunications operator. In *IEEE Vehicular Technology Conference (VTC)*, 2001.
5. Johansson P, Kazantzidis M, Kapoor R, Gerla M. Bluetooth: an enabler for personal area networking. *IEEE Network* **15**(5): 2001; 28–37.
6. Kalia M, Garg S, Shorey R. Scatternet structure and inter-piconet communication in the Bluetooth system. In *IEEE National Conference on Communications*, New Delhi, India, 2000.
7. Miklos G, Racz A, Turanyi Z, Valko A, Johansson P. Performance aspects of Bluetooth scatternet formation. In *ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2000.
8. Salonidis T, Bhagwat P, Tassiulas L. Proximity awareness and fast connection establishment in Bluetooth. In *ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2000.
9. Law C, Mehta AK, Siu K-Y. Performance of a new Bluetooth scatternet formation protocol. In *ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2001.
10. Ramachandran L, Kapoor M, Sarkar A, Aggarwal A. Clustering algorithms for wireless ad hoc networks. In *ACM DIAL M Workshop*, 2000, pp. 54–63.
11. Salonidis T, Bhagwat P, Tassiulas L, LaMaire R. Distributed topology construction of Bluetooth personal area networks. In *IEEE INFOCOM*, 2001.
12. Zaruba GV, Basagni S, Chlamtac I. Bluetrees—Scatternet formation to enable Bluetooth-based ad hoc networks. In *IEEE Int'l Conference on Communications (ICC)*, 2001.
13. Lilakiatsakun W, Seneviratne A. Wireless home networks based on a hierarchical Bluetooth scatternet architecture. In *IEEE Int'l Conference on Networks (ICON)*, 2001.
14. whitepaper AU-System Bluetooth Whitepaper 1.1, http://www.ausystem.com, January 2000.

Authors' Biographies

Ting-Yu Lin (tylin@csie.nctu.edu.tw) obtained her B.S. and Ph.D. degrees, both in computer science from the National Chiao-Tung University, Taiwan, in 1996 and 2002, respectively. She is currently an R&D engineer at the Computer & Communications Research Laboratories (CCL) of Industrial Technology Research Institute (ITRI), Taiwan. Her research interests include wireless communication, mobile computing, personal area networks, and energy conservative designs (software level).

Yu-Chee Tseng (yctseng@csie.nctu.edu.tw) is currently a full professor at the Department of Computer Science and Information Engineering, National Chiao-Tung University, Taiwan. Dr Tseng has served as a program committee member in several international conferences and as a guest editor in several journals, including *IEEE Trans. on Computers*, *Wireless Communications and Mobile Computing*, *Wireless Networks*, and *Journal of Internet Technology*. His research interests include wireless communication, network security, parallel and distributed computing, and computer architecture. Dr Tseng is a member of the IEEE Computer Society.

Keng-Ming Chang obtained his M.S. degree in computer science from the National Central University, Taiwan, in 2002. He is currently an engineer at the Institute for Information Industry of Taiwan.