Improving Forwarding Mechanisms For Mobile Personal Area Networks

Redouane Ali

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Electrical & Electronic Engineering

University College London

Supervisor: Dr. Miguel Rio

January 10, 2011

Declaration of Authorship and Originality

I confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Department of Electronic & Electrical Engineering

University College London

Torrington Place

WC1E 7JE London

UK

Email: rali@ee.ucl.ac.uk

Date: January 10, 2011

Redouane Ali

Abstract

This thesis presents novel methods for improving forwarding mechanisms for personal area networks. Personal area networks are formed by interconnecting personal devices such as personal digital assistants, portable multimedia devices, digital cameras and laptop computers, in an ad hoc fashion. These devices are typically characterised by low complexity hardware, low memory and are usually battery-powered. Protocols and mechanisms developed for general ad hoc networking cannot be directly applied to personal area networks as they are not optimised to suit their specific constraints.

The work presented herein proposes solutions for improving error control and routing over personal area networks, which are very important ingredients to the good functioning of the network. The proposed Packet Error Correction (PEC) technique resends only a subset of the transmitted packets, thereby reducing the overhead, while ensuring improved error rates. PEC adapts the number of re-transmissible packets to the conditions of the channel so that unnecessary retransmissions are avoided. It is shown by means of computer simulation that PEC behaves better, in terms of error reduction and overhead, than traditional error control mechanisms, which means that it is adequate for low-power personal devices.

The proposed C2HR routing protocol, on the other hand, is designed such that the network lifetime is maximised. This is achieved by forwarding packets through the most energy efficient paths. C2HR is a hybrid routing protocol in the sense that it employs table-driven (proactive) as well as on-demand (reactive) components. Proactive routes are the primary routes, *i.e.,* packets are forwarded through those paths when the network is stable; however, in case of failures, the protocol searches for alternative routes on-demand, through which data is routed temporarily. The advantage of C2HR is that data can still be forwarded even when routing is re-converging, thereby increasing the throughput. Simulation results show that the proposed routing method is more energy efficient than traditional least hops routing, and results in higher data throughput.

C2HR relies on a network leader for collecting and distributing topology information, which in turn requires an estimate of the underlying topology. Thus, this thesis also proposes a new cooperative leader election algorithm and techniques for estimating network characteristics in mobile environments. The proposed solutions are simulated under various conditions and demonstrate appreciable behaviour.

Acknowledgements

Above all, I would like thank my Supervisors Dr Miguel Rio and Dr John Pollard who have given me precious support and advice, without which it would never haved been possible to complete this thesis.

I will forever be indebted to my family who have always believed in me and supported me both morally and financially. To my father Mohamed, my mother Rabia, my sisters Karima, Amel, Dalila and Yasmine, and my brothers Sofiane and Salim I say that had it not been for your continuous support and encouragement, I would never have completed the PhD.

My sincere thanks go to Suksant Sae Lor, Dr Richard Clegg, Dr Raul Landa, Rabah Taleb Benouaer, Mussie Wouldeselasie, Dr Venus Shum and Dr Chibuzor Edurdu who have spent their valuable time and effort helping me with the thesis.

I will forever be beholden to my friends Riccardo Manzini, Thalis Tsailas, Doria Selmane, Nazim Boufis, Adel Abad, Nabil Yahiaoui, Viola Matusiak and Kerstin Zibirre who have supported me throughout the entire length of the PhD.

Last, but certainly not least, I cannot thank my fiancée Sarah Vernon enough for all her love, support, help and understanding, particularly in the last few months. No words can express my gratitude for the effort she made in correcting the thesis, and for keeping me motivated all this time. For these and for many other reasons, I am extremely grateful.

This PhD was funded by a grant from the Algerian Ministry of Higher Education

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivations

The increase in use of consumer electronics and the wide availability of networking-enabled personal devices have made personal area networking a very attractive way of organising and managing one's own network of personal equipment. A personal area network (PAN) is a collection of electronic devices, such as mobile phones, laptop computers, portable multimedia systems and Personal Digital Assistants (PDAs), belonging to the same individual or organisation linked together via an enabling wireless access technology. The concept of personal area networking was introduced by T.G. Zimmerman in his thesis entitled *Personal area networks (PAN): Near-field intra-body communication* [180]. In his work, the author considers a PAN to be composed of a number of personal devices scattered around an individual. This initial concept came to be known as Body Area Networks (BAN) [147]; however, this idea has evolved to mean short range ad hoc networks formed by interconnecting various personal devices [48]. Several PANs can also be interconnected in an ad hoc fashion, wherein multi-hop data transfer can take place, either within the individual networks themselves or to a remote destination via a Network Access Point (NAP). Moreover, an individual's devices can be interlinked to one another via a wide area network such as the Internet or UMTS, thereby constructing distributed personal networks where one is able to access a particular device remotely at any time and from anywhere [129]. The idea of personal networking as described in [129] is a natural evolution of personal area networks, wherein physical proximity is no longer a constraint. Nonetheless, the basic building block in personal networking consists of personal area networks.

The main difference between general Mobile Ad Hoc Networks (MANET) and PANs is that in the former, nodes are assumed to be symmetric in terms of capabilities, whereas in the case of PANs, nodes have different degrees of ability [45]. In addition, PANs are generally characterised by short range communications, whereas in MANETs this constraint is not usually applicable. In terms of processing and power constraints, PANs are closer to Wireless Sensor Networks (WSN) [7], in that relaying information

must take into account the capabilities of nodes and must aim at preserving energy. However, unlike WSN, which are designed to suit specific applications and form static topologies [11, 5], PANs aim to be more dynamic.

Just like any generic mobile ad hoc network, PANs do not rely on any underlying fixed infrastructure and nodes may displace, therefore the topology can change unexpectedly. Moreover, nodes may join or leave the network without warning. Nevertheless, because of the nature of the devices themselves, special attention needs to paid to the short transmission range, reduced buffering and processing capabilities, connectivity constraints and energy consumption. Consequently, some of the established mechanisms for general mobile ad hoc networks may not be directly suitable for PANs as they may not be energy efficient, may require more complex computations than devices can handle, or do not satisfy transmission characteristics. Similarly, routing protocols for wireless sensor networks, although energy-efficient, cannot be applied directly to PANs without alteration, as these protocols take advantage of the stability of the network and periodicity of transmissions, which contradict the pervasiveness of PANs. For these reasons, a new set of protocols and solutions need to be tailored to suit the specifics of personal area networks. This thesis confines itself to personal area networks; more precisely, to data forwarding issues. Figure 1.1 illustrates a typical setup for personal area networks.

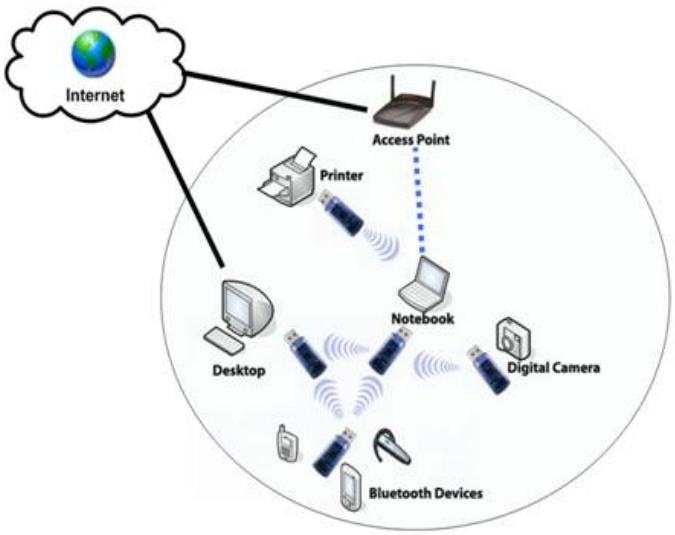


Figure 1.1: Personal Area Network

Arranging personal devices in a mobile ad hoc network has numerous advantages, including ease of access to data, carrying out distributed tasks such as monitoring, or distributing data storage. Uses of personal area networks are various, and typical usage scenarios include: home automation [29], health monitoring [94, 80], office networking [58] and disaster management [181]. Health monitoring, in particular, has had a tremendous impact on the development of PANs owing to the benefits they offer in

capturing and relaying information on behalf of patients without their direct participation, *i.e.,* even when patients are unconscious. In [80] the authors propose an architecture that establishes communication between ambulance crews and specialist doctors, who are often located in remote hospitals, so that patients can benefit from adequate treatment at the scene of an accident. The PAN equipment can capture and transmit patients' medical conditions to a data transfer unit, possibly located in the ambulance itself or in a hospital where doctors and specialist teams can assess the situation and advise on the necessary first aid treatment. In addition, these networks can be used to link patients at home to hospitals and analysis centres, thereby reducing the cost of post-trauma care.

PANs also play an important role in distributed personal networking. The idea being to enable access and interconnection of personal devices within close proximity or across a wide area network, such as a wireless data network [128]. Thus, a person can access his/her devices and transfer data, such as email or multimedia files, from anywhere and at any time without necessarily being within physical proximity. For instance, when an individual leaves his office and enters his car, a PAN is established with the in-car network system, allowing access to a wide area network, so that the individual can access other devices which remain in the office but are connected to a network access point. It is anticipated that distributed personal networking will play a major part in future communications systems, hence the need for networking personal devices into personal area networks.

There are a variety of technologies that could be used as the underlying bearer of this type of network: Bluetooth [24], Zigbee [153] or IEEE 802.11 [40][1]. In [90] the authors provide an overview of the different technologies that enable wireless personal area networking and identify two low power candidates: Bluetooth and IEEE 802.15.3. The latter presents higher data rates than the former but at the expense of covering smaller areas and draining more current. Nevertheless, when it comes to ubiquitous personal area networking, Bluetooth is one of the technologies of choice, as it is readily available in a multitude of devices and satisfies the requirements of such networks in terms of power, cost and availability. Bluetooth was specifically developed for short range communications between personal devices and is regarded as a universal enabler for personal area networking [78]. Moreover, as shown in [76], in noisy channel conditions, as the number of devices increases, Bluetooth performs better than IEEE 802.11 in terms of throughput, energy efficiency and medium access fairness.

Bluetooth[2] was initially devised as a cable replacement technology, however, its uses have evolved to include more sophisticated networking applications, such as multi-player gaming [13], home automation [150, 154] and social networking [97]. This thesis uses Bluetooth as the underlying technology for personal area networks. However, the findings and protocols can either be applied to other technologies

[1] WiFi Alliance announced WiFi Direct for personal area network for future release

[2] The term Bluetooth derives from the translation into English of the nickname given to King Blåtland of Denmark, who unified the kingdom under Christian law [112]. The Bluetooth Special Interest Group (SIG) code-named the project after King Bluetooth, and after its launch, the new wireless technology retained its name, hence Bluetooth was born.

that directly meet the requirements, or can be slightly modified to suit other architectures.

Since its launch, Bluetooth has witnessed tremendous popularity, thanks to its low manufacturing cost and great interest from electronics industry leaders. As of 2005 there were approximately 320 million active Bluetooth devices [156], and this number is expected to reach 2 billion by 2012, representing a sales figure of about $2.7 Billion [123]. Moreover, version 3.0 [23] and the enhanced version 4.0 [24] (released in April and December 2009 respectively) which offer augmented data rates, improved security and optimized power consumption, have rendered this technology even more attractive.

Another factor that has helped Bluetooth gain such remarkable market penetration is the new legislation regarding road safety. In the United Kingdom, a law banning the use of mobile phones while driving came into force on 1^{st} December 2003 [141]. Bluetooth manufacturers seized the opportunity to introduce hands-free kits that allow drivers to connect their handset to an earpiece via Bluetooth. In addition, many car makers integrated Bluetooth into their audio and voice over systems, hence reinforcing Bluetooth's position as the leader in short range personal communications.

All these factors lead the author of this thesis to believe that personal area networking will have a considerable impact on future communications, providing convenience and bringing tangible solutions to existing problems. However, achieving these goals comes with its own set of challenges and issues, which need to be addressed, specifically those relating to error control and routing. Currently proposed solutions are either incompatible with the strict constaints of PANs or are not optimised to suit their purposes.

1.2 Challenges

One particularity of PANs is the fact that they rely on low power, low transmission range devices; therefore requiring multi-hop communications. This, in turn, introduces the challenge of routing packets between source and destination as efficiently as possible in terms of delay and energy consumption, whilst dealing with mobility. Moreover, PANs operate in the Instrumental Scientific and Medical (ISM) unlicensed frequency band, hence they are susceptible to high error rates due to fading and mutual interference from coexisting networks. In addition, the choice of underlying technology adds its own constraints, as protocols and mechanisms need to comply with particular specifications. In the case of Bluetooth, the constraints imposed relate to very specific topology and transmission arrangements. Bluetooth relies on a master-slave concept, where direct slave-to-slave or master-to-master communication is disallowed. Furthermore, transmissions are based on a time-division schedule wherein a slave can only communicate with one master at any particular time and can only transmit or receive when instructed to do so by the master. All these challenges must be approached with care and the constraints taken into account.

A major concern in designing PANs is ensuring that transmission errors are dealt with effectively,

without burdening the devices with computationally demanding data coding or adding impractically large redundancy overhead. However, relying solely on retransmissions may add unacceptable delays for time-bound applications. The aim, therefore, is to devise error control solutions that satisfy the low power and low complexity requirements while maintaining acceptable levels of latency.

Personal area networks operate in an ad hoc fashion where information needs to be relayed from source to destination via other nodes acting as routers. However, due to the low power and low processing capacities of personal area network devices, a direct application of existing ad hoc routing protocols may prove inefficient. Traditionally, ad hoc routing solutions try to minimise the number of hops between source and destination, hence reducing end-to-end delay [42, 142, 2]. Nevertheless, taking the shortest routes may translate in longer transmission distances and consequently in higher power consumption. Therefore, the answer lies in designing a routing protocol that is light enough for nodes in terms of computations, yet optimised for energy efficiency and delay.

In order for PANs to be widely deployed, it is vital that transmission reliability and routing are improved and optimised for this particular type of network. Currently, the proposed solutions in the literature are not tailored for PAN optimality and do not account for their specific topology and transmission characteristics. Moreover, error control and routing are generally considered disjoined; while in actual fact these two areas are tightly linked to one another, as a channel's error behaviour should be an important factor in selecting appropriate routes over which to forward the data. For instance, routing via a wireless channel exhibiting high error rates, and without appropriate error control, would mean increasing packet retransmissions, consequently defeating the purpose of power-efficient routing. This thesis proposes integral solutions that aim at improving data forwarding in a multi-hop fashion, taking into account particular constraints of PANs.

1.3 Objectives

This thesis tackles forwarding issues in personal area networks. Given the aforementioned observations, it is evident that there are questions that remain to be answered with regard to transmission reliability and multi-path routing. There are numerous elements one could focus on in order to improve forwarding mechanisms for PANs; however, the work presented herein specifically addresses these two aspects, as they have a direct impact on the performance of such networks.

Firstly, error control is paramount to the sound operations of data forwarding in PANs, as unreliable transmissions result in increased delay, lower quality of service, increased link costs and complex data handling mechanisms. It is neither beneficial nor reasonable to try to forward data through paths that exhibit unreliable channels; therefore improving error control is a fundamental step towards improving forwarding mechanisms.

Secondly, routing must take into account the requirements of PANs in terms of low complexity,

energy efficiency, delay, and constantly varying network topologies. Owing to the fact that devices run on exhaustible battery power and that the user may not always have the option to recharge them, it is preferable to send data over paths which exhibit low energy consumption *i.e.,* those that use shorter distances between devices, and transmit over channels which exhibit lower error rates, even if this results in an increase in the number of hops. On the other hand, some applications may be bound by very strict delay constraints, therefore, very long routes may be impractical. Consequently, link costs must be calculated in such a way that routing yields an optimal solution.

The proposed solutions are evaluated through analytical methods and simulations. A simulator based on Matlab® [157] is built to assess the performance of the schemes in numerous scenarios with varying topology composition and node mobility. The simulator implements the relevant parts of the Bluetooth protocol stack and uses structures to represent the different network components: nodes, links and channels. On the other hand, time, data overhead and residual energy are implemented as variable discrete attributes belonging to the network structures. Simulation enables one to easily vary the different factors affecting the performance of the network, such as channel conditions and transmission coverage, thereby allowing the evaluation of a large number of settings.

1.4 Thesis Structure

This chapter has introduced the subject of the research project, its motivations and challenges. It also outlines the contributions to the field of personal area networking and has provided an overview of the methodology used for analysing and evaluating the proposed solutions. The remainder of the thesis is organised as follows:

Chapter 2 details the Bluetooth technology, its components and mechanisms. It examines the way the different layers of its protocol suite interact to form ad hoc personal area networks. This chapter also introduces some of the terminology used throughout the thesis.

Chapter 3 highlights pertinent background material related to ad hoc networking in general and personal area networking in particular. It shows how Bluetooth-based personal area networks are built and maintained, explaining their operations in some detail. It also outlines the technical challenges faced by such networks and summarises extant research in this field, including topology construction, scheduling and error control. It then discusses routing schemes for mobile ad hoc networks and Bluetooth-based personal area networks. These protocols fall into one of three groups: proactive, reactive and hybrid; this chapter gives examples of the predominant routing protocols for each of the three groups.

Chapter 4 goes on to present an adaptive error control mechanism for personal area networks. It outlines the scheme and provides a numerical analysis. The proposed error control scheme is a supplement to specification-defined forward error correction, which trades off between latency and effectiveness. In order to illustrate this trade off, several scenarios are considered and simulation results for each setting

are provided for comparison.

Chapter 5 proposes an adaptation of two network size estimation methods, random tour and gossip-based aggregation, to suit the specific constraints of master/slave personal area networks; showing that it is feasible to accurately estimate the size of such networks when topology changes due to mobility using both estimation methods.

Chapter 6 presents a cooperative leader election algorithm, which explores the particular topological characteristics of PANs in order to optimise efficiency. The algorithm demonstrates an improvement in terms of time complexity and response to node mobility in comparison to existing tree-based algorithms.

Chapter 7 outlines the proposed Centrally Coordinated Hybrid Routing (C2HR) protocol for PANs. C2HR belongs to the hybrid family of routing protocols, employing a proactive and reactive component. The chapter explains the mechanisms of the protocol and looks at its behaviour in various scenarios.

Chapter 8 summarises the findings of the thesis and discusses their applicability to future deployment. It also outlines suggested future work in the field of personal area networking, highlighting open research issues that require further exploration.

Chapter 2

The Bluetooth Technology

2.1 Introduction

This chapter provides the reader with an overview of the Bluetooth technology, in order to put the proposed solutions into context. It details the relevant components of the Bluetooth standards, presents the general Bluetooth operations and familiarises readers with no prior knowledge of the technology. This chapter also introduces the terminology and assumptions used throughout the thesis.

Bluetooth was initially developed as a means of replacing cables between electronic devices [56]; however, due to its wide availability and its capacity to support both data and voice traffic, its uses have evolved to include other applications such as multi-player gaming and peer-to-peer file sharing. The Bluetooth standard specifies its own protocol stack, which is compatible with the Open System Interconnect (OSI) model, ranging from physical to application layers and defining software/hardware interfaces between the various layers. In addition, it adopts many of the established networking, transport and application protocols such as IP, TCP, UDP or PPP, so that it can directly connect and interact with existing networks without any modifications [121]. The Bluetooth protocol suite is shown in Figure 2.1

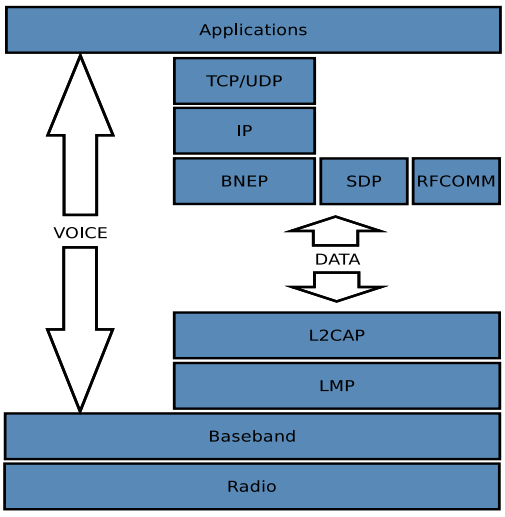


Figure 2.1: Bluetooth Protocol Stack

2.2 Bluetooth Radio Layer

The physical layer is the lower most layer of the protocol stack and defines transmission power, modulation and data encoding. Bluetooth operates in the Instrumental, Scientific and Medical (ISM) frequency band at 2.4 GHz and uses *Frequency Hopping Spread Spectrum* (FHSS) modulation technique, hopping at a rate of 1600 hops per second across 79 [1], 1 MHz wide frequency channels [24]. As per FHSS, each device is allocated a hopping sequence such that two non-communicating adjacent Bluetooth devices do not use the same sequence, hence ensuring that simultaneous transmissions from non-communicating nodes occur at different frequencies. However, for two devices to be able to communicate with each other they must have the same hopping pattern. This is achieved by synchronising the devices using a pre-determined pseudo-random sequence derived from MAC addresses and clock information.

The tight low power requirements for Bluetooth restrict the maximum transmit power to 20 dBm (the equivalent of 100 mW). This output power represents class 1. There are two other output power classes defined by the specifications: Class 2 with a maximum output power of 4 dBm (2.5 mW) and class 3 which allows a maximum transmit power of 0 dBm (1mW) [112]. Bluetooth devices typically use power class 3 as it requires inexpensive, low capacity batteries and complies with receivers' sensitivity [57]. Moreover, it is not generally recommended to use higher transmit powers in dense environments as it shadows other devices, thus preventing good network operations [57]. For this reason, it is common to assume a transmission range of about 10 meters, which corresponds to power class 3.

Besides high output powers, which may cause total signal jamming, Haarsten *et. al.* [57] have identified two other sources of interference in Bluetooth radio: One is inter-modulation distortion caused by the mixing of radio signals, and the other is the proximity of transmission frequencies of adjacent radios, which causes signal mitigation and therefore introduces errors. The latter can be dealt with by appropriate filtering and careful frequency hopping pattern selection.

Bluetooth radio allows rates of 1 Ms/s (M Symbols/s) [24], however the gross air data rate may vary depending on the modulation scheme in use. The first release of the Bluetooth standards (Version 1.0 [18]) specifies a basic data rate of 1 Mbps, by employing Gaussian Frequency Shift Keying (GFSK). Subsequent releases (Versions 2.0, 2.1) specify two Enhanced Data Rates (EDR) by changing the modulation from GFSK to Phase Shift Keying (PSK). The first employs $\pi/4$ Differential Quadrature PSK ($\pi/4$-DQPSK), resulting in a rate of 2 Mbps and the second uses 8 Differential PSK (8DPSK), thus resulting in a rate of 3 Mbps [22]. The latest versions of Bluetooth (versions 3.0 and 4.0 [23, 24]) attain gross data rates of up to 24 Mbps. This is achieved through the new Alternate MAC/PHY (AMP) feature which uses 802.11 radio when needed for increased data rates [2]. It is mandatory that each Bluetooth module supports the basic rate in order to ensure backward compatibility.

[1] In France, Spain and Japan the number of hopping frequency channels is restricted to 23 [69]

[2] Initially, Bluetooth SIG opted for Ultra Wide Band (UWB) for the AMP; however, UWB never made it to the market, hence the adoption of 802.11

2.3 Bluetooth Baseband Layer

The Baseband layer defines the way data is managed, divided into packets and sent between devices. It also deals with error control and basic security operations. Bluetooth relies on a time-division medium access scheme, where time is divided into slots of 625μs and each device can only send or receive in its allocated slots using the pre-defined frequency hopping pattern. These functions are managed by the *Baseband and Link Controller*, which interfaces between the radio and the baseband layers [120].

Bluetooth relies on a master-slave concept, wherein two slaves can communicate with each other only via the master node. Devices are arranged into clusters called *piconets* in which one of the devices assumes the role of master, which then admits up to 7 active slaves. Within a piconet, the master is responsible for scheduling transmissions and managing device memberships. A slave node can only start transmission in an odd time slot if it has received data from the master in the previous even slot. This particular mode of operation requires efficient scheduling and synchronisation between devices.

2.3.1 Baseband Channels

Bluetooth defines two types of baseband links: *Synchronous Connection-Oriented* (SCO) links, used mainly for transmission of circuit-switched audio data, and *Asynchronous Connection-Less* (ACL) links for packet-switched data [121]. These two links differ in the way data transmission is scheduled, and hence in the quality of service guarantees.

On SCO links, data is transmitted symmetrically at regular intervals. These intervals are pre-agreed between the communicating devices during link establishment through a slot reservation method which ensures that specific time slots are reserved for SCO transmission, regardless of whether there is data to be sent or not. In addition, SCO links can only support point-to-point transmission. ACL links on the other hand, support point-to-multipoint transmissions, wherein a master can send data simultaneously to more than one slave. Moreover, ACL links offer the option of retransmitting packets in case of failure, by appending redundancy check trails to packets. Another fundamental difference between SCO and ACL links, is that in the latter a slave can only transmit if it has received data or has been *polled* in the previous time slot; whereas in the case of SCO, a slave can only send in its reserved time slot.

ACL links are better suited to bursty data which is not time-bounded and in which strict packet ordering is not paramount, such as file sharing. Nevertheless, ACL links have proven adequate for audio transmission between Bluetooth devices [85, 8, 108] in terms of latency and data reliability. In [8] and [108], ACL links are used to convey voice over IP (VoIP) data as a substitute for the circuit switched SCO links, and show that ACL links are suitable, as far as audio quality is concerned.

2.3.2 Addressing

In Bluetooth, devices are identified by a number of addresses, depending on transmission characteristics and the state of the devices. There are four main addresses as defined by the specifications [24]: *Blue-*

tooth Device Address, *Active Member Address*, *Parked Member Address* and *Access Request Address* [120]. These are detailed below.

Bluetooth Device Address (BD_ADDR): This is a 48-bit long address that uniquely identifies every single manufactured Bluetooth device. It is hard coded into each module [120]. The BD_ADDR is divided into 3 parts: a 24-bit *Lower Address Part* (LAP), an 8-bit *Upper Address Part* (UAP) and a 16-bit *Nonsignificant Address Part* (NAP).

Active Member Address (AM_ADDR): Since a single master can only admit up to 7 active slaves at any one time, Bluetooth standards define a 3-bit long Active Member Address for master-slave communication within a single piconet. Each of the active slaves is assigned a unique intra-piconet identifier from the address range: 001 - 111. The address 000 is reserved for broadcast by the master [24].

Parked Member Address (PM_ADDR): Bluetooth allows additional devices to be registered and synchronised with the master without actively taking part in piconet communication [112]. These slaves are called *parked slaves*, which only listen to the master's broadcasts for synchronisation periodically. When a slave is in the park mode, the master assigns a unique 8-bit address to it, which is in turn used to call upon this slave for participation in the piconet (unpark procedure).

Reserved Addresses: In Bluetooth, there is a set of addresses that are reserved for discovering and/or inviting devices to establish a connection. When a master searches for devices in its vicinity it uses the universally known *General Inquiry Address* 0x9E8B33 [24]. However, if the master only targets a particular type of device, it uses a *Dedicated Inquiry Address*, drawn from the address space 0x9E8B00 - 0x9E8B3F [24] (excluding 0x9E8B33). Moreover, if the master needs to discover a specific device for a limited period of time, it uses a special dedicated inquiry address, referred to as the Limited Inquiry Address, which has a value of 0x9E8B00 [112].

2.3.3 Baseband Packets

At the baseband layer, packets are sent as Baseband Packet Data Units (BB_PDU) which contain actual user data and corresponding encapsulation and redundancy bits. There are two main baseband packet types: voice packets and data packets. The former are used on SCO links to carry voice information, whereas the latter are used for general data on ACL links [25]. Given that SCO links are used primarily for carrying voice data, ACL links are more common in Bluetooth personal area networks. Hereinafter, voice packets as referred to as SCO packets and data packets as ACL packets. Baseband packets normally include an access code, a packet header and the payload as shown in Figure 2.2; However, they may be comprised of 1) the access code on its own, 2) the access code and the payload header, or 3) the access code, the payload header and the user data [120].

The access code's main function is to provide synchronisation and piconet identification [103], and is derived from the LAP part of the BD_ADDR. There are five different access codes defined by

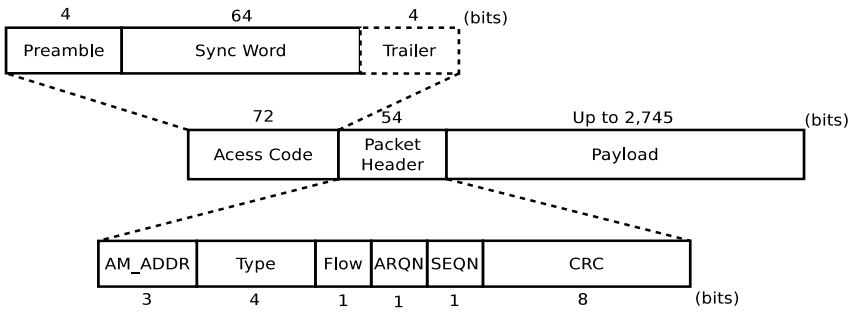


Figure 2.2: Baseband Packet Format

the specifications: the **Channel Access Code (CAC)** used with general user data, the **Device Access Code (DAC)** used in *paging* procedures, the **General Inquiry Access Code (GIAC)** used for *inquiry*, the **Dedicated Inquiry Access Code (DIAC)** for dedicated inquiries and the **Limited Inquiry Access Code (LIAC)** used in limited inquiry procedures[3]. The packet header identifies the type of data being carried as well as providing the device's local address. Because of the importance of the packet header, it is 1/3 PEC encoded, bringing its total length to 54 bits.

2.3.3.1 SCO Packets

In the literature, SCO packets are referred to as *HV* (High-quality Voice) packets. These are one slot-long containing 240 bits of payload, no payload header and no *Cyclic Redundancy Check* (CRC) trailer. The absence of a CRC means that no Automatic Repeat reQuest (ARQ) scheme is utilised, and hence SCO packets cannot be retransmitted in the event of failure, thereby reducing latency but at the risk of incurring higher data loss [120]. Three main SCO packets can be distinguished according to the *Forward Error Correction* (FEC) protection in use: HV1, HV2 and HV3. The last digit in these identifiers indicates the type of protection used in the payload [112]. HV1 packets use a (3,1) binary repetition forward error correction code, HV2 use a (15,10) shortened Hamming code, and HV3 are not FEC encoded. In other words, in HV1 packets 1 in 3 payload bits actually represent user data, in HV2 packets 2/3 of the payload is user data, and all the bits in HV3 packets are digitised voice data.

2.3.3.2 ACL Packets

These packets are destined for use with asynchronous data and are divided into two main subgroups: Data High-speed (DH) and Data Medium-speed (DM). DH packets use no FEC while a (15,10) shortened Hamming code is used for DM packets. A digit (1,3 or 5) following the DH or DM notation indicates the number of time slots that the packet spans. Therefore 6 different ACL packets can be distinguished: DH1, DM1, DH3, DM3, DH5 and DM5.

An obvious difference between DH and DM packets is that the latter offer more reliability because of their ability to recover from some errors, but at the expense of resulting in lower data rates. In

[3]Refer to Section 2.5 for details on Inquiry and Paging mechanisms

ideal error-free channels, packets that span more time slots exhibit higher data rates; however, due to the error-prone characteristics of the ISM band, packets that are not received correctly, *i.e.,* when FEC fails, are retransmitted, which means that there is a cut-off level of channel Bit Error Rates (BER) beyond which larger packets would not yield higher throughput because of the delay introduced by packet retransmissions.

There is another very important baseband packet type called the *Frequency Hopping Sequence* (FHS) packet, which is used to convey frequency hopping, addressing, synchronisation and class of service information during link establishment. FHS packets span a single time slot and, just like ACL packets, are FEC and CRC protected.

2.3.3.3 Baseband Packet Exchange

In Bluetooth operations, only the master is allowed to broadcast and can either communicate with a single slave or with multiple slaves simultaneously. A master sends data in even time slots and a slave replies in odd time slots, with each transmission occurring at a particular frequency of the hopping sequence. However, a slave can only transmit if it has received data or a *POLL* message in the previous even slot. This means that the minimum time taken for a master-slave exchange is $1.25ms$ (625μs in each direction in the case of single slot packets). In reality, the time it takes to transmit all of the packet's data is 366μs with the remaining 259μs reserved for frequency hopping operations [120]. Figure 2.3 illustrates the point-to-point, *i.e,* master-slave, data exchange mechanism in Bluetooth.

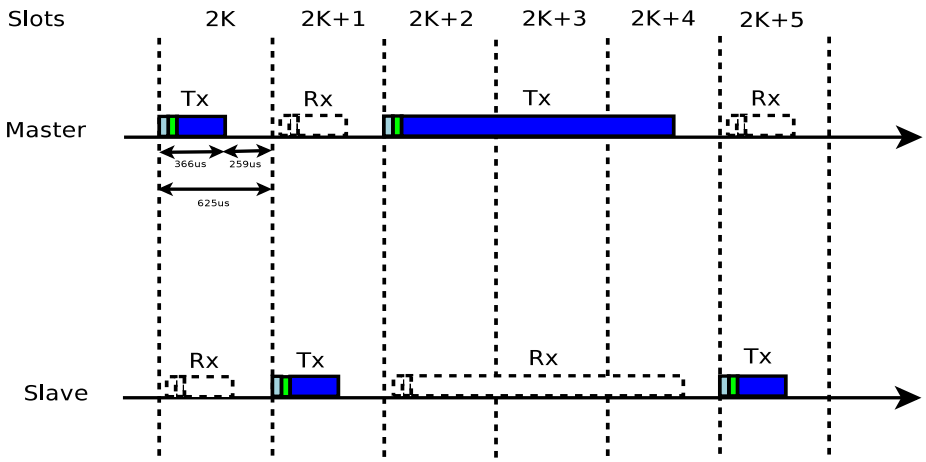


Figure 2.3: Baseband Packet Exchange - Point-to-Point

If more than one slave is present in the piconet, the master communicates with individual slaves either one at a time or via broadcast. However, if the communication requires the master to target individual slaves one at a time, it divides the time amongst all the slaves in the piconet via a time sharing technique. Naturally, in multi-slave operations, the throughput will also be split amongst the various devices. When a slave receives a packet, it decodes the access code and the packet header to check if

the packet is destined for itself and obtains the packet type and its time span. If the packet is intended for that particular slave, it leaves its transceiver on, otherwise, it switches it off until the next master-slave transmission. This mechanism ensures that devices do not leave their transceivers on unnecessarily and also reduces the possibility of packet collision. Although, at first glance, it appears that relying on time-division medium access yields low data rates, the unlikelihood of packet collisions within a piconet considerably reduces the number of retransmitted packets. Therefore, in dense environments Bluetooth may perform better, in terms of data throughput, than IEEE 802.11, as demonstrated by [76], while maintaining power consumption at comparatively low levels. Figure 2.4 shows a graphical representation of the multi-slave baseband packet exchange mechanism.

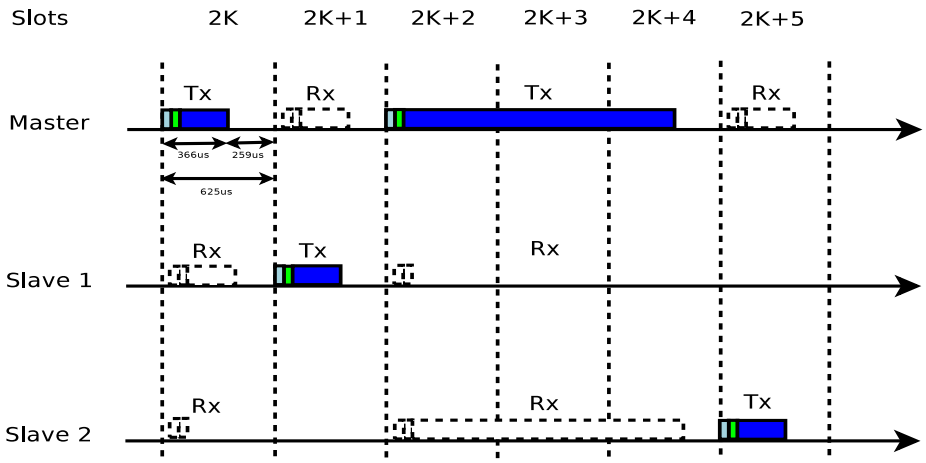


Figure 2.4: Baseband Packet Exchange - Multi-slave

As can be seen from Figures 2.3 and 2.4, each transmission uses a particular frequency of the frequency hopping sequence, which must remain the same for the entire duration of the packet transmission. Furthermore, regardless of the type and time span of the baseband packet, extra time (259μs) is allocated for frequency hopping operations so that the device is ready for the next reception.

2.4 Link Manager Protocol (LMP)

The *Link Manager Protocol* (LMP) is responsible for setting up, maintaining and terminating connections between communicating Bluetooth devices. When two devices need to establish a connection, the Link Manager (LM) modules on each device exchange special BB_PDUs (hereinafter called LMP packets) in a *command/response* process. During this procedure, link parameters, such as the type of link (*i.e.,* SCO or ACL), QoS parameters (*e.g.,* number of retransmissions allowed), or security settings (*e.g.,* authentication and encryption keys) are negotiated. LMP packets are transmitted as DM1 packets, have higher priority than user data and are not retransmissible [121]. The specifications define 55 different LMP packet types, each identifying a particular command [121]. These commands are identified by a

7-bit *OpCode* included in the LMP packet's payload, immediately followed by the command parameters, if they exist, as shown in Figure 2.5.

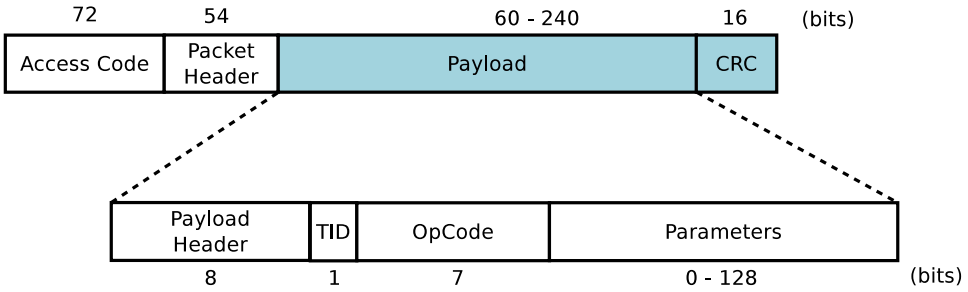


Figure 2.5: LMP Packet Structure

The TID bit indicates the originator of the LMP procedure. It is set to 0 if the link management procedure originates at the master, and it has a value of 1 if it originates at the slave [120]. When a device requires a particular link management feature, it sends a request to the other device's LM in an LMP packet. The receiver can then either accept the request by sending an `LMP_accepted` command or decline it by replying with an `LMP_not_accepted` message. If the request is declined the originator may then change the parameters and send another request as part of the negotiation process, until an agreement is reached.

Another very important function of the link manager protocol, as will be seen in section 2.6 is piconet management. In Bluetooth there are special states that allow devices to exist or enter a piconet temporarily, which are negotiated using LMP packets. These states are called low power modes and constitute a very useful feature of Bluetooth PANs.

2.5 Establishing Bluetooth Piconets

Before two Bluetooth devices can exchange data, they need to have knowledge of each other's presence and roles then establish a connection. This is achieved via the *inquiry* and *page* procedures. Inquiry is performed in order to discover new devices, and in the page phase, devices are invited to join a piconet. Nodes that initiate the inquiry or page process become masters, and the devices that respond to these invitations become slaves. Before the establishment of a piconet, the device that will assume the role of master is referred to as a prospective master (p-master) and the devices that are to become slaves are called prospective slaves (p-slaves) [120]. Each of the aforementioned procedures occur in pairs: *inquiry/inquiry scan* and *page/page scan*.

2.5.1 Inquiry

A p-master starts the discovery process by sending inquiry messages to potential respondents (p-slaves). A p-master has neither a priori knowledge of the addresses of the surrounding devices nor their hopping

frequency patterns. Thus, during inquiry, a p-master sends inquiry messages, containing only the GIAC, DIAC, or LIAC.

The p-master generates a sequence of 32 inquiry hopping frequencies, from the IAC[4], together with its native clock (CLKN) and corresponding hopping phase. It hops at double the nominal rate (*i.e.,* 3200 hops per second), so that the frequency changes every $312.5\mu s$ as opposed to the nominal $625\mu s$ [24, 120, 112]. A p-master sends two consecutive 68-bit IACs in an even time slot on two different hopping frequencies, $312.5\mu s$ apart, then switches to the two corresponding response frequencies to listen to the response from the p-slaves during the subsequent odd time slot. When a p-slave receives the IAC, it responds by sending an FHS packet in the corresponding reply frequency, from which the p-master can deduce the p-slave's BD_ADDR and CLKN for use in the page phase. However, for a p-slave to hear and to respond to an inquiry, it must be in the inquiry scan state at the same time a p-master is in the inquiry state. Bluetooth has a mechanism which deals with this timing issue; the idea being to have the inquirer (p-master) hop more frequently than the p-slaves so that both devices lock to the same inquiry frequency. The p-master divides its 32 hopping frequency sequence into two sets of 16, called *A-Train* and *B-Train*. Each train lasts for $10ms$ ($16 \times 625\mu s$) and must be repeated at least 256 times before switching to the other. The total duration that a p-master remains in the inquiry state is called $T_{w\_inquiry}$ [82]. In order for the p-master's transmit frequency to match the p-slave's, the latter must remain in the inquiry scan state for a duration equal to or greater than $10ms$; this duration is called $T_{w\_inquiry\_scan}$ [120]. Furthermore, a p-slave must be able to enter the inquiry scan mode regularly in order to check for inquiry messages from the p-master; this interval is referred to as $T_{inquiry\_scan}$. The permissible ranges for $T_{inquiry\_scan}$ and $T_{w\_inquiry\_scan}$, according to Bluetooth specifications, are $[10.625ms - 2560ms]$ and $[11.25ms - 2560ms]$. Because a p-master has no a priori knowledge of p-slaves, it assumes that $T_{inquiry\_scan}$ is set to the maximum, *i.e.,* $2.56s$, thus it repeats each train at least 256 times, as each train takes $10ms$ to complete.

When a p-slave receives the IAC, it backs off for a random number of time slots before sending its FHS to the p-master. This is done in order to avoid possible collisions with replies from other p-slaves which respond to the inquiry simultaneously. This random number is drawn uniformly from the range [0,127] if the inquiry scan interval is less than $1.28ms$, and from the range [0,1023] if larger than $1.28ms$ [82]. Figure 2.6 illustrates the Bluetooth inquiry mechanism.

2.5.2 Paging

The page procedure is used to invite a p-slave to join a piconet when a p-master has knowledge of the p-slave's BD_ADDR, which is either obtained through inquiry or is input manually by the user.

A p-master sends two page packets per time slot then switches to the corresponding response fre-

[4]For disambiguity, inquiry messages are referred to in this section as IAC for Inquiry Access Code

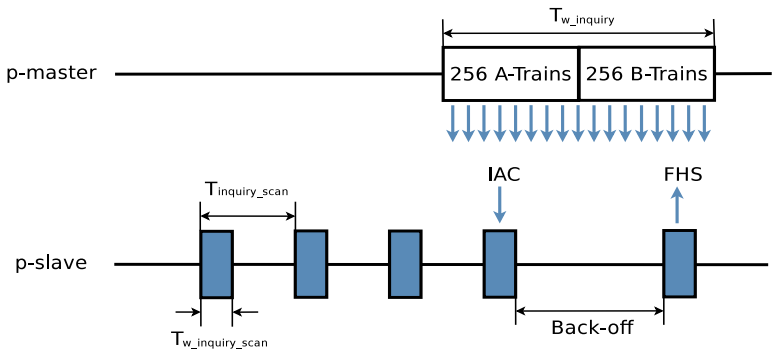


Figure 2.6: Inquiry Mechanism

quencies in the next odd time slot to hear the p-slave's reply if the latter happens to be in the page scan state. A nuance with the page mechanism, however, is that the p-master targets individual devices. It therefore sends a Device Access Code (DAC), to which only the targeted p-slave can respond [24]. The key however, is to coordinate the page and the page scan states so that when the p-master sends the page packet, the p-slave is ready to receive it.

Similar to inquiry, the p-master divides its hopping frequencies into two sets of 16, also called A-train and B-train and sends DACs at a much higher rate than the p-slave's hopping rate. Moreover, a p-master can estimate the hopping phase used by the p-slave by calculating an estimate of the p-slave's current CLKN. This is done by adding an offset to the p-slave's previous CLKN obtained during inquiry, with the aid of the FHS packet's time stamp. This new estimated clock (CLKE) is used to estimate the hopping frequency that the p-slave will be listening on when it enters the page scan mode. A p-slave enters the page scan mode every T_{page_scan} and spends $T_{w\_page\_scan}$ in it. A p-slave has a choice of three T_{page_scan} intervals: continuous, $1.28s$ or $2.56s$. Whichever interval is selected, the p-master will have knowledge of this choice as the p-slave would have communicated this information to the p-master in the *scan repetition* (SR) field in the FHS packet during the inquiry stage. From this information, the p-master can select the number of times a train is repeated before switching to the other train, so that if CLKE is accurate, then the p-slave's scanning frequency would coincide with one of the 16 frequencies that the p-master has selected.

Upon reception of the DAC, a p-slave responds to the p-master by sending the same packet back on the response frequency $625\mu s$ later. When the p-master hears this reply it sends an FHS packet to the p-slave, so that both devices can synchronise. Following this, the p-slave sends its DAC to the p-master as an acknowledgement. Upon successful reception of this last DAC, the p-master becomes master and the p-slave becomes slave. Both devices then switch to the channel hop frequency and the link is tested by exchanging a Poll/Null pair of packets.

2.6 Low Power Modes

The Bluetooth standards specify three different low power modes which allow devices to momentarily limit their transceivers' activity when not needed. These modes are *HOLD*, *PARK* and *SNIFF* [120]. The authors of [91] compare the three low power modes and show that the PARK mode has the lowest energy consumption, whereas SNIFF offers the best response time. Although low power modes were originally introduced as a way to control energy consumption within a piconet, they also have a fundamental use in scatternet operations.

2.6.1 HOLD Mode

When a slave does not need to participate in the current piconet, it enters the HOLD mode for a predefined duration, after which it resumes normal piconet activities. Before a slave enters the HOLD state, the master and the slave agree on the HOLD Timeout ($hold_{TO}$), which is the time interval during which the slave becomes unavailable. During this interval it may sleep, carry out inquiry/paging or attend another piconet; however, it cannot resume its active membership until the timeout has elapsed [30]. HOLD is a one off procedure, meaning that each time a slave needs to exit the piconet, a new $hold_{TO}$ is negotiated. During HOLD, a slave keeps its AM_ADDR, so that it can rejoin the piconet almost instantaneously [91]. This mode can also be used for the master to exit the piconet temporarily. This is done by placing all active slaves in HOLD, which allows the master to carry out other duties or attend another piconet from the time the last slave enters the HOLD mode until the first slave exits HOLD.

In scatternet operations, this low power mode can be advantageous as $hold_{TO}$ can be adapted to suit changing piconet load so that a device can share its membership between the various piconets efficiently. Examples of inter-piconet scheduling which use HOLD are proposed in [61] and [66]. One obvious drawback of using HOLD, however, is that a master cannot admit an extra member to participate in the piconet because the AM_ADDR is still occupied.

2.6.2 Park Mode

A slave in the PARK mode halts all its piconet activities and only wakes up at regular intervals to synchronise with the master [83]. It substitutes its AM_ADDR with two addresses: PM_ADDR and AR_ADDR so that it can be unparked [120]. The PM_ADDR distinguishes parked members and is used by the master when it needs to call upon the parked slave to join the piconet as an active member. The AR_ADDR, on the other hand, is used when the slave itself initiates the unpark procedure [24].

A master requests a slave to enter the PARK state by sending an `LMP_park_req` command to the slave. If the PARK request is accepted the master then sends an `LMP_park` command containing beacon controls [121]. These controls include the interval between beacon instants (T_B), beacon duration (Δ_B) and number of beacon messages during each Δ_B. The beacon messages, themselves, can be either ACL, SCO or NULL packets and are used for re-synchronisation, general broadcasting, altering beacon control

parameters and unparking a parked slave [120].

When a master needs to unpark a particular slave, it sends an `LMP_unpark_PM_addr_req` in a beacon. One particularly powerful feature of the PARK mode is the ability of slaves to initiate their own unpark procedure. This is done by allowing a parked slave to send an unparked request to the master during an *Access Window* following the beacon. Access window controls are also sent in the `LMP_park` message and contain the access window start instant, D_{access}, the duration of the access window, T_{access}, and the number of access windows allowed after each beacon, M_{access}. A slave-initiated unpark is only allowed after a broadcast beacon, since active slaves do not reply to broadcasts, thus minimising chances of collision. When a slave wishes to rejoin the piconet, it sends the master's DAC at the appropriate time slot, as an unpark request. Another advantage of PARK, is that a master can invite other devices into the piconet since the AM_ADDR is released by the parked slave. Nevertheless, when compared to HOLD, PARK mode is more complex in terms of data and time overhead due to the unpark procedure.

2.6.3 Sniff Mode

When a slave is in the SNIFF mode, it only activates its transceiver at regular intervals called *SNIFF Time Intervals* (T_{sniff}) [120]. Unlike normal piconet operations, SNIFF allows a device to wake much more infrequently. This mode allows devices to exit a piconet for up to $40.96s$, during which a slave can either sleep or join another piconet.

A master invites a slave to enter the SNIFF mode by specifying the SNIFF parameters [112]. The parameters needed, in addition to T_{sniff}, are $N_{sniff\_attempts}$, $N_{sniff\_timeout}$ and D_{sniff}. $N_{sniff\_attempts}$ indicates the number of time slots a slave needs to keep its receiver on each time it turns it on. If any data is received during this interval, the slave leaves its receiver on for a further $N_{sniff\_timeout}$. D_{sniff} indicates the number of time slots until the first SNIFF.

A slave in SNIFF mode keeps its AM_ADDR, therefore the device can reintegrate the piconet immediately. Furthermore, the regularity of the SNIFF time intervals is ideal for cyclic scheduling. However, it can also be used for adaptive scheduling as suggested by [12]. One drawback of SNIFF, nonetheless, is that it limits the number of possible participants in a piconet to 7.

2.7 Link Layer Control and Adaptation Protocol (L2CAP)

Data is often larger than can be supported by ACL packets. L2CAP provides a functionality for segmenting large streams of data and reassembling the packets at the receiving end. L2CAP introduces its own encapsulation overheads that identify one L2CAP stream from another, provide information about the size of the data, and specify QoS parameters. L2CAP can be thought of as a data conduit (channel) for ACL packets, such that baseband restrictions are seamlessly shadowed for higher layer protocols. The main functions of L2CAP are [25]:

1. **Protocol multiplexing**: One of L2CAP's functions is to multiplex various protocols transparently to the lower layers. Protocol multiplexing is achieved via virtual channels, wherein a channel is a conduit that links the higher protocols of the communicating devices via baseband communication.

2. **Segmentation and reassembly**: L2CAP segments long data frames into smaller blocks to fit into ACL packets. At the receiving end, L2CAP reassembles the data to re-form the longer frame.

3. **Group management**: L2CAP is capable of managing a group of devices within the piconet by mapping a particular protocol to a group of addresses (devices); for example managing a VoIP conference call.

4. **Quality of Service**: Part of the L2CAP channel set-up procedure is the negotiation of quality of service parameters. L2CAP also monitors these parameters for the duration of the channel.

Each L2CAP virtual channel is identified by a pair of 16-bit *Channel Identifiers* (CID): the *Destination Channel identifier* (DCID), which represents the receiver, and the *Source Channel Identifier* (SCID), used by the source to distinguish between the various outgoing virtual channels. SCIDs are used internally by devices to manage the numerous simultaneous virtual channels, whereas DCIDs are appended to the L2CAP frame as part of the L2CAP header to identify the conduit. Moreover, when a master-slave pair establishes an ACL link, an L2CAP signalling channel between these devices is also created. Figure 2.7 shows the format of an L2CAP frame.

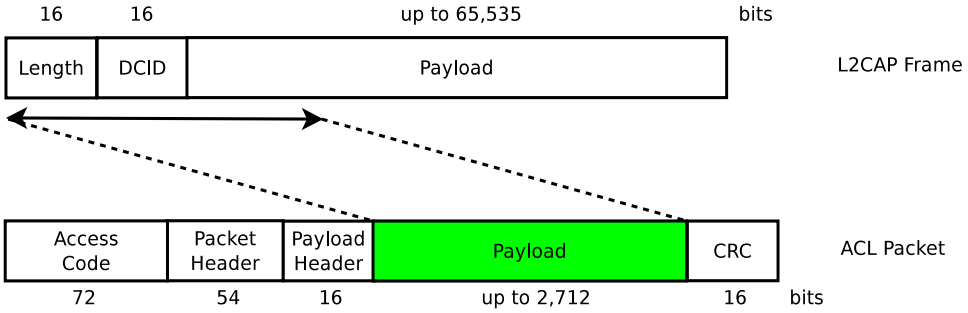


Figure 2.7: L2CAP Frame

As can be seen from Figure 2.7, segments of L2CAP data are carried over ACL baseband packets. Since L2CAP does not provide any mechanism that deals with channel reliability, it relies on the baseband for such functions [112]. It is important to note that L2CAP does not offer the possibility to interleave different frames, *i.e.,* a device can only begin a new frame once all the segments of the current frame have been transmitted.

2.8 Bluetooth Network Encapsulation Protocol (BNEP)

The Bluetooth Network Encapsulation Protocol (BNEP) [20] provides Ethernet functionality to Bluetooth PANs. It defines its own addressing scheme that is compatible with IEEE802.3 addressing [1]. The aim of BNEP is to provide networking capabilities to Bluetooth PANs, so that existing network protocols, such as the Internet Protocol (IP), can be seamlessly incorporated without any modification.

BNEP is used to transport both data and control packets directly over L2CAP. Data packets are used to carry user information, whereas control packets carry control messages that are essential for setting up Ethernet-like links between Bluetooth devices [20]. The format of the BNEP header depends on its type; however, every header contains a Type field that indicates the type of data being transported and an Extension flag.

2.8.1 BNEP Data Packets

These packets carry user data between Bluetooth devices in an Ethernet-compatible fashion, *i.e.,* using similar addressing scheme and protocol types as defined by the Ethernet specifications [1]. There are four different types of BNEP data packets depending on what addressing information is included; these are `BNEP_GENERAL_ETHERNET`, `BNEP_COMPRESSED_ETHERNET`, `BNEP_COMPRESSED_ETHERNET_SOURCE_ONLY` and `BNEP_COMPRESSED_ETHERNET_DEST_ONLY`.

The format of the general BNEP packet is shown in figure 2.8

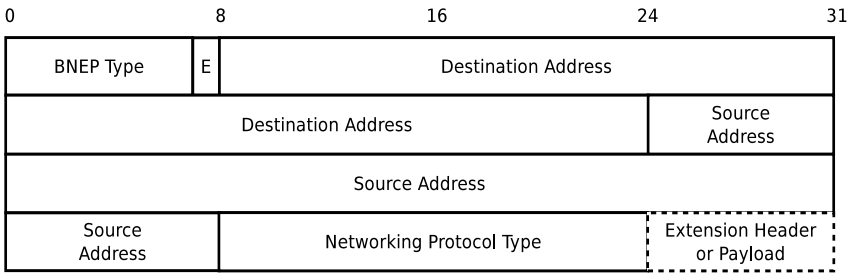


Figure 2.8: General BNEP Packet Format

BNEP Type: Identifies the type of BNEP header contained in the packet.

E: This flag indicates if an extension header follows the BNEP header.

Source Address: 48-bit Ethernet address of the source device.

Destination Address: 48-bit Ethernet address of the target device.

Networking Protocol Type: Identifies the networking protocol contained in the payload.

2.8.2 BNEP Control Packets

The main function of control packets is to manage BNEP connections between communicating devices. Setting up a BNEP connection involves the exchange of Request/Response commands that are contained entirely within the header, meaning that control packets do not carry any payload [20]. The generic

format of BNEP control packets are as shown in Figure 2.9

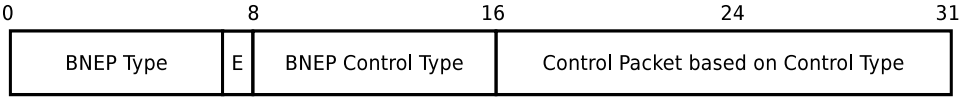


Figure 2.9: BNEP Control Packet

The BNEP Type field shall be set to 0x01 to indicate that it is a control packet. The 8-bit BNEP Control Type field indicates the type of command included in the header. The specifications define seven distinct commands, which leaves room for additional control commands to be specified. This feature is of relevance to this project as it allows the definition of new commands for routing management purposes.

2.9 Bluetooth Profiles

Bluetooth SIG has identified a number of usage scenarios for Bluetooth and grouped these models into *profiles* [121]. A profile encompasses several protocols and building blocks from the Bluetooth stack to enable devices to support certain applications such as telephony, fax, audio streaming and so on. Some of the defined profiles include File Transfer Profile (FTP), Hands-Free Profile (HFP) and Personal Area Networking Profile (PANP)[5]. The latter is highly relevant to the work presented in this thesis, as it defines the way devices discover each other, connect to one another and exchange data in an ad hoc fashion.

2.9.1 Personal Area Networking Profile

The Personal Area Networking Profile (PANP) resides within the Generic Access Profile (GAP). It describes a set of protocols and mechanisms for Bluetooth-enabled devices to search for other devices, establish connections and exchange Ethernet traffic using BNEP [127]. Because the PANP profile is part of the GAP profile, it inherits some of its characteristics. Therefore a PANP User (PANU) has the capability to set its discoverability, connectibility, pairability and security modes as required. The PANP profile defines two usage scenarios: *Group Ad Hoc Networking* (GN) and *Network Access Points* (NAP). In the former, devices have the ability to form a peer-to-peer ad hoc network, wherein each device acts both as a peer and as a relay; whereas in the latter, a network access point links a Bluetooth local area network with an external network such as the Internet, ISDN or UMTS. A Bluetooth NAP implements the bridging functions as set out by the 802.1D standards [68]. Of the the two scenarios, the GN is most relevant to this project as it deals with ad hoc personal area networking. The steps involved in such a process are described below [127]:

1. The initiating device sends inquiry and service dicovery messages to search for nearby devices that support the GN functionality.

[5]The Personal Area Profile is referred to in the literature using the acronym ”PAN”; however, in order to avoid confusion with the acronym used for Personal Area Network in this thesis, the author refers to the Profile as PANP

2. Once the target devices are found, the initiator requests the establishment of L2CAP channels. The Maximum Transmission Unit (MTU) of BNEP packets is negotiated at this stage.

3. A BNEP connection is then created through the exchange of BNEP control packets so that Ethernet traffic can flow between the communicating devices.

When a connection is no longer required or if the service requirements are no longer met, a device can request an L2CAP disconnection, which would disconnect the BNEP link. Furthermore, a connection is automatically dropped if the devices are no longer within communication range or if BNEP fails.

In a PAN, a device may join or leave the network arbitrarily. When a node joins the PAN, it is either accepted automatically or may first require authorisation. The profile defines three authorisation modes, which are described below:

Open PAN: In this mode a node may join the network without any authentication or authorisation from other PANUs.

Authentication Required: Before the new node is registered as a member of the ad hoc network, it needs to authenticate itself by responding to an LMP authentication request. The new entrant may also be required to provide authentication at the Ethernet or IP layer if necessary.

Authorisation and Authentication Required: In this mode of operation, a channel is not created until the new device authenticates itself and is authorised to join the network. Authorisation is performed either locally or via a server entity by consulting a device database of authorised users.

If either authentication or authorisation fail, the L2CAP channel set-up is terminated, thus no BNEP connection is established. In addition to authentication and authorisation, the PANP profile also specifies two modes of encryption that shall be enforced on all BNEP communications; these are:

Clear Mode:No encryption is used for data exchange.

Encrypted Mode: All communication within the PAN is encrypted. The encryption keys and other encryption parameters are negotiated through the exchange of LMP messages.

2.10 Summary

This chapter introduced the Bluetooth architecture, covering the major aspects of its protocol suite. It explained how Bluetooth devices connect and exchange data and presented the mechanisms governing medium access and link management. It also highlighted the elements relevant to this project and explained how specific components are used. In particular, it focused on how personal area networking over Bluetooth can be achieved. Although the material presented in this chapter does not cover every aspect of the Bluetooth technology, it is believed that it constitutes a good introduction for the novice and provides a general overview of the Bluetooth standards. For further detail on the material covered in this chapter, the reader is referred to the references provided herein.

Chapter 3

Ad Hoc Personal Area Networking

3.1 Introduction

This chapter provides the reader with the background necessary material to understand the proposed solutions and strengthens the motivations behind the research project. The material presented herein is based on a collection of previous published works on various aspects of ad hoc networking in general and on Bluetooth personal area networking in particular. The chapter begins by describing the particularities of Bluetooth personal area networks, covering topology construction, membership management, transmission scheduling and error control. It then discusses routing in wireless ad hoc networks and presents some of the predominant routing protocols, highlighting their characteristics, strengths and weaknesses.

3.2 Bluetooth Personal Area Networking

A Bluetooth personal area network (PAN) is a collection of Bluetooth devices arranged in an ad hoc peer-to-peer fashion without relying on any fixed infrastructure. As outlined in the previous chapter, Bluetooth devices are arranged in pico-cells, referred to as *Piconets*, which must contain a master and up to 7 active slaves. Several piconets may be interconnected to form larger networks called *Scatternets*, via bridge nodes referred to as *Participants in Multiple Piconets* (PMP). PMP nodes may either be slaves in all the piconets they participate in, or can be slaves in some piconets and master in their own piconet. A PMP, nonetheless, can only participate in one piconet at a time, hence it must share its membership among the various piconets in a time-division manner. In Bluetooth, a device has the ability to enter one of the specified *low power modes* that allow it to momentarily exit its piconet to participate in another piconet's activities, hence providing support for scatternet operations. Other influencing factors on the performance of personal area networks include network composition, scheduling and error control mechanisms. Figure 3.1 shows an interconnected Bluetooth scatternet.

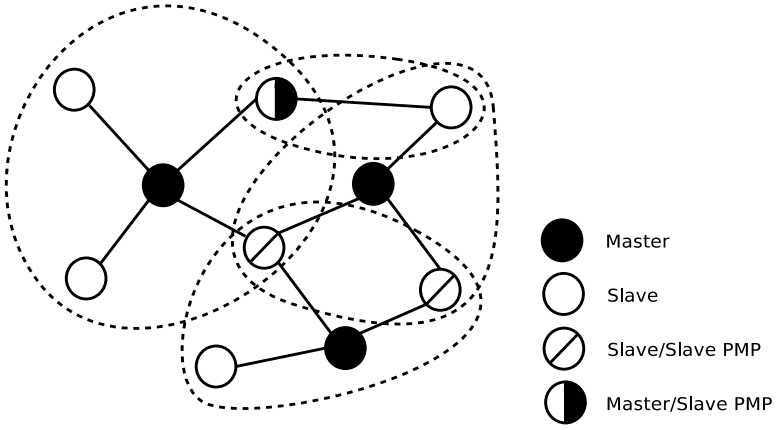


Figure 3.1: Bluetooth Scatternet

3.2.1 Network Formation

Scatternet formation and self organisation of Bluetooth devices are fundamental to the performance of PANs, as they dictate the way nodes communicate with one another, how efficiently data is relayed across various interconnected piconets and have a direct effect on routing performance [9, 15, 167, 87]. Miklós *et. al.* [111] have studied the correlation between the number of links and PMP degrees in Bluetooth scatternets. They found that there is an optimal point, given a certain number of nodes, at which traffic is maximised. The basic idea is that increasing the number of links would increase traffic but at the same time it increases the degree of PMP nodes, which in turn affects network throughput. Therefore, there is a tradeoff between throughput, latency and bridging overheads in Bluetooth scatternets as analysed in [84, 115]. Furthermore, the authors in [107] state that it is possible to find optimal Bluetooth scatternet topologies that maximise throughput while minimising energy consumption through centralised optimisation. Nonetheless, Chiasserini *et. al.* [36], compare centralised and decentralised scatternet formation approaches and argue that, although distributed algorithms result in sub-optimal performance when compared to centralised schemes, they are still preferable as they are more flexible and scale better with network size. There has been much work done in this direction and many algorithms have been proposed to build scatternets which are as efficient as possible, with each one presenting strengths and weaknesses. This section gives an overview of scatternet formation and describes some of the predominant algorithms.

An initial attempt to solve the network formation problem for Bluetooth PANs is presented in [145]. The proposed Bluetooth Topology Construction Protocol (BTCP) consists of three phases: 1) Coordinator Election, 2) Role Determination, 3) Connection Establishment. In the first phase, a network leader is elected, which gathers the number, identities and synchronisation information of all the nodes in the network. The leader is elected through iterative duels, wherein nodes compare each other's `VOTES` variable, which counts the number of duels won so far. The winning node is the one that holds the larger

value of VOTES in the one-to-one confrontation, stores the information of the loser and increments its VOTES variable. Eventually, there will be a single leader that holds information about every other node in the network. The information gathered during the election phase, is used by the coordinator (leader) to assign roles to the different nodes. The coordinator starts by calculating the number of masters, P, necessary to fully connect N nodes into a scatternet as $P = \lceil \frac{17-\sqrt{289-8N}}{2} \rceil$, provided $N \leq 36$, and assigns the role of master to both them and itself. The leader then calculates the number of PMP nodes needed as $\frac{P(P-1)}{2}$ and assigns the role to the corresponding nodes such that connectivity requirements are met. The remaining nodes are assigned the role of pure slaves. In the final phase the coordinator instructs the nodes to enter page and page scan accordingly in order to form the scatternet.

Bluetrees [177] is a self routing scatternet formation algorithm that builds a spanning tree rooted at the *blueroot*. The root node (blueroot) is assigned the role of master and pages its surrounding nodes to become its child slaves. These slaves then are assigned the role of masters to form their own subtree (piconet) and, in turn, page unassigned nodes to become their slaves. This process is repeated until the leaves are attained. Running the Bluetrees algorithm results in a connected spanning tree rooted at the blueroot with each intermediate child taking up the role of a master/slave bridge. A distributed form of the Bluetrees algorithm, Distributed Bluetrees, is also proposed in [177], which eliminates the issue of conflicts in simultaneous spanning tree expansions but admits an additional type of bridge node: M/S/S. Distributed Bluetrees works in two phases; phase I identifies a set of *init* nodes that take the role of master root of their own tree. The protocol for building these trees is similar to the Bluetrees algorithm except that a node may join a particular tree only if it is not part of another tree. At the end of phase I, the network will be composed of a number of distinctly disjointed trees, each rooted at its own init node. In the second phase these disconnected trees are joined together to form a connected scatternet, by considering each of the disconnected trees as a virtual vertex and running the Bluetrees algorithm, ensuring that connectivity requirements are not violated. An advantage of Bluetrees is that it guarantees connectivity of the network, however, it does not cope well with partitioning and node mobility as demonstrated in [9], and can only admit master/slave bridge nodes, which can be problematic in inter-piconet scheduling.

The algorithm proposed in [98] also results in tree scatternets. In this scheme, the network is initially composed of disjointed components, with a leader; these components are then merged to form the larger scatternet. A component may be a piconet, a scatternet or an isolated node and the leader of each component must be a master node. Each leader performs the procedure SEEK to inquire and page new devices while other devices perform the procedure SCAN, which are in essence the same as the inquiry scan and page scan procedures. When two components merge, they first try to form a single piconet if transmission range and connectivity restrictions allow. In this case, one of the leaders becomes the master of the new piconet and the other retires to become a piconet member; this is achieved by the

procedure MERGE. When components cannot merge into a single piconet, an unshared node becomes a bridge node linking the two segments and one of the two leaders becomes the leader of the merged component; the other leader retires and seeks connection to other devices to complete the tree; this is done through the procedures MOVE and MIGRATE. A strength of this algorithm is that it forms connected scatternets relatively quickly and with low time complexity. However, it requires nodes to be within communication range of one another, which is an significant handicap. Moreover, the scatternet formed is a tree and, like Bluetrees, creates bottlenecks in the network and does not cope well with mobility.

To alleviate the problem of network bottleneck, some authors propose mesh topologies. Petrioli *et. al* [137] introduce the *Bluestars* algorithm which constructs a mesh network of interlinked bluetooth nodes. The algorithm works in three phases: in phase I, adjacent nodes learn about each other's identities and *weights*. Using this information nodes are then arranged into piconets in phase II. In the final phase the individual piconets are interconnected to form the scatternet. The *weight* of a node dictates its ability to become the master of its piconet. A node becomes a master if it has the greatest weight amongst its neighbours, or if it learns that all its larger-weighted neighbours have become slaves in other piconets. Once the set of masters has been selected, they can invite slaves to join their piconet via paging. A slave chooses to connect to the master that has the greater weight amongst its neighbours. After completing phase II, the network would be composed of disconnected star-shaped piconets, thus the name Bluestar. These piconets are connected to one another to form a *BlueConstellation* (scatternet). This is done by selecting a set of intermediate nodes, which ensure that the entire network is connected. The advantage of Bluestars is that it guarantees network connection in a mesh; nevertheless, piconets might admit more than 7 slaves which means that some need to be parked and unparked regularly, which degrades the performance of the network [167].

The drawbacks of Bluestars are rectified in [138], by restricting the maximum number of slaves per piconet to 7. The proposed scheme, *Bluemesh*, is similar to the Bluestars algorithm in the sense that it is divided into phases: A neighbour discovery phase and a network connection phase. In fact, phase II of the Bluemesh algorithm (network connection) is sub-divided into a *role selection* element, in which nodes are arranged into piconets, and a *gateway selection* element, in which the piconets are connected to form the scatternet. This phase is, in essence, phases II and III of the Bluestars algorithm combined into a single phase. The neighbour discovery phase is, in turn, divided into two steps: in the first, a node discovers the identities and weights of its one-hop neighbours and in the second each node informs its adjacent neighbour of its own one-hop neighbour. Consequently, each node would have knowledge of the identities and weights of its two-hop neighbours. Once the selection phase is completed, the set of *init* nodes is determined. Init nodes are master nodes that have the largest weight among their neighbours. Each master from the init set, enters the page state and starts inviting (paging) slaves from its immediate neighbourhood to join the piconet, allowing a maximum of 7 slaves per piconet. This is

achieved through an iterative process, in which an init node repeats the paging process until either all of its immediate nodes join its piconet or it discovers that the neighbours have already been invited by two-hop away nodes. This method allows the construction of piconets of no more than 7 slaves, while ensuring that no unattached node is left disconnected. The weakness of Bluemesh lies in the fact that it needs a two-step discovery phase, which might be impractical for highly dynamic networks.

BlueNet [166] is another distributed algorithm that builds a mesh scatternet. This scheme is divided into three phases: 1) Initial piconet formation, 2) connecting unattached nodes, 3) interconnecting individual piconets. In the first phase, nodes build a local visibility graph by learning the identities of their neighbours through inquiry, then enter the page state randomly to form initial piconets, making sure that a piconet does not exceed a maximum number of slaves N_{max}. In order to avoid overlapping piconets, a slave does not respond to pages once it becomes connected to a master, unless instructed to do so. Moreover, in order to avoid constructing piconets within piconets, master nodes broadcast the identities of the slaves within their respective piconets. Upon completion of phase I the network will be composed of separate piconets, with some nodes remaining unattached. In phase II, the unattached nodes enter page and invite up to N_{max} of their neighbours to join their piconet. When a previously attached slave connects to the new piconet it becomes a PMP node. The various disconnected components are joined together to form the final mesh scatternet in phase III. In this phase, a master instructs some of its slaves to enter page scan to respond to pages and others to enter page to form their own piconets, thus linking isolated components. The number of slaves instructed to enter page or page scan modes in phase III may be either random or set in such a way as to satisfy some connectivity requirement. Some of the advantages of Bluenet include flexibility and simplicity when compared to other network formation protocols. In addition, it guarantees mesh connections, which is a desirable feature of Bluetooth scatternets as it allows resilient routing and alleviates the problem of network bottlenecks. The topology construction algorithm employed in the project presented in this thesis is a variation of the Bluenet algorithm; however, it differs in that the proposed scheme only admits slave/slave PMPs, as this facilitates interpiconet scheduling when compared to master/slave bridge nodes.

3.2.2 Transmission Scheduling

As previously mentioned, Bluetooth relies on a time-division polling scheme, wherein communication opportunities are divided into time slots and a slave can only communicate with one master at a time if it has been contacted in the previous time slot. This means that masters need an efficient way to schedule transmissions in order to maximise throughput while remaining fair. There are two types of scheduling: intra-piconet scheduling and inter-piconet scheduling, however, the two are closely related to one another. For example, a PMP that is polled for transmission in a given time slot in a particular piconet is unavailable for participation in another piconet, thus affecting inter-piconet scheduling. Sim-

ilarly, a device that is busy relaying data from a different piconet cannot participate in the intra-piconet communication of another piconet.

3.2.2.1 Intra-piconet Scheduling

Polling refers to the action of a master node calling a particular slave for transmission. A slave can only transmit if it has received data in the previous even-numbered time slot or if it has received a *POLL* packet, which is a baseband packet that contains no user data. The slave sends its data in the following odd-numbered time slot or simply sends a *NULL* packet (another baseband packet with no data) if it has no data to send. When the piconet contains more than one slave, the master needs an efficient way to poll its slaves one by one.

The simplest intra-piconet scheduling scheme is one where a master polls each slave in a cyclic fashion, allocating a fair share of the available slots, *i.e.* following a *Pure Round Robin* (PRR) schedule. In PRR, a master allows each slave to transmit a single slot packet in turn. The advantage of this scheme is that it ensures fairness when data transmission is symmetric, however, in most applications, transmissions occur asymmetrically, which renders PRR inefficient and unfair as a master would poll a slave regardless of its queue occupancy [41, 182].

There have been variants of PRR proposed to overcome its shortcomings; these include *Exhaustive Round Robin* (ERR), *Exhaustive Pseudo-cyclic Master Queue Length* (EPM), [31], *k-limited Round Robin* [168] and *Deficit Round Robin* (DRR) [161]. K-limited scheduling is a modification of PRR, in which a master allocates a varying number of time slots to each slave. In fact, PRR can be regarded as a special case of k-limited Round Robin when $k = 1$.

ERR introduces a degree of fairness to the cyclic scheme by moving to the next master-slave pair only when both the current master and the slave queues are empty. EPM is a more dynamic Round Robin-based schedule; it defines an order of visits at the beginning of each cycle based on a decreasing queue length and visits each slave only once in each cycle. Deficit Round Robin (DRR) is another RR variant, similar to PRR, but instead of serving each queue cyclically, it takes packet length into account. The scheme was developed for general queue management, however its applications to Bluetooth intra-piconet scheduling are evident. In DRR-based polling, each slave is assigned a time *quantum*, which represents the time allocated by the master for the slave to send a packet. If the packet is larger than the quantum then the queue is not served and the remainder of the quantum is added to the next poll. This means that deficits accumulated by slaves are compensated for in future rounds. Maalaoui *et. al.* [105] show that by incorporating QoS criteria, in addition to packet length, DRR can yield near optimum performance.

The main advantage of Round Robin scheduling is its simplicity, which satisfies Bluetooth's requirements, however, it still lacks adaptivity to varying traffic conditions. The work described in [41]

takes note of this fact and proposes three adaptive schemes that deal with varying traffic flows, namely *Adaptive Flow-based Polling* (AFP), *Sticky*, and *Sticky Adaptive Flow-based Polling* (StickyAFP). These algorithms rely on the traffic measure *flow* to adjust the polling interval for each slave. In AFP, a slave with $flow = 1$ is served more frequently than a slave with $flow = 0$, and if the slave fails to send data after being polled, the polling interval is doubled. In Sticky, a master serves a slave for num_sticky packets if $flow = 1$ and behaves like PRR if $flow = 0$. StickyAFP is similar to AFP except that if $flow = 1$ a slave is served for num_sticky packets.

Limited and Weighted Round Robin (LWRR) is another adaptive cyclic scheduler that takes into account previous queue occupancy. In this scheme, each slave is assigned a value Max_Priority (MP) which is incremented each time a slave has data to send when polled, and decremented otherwise. The frequency of visits, which has a minimum value of 1, is thus adapted to the MP value of each slave. The authors extend the idea of LWRR to a more dynamic Round Robin-based scheduling scheme. In [101], they propose the *Pseudo-Random Cyclic Limited and Slot-Weighted Round Robin* (PLsWRR) scheme, which polls slaves in a pseudo-random order and takes into account the traffic history of slaves. This scheme outperforms LWRR in terms of fairness and throughput.

In addition, [117] describes a scheduling scheme that uses traffic load and QoS parameters as polling criteria. In the *Adaptive Cyclic-limited Scheduling* (ACLS), slaves with high load are served with a larger portion of the cycle time than slaves with low load. Furthermore, ACLS also tries to maintain QoS guarantees by serving slaves that require strict polling intervals accordingly. In ACLS, slaves are served depending on their traffic load in the previous cycle, where a cycle is a round in which every slave is polled. The authors show that ACLS performs better than ERR in terms of throughput and delay, however, the algorithm does not have a mechanism to predict future loads, thus it lacks optimisation.

Cordiero *et. al.* [38] take a different approach to tackling the intra-piconet scheduling problem in personal area networks. The authors describe a method for assigning slots and partitioning the network dynamically in such a way as to provide QoS guarantees. In the proposed *Dynamic Slot Assignment* (DSA), a master/slave pair negotiate the establishment of a new connection that guarantees the desired QoS parameters, including slot and duration. If the QoS parameters cannot be satisfied via direct links then piconet partitioning takes place. This is achieved via The Enhanced DSA (EDSA), wherein two or more piconets are created to transmit data from source to destination via bridge nodes. Although DSA and EDSA are adaptive to quality of service, the creation of new links, and in particular the creation of new piconets, incur impractically large delays.

3.2.2.2 Inter-piconet Scheduling

In scatternet settings, a node may need to exchange data with another node that lies outside its piconet via PMP nodes. However, since a PMP can only participate in one piconet at a time either as a master or

as a slave, there must be a mechanism to schedule memberships among the various piconets as efficiently as possible. The measures for efficiency vary from one scenario to another; nevertheless, data throughput and bridging overhead are common criteria for measuring the behaviour of an inter-piconet scheduling algorithm [116]. Moreover, it has been found that slave/slave PMPs perform better than master/slave PMPs in terms of throughput but at the cost of slightly longer routing delays [116]. Inter-piconet operations are possible owing to the previously discussed Bluetooth low power modes (sniff, hold and park), where a node not participating in a particular piconet enters one of the three low power modes while joining another piconet as an active member. Consequently, PMP nodes become network bottlenecks, and their ability to switch from one piconet to another dictates the behaviour of routing and data forwarding in scatternets [149, 16].

The work presented in [77] introduces the concept of *Rendezvous Scheduling*, where a master/PMP pair schedule a meeting at a specific *Rendezvous Point* (RP) and stay connected for a period determined by the *Rendezvous Window* (RW). The authors propose the *Maximum Distance Rendezvous Point* (MDRP) algorithm, in which RPs and RWs are established in advance. MDRP uses the concept of a common superframe, which represents the RP for each PMP associated with a particular master and the pair's corresponding RWs. MDRP uses sniff timers to divide the superframe into regular meeting points (RP); the idea being to maximise the distance between RPs so that PMP nodes are served fairly. This scheme is quite simple and does not require complex management, however, it comes at the cost of not being able to adapt to traffic changes.

Another algorithm that uses sniff to schedule RPs is presented in [16]. The authors describe a solution for building connected scatternets and devising scheduling policies that are optimised depending on the resulting network formation. The proposed *SS-Blue* technique builds scatternets with as few bridges (PMPs) as possible then applies the inter-piconet scheduling algorithm. The scheduling algorithm, itself, is credit-based, meaning that bridge nodes connect to the master with whom they have the largest credit. Credits are accumulated by redistributing unused credits of a particular piconet to all PMP nodes and are spent on exchanging data. In SS-Blue, a master, and a corresponding bridge node, negotiate *Presence Points* (similar to Rendezvous Points) by setting sniff intervals in the superframe, and the decision regarding whether to switch to the other master or not at the Presence Point depends on the credit the PMP node holds with its corresponding piconets, *i.e.,* if the credit the bridge node holds with the new master is greater than the one it holds with its present master, it switches piconets, otherwise, it stays with in its current piconet. The authors show that the proposed algorithm ensures fairness and is more adaptive than MDRP, however, it is optimised for a particular network formation, therefore its performance cannot be guaranteed for arbitrary networks. This drawback is more apparent in situations where the topology changes due to node mobility.

The algorithm presented in [60] and [61] describes the *Load Adaptive Algorithm* (LAA), which is a

simple but efficient inter-piconet scheduling mechanism based on the hold mode. Hold is more flexible than sniff, in that the hold timeout can be set as required, whereas in sniff the intervals are fixed. As per LAA operations, a bridge node calculates the time required to connect to a different piconet according to the other master's queue size and traffic, setting the *Time Commitment* (TC), which is simply the hold timeout. However, because queues can be very large, LAA specifies the maximum queue size to be served (MQS) so that a PMP can exit the piconet before the queue is exhausted. Furthermore, because the nature of traffic can vary, a predictability factor, β is used to estimate the average packet size to be served in the queue so that TC can be adapted. In addition, a bridge cannot remain indefinitely connected to a particular master, therefore LAA defines *Max Time Share* (MTS) as being the maximum time a bridge node can participate in a particular piconet. These factors are used to compute the hold timeout, which is adapted to traffic patterns and queue size so that PMP nodes participate in their corresponding piconets fairly. The disadvantage of LAA is that it is limited to networks of two piconets, therefore, it is only effective in small-scale networks.

An improvement to LAA, *Predictive Inter-Piconet Scheduling* (PIPS) [67] is also based on the hold mode for scheduling PMP presence in multiple piconets; however, unlike LAA, it is designed to work on larger networks. PIPS estimates future traffic by gathering statistics on k previous cycles and adjusts the presence point and length of stay of a bridge node in a particular piconet accordingly. At each cycle, and for each piconet i that a PMP is a member of, it calculates a priority weight W_i based on traffic estimation and queue size so that it can schedule an order of membership from the highest to the lowest weight. Also, similar to LAA, PIPS defines a maximum time for a bridge node to remain connected to a particular piconet so that it can exit hold before the queue is exhausted, thus providing a degree of fairness. The authors show that, when compared to MDRP, PIPS performs better in terms of delay and throughput as traffic increases and scales better with network size (*i.e.,* number of piconets). A variant of the LAA and PIPS algorithms is presented in [125]. The *QoS aware Inter-Piconet Scheduling* (QIPS) scheme builds on the same principles as the previous two methods but adapts to quality of service rather than load. QIPS defines seven QoS classes, so that the priority weight, on which the switching decision is based, is calculated according to the type of service required.

Another similar approach to LAA and PIPS, which is based on the hold mode for inter-piconet scheduling is given in [144]. The Efficient and Fair Scheduling algorithm (EFSA) computes the probability, $P_{data\,i}$, of a master's queue being non-empty, coupled with the number of time slots that have elapsed since the last visit, N_i to schedule presence point with a particular master i. The parameters $P_{data\,i}$ and N_i represent measures for efficiency and fairness respectively, as an accurate estimate of the queue indicates adaptability to varying load, and adjusting the frequency of visits according to transmission opportunities means that the bridge gives every piconet a fair share of the slots available.

The authors in [75] exploit the concept of rendezvous scheduling to devise a new low power mode,

called *Jump* that enables inter-piconet scheduling. A node in the Jump mode (a jumping node) arranges its time into superframes and divides these frames into rendezvous windows of pseudo-random length. A jumping node can then disconnect from its current piconet until the next rendezvous window. The choice of pseudo random window length stems from the fact that fixed intervals require very strict co-ordination among the PMP nodes, which could become problematic if conflicts were to occur; whereas with pseudo random windows this problem tends to resolve itself. However, a jumping node must signal its rendezvous points to its respective piconets, so that it can participate in intra-piconet scheduling if required. There are two cases to consider: slave/slave PMP and master/slave PMP. In the former, the bridge node signals its presence by responding to a poll from the master of the particular piconet at the predetermined rendezvous point. This is achieved by informing all the corresponding masters of the superframe structure. In the case of a master/slave PMP, the master is not required to enter Jump mode in its own piconet since it is responsible for transmission scheduling and hence for polling slaves. However, it should respond to a poll from the piconet's master within the agreed rendezvous window. Jump is a flexible low power mode as it delegates piconet switching decisions to the PMP nodes themselves, hence adding adaptability to varying traffic and QoS conditions; however, it requires a modification to the Bluetooth standards.

3.2.3 Error Control

As explained in the previous chapter, baseband packets may include redundancy checks to verify the integrity of the data received, so that packets can be retransmitted if necessary. Although some packets are FEC-protected, corruptions may still occur from which the error protection cannot recover. While some applications, like real-time voice, can live with few packet losses, others may need very reliable channels, hence the need to re-send corrupted or lost packets. Retransmissions do indeed improve reliability, however, they introduce delays that may become impractically large. Evidently, there is a trade off between reliability and delay to be taken into account when designing retransmission schemes.

One way of improving the reliability of wireless links is to adapt packet sizes to channel conditions as proposed in [104] and [119]. Under high bit error rates (BER), it is sensible to reduce the size of packets and vice versa. If errors occur, only small packets need to be retransmitted, whereas if the packets are large, re-sending the data results in performance degradation. A study of the effect of packet sizes on throughput for Bluetooth links under varying channel conditions is given in [160] and [176]; it is shown that beyond a certain BER threshold, sending large packets results in lower throughput, essentially due to retransmissions of erroneous packets.

Another error control technique over wireless links is to adapt the number of *Automatic Repeat reQuest* (ARQ) transmissions to channel conditions [114, 113]. The aim being to find an optimal number of re-tries that satisfies applications' requirements given certain channel error rates. In ARQ-based

schemes, a packet is retransmitted until it is either received correctly or the maximum number of re-tries is reached, at which point the packet is dropped. However, the number of retransmissions allowed varies from one application to another and may not remain static at all times. Thus using an adaptive ARQ is a sensible approach.

Bluetooth uses a stop and wait ARQ error control mechanism, which re-transmits every packet until received correctly or the maximum number of allowed retransmissions is reached. The work presented in [34] describes an adaptive ARQ scheme that varies the number of retransmissions, *i.e., Retransmission Time-Out* (RTO), to suit channel conditions. In the proposed scheme, a node monitors the *Round Trip Time* (RTT) of the previous transmission and calculates a new RTO for the next packet. The idea is to increase the RTO when the previous packet experiences a delay and to decrease it otherwise, following a multiplicative increase/decrease method, meaning increasing RTO as RTT decreases and vice versa. The reasoning behind the protocol is that if a channel had experiences short RTTs, there would be space to increase the RTO for subsequent packets, thereby increasing channel reliability. The authors show that for real time audio applications, the adaptive scheme behaves well. Nevertheless, it requires a node to calculate RTOs quite frequently which might be power consuming. Moreover, it requires an application-aware link layer, which makes the scheme only applicable in certain scenarios.

Another application-specific error control mechanism is described in [140]. The Fuzzy Logic Control (FLC) ARQ scheme for video transmission over Bluetooth links adapts the ARQ timeout according to buffer fullness and head of line packet delay. The authors argue that buffer fullness is a more appropriate measure as opposed to packet loss, as it is directly available to the application via HCI and also reflects channel conditions. In the FLC ARQ scheme, a source node evaluates its buffer fullness and delay against the predefined membership functions and outputs an ARQ timeout function, which dictates the number of retransmissions allowed. The authors compare their adaptive FLC-based error control technique with the default infinite timeout on video transmission and show that the proposed scheme improves the video quality by up to 4dB. Moreover, fuzzy mapping is quite lightweight when compared to predictive computations, therefore, it is suitable for low power devices.

The work presented in [44] proposes a cooperative ARQ scheme for wireless networks, wherein, nodes cooperate in acknowledging/retransmitting packets. Nodes are arranged in *Cooperation Groups*, in which they are able to hear and monitor each other's transmissions. Each node within the cooperation group receives, decodes and stores the latest packet, and if it is erroneous it sends a *Negative ACKnowledgement* (NACK) back to the sender so that the packet can be retransmitted. This means that if the receiver fails to hear the packet or if its NACK is lost, the receiver will still obtain an acknowledgement of transmission failure from one of the cooperative nodes. Similarly, when a cooperative node hears a NACK from the receiver, it re-sends the packet which it had previously stored. When the receiver acknowledges the correct receipt of the packet, all cooperative nodes delete their stored copies of

the packet. Although this method improves transmission reliability, it does require the participation of all the nodes within the cooperation group, which is not power efficient, and therefore not suitable for personal area networks.

3.3 Routing in Personal Area Networks

When a node has data to send to another device, with which it does not have a direct wireless connection, it needs to rely on other nodes to convey the data in a multi-hop fashion until the destination is reached. Routing is, therefore, a mechanism by which intermediate nodes are selected as relays along the path between a source and a destination taking into account various factors specific to personal area networks, such as channel reliability, node mobility, energy restrictions and connectivity constraints. Routing over PANs can be categorised into three main groups: *proactive*, *reactive* and *hybrid* Routing. In the proactive routing, routes are known a priori and are maintained locally. In reactive routing, a route is set up and maintained only when required, *i.e.,* on demand; whereas hybrid routing combines elements of the previous two. This section only gives an overview of some of the predominant candidates in each goup, as a vast amount of work has been conducted in this direction (for a more complete examination of the extant literature, the reader is referred to [3, 63, 143, 178]).

3.3.1 Proactive Routing Protocols

In proactive routing, every node keeps routing information for all or part of the network. These routes are stored in the form of look-up tables, which are set up a priori and maintained periodically. The advantage of proactive routing is that paths are readily available when required. Nonetheless, they come at the cost of regular updates, which use a sizeable portion of the available resources. The various proactive routing protocols proposed in the literature differ in the way routing information is stored, the type of information needed and how routes are maintained [3]. Some of the predominant protocols include Destination-Sequenced Distance Vector (DSDV), Optimised Link State Routing (OLSR), Global State Routing (GSR), Fisheye State Routing (FSR), Source Tree Adaptive Routing (STAR) and Topology Dissemination Based on Reverse Path Forwarding (TBRPF), each of which is briefly described below.

DSDV [135] is a table-driven routing protocol, in which nodes store and maintain a single path to every destination in the network. The updates are sent at regular intervals through *full dump* and *incremental* packets. A full dump packet carries full topology information and is distributed to every node, whereas incremental packets carry the changes that have occurred since the last full dump packet. Full dump packets normally occupy several data units, as they carry full routing information, while incremental packets are carried in a single data unit. The use of incremental packets to signal a topological change reduces the overhead associated with route updates. However, in highly dynamic networks, DSDV may fail as full dump packets would need to be sent more frequently.

OLSR [70] is based on traditional link state routing, in that each node builds and maintains a topological view of the network, from which the shortest routes to each node are obtained and stored in a routing table. The novelty of OLSR is that it minimises the overhead associated with topology updates by allowing only a subset of the nodes to rebroadcast this information. These nodes are referred to as *Multi-Point Relays* (MPR), which are selected according to their degree centrality; meaning that the number of MRP nodes should be the minimum possible whilst also ensuring that topology updates reach every node in the network. Naturally, selecting MRPs requires regular exchanges of neighbour information amongst adjacent nodes. This is achieved through HELLO messages which, in addition to carrying neighbour information, are also used to check for node presence and link failures.

Global State Routing (GSR) [35] is another link state routing protocol that relies on a topological view of the network. However, in GSR, topology updates only occur locally, *i.e.,* amongst adjacent nodes, thus reducing the overhead significantly. The disadvantage of this protocol is that the update messages are relatively large, which makes it unscalable with network size. Furthermore, given the time-division nature of PANs, sending large packets would impair the performance of the network.

The issues introduced by GSR are rectified in the Fisheye State Routing (FSR) protocol proposed in [133]. Here, each node defines fisheye zones (in terms of number of hops), in which link state updates are exchanged at different frequencies, with nodes closest to the focal point benefiting from the most frequent updates. Contrary to common link state routing protocols, which flood topology updates whenever a change occurs, FSR relies on periodic notifications with varying frequencies. Moreover, in FSR, updates are exchanged locally amongst adjacent nodes, and each update is identified by a sequence number so that nodes keep up-to-date information. This means, however, that FSR trades lower message complexity for lower accuracy, as the update frequency may not match the frequency of topological changes, and consequently routes to remote nodes may become less accurate. Thus in highly dynamic environments, FSR might fail to provide reliable routes.

TBRPF [131] is another link state routing protocol that aims at reducing the amount of overhead associated with link state updates. TBRPF is based on the principle of *Reverse Path Forwarding* (RPF) to guarantee loop-free paths. Here each node builds and maintains a tree to every reachable node in the network based on topological information collected locally from neighbouring nodes. The approach of obtaining and maintaining network topology locally, without relying on flooding, decreases routing overhead significantly. The overhead is further reduced by only reporting the changes in neighbouring nodes' states rather than reporting on the entire topological view. However, the fact that TBRPF relies on periodic updates, means that transient and sudden changes in network topology might go unnoticed.

The STAR protocol described in [50] is a hierarchical link state routing protocol that relies on a partial view of the network to construct source trees from every source to every reachable destination. A major advantage of this protocol is that update overhead is reduced considerably. This is achieved

by only sending updates if certain conditions are met, for example all possible paths to the destination fail. STAR, therefore, trades optimality of routes for lower overhead. The authors argue that although routes may not be optimal, the protocol achieves lower levels of routing overhead when compared with on-demand approaches, while having the benefit of table-driven protocols, *i.e.,* routes are available when needed.

3.3.2 Reactive Routing Protocols

In reactive routing approaches, routes are set up on-demand when a node needs to exchange data with another node. The main advantage of this type of routing protocol is the reduced overhead associated with route maintenance. However, it comes at the expense of longer route set-up time, since a source node needs to search for a suitable path by sending route search requests and must wait for a reply before data exchange can take place. Moreover, on-demand routing protocols do not always guarantee optimal routes [50]. Reactive routing protocols are divided, in turn, into two subcategories: source routing and hop-by-hop routing [3]. In the first category, each packet header carries the addresses of all the nodes that the packet traverses along the path, whereas in hop-by-hop protocols, routes to destinations are stored as next hop entries at each intermediate node along the path. The most popular reactive routing protocols for PANs are Route Vector Method (RVM), Location Aware Routing Protocol (LARP), Relay Reduction and Route Construction Protocol (LORP) and Route Maintenance Algorithm (ROMA). Moreover, other reactive routing protocols, which were developed for general ad hoc networks, have also been deployed on PANs; these include Dynamic Source Routing (DSR), Ad-hoc On-demand Distance Vector (AODV), Temporally Ordered Routing Algorithm (TORA) and Location-Aided Routing (LAR).

RVM [17] is a routing solution that takes into account the specific topology characteristics of Bluetooth. In this method, a node wanting to establish a route to another node, sends out a route search packet, which is flooded throughout the network until the target is found. The destination or a node that has a route to the destination replies with a route reply packet indicating which route to follow. RVM belongs to the family of source routing protocols, *i.e.,* the entire path is included in the header of each packet. However, because Bluetooth device addresses are 48-bit long, RVM uses a mechanism to substitute BD_ADDR with a sequence <AM_ADDR, LoCID>, where the latter locally identifies a particular piconet. A packet sent from source, s to destination, d would include the ordered list of the addressing sequence such that the packet is forwarded to the correct PMP and, in turn the PMP forwards the packet to the correct piconet and so forth until the destination is reached. The pair <AM_ADDR, LoCID> is only 6-bit long and therefore introduces far less overhead than using the full BD_ADDR would. The issue with RVM, however, is that routes are not optimal in terms of number of hops, which may result in transmission delays and increased error probability.

The drawbacks of RVM are rectified in ROMA [175] by devising a protocol that removes unnec-

essary relays in the scatternet. The idea behind ROMA is that relays are bottlenecks in the network and contribute to long route discovery paths. Therefore, by removing the unnecessary bridge nodes, route set-up time and overhead can be reduced. To do this, each node maintains a *Connection Table* (CT) which lists the relays that its corresponding masters connect to. This information is provided by the masters to each relay upon request. After the PMPs construct their CT, they will have a connectivity view of the network and if they discover that connectivity is maintained through another relay, they change their role to pure slaves so that route search packets are not forwarded through them. The routing protocol itself is based on a route search/reply mechanism, in which a node wishing to establish a route to a remote destination floods the network with a route search packet and upon receipt of this packet by the destination, it replies with a route reply message via the reverse path. Each node transmitting the route reply packet includes its address and clock offset and enters page scan mode. When the destination receives the reply, it pages the nodes present in the reply path in the reverse order in order to establish a piconet with them, and hence reduce the number of hops. The authors provide simulation results that indicate good performance in terms of packet delay and packet loss, however, the delay associated with setting up new piconets and routing overhead are not factored.

DSR [79] is a simple reactive routing protocol, which consists of two phases: Route Discovery and Route Maintenance. In the first phase, a source node broadcasts a *Route Request* packet that is flooded throughout the network until it reaches the destination or a node that has a route to the destination. At this point a *Route Reply* message is sent back indicating the list of addresses of intermediate nodes through which packets should be routed. This means that in DSR, each packet carries the ordered list of addresses of nodes along the path, so that when an intermediate node receives the packets it knows which node to forward it to next. In order to reduce the overhead associated with route search, DSR employs a caching mechanism where each node stores the previously acquired routes in the form of address sequences. The route maintenance phase is concerned with detecting and repairing route failures. If a link present in the initial route breaks, the source is notified via a Route Error message and a new route search is initiated if no alternative is available in its cache. The advantage of DSR is that it does not require up-to-date route maintenance [178], as full routing information is contained within each packet. However, as the size of the network grows so does routing overhead, thus DSR does not scale well with network size. Nonetheless, in small to medium size PANs, DSR can be advantageous as it satisfies the low complexity requirements of this type of network.

The Temporally Ordered Routing Algorithm (TORA) [132], is based on the concept of route reversal over a *Direct Acyclic Graph* (DAG), the direction of which points to the destination in a decreasing manner. TORA is an on-demand protocol, in that, routes are set up and maintained on the request of source nodes when required. When a node needs to find a route to a remote destination, it broadcasts a *query* message (QRY), which is flooded throughout the network until the destination or a node that has a

route to the destination is reached. At this point, an *update* packet (UPD) is sent back to the initiator, following the reverse path and assigning directions to the traversed links, which essentially creates a DAG that points to the desired destination. To assign these directions, each node, i, is referenced by its *height* $H_i = (\tau_i, oid_i, r_i, \delta_i, i)$ so that the route from source to destination follows a lexicographically decreasing height path. The first three elements of the height are called the *reference level* and the remaining two are referred to as the *delta*. The reference level uniquely identifies a query, with δ_i representing a time tag, oid_i being the originator's ID and r_i being a flag that distinguishes between a particular height and its reflection. The delta indicates the height of a node with respect to a reference level, where δ_i is an integer and i represents the unique id of the node. During the route update phase, a node sets its height to be higher than its predecessor's and forwards the UPD packet, so that the DAG points to the destination in a lexicographically decreasing manner. Routes are maintained in a similar fashion, *i.e.,* by adjusting heights so that each next hop is a local minima. When a route is no longer needed, the source sends a *clear* (CLE) message to all the nodes along the path, which then reset their heights. Although TORA ensures loop free routing and is highly adaptive to topological changes, its applicability to low power personal area networks might be limited by the fact that it relies on flooding to search for routes and that adjusting and maintaining heights requires rather expensive computations.

AODV [134] belongs to the hop-by-hop category of on-demand routing protocols. When a source node needs to set up a path to a destination with which it does not have a route, it sends a *Route Request* (RREQ) packet to all its neighbours, which in turn forward it to their neighbours, until the destination is found. In order to reduce route set-up overhead, each RREQ is uniquely identified by the pair `<source_id, broadcast_id>`, so that a node which has previously received a RREQ with the same broadcast id generated from the same source drops the packet and does not rebroadcast it. The RREQ packet also contains a `source_seq_num` that indicates the freshness of the route search so that if an intermediate node receives a route request packet containing a smaller `source_seq_num`, it simply discards it. When the desired target node or a node with a route to the destination receives the RREQ packet, it send a *Route Reply* (RREP) packet back along the reverse path. The reverse path is established during the path search phase, by setting up a pointer to the predecessor from which a node had received the first RREQ. Similarly, during the traversal of the RREP packet, each intermediate node points to its predecessor so that the forward path for data transmission can be established. However, in order to ensure that the path is the shortest possible and to ensure route freshness, the RREP packet contains a `destination_seq_num` field that indicates the freshness of the route found and a `hop_count` that indicates that number of hops traversed by the RREP so far, so that an intermediate node only rebroadcasts the RREP if it contains a greater `destination_seq_num` or the same `destination_seq_num` with a smaller `hop_count`. After a path is established, the source node retains the route to the destination in a routing table for a period determined by `route_caching_timeout` beyond which the route

is considered invalid. The advantage of AODV is that topology changes not affecting an active path do not trigger route updates, which reduces route maintenance overhead when compared with common proactive protocols such as DSDV. However, if links fail, a *Route Error* (RERR) message is propagated back to the source so that it can initiate a new route search procedure. AODV has been tested over Bluetooth PANs [6, 62] and it was been shown that due to the limited processing capabilities and the nature topological constraints in these networks, this protocol exhibits limited efficiency.

The Location Aided Routing protocol (LAR) [93] uses node location information to direct route searches and replies. This information can be obtained through the use of GPS. When a node needs to establish a route to a particular destination, if it has information about the destination's previous location at time t_0 and speed, v, it can calculate at time t_1 a circular *expected zone* of radius $v(t_1 - t_0)$, in which the the destination is expected to be located. It also calculates a *request zone* that includes intermediate nodes that the route search must go through in order to reach the destination. LAR defines two methods of calculating the request zone. The first method generates a rectangular region; if the source is outside the expected zone then the request zone is the smallest rectangle that includes the expected zone with the source node being in one of the rectangle's corners. If, on the other hand, both source and destination nodes fall within the expected zone, then the request zone is the smallest square containing the expected zone. In the second method, the request zone is explicitly included in the route request packet in the form of distance to the target such that a node forwards the packet only if it brings it closer to the destination. If the source node has no information about the location of the destination, the request zone would span the entire network and the route request would be achieved via flooding. Once the destination node is reached, it replies with a route reply packet following the reverse path. The main advantage of LAR is that it shortens the time and overhead of the route discovery phase; however, this comes at the expense of relying on location systems, which might not always be available, particularly in personal devices. The work presented in [32] proposes the *Location Aware Routing Protocol* (LARP) that uses location information but is specifically adapted to Bluetooth PANs. In the proposed solution, nodes obtain their location information via a *Bluetooth Location Network* (BLN) [54] and re-arrange their piconet membership so as to minimise the distance between source and destination during the route search phase. This protocol suffers from similar drawbacks to LAR, in that it requires a location network, which may not always be possible.

3.3.3 Hybrid Routing Protocols

Hybrid routing protocols combine both proactive and reactive elements in order to optimise routing overhead and delay. Most of the proposed hybrid routing solutions are either zone-based or tree-based [3]. Zone-based schemes define zones in which routes are set up and maintained proactively, while routes to nodes outside the zones are set up on-demand. The difference between the various zone-based routing

protocols is the way in which zones are created and maintained. Tree-based protocols, on the other hand, arrange nodes in a number of sub-trees in which proactive routing takes place. Routing between sub-trees, however, uses a reactive approach. Representatives of zone-based routing protocols for general ad hoc networks are the Zone Routing Protocol (ZRP), Zone-based Hierarchical Link State (ZHLS) and Scalable Location Update-based Routing Protocol (SLURP). Examples of Tree-based hybrid protocols include the Distributed Spanning Tree (DST) routing protocol and the Distributed Dynamic Routing (DDR) protocol. Each of these is explained briefly in this section.

In ZRP [59], every source node defines a zone inside which routes are maintained proactively and any route to a node outside the zone is established reactively. Each zone has a diameter, ρ, expressed in terms of the number of hops, which is chosen in such a way as to optimise the trade off between overhead and route set-up latency. The nodes which lie strictly inside the zone are referred to as *interior* nodes, whereas, nodes that are exactly ρ hops away from the source are called *peripheral* nodes. ZRP does not specify which proactive and reactive protocols to use, but rather devises mechanisms to construct and maintain the zones. When a source node wishes to send data to a particular destination, it first checks if the route exists in its routing table, *i.e.,* if the route can be determined proactively using the *IntrA-zone Routing Protocol* (IARP). If the route is not found, it means that the destination lies outside the zone, thus the source instructs peripheral nodes to initiate a route search in accordance with the *IntEr-zone Routing Protocol* (IERP); this is referred to as *Bordercasting*. The advantage of ZRP is that it finds a trade-off between route discovery latency and routing overhead.

A routing protocol that builds on the idea of ZRP is proposed for PANs in [86]. The authors describe a scheme where each node maintains a proactive zone of diameter *MAXHOPS*, beyond which AODV is used to set up routes on-demand. This protocol is compatible with Bluetooth and does not require any modification to the specifications. The proactive part of the protocol stores routing information as linked lists. Each master maintains a table that lists the slaves and gateways it connects to. The gateway entries, in turn, contain lists of masters that the gateway themselves connect to. Similarly, each gateway itself, maintains a list of masters that it connects to, which, in turn, lists the gateways that the masters connect to and so forth. This structure allows nodes to know which nodes are included in the proactive zone. The size of the proactive zone's diameter is an important performance parameter for ZRP as too small a zone would render the protocol almost purely reactive and a large zone would make the protocol behave almost as a purely proactive protocol.

ZHLS [74] is another zone-based hybrid routing protocol, however, unlike ZRP, it is based on hierarchical routing, wherein nodes are arranged into non-overlapping zones, using nodes' knowledge about their location via GPS. Each zone is identified by a Zone ID and each node holds a node ID, such that a <Node ID, Zone ID> pair identifies a unique node within a particular location. If both the source and destination lie within the same zone then routing is done proactively. This means that each

node knows the members of its own zone. If the target node is outside the zone, the source initiates a location search to find the zone ID of the destination. This means that nodes must also have a high level knowledge of the topology zone map. ZHLS does not rely on any cluster head for inter-zone routing, instead it relies on a peer-to-peer approach to route packets between the different zones. The advantage of this protocol is that it reduces the amount of overhead associated with route search since it only requires nodes to find the zone to which the destination belongs rather than flooding the entire network looking for a specific address. The drawback, however, is that it requires GPS to obtain location information in order to map the location to a particular zone. Furthermore, the zone map must be preloaded into every node, which could be problematic in highly dynamic networks.

Woo *et. al.* [170] propose the Scalable Location Update-based Routing Protocol (SLURP), which, like ZHLS, maps nodes into non-overlapping regions. Regions are generated from device addresses using the mapping $f(NodeID) \rightarrow RegionID$. Routing within a region is performed proactively, whereas inter-region routing is done reactively. The novelty of SLURP is that it eliminates global route discovery. This is achieved by defining a home region for each node, which is obtained from the aforementioned mapping. If a node moves it updates the nodes within its region with its new location. When a node needs to establish a route to another node that lies outside its zone, it calculates the destination's home region from the destination address and then sends a query to that home region to obtain its location. Once the location is found, packets can be forwarded using the *Most Forward with fixed Radius* (MFR) routing protocol, which selects next hops according to their proximity to the target node. The disadvantage of SLURP is that it relies on location systems (GPS) as well as pre-programmed zone maps. In addition, in highly dynamic environments, location updates may generate large overhead.

DST [139] is a tree-based hybrid routing protocol that arranges nodes into trees, which are joined together via bridge nodes. The idea behind DST is that connectivity cannot be guaranteed to be stable at all times, therefore the authors propose two routing methods: *Hybrid Tree Flooding* (HTF) and *Distributed Spanning Tree Shuttling* (DST). In HTF, a source node sends the packet along all tree edges using flooding, and each node receiving the packet stores it for a certain period called the *Holding Time*. The reason behind the Holding Time is to allow nodes to send the packet to the destination if the connectivity increases, without having to regenerate the packet at the source. In the second approach, *i.e.,* DST, a packet is sent along a tree branch then when it reaches a leaf node it is sent back up the tree until it reaches a height referred to as the *Shuttling Level*, from which the packet follows a different branch. Naturally, Shuttling introduces far less overhead than flooding, however, it incurs more delays in determining routes. When a source node needs to send data to another node to which it does not have a route, it first tries to route via HTF; if it fails after the Holding Time expires, packets are sent via Shuttling. The advantage of this protocol is that it is highly adaptable to topological changes, however, maintaining trees introduces large overhead.

Distributed Dynamic Routing (DDR) [130] is another tree-based hybrid routing protocol, which is compatible with the topological characteristics of PANs. DDR arranges nodes in non-overlapping zones, each of which is a tree. Trees are constructed in a distributed fashion and a collection of trees forms a forest. Trees are interconnected via gateway nodes, *i.e.,* nodes that are within transmission range of each other but belong to different zones. Each tree is identified by its zone ID (ZID) which is obtained from applying a function to the tree members' identities. This means that DDR does not rely on location information of nodes, however, it does require a view of the topology, which is obtained by exchanging beacon messages among neighbours and storing the information in tables. In addition to these tables, each node maintains an inter-zone table that lists the gateway nodes linking a particular tree together with a zone stability factor (Z_Stability) that indicates the Euclidean distance between the IDs of adjacent zones. If the source/destination pair belong to the same zone, then DDR uses the tables built during the tree construction phase. However, if the destination is in another zone, DDR employs a reactive routing protocol to find the route.

3.4 Summary

This chapter explained the issues linked to personal area networking, covering network formation, error control and transmission scheduling. It also covered some of the predominant routing protocols for PANs. The reviewed literature helped to identify areas that need improvement and rectification in order to comply with the strict requirements of personal area networks. Moreover, some routing protocols presented herein are either computationally complex or not optimised to suit the constraints of PANs. The aim of this project is to design PAN-compliant protocols which improve the performance of PANs while remaining simple and energy-efficient.

Chapter 4

Packet Error Correction (PEC)

4.1 Introduction

This chapter presents a novel adaptive error control scheme that improves on data delivery while introducing low overhead. The idea of the proposed *Packet Error Correction* (PEC) technique is to arrange packets into bins, where bins are buffer spaces, in which packets are stored for a predetermined period of time, and to retransmit only a subset of these packets if necessary. Naturally, in channels characterised by high *Bit Error Rates* (BER), not every packet maybe retransmitted; therefore PEC is best suited for multimedia applications, wherein the loss of few packets are not of major concern (as far as user perception is concerned). Nonetheless, PEC also offers the possibility to adjust the number of re-transmissible packets per buffer (bin), so that more transmission opportunities are allowed if required by a particular application.

Error control over wireless links is necessary because these channels are characterised by high BER. These errors are introduced due to various factors including interference and signal fading. In Bluetooth-based PANs in particular, noise results from mutual interference between adjacent piconets [65] and interference caused by frequency hopping dependencies [136, 124]. Channels' BER has a direct relationship to the *Packet Error Rate*, which is the ratio of packets received in error to the total number of transmitted packets within a given time period. The authors of [171] show that the relationship between PER and *Signal to Noise Ratio* (SNR) follows an exponentially decaying curve; meaning that a slight increase in noise greatly influences the PER. The difference between BER and PER is that the former is a measure of the channel's error characteristics whereas the latter gives an indication of the performance of a particular system on packet delivery rate. The two, however, are tightly linked to one another, in that an increase in BER results in an increase in PER and vice-versa. In this chapter a clear distinction between the two measures is made, and selecting the appropriate number of re-transmissible packets relies solely on BER estimation.

This chapter is only concerned with ACL packets, as SCO packets cannot usually be retransmitted[1].

[1] Later versions of Bluetooth define eSCO, which can be retransmitted under special conditions [22]

Each ACL packet carries a CRC trail that is used to check for the presence of errors, thereby allowing a node to request the retransmission of individual packets if the receiver is not able to recover from the errors incurred during transmission.

The proposed PEC scheme is assessed by simulations and mathematical analysis. Although it was simulated on Bluetooth, PEC can be applied to any wireless system that is able to request individual packets to be resent. Moreover, a general formula is derived, which can be applied to any other system by simply altering the parameters.

PEC requires nodes to have an estimate of channels' BER, which are then mapped to a link quality measure; this is outlined in section 4.3. Details of the PEC algorithm and the analysis of its operations are covered in section 4.4. The scheme is then evaluated by simulation in section 4.5, and section 4.6 summarises the findings.

4.2 System Overview

Bluetooth MAC normally relies on ARQ techniques to retransmit every corrupted packet until a timeout, T_O, is reached, which by default is set to infinity, *i.e.,* a packet is never dropped [34]. This method, introduces large delays especially in very noisy channels where the probability of error is high. In the solution proposed herein, a receiver buffers σ packets in its memory and asks the sender to retransmit ρ packets such that $\rho < \sigma$. This scenario is shown in Figure 4.1.

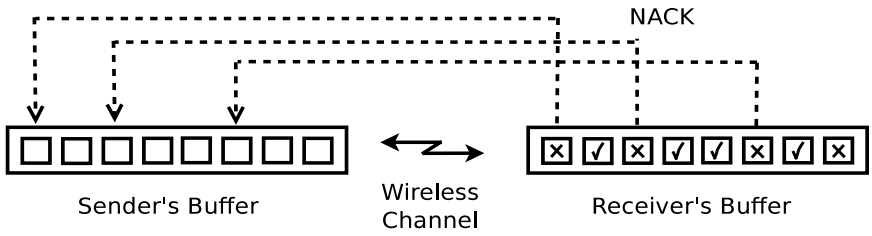


Figure 4.1: Packet Error Correction operation

In PEC, the number of re-transmissible packets, ρ, is adjusted according to channel conditions, *i.e.,* ρ is decreased as channel conditions improve and increased otherwise. However, ρ is always strictly lower than σ. The scheme relies on *negative acknowledgement automatic repeat request* (NACK-ARQ), so that the receiver can request individual packets to be retransmitted if necessary.

A fundamental difference between PEC and other error control schemes proposed in the literature [34, 113, 114], is that instead of adapting the retransmission timeout (T_O) to channel conditions for every packet[2], PEC varies the number of allowed retransmissions in a given set, such that a packet may be resent between 0 and ρ times, depending on the number of packets in error in the set σ. If the number

[2] T_O can either be a time window dedicated solely for retransmissions following the transmission of each packet, or can represent a limit in time beyond which a packet is dropped

of packets in error is lower than the number of allowed retransmissions, then each erroneous packet is given at least one retransmission opportunity. If the number of erroneous packets in a buffer is higher than the number of allowed retransmissions, then the packets to be given retransmission opportunities are chosen randomly, such that no packet can be resent more than once (for further details regarding the retransmission mechanism refer to Section 4.4.2)

4.3 Channel Monitoring

For the PEC scheme to work, the sender and the receiver must have an estimate of the wireless link quality. Devices, by default, can compute the *Received Signal Strength Indicator* (RSSI), which gives some indication of the quality of the link [64]. Relying on this measure, however, has proven to be insufficient [164]. Therefore, a more reliable way of obtaining link quality is to use *Bit Error Rate* (BER). In order to comply with the low complexity requirements, personal devices estimate link quality (in terms of BER) by simply monitoring the received bits over a period of τ and estimating the BER as

$$BER_\tau = \frac{rcv\_err}{rcv\_total} \tag{4.1}$$

where rcv_err represents the number of bits in error and rcv_total is the total count of received bits over the period τ. This estimate is only possible if the receiver employs forward error correction on blocks of data, or checks the received stream against a known training sequence. The latter is a simpler method, traditionally performed off-line, *i.e.,* in the absence of any data transmission. However, for accurate estimates, the sequence may be impractically long. In order to overcome this drawback, the channel access code can be used as the training sequence. The advantage of using the access code is that it is known to every node in the piconet and is included with every packet, thus a receiver can estimate the BER on-line.

The choice of the lag τ is application-dependent; nonetheless, considering that typical BER values in the ISM band are around 10^{-3}, and that for most applications a BER beyond 10^{-5} is insignificant [121], a node is required to monitor 100kb of access code data. Thus, τ corresponds to $14,286$ time slots. However, because of Bluetooth scheduling mechanisms, wherein nodes may not receive the required amount of data, the PEC algorithm sets a timeout, T_{out}, after which it generates a BER as an average over p previous readings to be the current estimate until the next τ has elapsed, as given by Equation 4.2.

$$BER_\tau = \frac{1}{p} \sum_{i=t}^{i=t+p} BER_i \tag{4.2}$$

Where t is the time instant of the estimate. BER_τ represents the expected value of the channel's BER for the time lag τ, and is only generated if a node fails to receive enough data, before T_{out} has elapsed, to

be able to estimate the BER. In PEC T_{out} is assigned a value of 30 seconds. This choice stems from the fact that the time taken to receive sufficient access code data is close to 20 seconds (14,286 time slots), to which an extra 50% margin is added, thus resulting in a value of 30 seconds.

4.3.1 Link Quality

When a node obtains the link's BER, it generates a *Link Quality* (LQ) metric. LQ is an 8-bit mapping of the BER and is chipset-specific [64]. This means that two communicating devices may compute different LQs and hence interpret the BER differently. In this project, another LQ mapping is imposed on every module for the purpose of error control and requires only 3 bits.

As stated above, in most applications, bit error rates lower than 10^{-5} are negligible, especially if forward error correction is used. Furthermore, Bluetooth specifications consider a link with BER above 10^{-1} to be unworkable, *i.e.,* the link is considered non-existent. Therefore, there is a region, r, such that $10^{-5} < r < 10^{-1}$, over which fluctuations in BER are more significant. This means that lower and upper BER values should correspond to coarser LQ mapping, which in turn means that the function f, such that $LQ = f(BER)$, is non linear. A non linear mapping that satisfies such a relationship is an s-curve as given by equation 4.3.

$$LQ = \frac{8}{1 + e^{-100(2 \times ber - 0.02)}} - 1 \tag{4.3}$$

Equation 4.3 indicates that the maximum output of the mapping function is 7, which means that a 3-bit quantisation on integer outputs suffices to represent 8 different LQ levels. Figure 4.2 shows the output of the quantiser [3] for $BER \in (10^{-4} - 2 \times 10^{-1}]$.

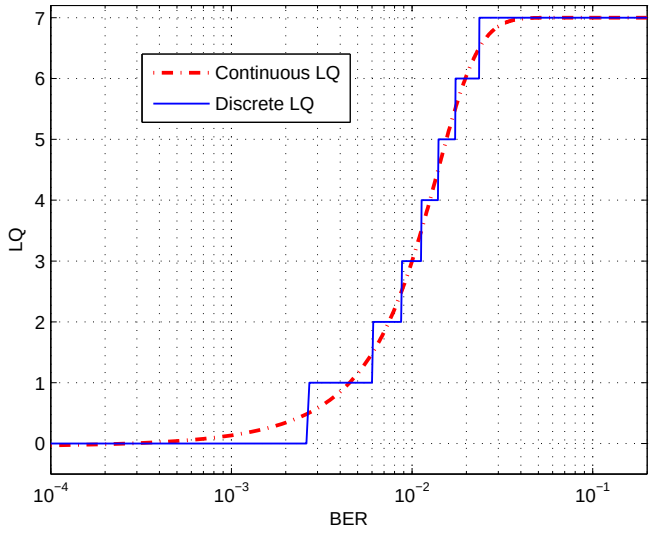


Figure 4.2: BER to LQ Mapping

[3] The quantiser is software-based, therefore does not require any additional circuitry

4.4 Packet Error Correction (PEC) Scheme

The aim of the adaptive retransmission scheme is to retransmit only a subset of the buffered data when individual packets are acknowledged to have been received incorrectly. The number of allowed retransmissions is governed by the LQ value, which translates the quality of the link in terms of BER into 8 distinct levels, where a low LQ value means low BER and vice versa. The number of allowed retransmissions, ρ, per buffer size, σ, increases linearly with LQ, which means that as BER conditions improve, there is a need for fewer retransmissions. Before calculating the required ρ, a node first computes $\widehat{lq}$, as given by equation 4.4, which is the reverse of LQ.

$$\widehat{lq} = |LQ - LQ_{max}| \tag{4.4}$$

In equation 4.4, LQ_{max} is the highest LQ level, which in this case is 7. The number of allowed retransmissions is then obtained as given by equation 4.5

$$\rho = \lfloor (\sigma - \beta)\,(\widehat{lq} + 1)^{-1} \rfloor \tag{4.5}$$

Where $\beta \in \mathbb{N}^{+}$ is the number of additional packets that are non-re-transmissible to ensure that the condition $\rho < \sigma$ is met. β is selected by each node independently, although, it is tightly linked to the node's error reduction target. The reason for taking the floor of the product term in equation 4.5 is to ensure that the number of retransmissions allowed is always a whole number, while ensuring that the number of unnecessary retransmissions is reduced. Equation 4.5 indicates that the number of retransmissions allowed is a fraction of the buffer size and depends on the measure LQ, which in turn reflects the channel's estimated BER. Higher BERs are mapped into higher LQ values, which result in lower $\widehat{lq}$ (since $\widehat{lq}$ is the reverse of LQ), thus increasing the number of re-transmissible packets. Similarly, lower BERs correspond to lower LQ, and consequently higher $\widehat{lq}$, thus resulting in fewer possible retransmissions. Algorithm 1 describes the operations of the adaptive retransmission scheme.

This algorithm applies to every master/slave pair, wherein, a node (either the master or the slave), generates an estimate of the channel's BER by monitoring the received access codes over a period τ. A node first reads the packet (pkt) from its receiver queue and obtains its length (ℓ) in terms of time slots. It also extracts the access code ($access\_code$) from the packet and stores it in the rcv_total variable. From the extracted access code, the node records the number of errors by running the $ExtractErr()$ method, and increments the count, err_total, if errors are found. Here error refers to a packet being in error, *i.e.,* if the packet is corrupted. After τ time slots have elapsed, the node computes an estimate of the BER (ber) and maps it to a link quality measure (lq), from which the number of retransmissions, ρ is calculated through method $Set()$, according to equations 4.4 and 4.5. If, however, a node does not

Algorithm 1 PEC Adaptive Error Control

$\tau \leftarrow 14,286$
while $\tau > 0$ **and** $T_{out} < 30s$ **do**
  $pkt \leftarrow GetPacket(input\_queue)$
  $\ell \leftarrow GetLength(pkt)$
  $access\_code \leftarrow Extract(pkt)$
  $rcv\_total \leftarrow rcv\_total + access\_code$
  $err \leftarrow ExtractErr(access\_code)$
  $err\_total := err\_total + err$
  $\tau := \tau - \ell$
  $T_{out} + +$
end while
$ber \leftarrow Estimate(rcv\_total, err)$
$lq \leftarrow Map(BER)$
$\rho \leftarrow Set(LQ)$

receive enough data to compute the BER after T_{out}, the BER is estimated according to equation 4.2.

4.4.1 Buffer Management

The adaptive retransmission scheme described above, suggests that both the sender and the receiver must agree on the buffer size, σ and the duration for which it is valid, T_σ. On expiration of T_σ, the receiver sends an `LMP_buff_request` to the sender's link manager, requesting the size of the buffer that it wishes to use and the time duration for which the requested buffer size is required. If the request is accepted, the sender replies with an `LMP_accepted` PDU. If the request is not granted, due to capacity limitations at the sender, the sender reduces the size of the requested buffer by a factor of ϕ and resubmits the request. This process continues following LMP protocol operations, until either a buffer size is agreed upon, or, if not, both sender and receiver use the default values for σ and T_σ, which are set to DH5 and 5 time-slots respectively. Naturally, there is a correlation between the size of a buffer and its duration, since ACL packets are defined by their time span. Therefore each `LMP_buff_request` must satisfy the condition $T_\sigma \geq \sigma$. `LMP_buff_request` uses one of the reserved LMP OpCodes defined in the Bluetooth specifications [23]. In this project it is given the value 0x6F and its format is illustrated in Figure 4.3.

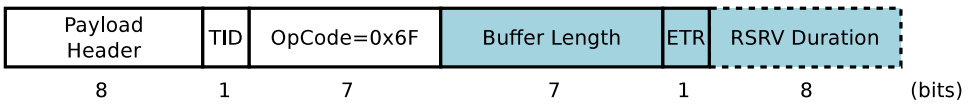


Figure 4.3: LMP_buffer_request Packet Format

The `LMP_buff_request` PDU is of variable size depending on whether the time required for the new buffer size has changed or not. It follows the general LMP format, *i.e.,* the OpCode identifying the LMP type, followed by its Parameters. In Figure 4.3, the shaded area represents the message's parameters; these are described briefly below.

Buffer Length: This field indicates the requested buffer length in time-slots. A node can request a buffer

size of up to 127 time-slots, which can hold up to 25 5-slot packets.

ETR: The Extension Time Request flag indicates whether a new T_σ is required for the new buffer size. If ETR is set to 0 it indicates that T_σ should remain the same as the previous one.

RSRV Duration: Indicates the requested duration (in time-slots) to be reserved for the buffer, so that the sender allocates memory space for the requested duration.

The time lag T_σ and buffer size, σ, depend on nodes' buffer space capacity, jitter conditions and application. For instance, in real time applications, nodes may need to buffer up a few more packets before processing in order to compensate for the jitter incurred during transmission. In this case the receiver selects a new value of T_σ and a new buffer size and sends an `LMP_buff_request` to the sender. Furthermore, nodes also store the previous buffer, in addition to the current one, in order to cater for out-of-sync data reception, so that a node can request the retransmission of a particular packet from the previous buffer if necessary. This means that after expiration of T_σ, the current buffer is stored for a further T_b seconds. The value of T_b may be negotiated through LMP or may be specified by the application.

4.4.2 Retransmission Mechanism

As illustrated in Figure 4.1, a receiver running the PEC scheme, buffers σ packets and sends a NACK to the sender requesting a particular packet to be retransmitted if it is corrupted. The number of possible retransmissions, ρ, is spread uniformly amongst all σ packets; this means that each erroneous packet is given, when possible, an equal retransmission opportunity. Evidently, if the number of packets in error, e, exceeds the number of allowed retransmissions, then the packets to be resent are chosen randomly. Similarly, if ρ cannot be divided equally among the packets in error (*e.g.* $e = 3$ and $\rho = 5$), then the packets to be given extra retransmission opportunities are also selected randomly from amongst the set of erroneous packets. PEC's retransmission operations are summarised in Algorithm 2

A receiver reads each packet (pkt) from its buffer ($InBuffer$) and checks if it is in error through the method $CheckPacket()$. If it is, the packet is marked via the method $MarkPkt()$ and stored in the array $ErrSet$ which holds a list of erroneous packets. The algorithm distinguishes between two cases:

- Case 1 - There are fewer packets in error in the set than the number of possible retransmissions: In this case, the receiver requests each packet to be retransmitted by sending a NACK back to the sender ($Request()$ in Algorithm 2).

- Case 2 - The number of erroneous packets is greater than the number of allowed retransmissions: In this case, the receiver selects a subset of $ErrSet$ randomly and stores it in $RetrSet$ via the method $Select()$ such that the number of packets in $RetrSet$ is exactly ρ. The receiver then requests each packet in this subset to be resent.

Algorithm 2 PEC Retransmission - Receiver

while $\rho > 0 \,||\, e \neq 0$ **do**
  $e \leftarrow Reset(e)$
  $j \leftarrow Reset(j)$
  for all $pkt \in InBuffer$ **do**
    $e \leftarrow CheckPacket(pkt)$
    if e is **TRUE then**
      $ErrSet[j] \leftarrow MarkPkt(pkt)$
      $j := j + 1$
      $e := e + 1$
    end if
  end for
  if $e < \rho$ **then**
    for all $pkt' \in ErrSet$ **do**
      $pkt' \leftarrow Request(pkt')$
      $\rho := \rho - 1$
    end for
  else
    $RetrSet \leftarrow Select(ErrSet)$
    for all $\overline{pkt} \in RetrSet$ **do**
      $\overline{pkt} \leftarrow Request(\overline{pkt})$
      $\rho := \rho - 1$
    end for
  end if
end while

This process is repeated until either ρ is exhausted, or there are no more errors present in the buffer after retransmissions. Because of the particular setting of the PEC scheme, NACKs can be sent simultaneously in a single LMP packet, specifying the buffer and sequence numbers of the packets to be resent. The OpCode used for the NACK packet bears the value 0x43 since it is unused in the specifications. This method reduces the overhead even further as individual NACKs are avoided. The format of the NACK packet is shown in Figure 4.4.

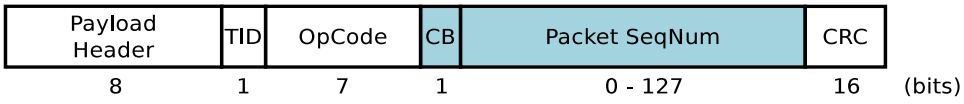


Figure 4.4: NACK Packet Format

The structure of the NACK indicates that it can be sent in a single-slot packet; more precisely as a DM1 packet, where the 20 Bytes of payload are 2/3 FEC encoded to make up the required 30 Byte length. The format of the NACK is described below:

CB: The Current Buffer (CB) flag indicates the targeted buffer. A value of 1 indicates the current buffer, whereas a 0 means the previous one.

Packet SeqNum: This field holds a sequence of 1s and 0s, wherein a 1 indicates a retransmission request for the packet whose sequence number corresponds to its position in the Packet SeqNum sequence. If `Packet SeqNum` is empty, it indicates that every packet was received correctly.

Because channels are characterised by high bit error rates, a NACK might be corrupted or lost during transmission and therefore the sender would not know which packets to retransmit. If the sender does not receive the NACK when it is supposed to, *i.e.,* after sending the last packet in the buffer, it continues sending subsequent data. The receiver realises that the NACK has not been received and sends another one requesting the missing packets. The advantage of the adaptive PEC scheme over incremental ARQ windowing, such as the method proposed in [34], is that it adapts to sudden changes in channel conditions. Moreover, the proposed method retransmits only a portion of the packets sent, thereby reducing error control overhead and delay. Although PEC may seem counter-intuitive, since fewer packets are allowed to be resent, in reality, it presents better correction characteristics than adaptive ARQ schemes (see section 4.5).

4.4.3 Analysis of the PEC Scheme

PEC relies on retransmitting packets that get corrupted when passing through a channel with a particular BER (or probability of error, p). It is assumed that errors are Poisson distributed in time, therefore the probability of error is given by:

$$P_e = \frac{e^{-\lambda_\tau}(\lambda_\tau)^k}{k!} \tag{4.6}$$

For a finite set of size σ, the probability of having e errors, given a channel error probability, p, is defined by the binomial expression instead, as given by Equation 4.7

$$E_\sigma[e] = \binom{\sigma}{e} p^e (1-p)^{\sigma-e} \tag{4.7}$$

Where $E_\sigma[e]$ denotes the probability of e errors being present in a set of size σ and $\binom{\sigma}{e} = \frac{\sigma!}{e!(\sigma-e)}$. Here the probability of error is expressed as *Packet Error Rate* (PER), hence an error, e, means a packet being in error.

Proposition 4.4.1. *The probability of e errors remaining in a set of size σ after ρ retries is expressed as:*

$$P_\sigma[e][\rho] = \begin{cases} eE_\sigma[e]p^{\rho-e+1}, & \text{if } \rho \geq e \\ E_\sigma[e](\rho.p + e - \rho), & \text{if } \rho < e \end{cases} \tag{4.8}$$

Proof. **Case 1:** $\rho \geq e$

After retransmitting e packets once, the probability of remaining errors in the set σ is:

$$E_\sigma[e] \times \left(p. \sum_1^n e_n \right) \tag{4.9}$$

Where $e_n \in e$ (the set of erroneous packets); thus $\sum_1^n e_n = e$, and therefore the probability of packets

remaining in error is $e.E_\sigma[e].p$

Also, since $\rho \geq e$, after e retries, the remaining number of retransmissions is $\rho - e$, which means that the probability of remaining errors is reduced by a further factor of $p^{\rho-e}$, so:

$$P_\sigma[e][\rho] = e.E_\sigma[e].p.p^{\rho-e} \Leftrightarrow P_\sigma[e][\rho] = e.E_\sigma[e].p^{\rho-e+1} \tag{4.10}$$

Case 2: $\rho < e$

The probability of the retransmitted packets to remain in error after ρ retries are exhausted is:

$$\bar{P_\sigma}[e][\rho] = E_\sigma[e].p.\rho \tag{4.11}$$

However, since the number of errors exceeds the number of retries, $e-\rho$ packets will not be retransmitted. This means that the error probability of these packets after retransmission is the same as the initial probability, *i.e.:*

$$P_{\hat{\sigma}} = (e - \rho)E_\sigma[e] \tag{4.12}$$

The total error probability when $\rho < e$ is thus the sum of the probability that errors remain in the set σ after ρ retries, and the probability that the non-transmissible packets are initially in error such that:

$$P_\sigma[e][\rho] = \bar{P_\sigma}[e][\rho] + P_{\hat{\sigma}} \Leftrightarrow P_\sigma[e][\rho] = E_\sigma[e](\rho.p + e - \rho) \tag{4.13}$$

□

The formula given in 4.8 allows an analytical comparison between a channel's initial error probability and the resulting probability of errors after retransmission, for various buffer sizes and number of retries, thus providing a performance evaluation of the error control scheme prior to implementation. PEC presents interesting probabilistic characteristics, in that the possibility of having more than e errors in a set of size σ is unlikely even at high BER; therefore after ρ retries, the probability of remaining errors is very slim. Moreover, adapting the number of retries per set to channel quality, ensures that the overhead and delay associated with error correction are reduced, while ensuring that PEC's error reduction properties are maintained. An evaluation, via simulation, of the proposed scheme is given in the next section.

4.5 Evaluation

The proposed PEC scheme is simulated under various parameters and channel conditions in order to assess its performance. PEC is also compared to the specification-recommended retransmission scheme. The simulator is written in Matlab, wherein nodes are objects, with associated attributes and methods,

and channels are filters with additive error terms.

4.5.1 Simulation Settings

PEC is evaluated on audio over ACL applications [8, 85], wherein, unlike SCO, packets which are not received correctly may be retransmitted. The audio data source can either be real time recording or a stored file. For the purpose of the simulation, the source is a WAV file, which is PCM-encoded to make up the raw audio data. The transmitter applies the necessary encapsulation and the data is sent in single slot packets (DM1). The packets go through a channel that introduces errors, such that error patterns are Poisson distributed. Upon receiving a packet, the receiver checks for errors, records the packet's sequence number, copies the access code for BER estimation and stores the packet in the buffer. Every σ packets, the receiver sends a NACK to the sender requesting the retransmission of erroneous packets as set out by ρ. The data is then de-encapsulated, decoded and processed for output. The operations of the simulator are summarised in the block diagram of Figure 4.5

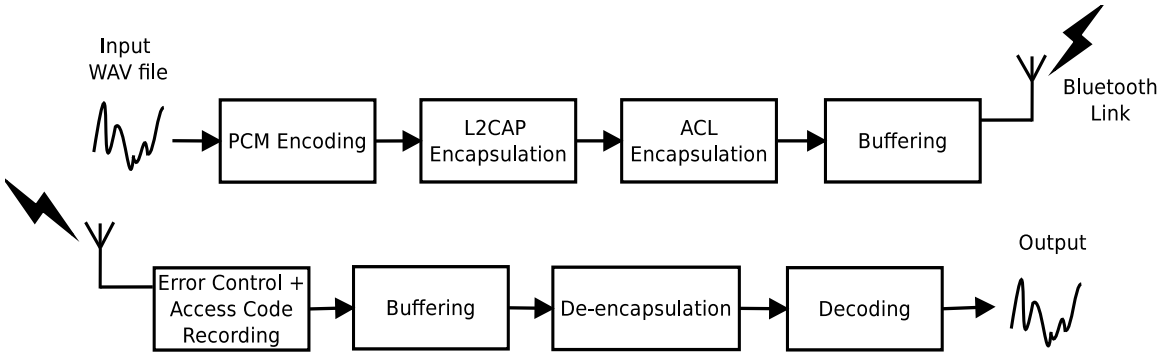


Figure 4.5: PEC Simulator

In the simulator, errors are obtained by generating an array of natural numbers of length L (where L = channel error probability $\times$ total number of bits)using the Poisson formula as given by Equation 4.6. Each entry in this array represents the distance between error-free bits. These L numbers are used as pointers to indicate the position at which an error is forced, thus emulating the effect of noisy channels. In the simulation, the sender and receiver queues are assumed to be infinite, therefore the simulator does not handle buffer overflows. One assumes that this functionality is carried out by upper layer network protocols such as TCP.

4.5.2 Metrics

The assessment of the proposed error correction scheme is based on the following performance metrics:

- **Packet Delivery Rate**: This metric indicates the resulting data error rate after retransmission of erroneous packets, in accordance with σ and ρ; that is the ratio of the number of packets that remain in error to the total number of packets sent, after applying the retransmission mechanism.

it indicates the extent to which the proposed scheme improves on data loss rates, thus improving transmission reliability. Furthermore, the performance of PEC, in terms of packet delivery rates, is closely linked to the ability of a node to accurately estimate channel conditions, as the number of allowed retransmissions depends on the channel's BER. Therefore, an evaluation of the estimator's behaviour is also given under this heading.

- **Data Overhead**: The overhead indicates the extra amount of data introduced by the retransmission scheme. This includes the actual resent packets as well as the associated control data, *i.e.,* NACKs. The overhead is defined as the ratio of the extra data to the overall data size such that $\Omega = \frac{\Psi}{data\_size}$, where Ψ indicates the redundancy introduced by the retransmission mechanism and $data\_size$ is the size of the transferred data. Note that Ψ does not include FEC and CRC redundancies, as the aim of the measure Ω is to provide an indication of the "extra" overhead introduced by the proposed retransmission method.

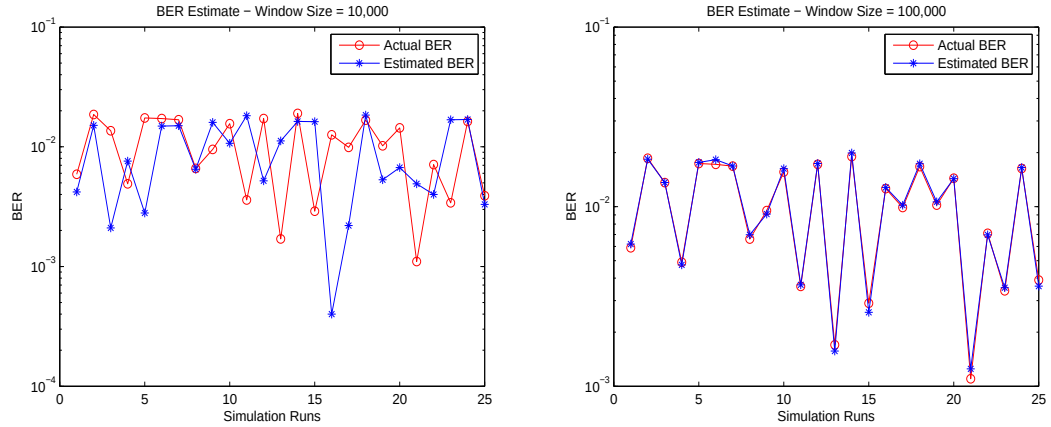
- **Audio Quality**: Although PEC is not restricted solely to audio, the quality of the received audio data gives an indication of the scheme's performance. The difficulty with using audio quality as a performance metric, nonetheless, is that quality is subjective to human perception, therefore it is unclear what would be deemed as "acceptable". However, it is possible to show deviations from error-free quality before and after retransmissions, which provide a visual means of the performance of PEC.

- **Delay**: Buffering and retransmitting packets introduce delays in processing the data. This is more of an issue in real-time applications, where ordered and time-tight processing is required. In some instances, large delays are unacceptable even if it means that the data can be recovered. Assessing the delay introduced by PEC is therefore essential in evaluating the performance of the scheme, as it indicates it suitability for given applications.

4.5.3 Packet Delivery Rate

PEC is simulated for various buffer sizes and BER values while varying the number of non-retransmissible packets per buffer, in order to assess the impact of these parameters on the performance of the scheme. PEC adapts the number of retransmissions per buffer to the quality of the link (in terms of BER), therefore an accurate estimate of error rate is essential. Figures 4.6(a) and 4.6(b) show the estimate of the BER obtained from the access code, compared with actual BER enforced by the simulator for an estimation window of $10,000$ and $100,000$ bits respectively.

Figure 4.6 suggests that an estimate of the channel BER is possible online, using the access code, without the need for a training sequence. The drawback of this method, nonetheless, is that access codes are small in size (72 bits); therefore, before a node can compute an estimate of the BER, it first

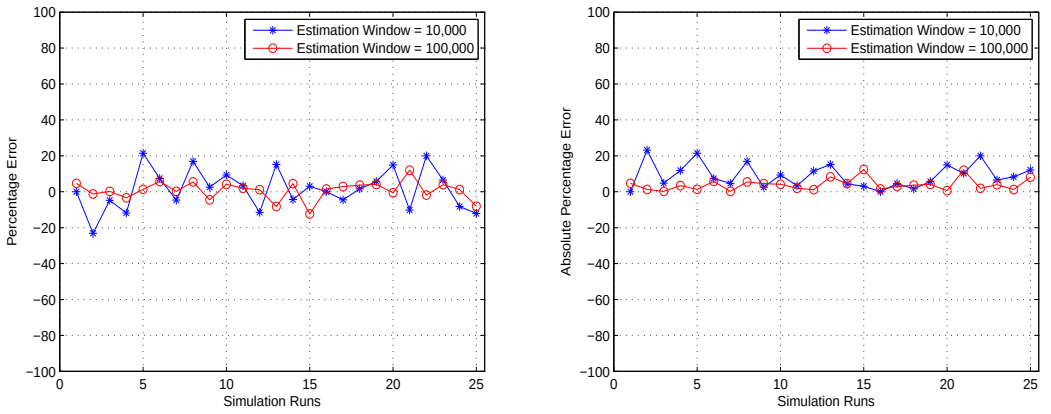


(a) BER Estimate - Window Size = 10,000

(b) BER Estimate - Window Size = 100,000

Figure 4.6: PEC's Estimate of Channel BER

needs to gather sufficient data, which may incur delays. This issue is rectified by considering the current estimate to be the average of the previous readings in case a node fails to receive sufficient data within a predefined timeout. Another way of assessing the performance of the estimator is to look at error performance metrics. The measures used in this chapter are the *Percentage Error* (PE) and the *Absolute Percentage Error* (APE). The former represents the deviation (as a percentage) of the estimate from the actual BER, whereas the latter takes the absolute value of this deviation, such that $pe = 100 \times \frac{\epsilon}{ber}$ and $ape = |pe|$, where $\epsilon = estimated\_ber - actual\_ber$. The error graphs corresponding to the estimates given in Figure 4.6 are shown in Figure 4.7.



(a) Percentage Error

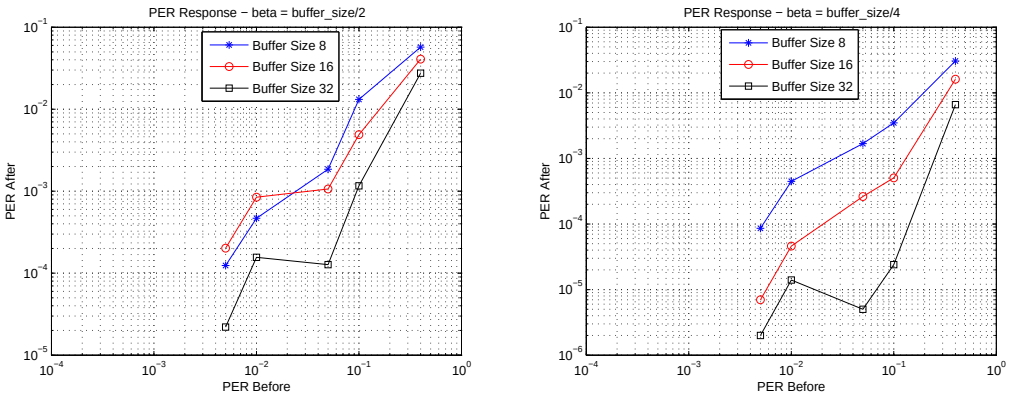
(b) Absolute Percentage Error

Figure 4.7: BER Estimator's Error Performance

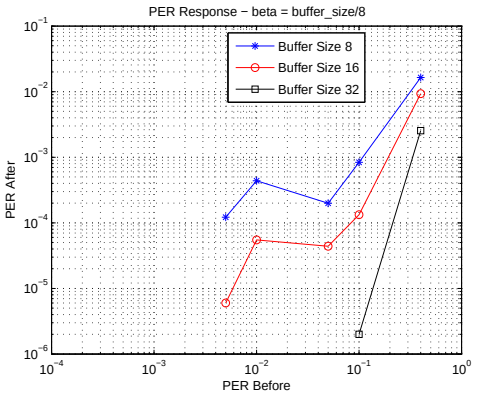
It can be seen from Figure 4.7 that the accuracy of the BER estimate increases with estimation window size. This means that for very precise estimates a large window size is required. A larger

window size, however, introduces the issue of estimation delays, thus changes in channel conditions between consecutive estimate instants could be missed. Therefore, it is important for the estimator to be application-aware, so that the estimation window is adjusted accordingly.

The results given in Figure 4.8 show the performance of the PEC scheme in terms of error rate response for various buffer sizes, σ, and number of non-re-transmissible packets per buffer, β.



(a) $\beta = \frac{\sigma}{2}$ (b) $\beta = \frac{\sigma}{4}$



(c) $\beta = \frac{\sigma}{8}$

Figure 4.8: Packet Delivery Rate

The graphs shown in Figure 4.8 indicate that applying the PEC retransmission method improves the delivery rates by several orders of magnitude, which proves the adequacy of the scheme. Moreover, it can be seen, by comparing Figures 4.8(a), 4.8(b) and 4.8(c) that the response is improved by reducing β. Lower values of β translate into more packets being retransmitted within the set, thereby improving the performance. Furthermore, the relationship between the data error rate before and after retransmission is non-linear, owing to the non-linear mapping of the BER to the number of retransmissions.

Another observation is that the resulting error rate after retransmission is inversely proportional to buffer size. This is due to the fact that the number of retransmissions is also proportional to the size of the buffer; thus increasing σ would increase ρ, thereby more packets can be retransmitted, which results

in a better response. Increasing the size of the buffer, however, comes at the cost of increased delays (as outlined in 4.14), which in some applications can be intolerably long. This last observation is true for any retransmission scheme, wherein increasing the number of retransmissions increases delay. The advantage of PEC over other error control schemes, such as the one defined in the Bluetooth specifications [24], however, is that even when restricting the number of retransmissions to a subset of the packets, the resulting error rates are satisfactorily low. Bluetooth recommends retransmitting every packet that is in error, which can be thought of as PEC where $\sigma = 1$ and $\rho = 1$. Figure 4.9 shows a performance comparison between PEC and the retransmission scheme defined by the specifications.

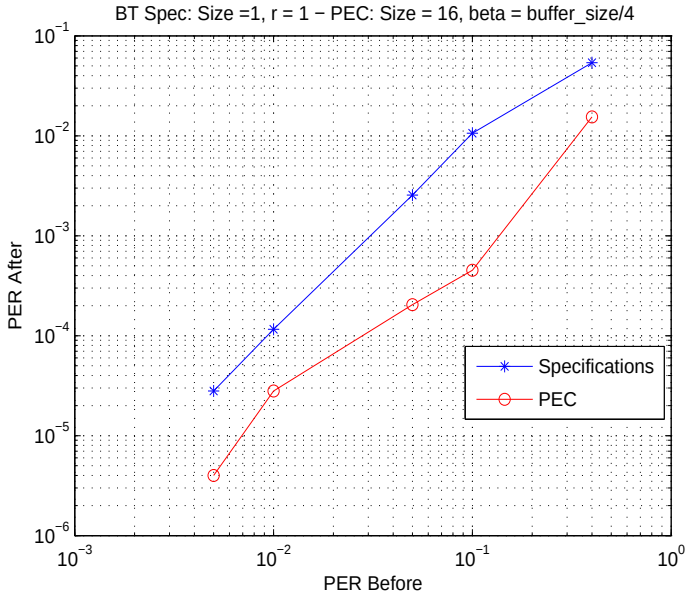


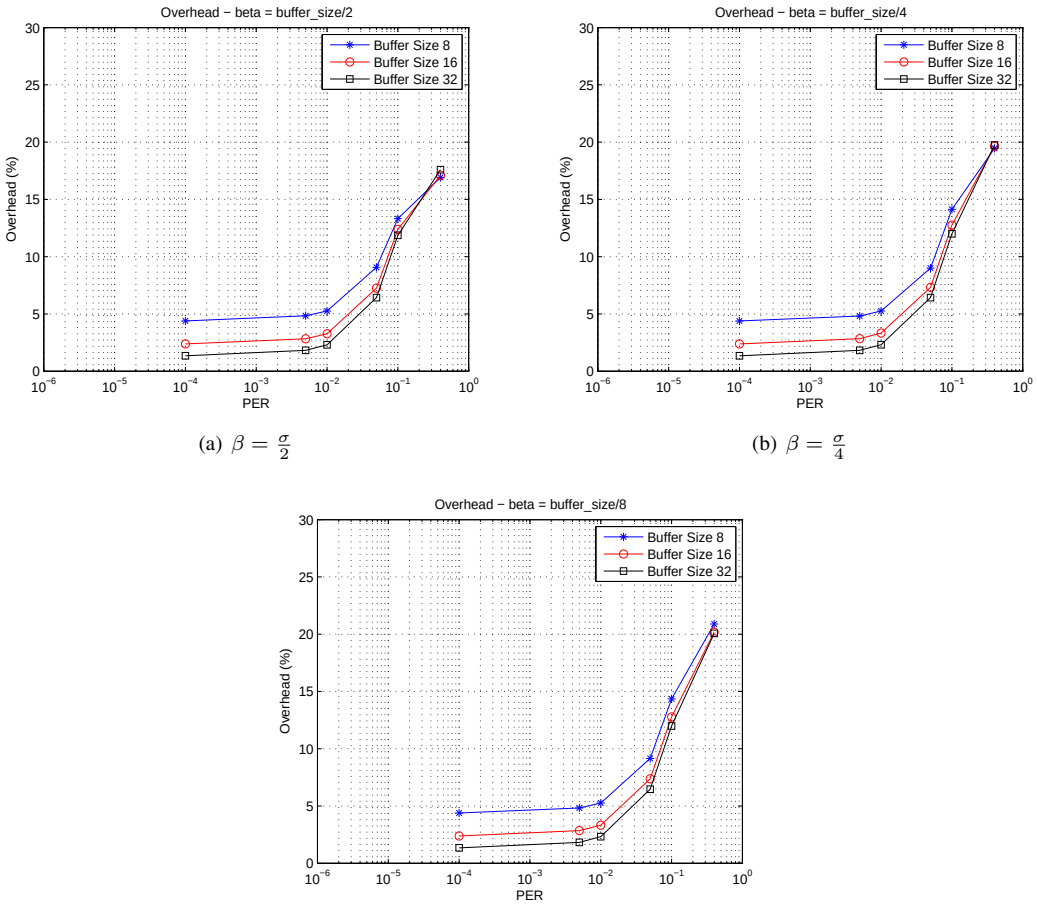
Figure 4.9: Packet Delivery Rate - Comparison between PEC and Specification-recommended Scheme

By looking at Figure 4.9, it is clear that PEC outperforms the retransmission method recommended by the specifications, while incurring far less overhead. For instance, for an initial error rate of 10^{-1}, PEC reduces this rate to about 3×10^{-3} with just 75% of packets retransmitted. For a similar result, the specification-recommended method requires an initial error rate of about 2×10^{-2} with 100% retransmission rate.

4.5.4 Data Overhead

The overhead incurred by the retransmission scheme includes both NACK packets and the actual retransmitted data. This measure also gives an indication of the data throughput degradation introduced by PEC. Figure 4.10 below shows the overhead introduced by PEC as a percentage of the total data sent.

The first notable point when examining Figure 4.10 is that overhead decreases as error rates decrease. This is because at low error rates, the number of packets in error is low, resulting in fewer retransmissions and thus lower overhead. Furthermore, because a NACK is sent every σ packets, the



(a) $\beta = \frac{\sigma}{2}$

(b) $\beta = \frac{\sigma}{4}$

(c) $\beta = \frac{\sigma}{8}$

Figure 4.10: Overhead Response

overhead decreases as the buffer size increases. By comparing Figures 4.10(a), 4.10(b) and 4.10(c) it is also evident that the overhead is inversely proportional to β, which is due to the fact that lower values of β translate into more packets being retransmitted, therefore resulting in more overhead. On the other hand, a lower value of β means higher correction probability. Similarly, increasing buffer size decreases overhead but increases delay. Therefore, there is a clear trade off between overhead, error rate response and delay.

PEC's overhead response is also compared to the retransmission scheme specified by the Bluetooth standard in order to demonstrate its advantages. The simulation results are shown in Figure 4.11 below. These are obtained by applying PEC, with $\sigma = 16$ and $\beta = \frac{\sigma}{4}$, and compared with the specification-recommended retransmission method, in which $\sigma = 1$ and $\rho = 1$, *i.e.,* the receiver requests the retransmission of every packet that is received in error.

Figure 4.11 shows that the overhead incurred by PEC is much lower than the one resulting from the specification-recommended scheme. Such a large gap is explained by the fact that the latter needs to acknowledge every packet so that the receiver can request an erroneous packet to be retransmitted,

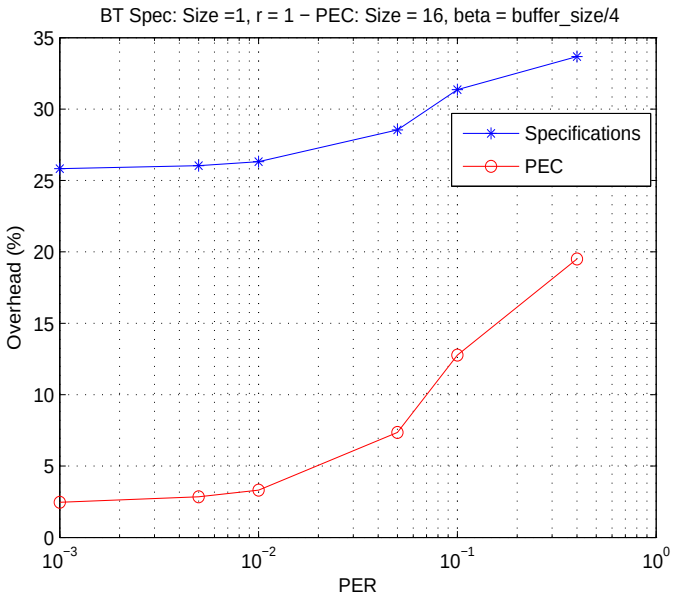


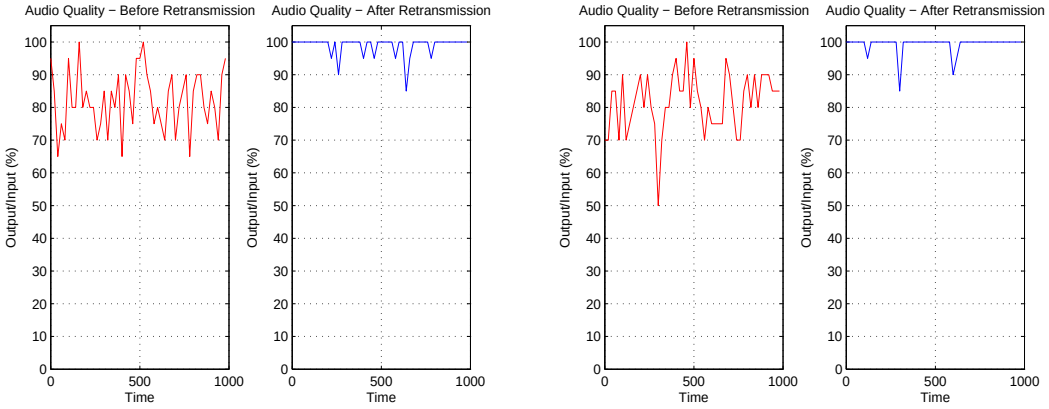
Figure 4.11: Overhead Response - Comparison of PEC and Specification-recommended Scheme

whereas PEC only sends a NACK every σ packets. In addition, PEC presents better PER response, which means that the number of retransmitted packets is lower for PEC than it is for the scheme defined in the standards, thus PEC results in lower overhead. As error rates decrease, and consequently the number of retransmissions decreases, the overhead would mainly be the result of acknowledgements, which explains the quasi-constant tendencies towards low error rates. The disadvantage of PEC, however, is that it incurs larger delays as explained in 4.14.

4.5.5 Audio Quality

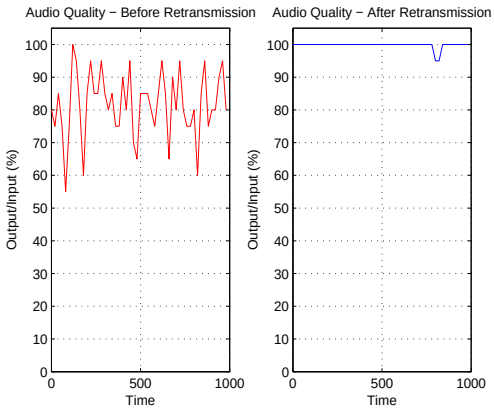
The quality measures of the audio data used in the simulation indicate, to some extent, the performance of PEC in terms of error reduction. One way of measuring audio quality is to show the deviation from a reference point, which in this case is the error-free audio file. This is achieved by dividing the resulting data, after applying the PEC retransmission, into bins representing $10ms$ each and plotting the deviation (as a percentage) from the original error-free audio data. The results obtained for three different buffer sizes, namely 8, 16 and 32, with PER = 20% and $\beta = \frac{\sigma}{8}$, are illustrated in Figure 4.12. In this figure a 100% mark suggests no errors, whereas a difference of percentage of Δ indicates a reduction of quality by $\Delta\%$. Each plot is compared to the quality obtained from sending the data through the channel with no retransmission in order to illustrate the improvement attained by PEC.

For ease of illustration, Figure 4.12 only shows 1000ms of the audio under test. Looking at Figure 4.12, the first observation one can make is that applying PEC does indeed improve the quality significantly, from a barely audible level to a level exhibiting only a few errors. The term error here means the number of erroneous packets present in each bin, which in turn represent 10ms of audio data. It is



(a) $\sigma = 8$

(b) $\sigma = 16$



(c) $\sigma = 32$

Figure 4.12: Audio Quality

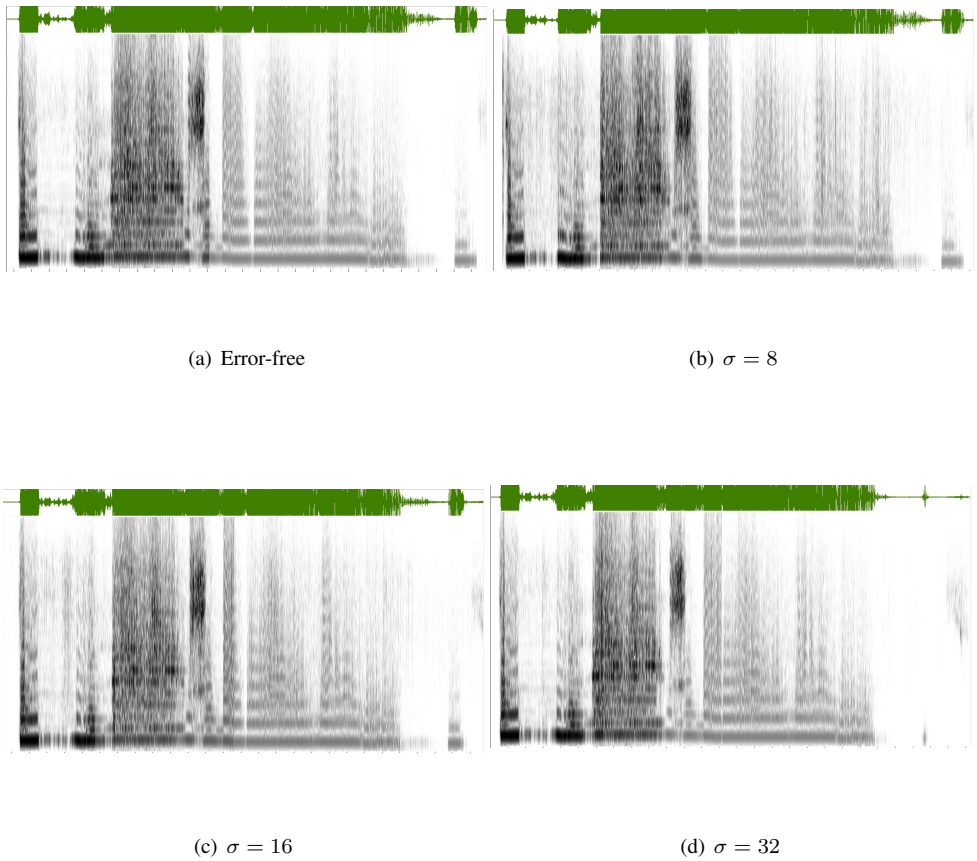
also easy to see that as the size of the buffer, σ, increases, so does the quality. This confirms the findings illustrated by Figure 4.8, which suggest a better error rate response for increased buffer size.

As previously mentioned, quality is a subjective measure, thus illustrating it visually is not straight forward. However, in the case of audio, it is possible to compare spectrograms, which provide a visual means of assessing the quality of the data [47]. A Spectrogram is the square of the magnitude of a signal's *Short-Term Fourier Transform* (STFT) such that

$$spectrogram(s(t)) = \left| \int_{-\infty}^{\infty} s(t)w(t-\tau)e^{-j2\pi f_s t}dt \right|^2 \tag{4.14}$$

Where $w(t-\tau)$ is a window function and $s(t)$ is the actual audio signal. Figure 4.13 shows the spectrograms generated from sending audio data through a noisy channel with PER=20% and with $\beta = \frac{\sigma}{8}$. These results are obtained using the RTGram tool [159], where the horizontal axis is time and the vertical axis represents frequency.

In phonetic and acoustic sciences, a spectrogram provides a way of representing particular sounds



(a) Error-free (b) $\sigma = 8$

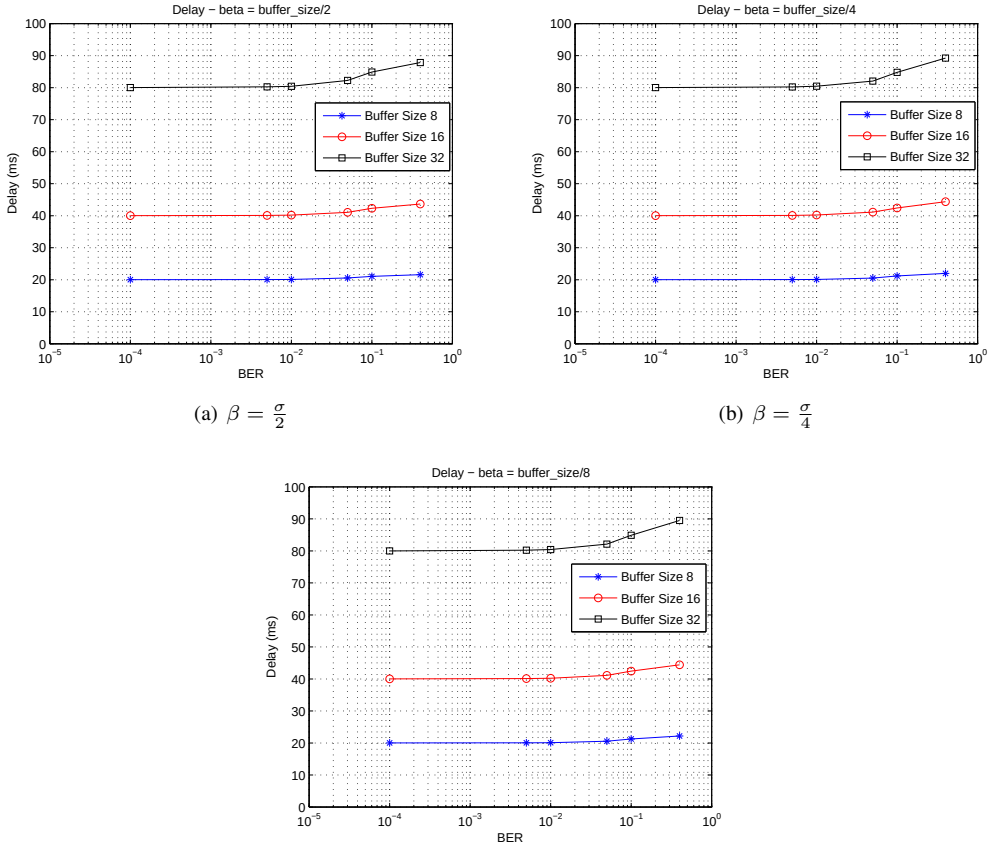
(c) $\sigma = 16$ (d) $\sigma = 32$

Figure 4.13: Audio Quality - Spectrograms

through their frequency "formants" and "bands". A formant identifies a particular frequency concentration, meaning a combination of harmonics, which in turn identifies a particular sound. These are represented by the dark areas in Figure 4.13, whereas the band points to the frequency tone, *i.e.,* the intensity of the sound, which is indicated by the corresponding frequency on the vertical axis. Packet errors introduce undesired sounds, and consequently, a shift in the signal's frequency components, which naturally alters how these change over time. A spectrogram can therefore be used to assess the quality of the received audio data in the presence of errors. A detailed analysis of the spectrograms given in Figure 4.13 is beyond the purpose of the this evaluation (for further reading on acoustic science and spectrography refer to [102, 81]); it can be seen, however, that all four of the figures above show close similarities, in terms of formants and bands, which proves PEC's ability to recover from errors even at high error rates. Furthermore, the figure providing the closest alignment with the error-free data is 4.13(d), which confirms that larger values of σ result in better error recovery.

4.5.6 Delay

The meaning of delay in this evaluation is the time taken by the receiver to process the data. This includes buffering time and delay introduced by the retransmission of erroneous packets. Data propagation delays and processing delays due to circuitry are not taken into account, as the aim of this section is only to show the impact of the retransmission scheme. Therefore the term "delay" here means the extra time introduced by the PEC scheme. Simulations are conducted on the audio data for various packet error rates, buffer sizes and values of β, wherein the delay is measured as an average per buffer. The results are shown in figure 4.14 below.



(a) $\beta = \frac{\sigma}{2}$ (b) $\beta = \frac{\sigma}{4}$

(c) $\beta = \frac{\sigma}{8}$

Figure 4.14: PEC Delay Response

As shown in Figure 4.14, the larger the buffer size the longer the delay, which is expected, as buffering more packets introduces more delay. The slow decaying trend that the curves follow is due to the delay added by retransmitting erroneous packets. At high error rates, the rate of change of the curves is higher, due to the fact that the number of erroneous packets is larger, hence incurring more retransmissions. Moreover, the number of re-transmissible packets, ρ, increases as BER increases, therefore more delay is introduced at higher error rates. As BER decreases, and consequently the number of packets in error decreases, delays tend towards a constant level, which is, essentially, buffering delay.

When comparing Figures 4.14(a), 4.14(b) and 4.14(c), it appears that varying the number of non retransmissible packet, *i.e.,* β, has no major effect on the delay. This is because the significant component making up delay values is buffer size. As far as real-time audio is concerned, buffer sizes of 32 packets are not an adequate choice, as they introduce large delays that would degrade user experience [89]. Furthermore, these values are for single hop communications; in a multi-hop environment these delays would be larger due to queue overflows and other effects such as congestion control introduced by higher protocols. It is therefore essential when deploying PEC to consider what an "acceptable" delay is (given a particular application) so that user experience is not degraded.

4.6 Summary

This chapter detailed a simple and efficient adaptive error correction scheme, PEC, that complies with the low complexity requirements of personal devices. It was shown that PEC behaves better than the retransmission scheme defined in the specifications in terms of error reduction and overhead response. Improving error response, however, comes at the cost of increased delays. There is, therefore, a trade off between latency, error rate response and overhead, which must be adjusted to suit the applications' requirements. This can be achieved analytically, through the proposed expressions, which allow the selection of the required parameters prior to implementation. Moreover, these parameters can be passed online through HCI commands or by defining new LMPs so that PEC is altered to satisfy the requirements. This allows seamless implementation without modifications to the protocol stack or hardware.

Chapter 5

Network Size Estimation

5.1 Introduction

This chapter proposes techniques for estimating the size and other topological properties of the underlying graphs, specifically adapted to suit the particular constraints of personal area networks, *i.e.,* master/slave settings. Since it is not practically feasible to maintain an accurate view of the underlying network at all times due to the unpredictable and constant changes in topology, some applications rely on the knowledge of its approximate size instead.

Network size estimation is a well studied area in large *peer-to-peer* (P2P) networks; two main techniques can be distinguished: random walk-based methods [109, 51] and gossip-based aggregation methods [148, 73]. In the latter approach, nodes average a value each time they communicate with each other, so that if the process is repeated sufficiently, nodes would obtain an average, from which the size of the network can be inferred. It was shown in [99] that for network sizes of up to $1,000,000$ nodes, gossip-based aggregation methods obtain an accurate estimate of the network size after just 40 rounds. Here a round is a repetition of the aggregation after a certain time (called an *epoch*) has elapsed. Random walk methods, on the other hand, trade accuracy for lower cover time. These techniques rely on passing a token around, gathering network measurements and estimating the size of the network upon return of the token to the initiator. This chapter investigates the applicability of these two techniques in mobile personal area networks and proposes adaptations to suit their particular topological and mobility constraints.

This chapter begins by introducing the network formation algorithm used throughout the thesis in section 5.2, it then describes and evaluates the random walk and the gossip-based network size estimation algorithms in sections 5.3 and 5.4 respectively. Finally it summarises the findings in section 5.5.

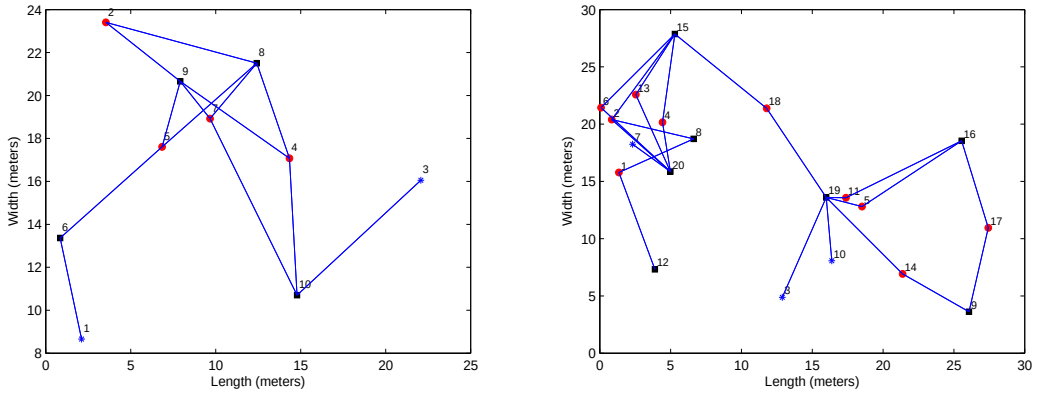
5.2 Network Formation

As mentioned in chapter 3, network topology is a fundamental factor of the performance of PANs. On the one hand, data forwarding is facilitated if nodes are well connected, but on the other hand, large degrees

increase bridging overhead and latencies. Therefore, there is an optimal topology, for a given scatter of nodes, for which throughput is maximised [111]. Achieving an optimal topology, however, requires extra overhead, which in turn, deteriorates the performance of the network. Consequently, in this project, each node selects its degree randomly, which according to the *No Free Lunch* (NFL) optimisation theory [169] results in a quasi-optimal topology.

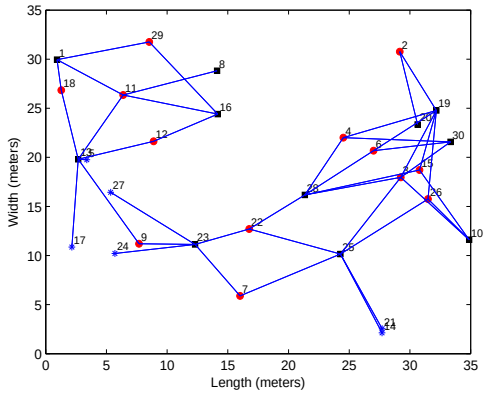
In the work presented in this thesis, the network is composed of master nodes, slave nodes and PMP nodes. However, PMP nodes can only be slave/slave nodes as this configuration facilitates inter-piconet scheduling [116]. The network formation employed herein is similar to the BlueNet algorithm [166], except that it only admits slave/slave bridge nodes and connections are established randomly. Similar to BlueNet, the proposed topology construction algorithm is divided into three phases. In phase I, some nodes (say m nodes) enter the page mode (p-masters) and start inviting other nodes to join their piconets, after having obtained the identities of their neighbours through the inquiry procedure. Each p-master, i, sets a maximum number of p-slaves, s^i_{max}, that it admits to its piconet in phase I, drawn randomly from the integer set $\mathbf{S} = \{1, 2, ..., 6\}$, so that it only accepts the first s^i_{max} page replies (DACs). In this phase, a p-slave that has already accepted a page from a previous p-master, ignores all other invitations, and in the end there would be m isolated piconets and some unconnected nodes. After phase I is completed, the masters instruct their respective slaves to enter page scan mode, in order to listen to other pages and start paging again. A slave, j, only replies to a page if it originated from a different master than the one it initially connected to, and if its degree is less than m^j_{max}. The latter represents the maximum number of piconets a slave node can be connected to in phase II, and is also selected randomly from the set $\mathbf{S}$. In phase III, the unattached nodes enter the page mode while the connected slaves enter page scan for a pre-defined timeout, t_o, during which the isolated nodes invite the connected slaves to join their piconets as PMPs. When a previously isolated node manages to connect to an existing slave, it pages its neighbours for a further t_o, in order to connect nodes that were unable to find a slave in their vicinity. If an isolated node cannot join the network, it enters page scan mode so that it can be invited by a recently connected master. If the last step fails, the isolated node signals to its neighbouring masters to page it so it can join the network. These procedures result in fully connected, quasi-optimal topologies that only admit slave/slave PMP nodes. Some of the randomly generated topologies are given in Figure 5.1. This attachment algorithm is used throughout this project.

The topologies given in Figure 5.1 are shown in two dimensional planes for ease of illustration. However, the evaluation is performed on nodes scattered across three dimensional spaces to reflect a realistic set up of PANs. In the figure, the star-shaped nodes represent pure slave nodes, the black squares are master nodes and the circular nodes are PMPs. It should also be noted that only slave nodes interconnect multiple piconets. The results given in Figure 5.2 illustrate some of the properties of the generated scatternets.

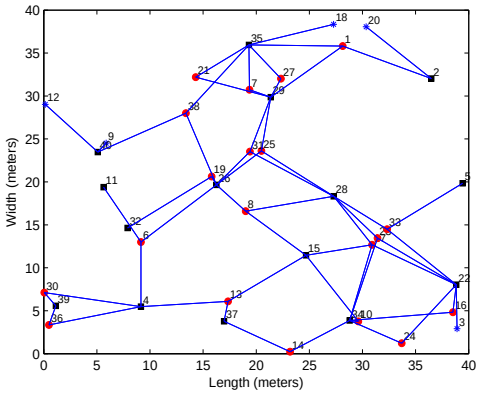


(a) n=10 nodes

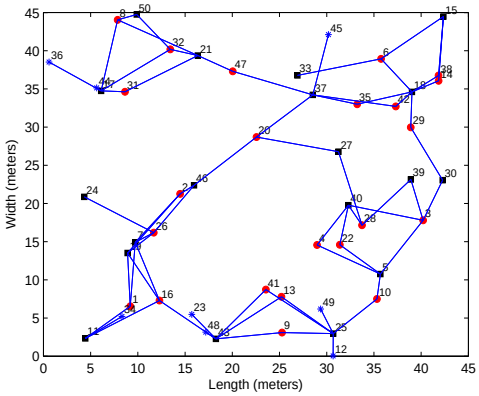
(b) n=20 nodes



(c) n=30 nodes



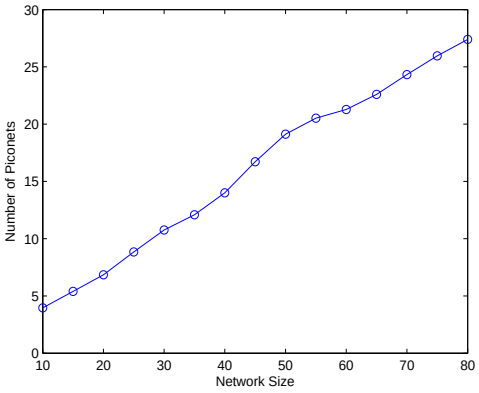
(d) n=40 nodes



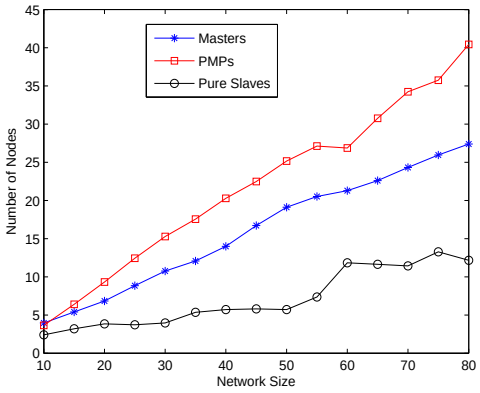
(e) n=50 nodes

Figure 5.1: Scatternet Topologies

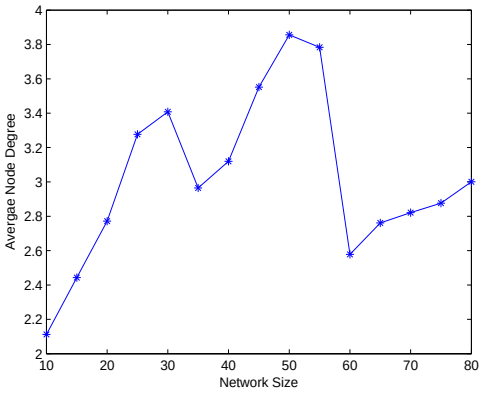
Figure 5.2 indicates that the number of masters, and hence piconets, increases linearly with the number of nodes in the network. The number of PMP and pure slave nodes also increases linearly; however, PMPs represent the largest portion of the nodes, which indicates good inter-piconet connection. Figure 5.2(c) shows no evident correlation between the number of nodes and average node degree, which is the result of the random attachment method used in this work.



(a) Number of Piconets



(b) Distribution of Roles



(c) Average Degree

Figure 5.2: Scatternet Properties

5.3 Random Walk-based Network Size Estimation

The random walk estimation scheme used in this work is based on the random tour method proposed in [109], which proposes a random tour technique for estimating the size of large peer-to-peer networks. In this method, an originator node, i, gathers information along the walk and estimates the size of the network upon return of the walk to this originator.

The originator, i, initialises a counter X to $\frac{1}{d_i}$, where d_i is the degree of node i, and forwards the message to one of its randomly selected neighbours. Upon receipt of the message, a node j, $(j \neq i)$, increments the counter by $\frac{1}{d_j}$, d_j being the degree of node j, and forwards it to one of its immediate neighbours, again chosen randomly. When the message returns to the originator, this node estimates the size of the network as: $\Phi = d_i X$

Random walk methods also allow nodes to gather other network statistics [39] such as node degrees and roles, which are key ingredients in the proposed algorithm. The Random Tour method [109] can, therefore, be adapted to suit the specific attributes of master/slave settings in order to yield more accurate estimates of the size of the network.

A master/slave PAN forms a bipartite graph with vertices of colours *m* and *s* for masters and slaves respectively. Let $G(V, E)$ define a master/slave PAN, in which the set of vertices $V(G) = M(G) \cup S(G)$, where $M(G)$ and $S(G)$ are the set of master nodes and slave nodes respectively. $S(G)$ is comprised of two subroups, namely $PS(G)$ and $P(G)$ such that $S(G) = PS(G) \cup P(G)$ where $PS(G)$ is the set of all s-coloured vertices whose degree is exactly one, *i.e.,* pure slaves, and $P(G)$ is the set of all s-coloured vertices whose degree is greater than one, *i.e.,* PMPs.

Let $N = |V(G)|$, $M = |M(G)|$, $S = |S(G)|$, $P = |P(G)|$ and $PS = |PS(G)|$. Consequently $N = M + S$ and $S = P + PS$. The values $\overline{d_m} = \frac{\sum_{v \in M(G)} d(v)}{M}$ and $\overline{d_s} = \frac{\sum_{k \in S(G)} d(k)}{S}$ are defined as the average node degrees of *m*-coloured and *s*-coloured nodes in $V(G)$ respectively. Similarly, the average degree of vertices in subgroup $P(G)$ is defined as $\overline{d_p} = \frac{\sum_{n \in P(G)} d(n)}{P}$

Proposition 5.3.1. *The total number of nodes (N) in a connected component within the network is:*

$$N = M \cdot (\overline{d_m} + 1) - P \cdot (\overline{d_p} - 1)$$

Proof. Because the graph is bipartite, every edge in $E(G)$ is incident in $M(G)$ and in $S(G)$; therefore:

$$e(G) = \sum_{v \in M(G)} d(v) = \sum_{k \in S(G)} d(k)$$

where $e(G) = \|E(G)\|$ is the number of edges in G.
Bearing the previous definitions in mind, the following holds true:

$$S = M \cdot d_m - S \cdot (\overline{d_s} - 1) \tag{5.1}$$

$$\sum_{k \in S(G)} d(k) = \sum_{n \in P(G)} d(n) + \sum_{i \in PS(G)} d(i) \tag{5.2}$$

$$\overline{d_s} = 1 \Rightarrow \sum_{i \in PS(G)} d(i) = PS \tag{5.3}$$

it follows from (5.1), (5.2) and (5.3)

$$S \cdot (\overline{d_s} - 1) = P \cdot (\overline{d_p} - 1) \Rightarrow S = M \cdot \overline{d_m} - P \cdot (\overline{d_p} - 1)$$

and finally

$$N = M \cdot (\overline{d_m} + 1) - P \cdot (\overline{d_p} - 1)$$

□

The function of this estimator is to obtain an accurate size of the network (the exact value of which is N) as well as topology characteristics. The network size estimation process is performed by each master node, which upon completion, broadcasts the estimate to its neighbours so that every node in the network has knowledge of the estimate.

At any non-initial state s, a node j, ($j \neq i$) selects one of its outgoing links at random with probability $p_j = \frac{1}{d_j}$ and forwards the estimate message to the corresponding node. The estimate message is the tuple $\{(N_m,D_m),(N_p,D_p)\}$, where N_m is a counter for the number of master nodes discovered so far and N_p is a counter for the number of PMP nodes. D_m and D_p represent the respective cumulative counts of the degrees of masters and PMP nodes encountered so far during the walk. The estimate message is initialised at the originating master i to $\{(1,d_i),(0,0)\}$ to indicate that, thus far, there has been only one master node discovered with degree d_i and no PMP nodes. Upon receipt of the message, a receiving node increments the counter N_m by 1 and adds its degree to the counter D_m if it is a master node, or updates N_p and D_p in the case of a PMP; the message remains unchanged if the node is a pure slave. When the message is returned to the initiating node, *i.e.,* when the tour is completed, that same node computes the average master and PMP nodes' degrees as $d_m = \frac{D_m}{N_m}$ and $d_p = \frac{N_p}{D_p}$ respectively, then estimates the size of the network according to proposition 5.3.1. The operation of the network size estimation is similar for both static and dynamic cases in the absence of node crashes; meaning that node movement may change the topology but the network remains connected.

5.3.1 Dealing With Mobility

In [46] the authors argue that a change in topology could be viewed as a change in the probability of choosing the next node to forward the message to. They prove that the walk completes in $O(n \log n)$ for a complete graph. However, in the case of a personal area network, the graph is not complete and the topology is governed by strict connectivity rules, therefore the assumption that the algorithm would eventually converge does not necessarily hold true in such scenarios. Furthermore, neither [109] nor [46] consider the case when it is impossible to forward the message (token) if, for instance, the node that currently holds the token crashes before forwarding it or moves out of range of the next node to receive the message. To account for these facts, the following modification to the random tour method is proposed:

When a non-initial node, i, receives the token, it is marked as having been visited and forwards the message to one of its immediate neighbours, node j for example, selected randomly with probability $\frac{1}{d_i}$. Node j then forwards the message to node q, again selected randomly, and sends an acknowledgement back to node i indicating that is has forwarded the message. If node i fails to hear the acknowledgement from node j, within a predefined timeout, it selects another node to forward the message to. When node i does not find any unmarked neighbouring node, it assumes that the walk cannot travel any further, in which case it returns the token to the initiator by flooding the network. This mechanism ensures that tokens are not lost or do not remain in infinite loops[1] without ever returning to the originator.

[1]The term loop refers to the repeated process of passing the token from one node to another. Infinite loops occur when the selected next hop node never coincides with the originator

5.3.2 Algorithm Evaluation

Figure 5.3 shows simulation results of network size estimations obtained through the random tour technique described herein, compared with the method proposed in [109], and the actual sizes of various randomly generated networks with different assumptions of node mobility. To simulate arbitrary movement in the network, a mobile node, changes its location coordinates at random and connects to its neighbours, according to the attachment mechanism described in Section 5.2. For simulation purposes, however, it is assumed that the random movement of nodes does not partition the network.

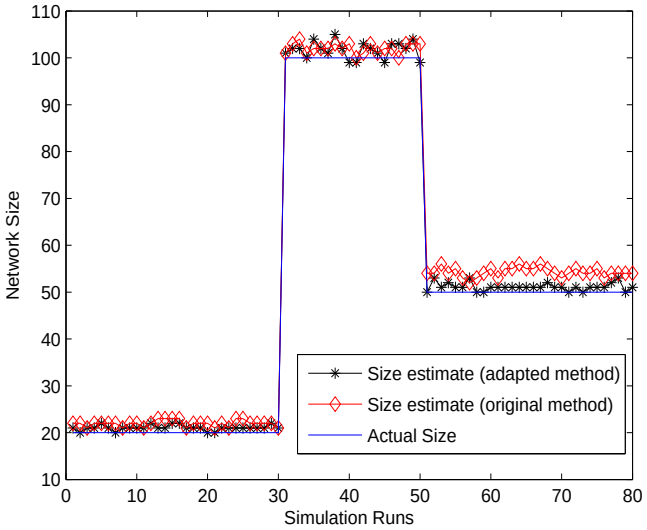
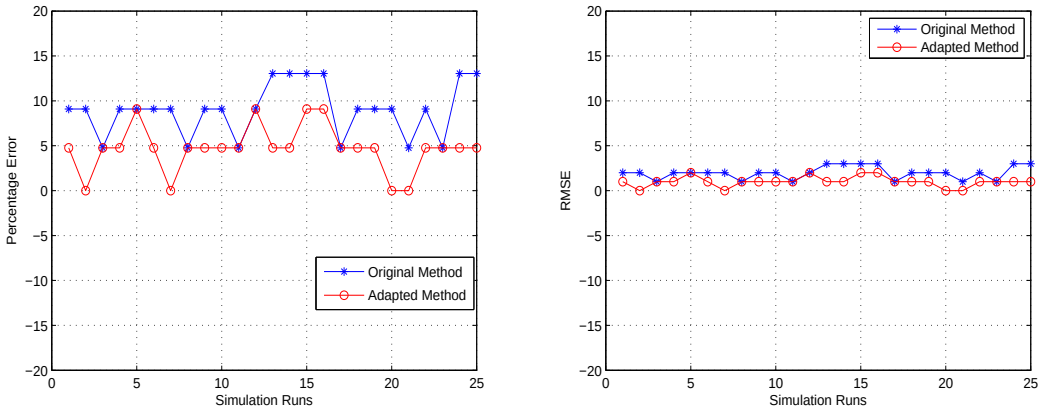


Figure 5.3: Network Size Estimations (Random Tour Method)

It can be seen from Figure 5.3 that the proposed adaptation yields better estimates than the original random tour method proposed in [109], which proves the adequacy of this technique. Moreover, the accuracy of the network size estimate also reflects the accuracy of network characteristic estimates, which are essential to the leader election algorithm (see chapter 6). Both methods, however, show an estimation bias; this is caused by the attachment characteristics of the graphs, which connect nodes that are within close proximity; as opposed to a fully connected graph, such as the ones considered in [109]. However, the proposed random tour method exhibits less bias than the original algorithm. This fact can also be verified by looking at the error graphs shown in Figure 5.4, which illustrates the Percentage Error and *Root Mean Square Error* (RMSE) performances, for both estimators, for network sizes varying between 20 and 100 nodes.

Figure 5.4(a) clearly shows an estimation bias, albeit with relatively low error terms. Moreover, it can be seen that the proposed adapted method presents less deviation from the actual network size than the original method (about 10% for the proposed technique contracted with 15% for the algorithm described in [109]). Figure 5.4(b) indicates the deviation from the mean; again, it is shown that the



(a) Percentage Error

(b) Root Mean Square Error

Figure 5.4: Random Tour-based Estimator's Error Performance

estimates tend towards a positive bias, although both are near zero, with the adapted method presenting less deviation. This indicates that the proposed method is better suited for master/slave personal area networks.

5.4 Gossip-based Network Size Estimation

In gossip-based aggregation algorithms, an initiating node sets a variable, avg, to 1 with all other nodes having this variable set to 0. The initiator, i, chooses one of its outgoing links randomly with probability $\frac{1}{d_i}$, where d_i is the outdegree of node i, and exchanges information with the corresponding next hop neighbour, p_{next}. Both nodes then average their avg variable as $\frac{avg_i + avg_{P_{next}}}{2}$ and store the resulting value locally. Every time two nodes communicate, they average their values. It is evident that if this process were to be repeated infinitely then every node in the network would have its avg variable set to exactly $\frac{1}{N}$, where N is the number of nodes in the network.

In PANs, however, one can take advantage of the particular master/slave topological characteristics of such networks. Nodes are arranged in pico-networks, each of which is coordinated by a master node, which in turn, serves only the slave nodes with which it scheduled a rendezvous point. In the proposed protocol, the master collects the averages from all currently connected slaves, performs the averaging, then broadcasts the results to every slave node currently synchronised with the piconet. If this process is repeated for a sufficiently large number of rounds then every node in the connected component will be able to accurately derive the size of the network (N). Here a round is defined as the process of collecting, averaging and re-distributing the information. The operations of the proposed gossip-based aggregation technique are given in Algorithm 3, which shows the process run by each piconet.

Table 5.1: Gossip-based Aggregation Algorithm - List of Notations

M	master node.
S_M	set of next hop slave nodes to master M.
$S_c(M)$	slaves currently participating in M's piconet.
S	slave node.
$avg(i)$	aggregate average held by node i.
$POLL(M, S_M)$	polls slaves from set S_M for communication with master M during reserved time-slots.

Algorithm 3 Gossip-based Aggregation Network size Estimation Algorithm

while 1 **do**
  $S_c(M) \leftarrow POLL(M, S_M)$
  for all $S \in S_c(M)$ **do**
    $avg(M) \leftarrow avg(M) + avg(S)$
  end for
  $avg(M) \leftarrow \frac{avg(M)}{(||S_c(M)||+||M||)}$
  for all $S \in S_c(M)$ **do**
    $avg(S) \leftarrow avg(M)$
  end for
end while

Every master node, M, polls the set of slaves, S_M, with which it scheduled a rendezvous point, one at a time, via the function $POLL$. The master then collects the avg variable stored at each slave and increments the aggregate average. The piconet average is then obtained by dividing the aggregated averages stored at the master by the number of nodes in the piconet (master plus slaves). Finally the average is distributed to all the nodes in the piconet via broadcast. This process is repeated either indefinitely or until a predefined number of rounds is reached.

5.4.1 Effect of Node Mobility

An obvious advantage of this technique is that node mobility does not deteriorate the precision of the estimate, as regardless of the location of the nodes in the network, the sum of all the averages will always converge to unity, that is:

$$\sum_{i \in V(G)} avg_i = 1 \tag{5.4}$$

from which it can be deduced that:

$$\forall i \in V(G) : \lim_{k \to \infty} \left(\frac{1}{avg_i(k)} \right) = ||V(G)|| \tag{5.5}$$

where $V(G)$ is the set of nodes in the connected component, avg_i is the average at node i, and k is the number of aggregation rounds. Because the number of rounds needed to find the exact network size can, in theory, be infinite, in practice algorithms are considered converged if the estimate reaches a certain precision level [148]. Since the protocol is applied to ad hoc networks, with much fewer nodes than

large peer-to-peer networks, the algorithm does converge towards the exact network size in a finite time regardless of the mobility pattern.

5.4.2 Evaluation

The gossip-based aggregation method is simulated on a number of random networks generated using the algorithm described in 5.2. Figure 5.5 shows simulation results for the number of rounds taken to find the network size estimate for different estimation precisions and under various assumptions of node mobility conditions. Here, nodes do not follow a particular mobility model, instead they move randomly and with varying speeds. Node mobility may partition or merge network components and nodes may change their roles to suit the connectivity characteristics of the proposed attachment scheme.

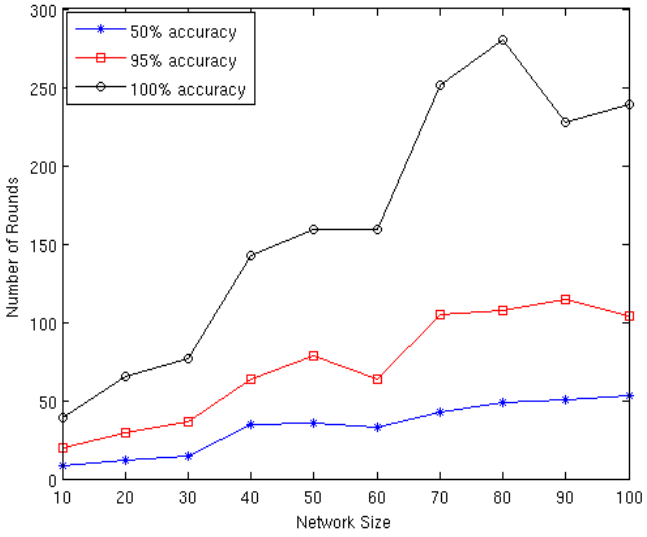


Figure 5.5: Gossip-based Aggregation Method - Rounds vs. Precision Comparison

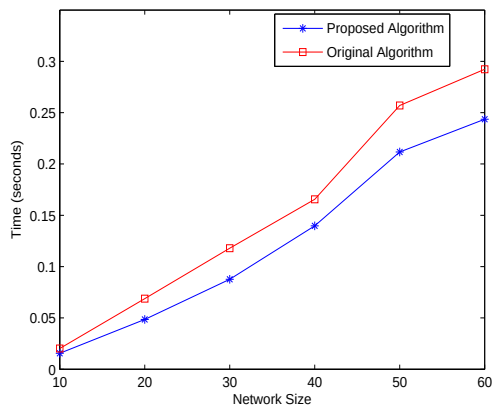
The fist observation that can be made by looking at Figure 5.5 is that the quality of the estimate does not follow a linear trend with the number of rounds. This indicates that for very accurate estimates, the algorithm needs to run for a large number of rounds. Another observation is that curves exhibit fluctuations, which are due to the randomness of the underlying topologies. In well connected networks, the number of rounds taken to reach a given precision level is less than that of a dispersed network. This means that a major factor governing the convergence time (in terms of number of rounds) of the gossip-based network size estimation algorithm is the degree centrality of nodes. However, the technique proposed in this thesis requires far fewer rounds than the gossip-based aggregation method, proposed in [73] does. Table 5.2 shows a comparison between the original and the adapted algorithm for 100% precision level for different network sizes.

The significant difference shown in the table above, stems from the fact that in each round, the adapted method produces the average of all participating nodes in the piconet, whereas in the original

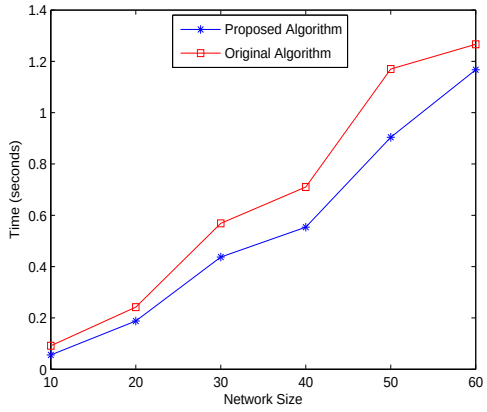
Network Size	20	60	100
Number of rounds (original algorithm)	2544	5035	7521
Number of rounds (proposed algorithm)	123	232	277

Table 5.2: Comparison Between Original and Adapted Gossip-based Aggregation Algorithms

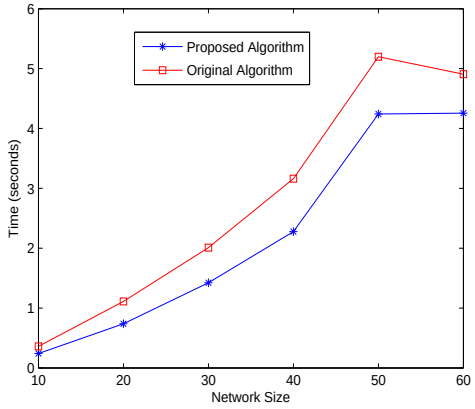
approach the aggregation concerns only a single master-slave pair. However, because of the TDD transmission characteristics of PANs, a round in the adapted method is expected to be longer than that in the original method. This is because, in the adapted method, a master sequentially collects the average from each slave currently participating in the piconet then broadcasts the value; whereas in the original algorithm a round only represents a poll/response cycle. It would therefore be fairer to compare the time taken to reach a given precision level in both algorithms rather than comparing the number of rounds. This is given in Figure 5.6



(a) 50% Precision



(b) 90% Precision



(c) 100% Precision

Figure 5.6: Gossip-based Aggregation Method - Estimation Delays

In the simulations, nodes are static; however, mobility would have no effect on the behaviour of the algorithms since the aggregation process occurs between any communicating nodes, thus it is indepen-

dent of the location of the nodes. Figure 5.6 shows that the original gossip-based estimation method takes longer to reach the desired precision level than the proposed adaptation. This diffence stems from the fact that the original method proposed in [73] requires a master-slave-master exchange in each round, *i.e.,* a poll-response cycle, whreas in the case of the method proposed herein, the exchange relies on broadcast which does not necessitate polling each slave at a time in order to send the aggregate averages, thereby resulting in quicker convergence time. The time complexity given herein shows the same trend as the number of rounds given in Figure 5.5, where an increase in precision level does not follow a linear increase in time, which again confirms that for precise estimates, gossip-based aggregation algorithms require large runtimes.

5.5 Summary

This chapter has adapted two network size estimation methods to suit the master/slave architecture of personal area networks and has shown that both methods can accurately estimate the size of the network. The gossip-based aggregation method exhibits better accuracy than the random tour method but at the expense of longer convergence time. In the work presented in this thesis, the random walk estimation method is preferred as it allows the collection of other network statistics such as the number of masters, the number of PMPs and the average node degree, which are all essential to the leader election algorithm described in the next chapter. However, that being said, both methods cope with random topology changes, which makes them suitable for mobile personal area networks.

Chapter 6

Cooperative Network Leader Election

6.1 Introduction

In distributed computing, a network leader is a node responsible for carrying out a particular task such as membership management, data collection, payment management or routing control. The leader election problem has been studied extensively, particularly in the field of distributed computing [151], and many algorithms have been devised to elect a unique coordinator in the network. The complexity of leader election in ad hoc networks, however, is augmented by the dynamic and unpredictable nature of the environment and can therefore, yield large convergence times, large overheads, and in some cases impossibility of consensus. There are many approaches to the leader election problem in ad hoc networks; three of the predominant ones are tree-based [163, 100], routing-based [106, 43] and probabilistic algorithms [55, 4]. Regardless of which approach is taken, or the role of the leader, the leader election problem must satisfy two fundamental conditions:

1. There is only one leader in a connected component.
2. Every node in the connected component has knowledge of the identity of the leader.

The proposed leader election algorithm satisfies these two fundamental conditions. It falls within the category of gossip algorithms, in that a gossip (the identity of the leader) will be disseminated to the entire network through local exchange of information. Each master gathers information from its currently connected neighbours, compares this information with its own, and re-distributes the best found value via broadcast. A fundamental question that must be answered, however, is: "How many times must a master repeat the process of collecting, comparing and redistributing the information before a unique leader is known to every node?" The answer to this question is strictly linked to the underlying topology characteristics of the mobile network. For the algorithm to converge, nodes require knowledge of the connectivity characteristics of the underlying topology, namely the estimated number of PMP nodes in the network and their corresponding average degree. This is achieved via the random tour network size estimation method described in Chapter 5.

6.2 Cooperative Leader Election

In the proposed cooperative leader election algorithm, any node in the network is a candidate, *i.e.,* it is a potential leader, and the node to become the leader is elected on its ability to carry out the required tasks. Each node computes a *value* that identifies its capacity to become a leader. In the case of routing management tasks, which are themselves dependent on various criteria, the value of a node i is the maxima of the utility, μ_i such that

$$v_i = max(\mu_i) = max\{f(\delta_i, \varepsilon_i, \eta_i)\} \tag{6.1}$$

Where δ_i is the degree centrality of node i, ε_i is its residual energy and η_i is a measure of stability, *i.e.,* how immobile the node is, on average, over time. The degree centrality indicates the node's ability to reach other nodes in the network so that dissemination is optimised, the residual energy is a measure of the life time of the node, and the stability indicates that routes to the potential leader change less frequently.

The proposed leader election algorithm relies on cooperation amongst nodes, in that the identity and value of the leader is spread by exchanging and updating local information until there is only a unique leader left in the network.

In Bluetooth scatternets, a time-based scheduler ensures that a slave PMP maintains only one active connection with one of its neighbouring masters at any one time [174]. As a result, at any given time, the network would be composed of independent and disconnected clusters, which, in a static setting would cycle back to the original state as shown in Figure 6.1.

Bluetooth relies on a TDD polling scheme, where a slave node can only communicate with its master if it has received a message in the previous time-slot. There are various scheduling mechanisms that could be employed for this purpose (see section 3.2.2 in Chapter 3); however, in order to keep operations simple and to ensure fairness, a PRR scheduler is employed. This scheduler polls each slave in turn in a cyclic manner to exchange information within a dedicated time-slot. In some applications, PRR is not the most efficient scheme, as nodes might be polled without having anything to send [172], [118]. In the proposed leader election algorithm, however, PRR is perfectly adequate, as the algorithm generates low traffic and requires continuous updates from all slaves [174]. In scatternets, the scheduling becomes more complex and the mechanism must ensure that no transmission loops occur [12], [88]. The inter-piconet scheduling employed in this work is based on a leasing mechanism, in which a master node reserves a time-slot for communication with a particular slave in the piconet. Consequently, this slave becomes unavailable to any other master during that time-slot and is only released upon completion of the update.

The proposed leader election algorithm makes use of this transmission mechanism in order to spread

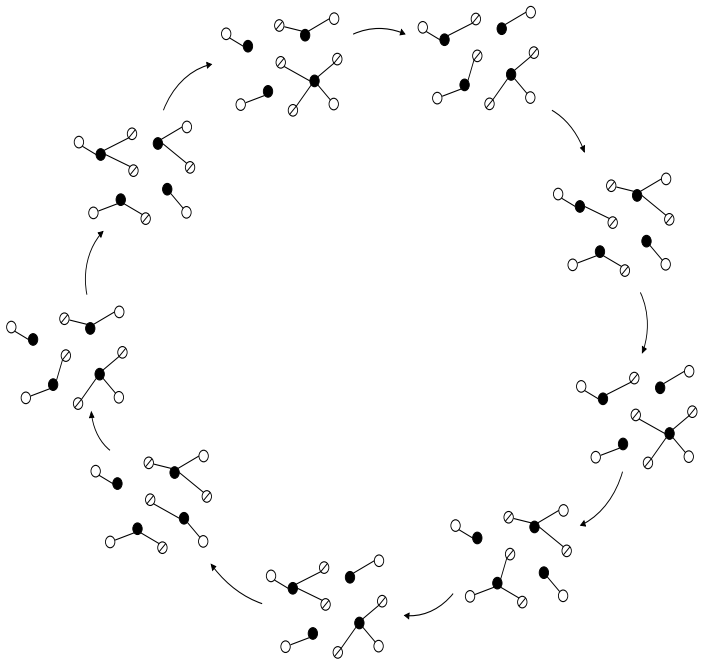


Figure 6.1: Independent Network States - Scatternet Scheduling

the identity of the unique leader in the connected component. The algorithm works as follows:

A leader election process is triggered when nodes are notified of the failure of the current leader, either due to its departure or its value dropping below a predefined threshold. Every node in the network keeps a variable tuple, (L_{id}, L_{val}), that indicates the id and value of the leader known so far, and initiates itself as a candidate. At the start of the algorithm, every node sets the leader id and value to its own. At each iteration, when a slave within a piconet is polled, it transmits the id and value of the leader it holds to its master; the master then compares the information it receives from the slave with the local one. If it learns of a better-valued leader, it updates its leader information (L_{id} and L_{val}) or discards it otherwise. After the completion of the Round-Robin tour, the master node broadcasts the information to the entire piconet.

A PMP that was previously connected to a particular master would join another piconet at the subsequent iteration. If this PMP holds a better leader value, it will update the master, and corresponding slaves participating in that piconet, with the new leader information, otherwise, it will update its own leader information. After going through this process several times, the unique identity of the leader will eventually be known to every node in the network. The process run by each master is illustrated in Algorithm 4.

Table 6.1: Leader Election Algorithm - List of Notations

M	master node.
S_M	set of next hop slave nodes to master M.
$S_c(M)$	slaves currently participating in M's piconet.
itr	number of iterations remaining.
S	slave node.
$L_{id}(M)$	id of the leader currently held by master M.
$L_{val}(M)$	value of the leader currently held by master M.
$L_{id}(S)$	id of the leader currently held by slave S.
$L_{val}(S)$	value of the leader currently held by slave S.
$POLL(M, S_M)$	polls slaves from set S_M for communication with master M during reserved time-slots.

Algorithm 4 Leader Election Algorithm

$S_c(M) \leftarrow POLL(M, S_M)$
while $itr > 0$ **do**
  for all $S \in S_c(M)$ **do**
    if $L_{val}(S) > L_{val}(M)$ **and** $L_{id}(S) > L_{id}(M)$ **then**
      $L_{id}(M) \leftarrow L_{id}(S)$
      $L_{val}(M) \leftarrow L_{val(S)}$
    end if
  end for
  for all $S \in S_c(M)$ **do**
    $L_{id}(S) \leftarrow L_{id}(M)$
    $L_{val}(S) \leftarrow L_{val}(M)$
  end for
  $itr \leftarrow itr - 1$
end while

Proposition 6.2.1. *The number of rounds needed for the gossip to be spread to the entire network has an upper bound determined by:*

$$itr_{max} \leq 2 \times \left(\sum_{i \in P(G)} d(i) - P \right) + 1 \tag{6.2}$$

Where $P(G)$ is the set of PMP nodes in the network G, P is the number of PMP nodes and $d(i)$ is the degree of PMP i. itr_{max} represents the maximum number of iterations that a master repeats the process of collecting and re-distributing leader information to satisfy the two fundamental conditions of the leader election problem.

Proof. Consider a star network with a single PMP node, p_1, and up to seven masters $\{m_1, m_2, ...m_7\}$. The maximum number of iterations it takes for the PMP node to communicate with a particular master, assuming a PRR schedule is employed, is d_{p_1}, where d_{p_1} is the degree of the PMP node. The maximum number of rounds it takes for two master nodes to exchange information is $d_{p_1} + d_{p_1} - 1 = 2 \times d_{p_1} - 1$.

If there is now another star network, with single PMP node p_2 of degree d_{p_2} linked to the first network via a bridge master node, then the maximum number of rounds it takes to convey information from one end to the other is:

$$(2 \times d_{p_1} - 1) + \{(2 \times d_{p_2} - 1) - 1\} \tag{6.3}$$

If n subnetworks are cascaded in a similar arrangement, then the upper bound is given by:

$$\begin{aligned} itr_{max} = &(2 \times d_{p_1} - 1) + \{(2 \times d_{p_2} - 1) - 1\} + \cdots \\ &+ \{(2 \times d_{p_n} - 1) - 1\} \end{aligned} \tag{6.4}$$

Equation 6.4 becomes

$$itr_{max} = 2 \times \left(\sum_{i \in P(G)} d(i) - P \right) + 1 \tag{6.5}$$

□

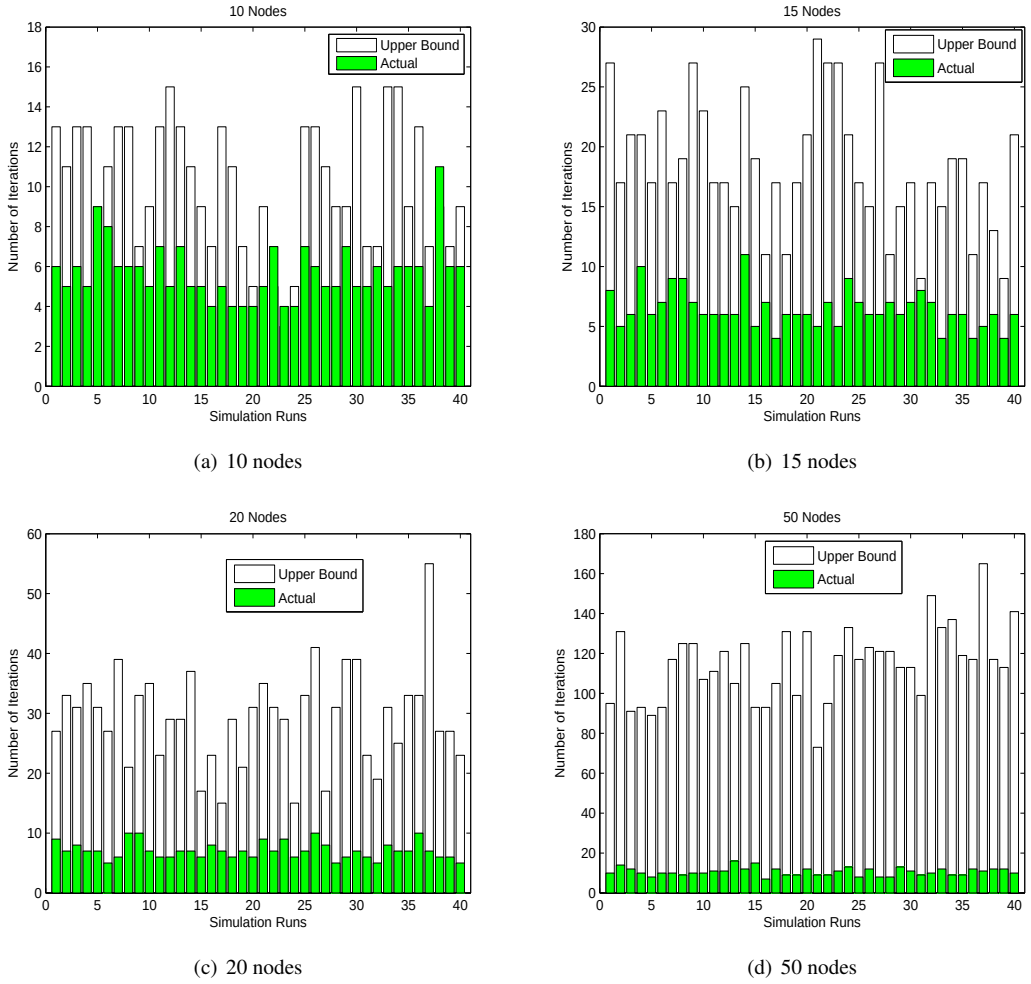
Equation 6.5 defines an upper bound for the number of rounds a master needs to exchange information with its neighbours, which can be approximated to:

$$2 \times \hat{P}(\overline{d_p} - 1) + 1 \tag{6.6}$$

where $\hat{P}$ is the number of estimated PMP nodes and $\overline{d_p}$ is the average degree of the estimated PMPs, both of which are obtained from the network size estimation method described in Chapter 5.

This upper bound is reached in two particular cases: when nodes are arranged in a star topology as described above and in a line arrangement, wherein the gossip needs to traverse the entire network, from one end to the other. This is due to the fact that conveying a single piece of information from one node to another, in both settings, could, in the worst case, involve every node in the network. In realistic networks, however, this limit is rarely reached, because of the connectivity properties of nodes. Figure 6.2 shows a comparison between the actual number of rounds it takes to elect a leader and the corresponding upper bound limit as expressed by equation 6.5, for randomly generated scatternets networks.

It can be seen from Figure 6.2 that the number of iterations it takes to elect a leader, in realistic networks, rarely reaches the theoretical limit. This is certainly true for medium and large size networks. However, for smaller networks, in which nodes are more likely to be arranged in star or line topologies, the upper limit has higher chances to be attained, although these cases still remain relatively infrequent (as shown in Figure 6.2(a)). This proves that line or star topologies (or a combination of both) present



(a) 10 nodes

(b) 15 nodes

(c) 20 nodes

(d) 50 nodes

Figure 6.2: Comparing Actual and Upper Bound itr Limit.

worst case arrangements in terms of the number of rounds. Another observation is that the as the size of the network increases, so does the difference between the actual and the worst-case limits. The network size estimation methods described in Chapter 5 can be used to adjust the number of iterations according to the estimated size of the network.

6.2.1 Dealing with Node Mobility

Unlike other leader election schemes, the cooperative leader election algorithm does not require significant longer election time when there is node mobility. If the moving node holds the information on the best leader, it propagates this information to other nodes in the new area. Similarly, if a node does not possess the knowledge of the unique leader and moves within proximity of nodes that do have knowledge of the leader, it will be updated. Therefore, as the number of moving nodes and the velocity increase, one would not expect to see a degradation of performance of the algorithm in terms of the election time.

Under cooperative leader election, node arrivals are handled slightly differently. When a node joins a network, it would have its leader information set to its own id and value. There are two cases to

consider:

Case 1) The network already has a leader: if the joining node has a lower value than the leader of the network, it simply adopts that leader as its own. However, if its own value is greater than that of the current leader, it informs its neighbours, which subsequently inform their own in a flooding process so that the leader information is propagated to the entire network. Updating through flooding is a more sensible approach in this case, as initiating a new leader election process would be wasteful since there is only one candidate.

Case 2) The network has yet to establish its leader: when a node, n, joins the network during the election process, it learns the current iteration counter from its neighbours and updates it according to its connection information. If it is a master node then it increments the counter by 1 or if it joins as a PMP node, it adds $2 \times (d_n - 1)$ to the counter (where d_n is its degree), and updates its neighbouring masters with the new counter.

6.2.2 Performance Evaluation

The criteria used to evaluate the performance of the leader election algorithm are the time it takes to find the leader, its scalability and its response to dynamic changes in a network topology. The algorithm is simulated on various randomly generated networks and under different assumptions of node mobility conditions.

The algorithm is first simulated in a static environment, where nodes do not move and no new node joins or leaves the network. Figure 6.3 shows the election convergence time of the cooperative leader algorithm as a function of the size of the network, and compares its performance with tree-based election algorithms.

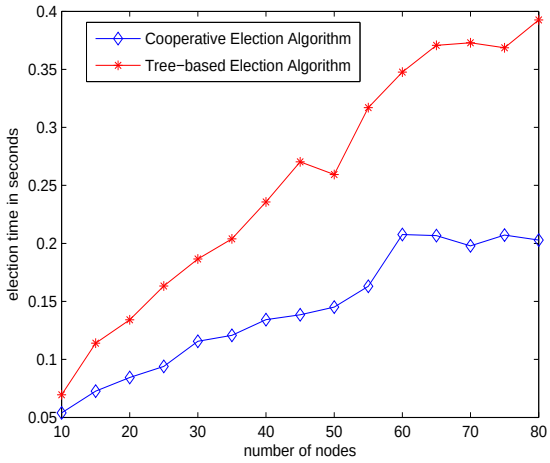
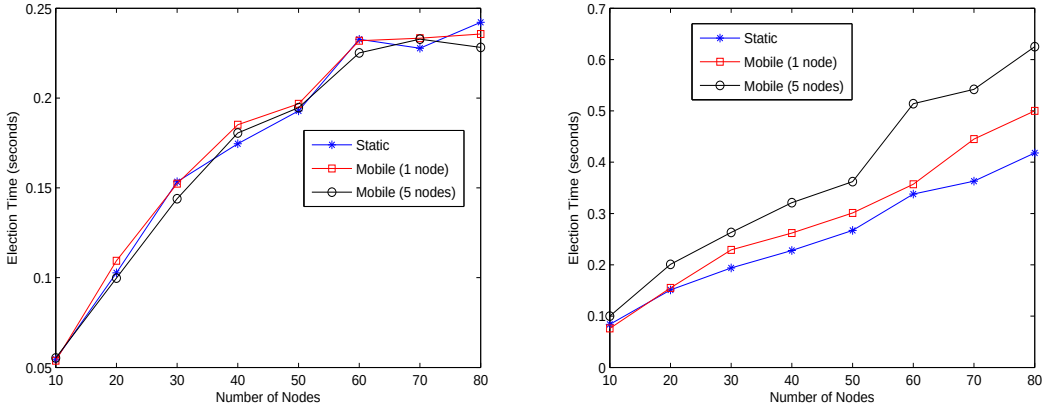


Figure 6.3: Leader Election Convergence Time

Figure 6.3 shows that a key difference between tree-based algorithms and the proposed cooperative

approach is that the former increases linearly with the size of the network, whereas the latter tends towards saturation. The gap between the two becomes more apparent as network size increases. This is explained by the fact that expending and shrinking a tree increases linearly with network size, whereas in the case of the cooperative leader election algorithm, the convergence time is governed by degree centralities and node roles as indicated by Equartion 6.2. The tendency for the convergence time to plateau in cooperative election can also be verified by Figure 6.2, which shows that the increase in network size does not follow a linear increase in the number of rounds. This is a noteworthy result, which proves that the cooperative election algorithm scales well with network size.

The algorithm is also simulated in a mobile setting, varying the velocity of nodes and the number of moving nodes. In order to evaluate the impact that frequent and unpredictable changes in network topology have on the performance of the algorithm, the simulation is performed for the exact time it takes to elect a leader when nodes move freely. Figure 6.4(a) illustrates a comparative performance result of the cooperative leader election algorithm as a function of network size under different assumptions of node mobility: static setting, mobile setting with a single node moving at a time, and mobile setting with 5 simultaneous node movements in the network. Nodes are assumed to move arbitrarily with various speeds and in any given direction. The same scenarios are repeated for the tree-based election algorithm and the results are given in Figure 6.4(b) for comparison purposes. The delays introduced by node mobility and link establishment are not taken into account as the aim is to show the impact of mobility on the performance of the algorithms.



(a) Cooperative Method

(b) Tree-based Method

Figure 6.4: Leader Election Convergence- Response to Mobility

As can be seen from Figure 6.4, node mobility does not deteriorate the performance of the cooperative leader election algorithm, in the sense that an increase in mobility does not yield longer election time, whereas the impact of mobility on tree-based election is noticeable. In the cooperative case, when a particular node that holds the identity and value of the true network leader moves to within an area

where nodes have yet to acquire knowledge of the leader, it updates those nodes. Similarly, if a node that has not yet been updated with the leader information moves to within proximity of nodes whose leader information corresponds to that of the actual leader, it will be updated. On the other hand, in the tree-based case, when a node moves, the tree needs to be updated, which introduces extra delays. This suggests that the cooperative election method is more suitable for dynamic environments than tree-based approaches.

6.3 Summary

This chapter presented a novel leader election algorithm for master/slave personal area networks and showed that it scales well with network size and node mobility. The algorithm exploits knowledge of the underlying topology, which is obtained through a random tour technique, in order to estimate the number of rounds a master needs to repeat the election procedure before a leader is elected. When compared to tree-based algorithms, it was shown that the cooperative approach performs better in terms of response to node mobility and convergence time. This makes the cooperative leader election approach more suitable for ad hoc settings that exhibit spontaneous changes in topology. Furthermore, the cooperative election algorithm can be applied to other types of ad hoc network without major modifications.

Chapter 7

Routing for Mobile Personal Area Networks

7.1 Introduction

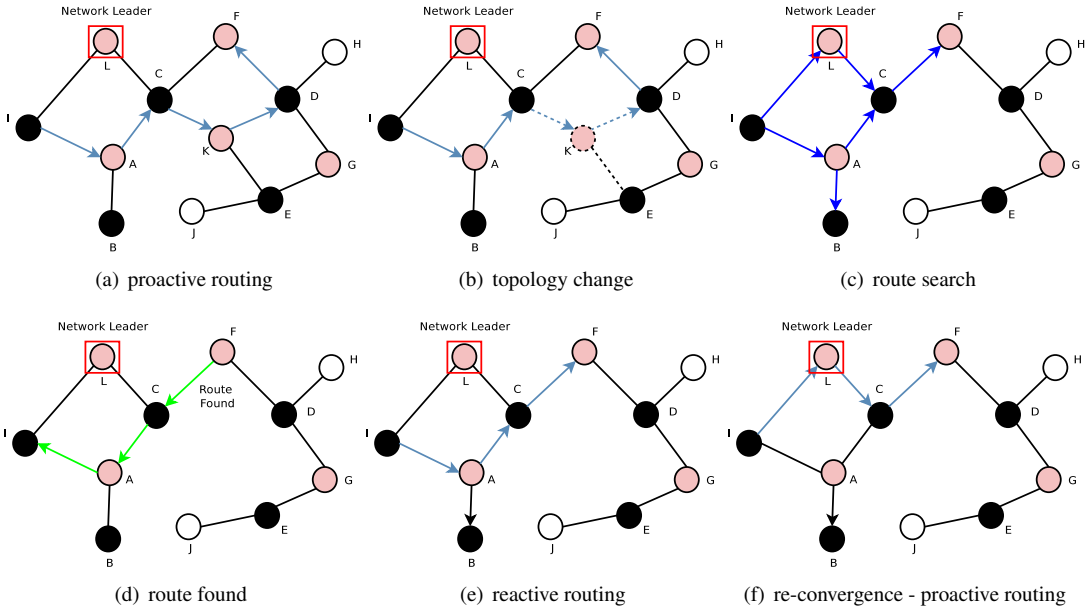
This chapter describes a novel hybrid routing protocol for mobile personal area networks: the *Centrally Coordinated Hybrid Routing* (C2HR) protocol. Unlike other proposed hybrid routing protocols, which are either zone-based or tree-based, the proposed protocol uses table-driven routing in steady states, and reactive routing when links fail. This means that when a route fails, C2HR searches for an alternative path through which to route packets. When the routing re-converges, *i.e.,* after the new lookup tables are computed, routing resumes using the proactive component. Another feature of C2HR is that it is an energy-efficient protocol which aims at prolonging the network lifetime. It achieves this by selecting the most energy-efficient paths as viewed by the collective of nodes rather than just the source-destination pair. Given that PANs are usually charaterised by small to medium size networks (less than 100 nodes), the overhead incurred due to topology maintenance is not subtantial enough to impair the performance of the protocol. Moreover, the reactive element of C2HR allows the transmission of data which would not otherwise be possible in purely proactive protocols in the case of link failures.

In C2HR, a network leader (coordinator) is responsible for collecting, building and distributing topology information, so that each node has a global view of the network, from which the routing tables are generated. The network leader is also responsible for carrying out route maintenance functions. Relying on a central node is a sensible choice for PANs [122] as it complies with the low complexity requirements and scarce resources in this type of network, wherein heavy computations are delegated to a more capable node.

7.2 Basic Operations

C2HR is a hybrid routing protocol in the sense that it uses both proactive and reactive routing components. When a source node, s, wishes to route a packet to a remote destination, d, it uses its local look-up table to forward the packet to the corresponding next hop. Each intermediate node that receives the packet, forwards it through to the appropriate next hop until the destination is reached. When a link

along the path fails, the detecting node notifies the source so that the latter initiates a route search, by sending a `Route Search` packet to look for an alternative path to the destination. Once the path to the destination is found, a `Route Found` message is sent in the reverse direction. After the routing re-converges, *i.e.,* when the lookup tables are re-computed, routing resumes using the proactive component. To illustrate this idea, consider the scenario shown in Figure 7.1, where solid black nodes are masters, white nodes are pure slaves and the shaded nodes are PMPs.



(a) proactive routing (b) topology change (c) route search

(d) route found (e) reactive routing (f) re-convergence - proactive routing

Figure 7.1: C2HR Basic Operations

When node I (source node) needs to send a packet to node F (destination node), it uses the lookup table stored locally to find the next hop node that corresponds to that particular destination, which in this case is node A. Similarly, upon receiving the packet, node A forwards it to its next hop along the path, *i.e.,* node C, and so forth. The full path between I and F is thus: I $\rightarrow$ A $\rightarrow$ C $\rightarrow$ K $\rightarrow$ D $\rightarrow$ F. This is depicted in Figure 7.1(a). If node K suddenly becomes unavailable due to its departure or battery drainage, the route from I to F fails (Figure 7.1(b)). At this point, nodes I and L (the network leader) are notified of the failure so that I initiates a route search and L readjusts the topology matrix (Figure 7.1(c)). As illustrated in Figure 7.1(d), once the *Route Search* packet reaches the target node, *i.e.,* node F, the latter sends a *Route Found* message back to the source, so that packets can follow the reactive path: I $\rightarrow$ A $\rightarrow$ C $\rightarrow$ F, while waiting for the routing to re-converge (Figure 7.1(e)). After recalculating the new lookup tables, routing resumes on the proactive path, which in this case is I $\rightarrow$ L $\rightarrow$ C $\rightarrow$ F, as depicted in Figure 7.1(f).

7.3 C2HR Routing

The proposed routing protocol is made of two components: proactive routing and reactive routing. In the proactive part, routes are established beforehand in the form of routing table entries; whereas in the reactive routing, routes are established on-demand when the proactive paths fail. The advantage of C2HR is that when a node suspects a route failure, it can immediately look for an alternative route without having to wait for the failure to propagate to the entire network and new routes to be recalculated. This is particularly advantageous when route failures are transient or when failure detection/notification is suppressed for long periods. In wireless networks, using fast re-route algorithms via alternate paths [95, 96, 126] is not an option, as nodes move frequently and therefore back up paths may not always be readily available. Moreover, in pure link state protocols, routing can only resume after the network recovers from the failure. This shortfall is overcome in C2HR by introducing on-demand routing when the link-state protocol fails.

7.3.1 Proactive Routing

The proactive routing component of C2HR, is similar to OLSR [70], in that each node builds a routing table based on a topological view of the network. Each node maintains a next hop entry to each destination as well as the associated total cost of the path, generated using Dijkstra's shortest path algorithm [52]. Since the algorithm is designed to suit low power personal devices, links are selected in a way that minimises energy consumption while also minimising the number of hops. In wireless communications, however, sending data through a least-hop path, implies an increase in the transmission range, which in turn, means an increase in power consumption [53]. On the other hand, relying on longer paths may deteriorate the quality of service. There is, therefore, a trade-off between energy conservation and acceptable path lengths that a packet should follow. The length of a path considered to be "acceptable" differs from one QoS class to another; this information can be extracted from the application and used to set a threshold for the maximum number of hops that a packet is allowed to traverse from source to destination. The idea being to use suboptimal routes in order to save energy and prolong the lifetime of the network, while ensuring the delay introduced by multi-hop routing remains at a tolerable level.

7.3.1.1 Link Costs

The cost of the link between node i and node j (denoted as C_{ij}) indicates the energy required to send the data via that link. This metric depends on a number of parameters, namely transmission energy (E_{tx_i}), residual energy (E_{r_i}) and channel bit error rate (ber_{ij}). E_{tx_i} denotes the energy required by node i to transmit a single data unit, which is a function of the distance between i and j; E_{r_i} indicates the battery power remaining at i, and ber_{ij} gives an indication of the number of times a packet needs to be retransmitted in order to ensure it is successfully received. Sending the same packet r times requires a transmission energy of $r.E_{tx_i}$, therefore ber_{ij} is an important parameter when computing the energy

required to forward data on the link ij.

As noted in [33], the energy metric varies with time and as nodes become more involved in network activities, their residual energy decreases. When all nodes have a high level of residual energy, it is preferable to route via the paths which incur the least transmission energy. However, when the battery power of some nodes falls below a critical level, it is advisable not to include them in the routing path. This achieved by increasing the link cost associated with sending data through those nodes. A link cost function that satisfies this paradigm is expressed as:

$$C_{ij} = a.E_{tx_i}^{\alpha} b.E_{r_i}^{\theta} + \phi.ber \tag{7.1}$$

where α and θ are binary factors, and a, b and ϕ are normalising factors. If α is 0 and the channel's bit error rate is negligible, the deciding factor would be the residual energy. Similarly, if θ is 0, then routes are chosen so that transmission energy is minimised.

All of the required parameters for computing the link cost are readily available to devices from local measurements. E_{tx} is extracted from the transmit power level set through LMP, E_r is obtained by monitoring battery level, and the bit error rates can be measured as described in Chapter 4

7.3.1.2 Topology View Generation

When nodes have no knowledge of the network topology, the leader builds a tree rooted at itself by sending a topology query message, to each node in the connected component, which is relayed from parent to child until it reaches the leaves in a tree expansion process. If a node receives a duplicate query message from another parent, it signals that it is already connected to a parent so that it is removed from the list of children, thus ensuring a node only has a single parent. When the message reaches a leaf node, the latter generates a query reply packet containing, in addition to its address and residual energy, its *adjacency information table*, which includes the node's one-hop neighbours, together with their associated transmit powers and bit error rates. Table 7.1 gives an example of the adjacency information table stored at node 4 in Figure 5.1(a).

Neighbour Count	Address	E_{tx}	ber
3	8	$E_{tx}^{4,8}$	$ber_{4,8}$
	9	$E_{tx}^{4,9}$	$ber_{4,9}$
	10	$E_{tx}^{4,10}$	$ber_{4,10}$

Table 7.1: Adjacency Information Table

Each non leaf node receiving the topology reply message forwards it up the tree, after appending its information, until the message returns to the leader. In order to ensure that the topology query message reaches every node in the connected component, a node does not send the topology reply packet unless it is either a leaf node or it receives a reply from all of its children within a pre-defined timeout. The

format of the topology query and topology reply packets are shown in Figure 7.2 and 7.3 respectively.

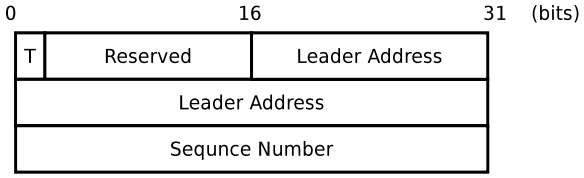


Figure 7.2: Topology Query Packet Format

The different fields making up the topology query message are described below:

- **T**: This field identifies the type of topology view generation message. In the case of topology query, it is set to 0.
- **Leader Address**: This is the 48-bit BD_ADDR of the leader.
- **Sequence Number**: The sequence number is a monotonically increasing variable that is incremented by 1 each time the leader initiates a topology query message.

The pair <Leader Address, Sequence Number> uniquely identifies a topology query message, such that if the leader's identity changes, the new leader can initiate its own topology view tree if required.

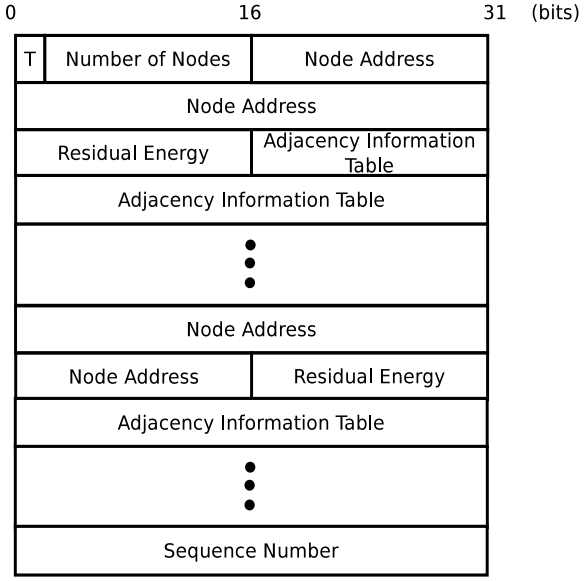


Figure 7.3: Topology Response Message Format

Figure 7.3 indicates that the topology reply message carries the entire topology information of the network, which is relayed from child to parent until it reaches the root, *i.e.,* the leader. As the number of nodes increases, so does the size of the reply packet, which means that for large networks, the topology reply message needs to be transported in several baseband packets. This method incurs the same overhead as relying on individual replies; however, nodes do not need to continuously relay the

information, therefore nodes can enter one of the power saving modes while waiting for the message to reach them, thereby saving energy.

Once the leader gathers the necessary information from all the nodes, it builds an $N \times N$ connectivity matrix, where N is the number of nodes, whose (i, j) entry represents the link cost between i and j as described by Equation 7.1, then distributes this matrix along the tree so that nodes can generate their routing tables using Dijkstra's shortest path algorithm.

7.3.1.3 Topology Maintenance

In an ad hoc network, energy and channel conditions vary with time, meaning that link costs also vary with time. Nodes need, therefore, to update the leader with these variations so that all routes are up-to-date. For this purpose, the leader generates a topology query message every T_u time slots and repeats the tree expansion/shrinking process as described above. Furthermore, in order to maintain a proximity view, each node exchanges `hello` beacons with its immediate neighbours at regular intervals. In addition to signalling their presence, nodes use the `hello` messages to convey other information such as a change in residual energy or the estimated channel error rates. These packets are transported as DM1 packets and include the address of the originator as well as parameter updates. In Bluetooth PANs, the beacons are exchanged when the master polls its slaves or the updates can be obtained directly from the low power mode beaconing mechanism, both of which can be easily implemented in LMP. When a node fails to hear the beacon from one of its adjacent nodes after a predefined timeout, it assumes that the link is lost and informs the leader so that the connectivity matrix can be updated and the updates redistributed. If topological changes occur while data is being routed, the detecting node informs the leader and the source node, so that the latter can look for an alternative route by initiating a route search procedure.

7.3.2 Reactive Routing

The reactive component of C2HR is used when proactive routes fail, so that routing can resume on-demand while the routing re-converges. As illustrated in Figure 7.1, when a node detects a link failure, it informs the sender and the leader by sending a `Link Error` message, in order to enable the source to look for an alternative route. Once the route to the destination is found, packets are sent along the new path unless explicitly instructed otherwise. Nodes need to know whether a packet should be routed via the primary proactive path or via the alternate on-demand one. For this purpose, packets are marked with a single bit to indicate which route an intermediate node must forward the packet to. Bluetooth headers do not provide room for packet marking, therefore, this marking must be included in the IP header.

7.3.2.1 Route Search

As soon as a source node is notified of a link failure after receiving a `Link Error` message, it generates a `Route Search` packet including the address of the target node and broadcasts the information to its neighbours. This packet is flooded throughout the network until it reaches the destination. When a node

receives a duplicate of the packet, it simply discards it. Route search is carried in a BNEP control packet as illustrated in Figure 7.4

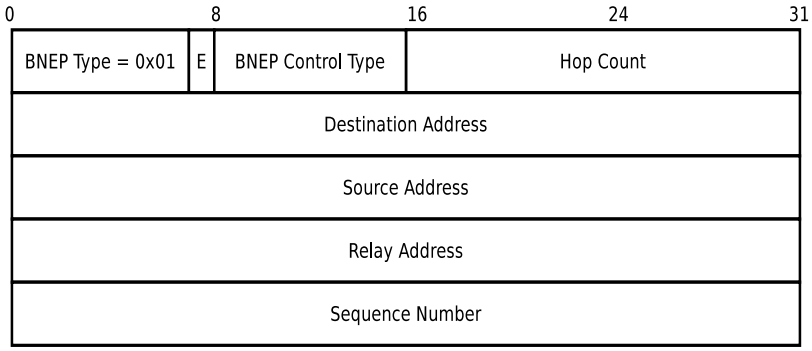


Figure 7.4: Route Search Packet Format

The field `BNEP Type` carries the value 0x01 to indicate that it is a control packet, and the value used for the `BNEP Control Type` field is 0x07, as it is unallocated by the specifications [20].

- **Hop Count**: This counter is incremented each time a node forwards the `Route Search` packet.
- **Destination Address**: This field indicates the address of the target node.
- **Source Address** This is the address of the source node, *i.e.,* the node that initiated the route search.
- **Relay Address**: This is the address of the current node relaying the Route Search packet. It is used to set up the reverse path in the Route Found phase.
- **Sequence Number**: This is a monotonically increasing number, which together with the source address, uniquely identifies a Route Search packet.

Each node receiving the Route Search packet, checks the destination address, and if it does not match its own, it rebroadcasts the packet, provided it has not previously received the same packet. This is achieved by comparing the pair <`Source Address`, `Sequence Number`> stored in the cache with that of of the received packet. Before re-broadcasting the packet, a node stores the `Relay Address` of the packet which presents the least number of hops to point to its predecessor and replaces that field with its own address to indicate to the next node that it is, itself, a possible predecessor. In Bluetooth, however, slave nodes do not have broadcast capabilities, therefore a master collects the Route Search packet from the first slave that received it and rebroadcasts it within the piconet. The message propagates when a PMP node connects to another piconet, repeating the same broadcast procedure until it reaches the target destination.

7.3.2.2 Route Found

When the destination node receives the Route Search packet, it replies by sending a `Route Found` packet following the reverse path as set by the predecessor pointers (*i.e.,* `Relay Addresses`) during

the route search phase. Because reactive routing is only used temporarily, *i.e.,* when the proactive component fails, nodes do not cache previous routes, thus a `Route Found` packet is only sent once the search reaches the destination. Moreover, the proactive routing table shall not be used to complete the route to the destination, as this creates routing inconsistencies; therefore a node shall not send back a `Route Found` message unless it is the target node. Another reason for not caching previous routes, is to avoid reply storms when various nodes which have routes to the destination send back a reply message. `Route Found` is also sent as a BNEP control packet and its format is given in Figure 7.5.

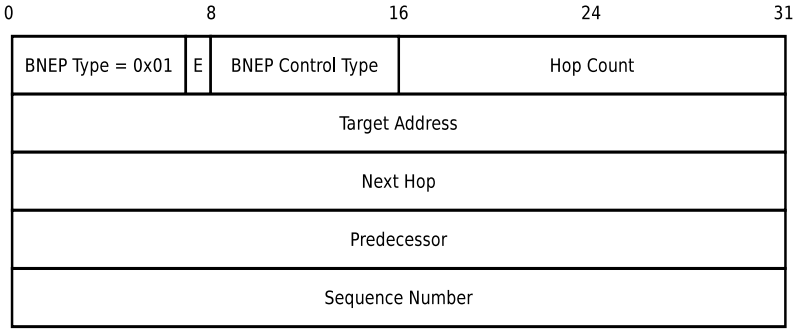


Figure 7.5: Route Found Packet Format

As mentioned above, `Route Found` messages are carried over BNEP control packets, therefore the `BNEP Type` field holds the value 0x01. `BNEP Control Type` indicates the purpose of the control packet and carries the value 0x08, as it is unused in the specifications. The other fields making up the `Route Found` packet are as described below:

- **Hop Count**: This field carries the number of hops left to traverse until the packet reaches the initiator of the route search. It is decremented each time it is forwarded on the reverse route.
- **Target Address**: This is the address of the initiator, *i.e.,* the source node in the reactive routing.
- **Next Hop**: This the address of the node relaying the `Route Found` packet.
- **Predecessor**: This field represents the address of the predecessor along the reverse path, *i.e.,* the address of the next node to receive the reply packet.
- **Sequence Number**: The Sequence Number is increased each time the target node generates a new `Route Found` message. The pair <`Target Address`,`Sequence Number`> uniquely identifies a `Route Found` packet.

The destination might receive a number of `Route Search` packets from different adjacent nodes; in this case it selects the route that presents the least number of hops and forwards the reply packet to its predecessor. Each intermediate node receiving the `Route Found` packet, stores its predecessor as the next hop address by replacing the address stored in `Predecessor` with its cached predecessor's

address, then forwards the packet. If the reactive route fails before the proactive route is repaired, the detecting node sends a `Link Error` message to the source node so that the latter initiates a new route search procedure.

7.4 Support for Multicast Routing

In many instances, a node may need to send the same information to a number of nodes belonging to a certain group (multicast group). One solution is to have the source node send the same packet to each node individually using either the proactive or the reactive routing component of C2HR. Obviously, this approach is wasteful as a node that is part of the multicast group may receive the packet numerous times if it is part of the route. Attractive alternatives are to use trees to send the data to the target recipients or to use a mechanism that can piggyback the data over network-wide control messages. In addition, for a node to join a multicast group, it requires prior permission from a network authority through a group management protocol such as the *Internet Group Management Protocol* (IGMP) [162].

C2HR exhibits desirable characteristics for supporting multicast routing, without the need for any modification to the routing protocol. The network leader can act as a group management authority, as it has a global view of the network and is universally known by every node in the network. Therefore, when a node wishes to join a group, it sends a request to the leader, which then assigns a multicast address if the subscription conditions are met. Moreover, as part of its routing functions, the network leader maintains an up-to-date tree to all nodes within the network, hence, sending the same information to a group of nodes becomes a simple task, as it merely requires forwarding along the tree. If the tree fails due to topological changes, the multicast information can be carried along the `Route Search` packets so that members of the multicast group can receive it. The latter method is less efficient, however, as reactive routing is a transient state, it is not of major concern. Although the use of shared trees for multicast communications over wireless networks has been criticised [49], in the case of PANs, relying on trees can be advantageous owing to the simplicity they offer [173], thus maintaining energy-efficiency.

7.5 Evaluation

C2HR is simulated on Bluetooth scatternets, generated using the network formation algorithm described in Chapter 5, for various performance metrics and under a number of assumptions of node mobility and network sizes. Nodes are uniformly scattered around an indoor space of dimensions $L \times W \times H$, where $L = W$ and varies between 25 meters and 45 meters; H is taken to be 2.4 meters (standard ceiling height).

7.5.1 Assumptions

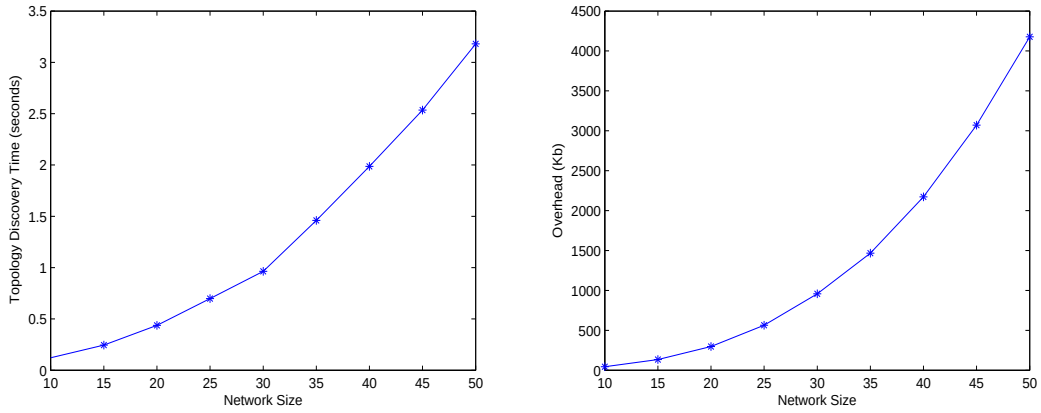
Before embarking on a detailed evaluation of C2HR, certain assumptions should be taken into consideration:

- Each node is identified by a unique identifier: BD_ADDR or a local IP address.
- Devices are assumed to know the identities of their neighbouring nodes, which are obtained during link set-up.
- Nodes can move arbitrarily in any direction and with varying speeds.
- Nodes may join or leave the network at any given time, hence causing network partitioning and mergers.
- Delays and overheads introduced by link establishment are not taken into account.
- A node may change its role to satisfy the network formation constraints, either because it moved or because it acquired new links.

7.5.2 Topology Discovery

When a network is initially powered on and a new leader is elected, it builds a tree routed at itself and sends out topology query messages to which the nodes respond by sending back their topology information. Naturally, this process incurs overhead and delay; however, this phase is only required when nodes do not have any knowledge of the network, as any subsequent changes (either topological or the election of a new leader), only result in updates to the connectivity matrix available to the nodes. It is therefore safe to assume that topology discovery would not impair the performance of the routing protocol. Figure 7.6 shows the average simulation results, over 100 runs, for the time and data overhead incurred during the topology discovery phase. In the simulation, nodes are immobile and the intra-piconet scheduling employed is exhaustive round robin, wherein a master polls the next slave, in sequence, only after the current one has sent all of its topology information. Furthermore, the baseband packet types used to transport the information are DHi, where i is chosen such that sending x amount of data requires the minimum number of time slots.

In Figure 7.6, data overhead refers the extra data sent during the topology discovery phase, including dissemination of the connectivity matrix, and excluding any other user information. The delay is the time interval from when the leader initiates the discovery process until every node receives the topology view from the leader, *i.e.,* expanding and shrinking the tree, and distributing the connectivity matrix. By looking at Figures 7.6(a) and 7.6(b), it is apparent that delay and data overhead follow an exponential relationship with network size, which is expected, as the size of teh connectivity matrix is the square of the number of nodes. In addition, there is a strong correlation between the data overhead and completion time. This is due to the fact that Bluetooth relies on time-division transmission schemes; therefore the delay increases as the amount of data increases.



(a) Completion Time (b) Data Overhead

Figure 7.6: Topology Discovery Overheads

7.5.3 Route Recovery

In mobile environments, it is common for multiple topological changes to occur concurrently as nodes move, thus several links are lost and others are established. Therefore, the protocol requires a mechanism to propagate simultaneous changes throughout the network. When a node detects a topological change, either the removal or addition of a link, it reports it to the leader, so that the latter can redistribute the updates. In order to cater for multiple changes, nodes that receive various notification messages, combine the information and forward it on. After the leader receives the notifications and updates all the nodes with the changes observed, each node can re-compute its routing table according to the new connectivity matrix. The time it takes to propagate topological changes to the entire network and to re-compute the routing tables is given in Figure 7.7, which plots the proactive route recovery delay as a function of the number of changes.

In this instance, only link failures are considered; however, the creation of new links has exactly the same effect. The simulation is performed on randomly generated networks comprised of 50 nodes, using the attachment mechanism described in chapter 5. The delay is measured from the instant a node detects a topology change until the time the last node re-computes its routing table, *i.e.,* until the routing re-converges. It is assumed that multiple nodes detect the changes simultaneously, and immediately start the notification process. If a node receives notification of multiple failures, it combines the information and forwards it on. When the leader receives notification of a link failure, it sets the cost of that link to *infinity* and sends out an update in the form of connectivity matrix coordinates. If the change consists of the addition of links, the leader sends out the coordinates with the associated newly computed link costs. It can be seen from Figure 7.7 that the time it takes for the routing to re-converge increases linearly with the number of simultaneous failures. This linear relationship is the result of the overhead associated with updating the connectivity matrix, as an increase in data overhead translates into longer delays.

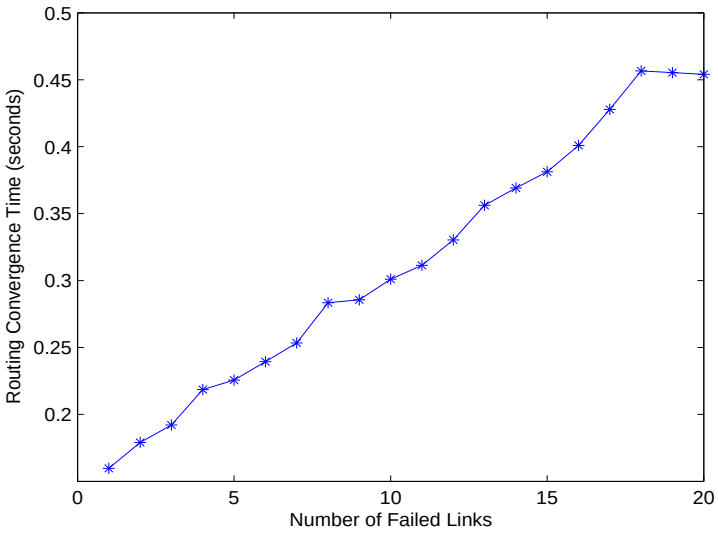


Figure 7.7: Proactive Route Recovery Delay

In the simulation, propagation and buffering delays are not taken into account, therefore the time it takes for the routing to re-converge after the detection of a topological change would be greater than the that given in Figure 7.7. In addition, it is assumed that failures are instantaneously and simultaneously detected, which in reality is not the case. A way of reducing route convergence time is to broadcast route updates frequently; however, this method is wasteful in terms of bandwidth and energy [158], hence the need for an on-demand mechanism, which provides a level of resilience while the routing re-converges. Figure 7.8 illustrates the route repair time of the on-demand component of C2HR, wherein the time represents the delay from the instant the source node receives a failure notification until the moment the source node receives the `Route Found` packet. In this scenario, several links may fail simultaneously as a result of node mobility, and the plot only concerns a single source-destination pair.

It can be seen from Figure 7.8 that the delay for a source node to find an alternative route reactively is well below the time it takes for the routing to re-converge after links fail, even for a small number of simultaneous failures. This shows the benefit of C2HR over traditional pure link state routing protocols, wherein, packets sent during routing re-convergence are lost, whereas in C2HR this gap is tolerable. C2HR's reactive component may not be optimal in terms of energy efficiency; however, because it is only temporary, the performance of the protocol is not impaired. On the contrary, it offers alternative routing paths for a period, during which, routing is not possible otherwise.

7.5.4 Excess Cost

The reactive routing component of C2HR does not guarantee the most energy-efficient path, as the aim is to quickly find an alternative path, while the primary proactive paths are re-computed. It is important, however, to analyse the deviation from the optimal (least cost) paths, which the proactive component

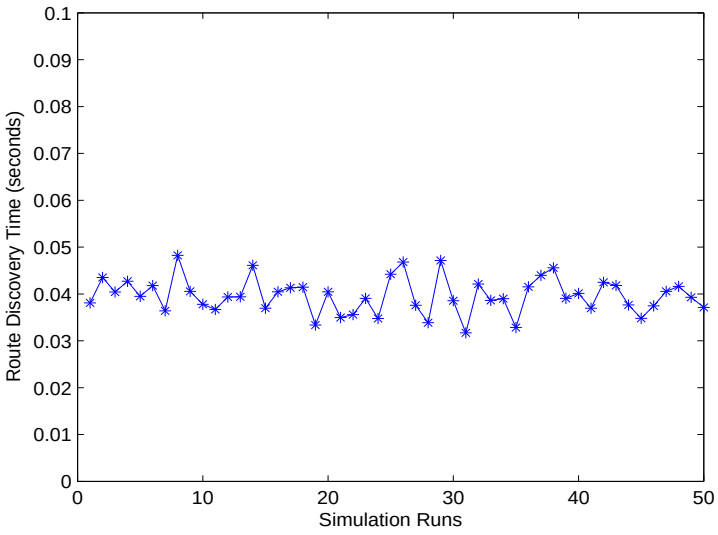


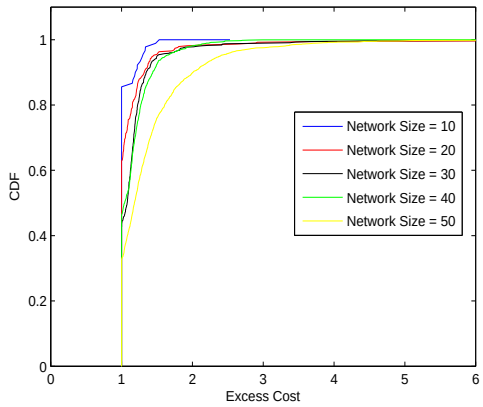
Figure 7.8: On-deman Route Repair Delay

guarantees. This measure is referred to as the *Excess Cost* and is measured as the ratio of the total cost of the reactive path to the total cost of the proactive path after re-convergence. If, for a given source-destination pair, the cost of routing through the on-demand path is C_{rs}, and the total proactive path cost after re-convergence is C_{rt} then the excess cost, C_{ex}, is given as:

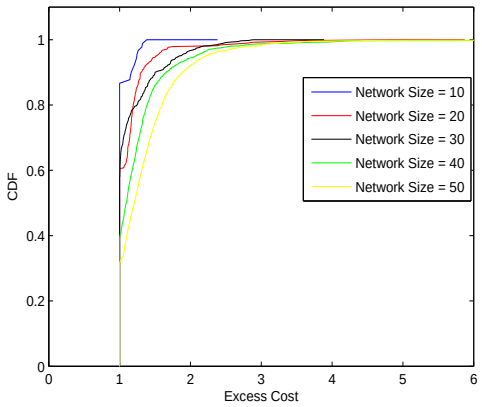
$$C_{ex} = \frac{C_{rs}}{C_{rt}} \tag{7.2}$$

It should be noted that $C_{rs} \geq C_{rt}$, therefore $C_{ex} \geq 1$. Figure 7.9 shows the excess costs for various assumptions of node mobility and network sizes.

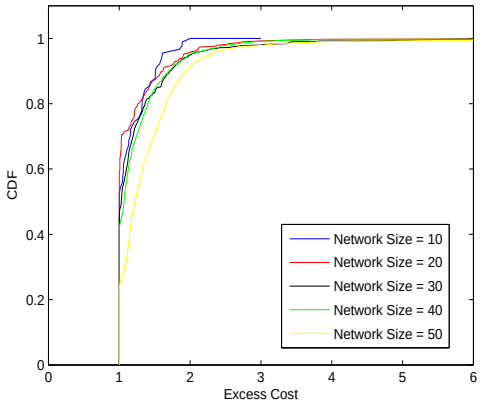
Figure 7.9 represents the cumulative occurrence of excess costs for various network sizes and under different assumptions of node mobility. It can be seen that as the size of the network increases, the number of suboptimal alternative routes also increases. This is an expected result; the larger the network, the greater the number of links, which in turn results in an increase in the probability of selecting a suboptimal reactive path. Similarly, the increase in the number of simultaneous mobile nodes translates into an increase in the number of link removals/additions, which in turn results in fewer optimal alternate paths, as observed in Figure 7.9. A notable observation is that for small network sizes and low mobility, the portion of optimal alternate paths is high (over 80%), which suggests that for PANs the on-demand component of C2HR does not significantly affect the energy conservation characteristic of the routing protocol.



(a) Number of Mobile Nodes = 1



(b) Number of Mobile Nodes = 2



(c) Number of Mobile Nodes = 3

Figure 7.9: Excess Costs

7.5.5 Throughput

The basic aim of every routing protocol for mobile ad hoc networks is to successfully deliver data from a particular source to a particular destination while dealing with topological changes. However, the longer the paths, the less the throughput as the data need to traverse a number of hops, which introduces transmission and buffering delays, thus resulting in lower throughput. Moreover, when a path from the source to the destination is not available, no data can be transmitted until the route is repaired, which, in turn introduces extra delays. It is, therefore, crucial to assess C2HR's behaviour in terms of throughput as it indicates its ability to successfully deliver data and recover from failures, *i.e.,* it indicates its employability as a routing protocol.

Figure 7.10 shows the throughput response of C2HR exhibited by different baseband packet types for a single source-destination pair, selected at random amongst a network of 40 nodes.

In addition to the assumptions set out in section 7.5.1, the simulation herein assumes that the following also holds true:

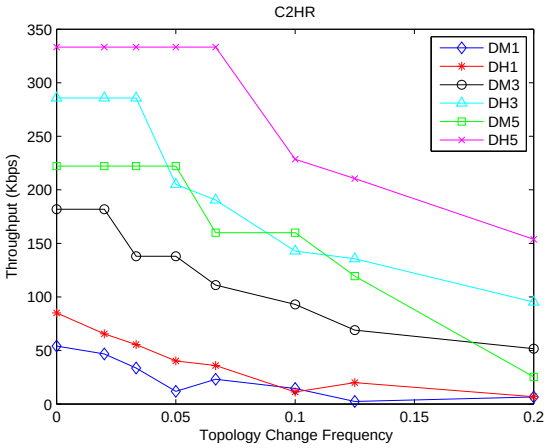
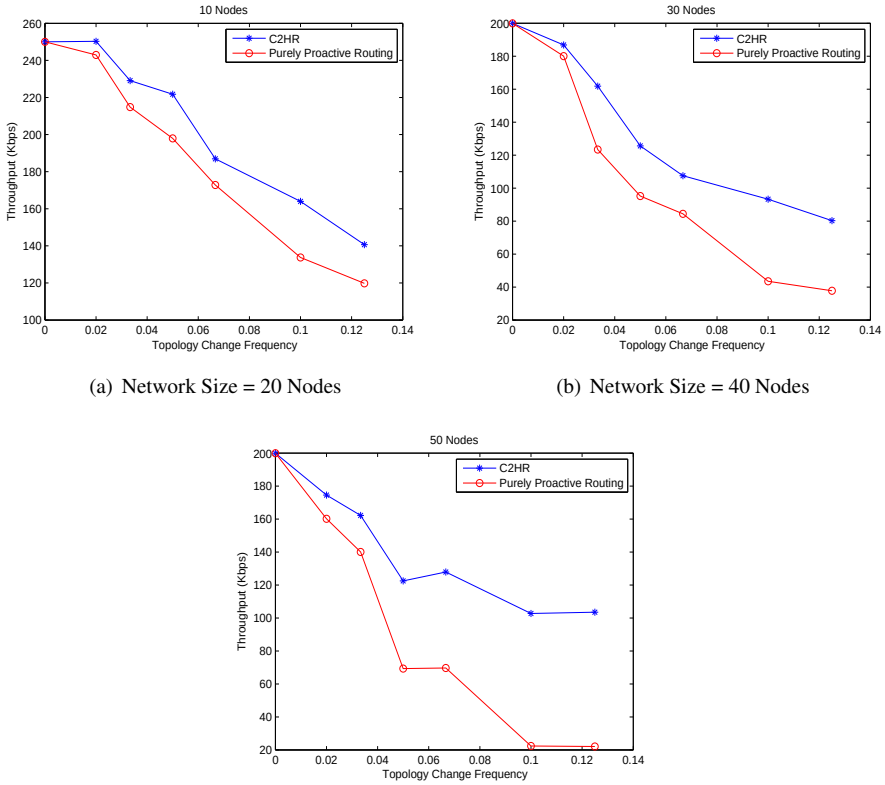


Figure 7.10: C2HR Throughput Response

- Links are uncongested and error-free.
- Only a single node moves at a time, resulting in multiple simultaneous link failures.
- Masters poll their slaves according to the Exhaustive Round Robin scheduling scheme.

When the network is static, the throughput is only affected by the path length, wherein each hop introduces master-slave or slave-master transmission delay, with DH5 packets presenting the highest throughput. However, as the frequency of topological changes increases, the throughput decreases. This is the result of the delay introduced by the failure detection/notification and the route search/reply phases. Here, the frequency of topological changes is the inverse of the mobility pause time, which indicates the frequency at which an arbitrary node changes location in the network. As can be seen from the figure above, the throughput may be higher at higher mobility frequencies. This is due to the fact that mobility may bring the destination closer to the source (in terms of number of hops), thus resulting in less transmission delay. A general trend, nonetheless, is that the higher the mobility the lower the throughput, however, as demonstrated earlier, the time it takes for the proactive routing to re-converge after a failure is higher than the delay introduced by the reactive component of C2HR. Therefore, it is expected that the throughput resulting from employing C2HR will be higher than a purely proactive routing protocol. This is demonstrated in Figure 7.11, which compares the throughput resulting from employing the two routing methods for an arbitrary source-destination pair when DM3 packets are used.

Figure 7.11 indicates that the difference in throughput between C2HR and the purely proactive routing protocol increases as the number of nodes in the network rises. This difference is the result of the increased computational complexity when re-computing the routing tables, *i.e.,* applying Dijkstra's algorithm, which is known to have a complexity of $O(|E|log|V|)$ [14], where $|E|$ is the number of edges and $|V|$ is the number of vertices in the network. Generally, however, C2HR presents better throughput



(a) Network Size = 20 Nodes

(b) Network Size = 40 Nodes

(c) Network Size = 50 Nodes

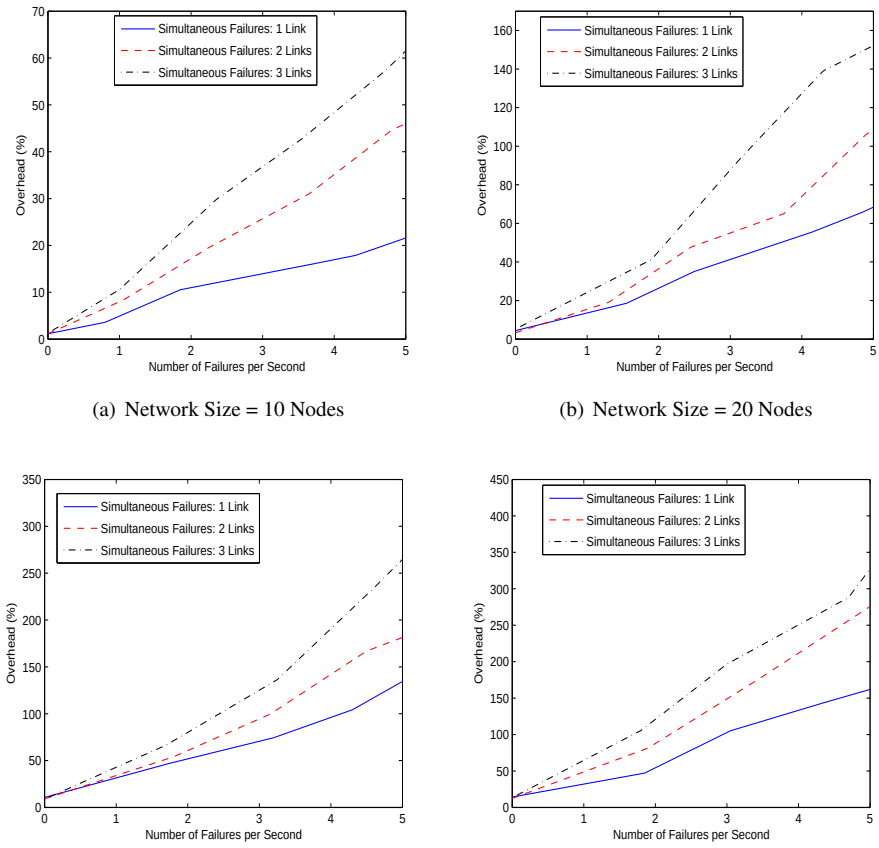
Figure 7.11: Throughput: Comparison between C2HR and Purely Proactive Routing

response; this is because C2HR allows data to be transferred when the proactive routing tables are not available.

7.5.6 Routing Overhead

Routing overhead is the "extra" information introduced by the routing mechanism, including signalling and retransmission of lost data due to route failures. Proactive routing has been criticised for incurring large overheads caused by regular topology updates. However, for small to medium size networks, which is the case for PANs, the routing overhead does not impair the performance of the protocol. Figure 7.12 shows the overhead incurred by C2HR as a function of topology change frequency. The overhead is taken to be the ratio of the extra data to the total data sent.

As can be seen from Figure 7.12, the overhead increases linearly with failure frequency. Moreover, as the size of the network increases so does the overhead. This is because an increase in network size results in longer paths, consequently incurring larger transmission overhead. Furthermore, the overhead is augmented by a greater number of hello messages associated with an increase in the number of nodes. The simulations are performed for very frequent failures in order to illustrate the relationship between the overhead and failure frequency. In reality, however, failures occur at much lower rates, which means that the overhead would be much lower than that given in Figure 7.12. The overhead presented by C2HR is



(a) Network Size = 10 Nodes

(b) Network Size = 20 Nodes

(c) Network Size = 40 Nodes

(d) Network Size = 50 Nodes

Figure 7.12: C2HR Overhead Response

higher than that of a purely reactive or purely proactive protocol, as it combines both; however, because PANs are characterised by a small to medium number of nodes, this difference is not significant as far as the overall performance is concerned, given that the proposed routing protocol has the advantage of improving throughput and network longevity.

7.5.7 Energy Efficiency

The aim of C2HR is to preserve the residual energy of devices, and thus to prolong the path lifetime. The lifetime is taken to be the duration from initial path set-up until the route from the source node to the destination is no longer available. At bootstrap, each node i is assigned a value of residual energy, E_{r_i}, which is decreased as the node is involved in data forwarding activities. It is assumed that a node can be in one of three states: `ACTIVE`, `INACTIVE` and `IDLE`. In the `ACTIVE` state a node has either its transmitter or receiver switched on. A node in the `INACTIVE` state is switched off, and in the `IDLE` state, a node only switches on its transceiver for periodic exchanges of control information. The energy

required to send a packet from node i to node j is denoted as E_{tx}^{i} and is defined as:

$$E_{tx}^{i} = s_p \times (\eta.E_b.d_{ij}^{\alpha} + E_a) \tag{7.3}$$

Where s_p is the size of the packet, η is a scaling factor, E_b is the energy required to send 1 bit (due to circuitry), which assumed to be constant and unique for every node, d is the transmission distance, α is the loss factor ($\alpha = 2$ in free-space loss model) and E_a is the internal energy consumed by a node during the `ACTIVE` state.

The energy required for node j to receive a packet, p, from node i is denoted as E_{rx}^{j} and is defined as:

$$E_{rx}^{j} = s_p \times (E_{lna} + E_a) \tag{7.4}$$

Where E_{lna} is the energy required by the low noise amplifier at the receiver to process 1 bit, which is assumed to be constant and the same for all nodes. The energy required to send a packet from source to destination via path $\wp$ is thus given by:

$$E_{\wp}^{p} = \sum_{i \in \wp} (E_{tx}^{i} + E_{rx}^{i}) \tag{7.5}$$

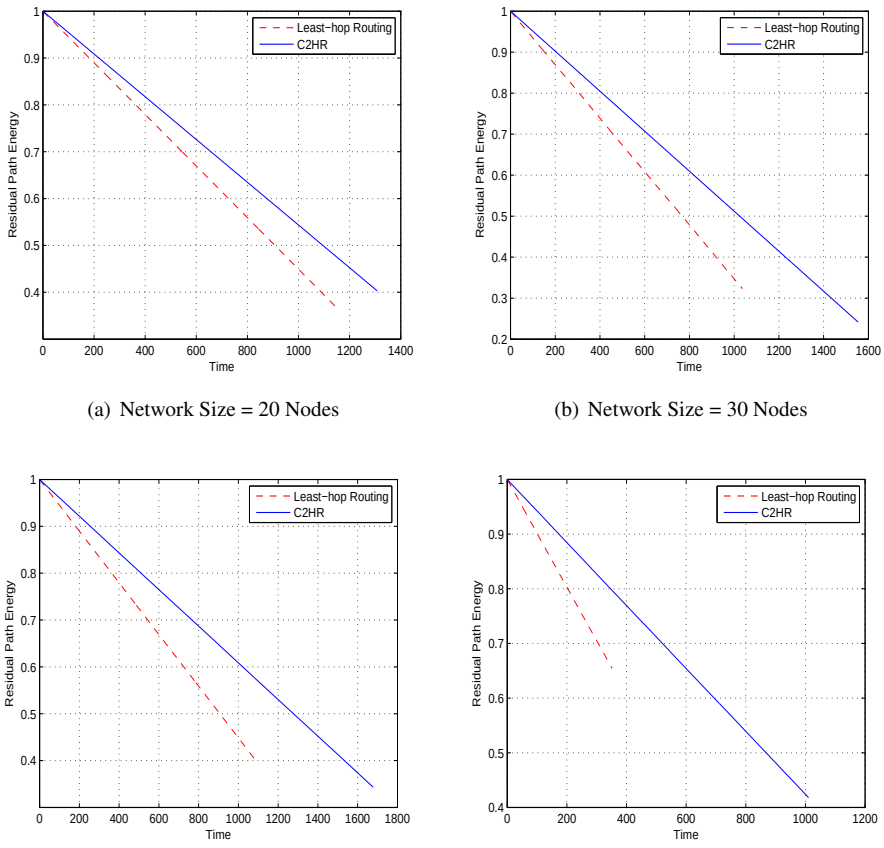
Similarly, the residual energy $E_{res}^{\wp}$, of a path, $\wp$ is defined as the sum of the residual energies of all the nodes along $\wp$. If the initial residual energy at node i is E_{r_i}, then the path's residual energy is given as:

$$E_{res}^{\wp} = \left(\sum_{i \in \wp} E_{r_i} \right) - N \times E_{\wp}^{p} \tag{7.6}$$

Where N is the number of packets.

Assuming that a source node sends a stream of information at a constant bit rate (DM1 packets) and that nodes are static, the residual path energy (expressed as a percentage) for an arbitrary source-destination pair is given in Figure 7.13.

The results given in Figure 7.13 compare the residual energy response of C2HR with that of least hop routing for an arbitrary path. In the latter, routes are chosen such that the number of hops between source and destination is minimal, whereas the former selects paths such that the residual path energy is maximised. As can be seen, the difference in energy increases as the size of the network increases. This is due to the fact that larger networks present longer paths, thus increasing energy consumption. In all cases, nonetheless, the path lifetime is larger for C2HR than it is for least hops routing, which suggest its suitability for power-constrained personal devices.



(a) Network Size = 20 Nodes

(b) Network Size = 30 Nodes

(c) Network Size = 50 Nodes

(d) Network Size = 80 Nodes

Figure 7.13: Energy Efficiency of C2HR

7.6 Summary

This chapter presented a novel hybrid routing protocol for personal area networks that aims at maximizing the network lifetime, by preserving the residual energy at each node along the routing path. The proposed Centrally Coordinated Hybrid Routing (C2HR) protocol uses proactive routing when the network is stable *i.e.,* in the absence of failures, and when a route fails it searches for an alternative route reactively. It was shown by simulation that C2HR is suitable for low power personal devices, as it helps nodes to preserve their energy, while resulting in higher throughput and incurring acceptable overhead.

Chapter 8

Conclusions and Future Work

This chapter concludes the thesis by highlighting the contributions and findings of the work presented herein, as well as suggesting future work to be carried out in order to improve on the proposed solutions.

8.1 Conclusions

The aim of the present work is to improve forwarding mechanisms for PANs, taking into account the low complexity and low power requirements of personal devices. This thesis considers Bluetooth as the enabling technology for personal area networking, as it is readily available in most personal devices such as mobile phones, laptop computers and portable multimedia systems. However, the proposed solutions can be adapted to other technologies that are capable of internetworking devices in an ad hoc fashion without major modifications. The thesis focuses on two main areas, namely error control and routing, which are highly influential in the performance of PANs. An efficient error control scheme which alleviates the burden of heavy data coding is particularly important in personal area networking, as relying solely on forward error correction translates into increased complexity and reduced efficiency. The proposed Packet Error Correction (PEC) scheme satisfies the low complexity requirements of PANs while achieving good response in terms of error reduction and overhead; thereby improving point-to-point transmission. Furthermore, the proposed C2HR routing protocol aims at preserving the overall energy of the network by forwarding data along the most energy-efficient paths whilst improving on throughput by allowing reactive routing in the event of failures.

It was demonstrated via simulation that the proposed techniques and algorithms do indeed improve forwarding mechanisms for PANs. The evaluation of PEC shows a better response than the scheme recommended by the specifications and incurs far less overhead. Improving the per-hop transmission enhances the multi-hop forwarding operations of the overall network. Furthermore, C2HR provides a network-level improvement to the forwarding operations in PANs. The simulations show a better response in terms of throughput and energy efficiency than found in other routing protocols, while incurring satisfactorily low data overhead. It is therefore safe to say that the proposed solutions help to

improve forwarding mechanisms in personal area networks, taking their particular constraints into account. In addition, the simplicity of the proposed solutions, makes them suitable for personal devices, which are characterised by low complexity hardware, low memory and power-constraints.

8.2 Contributions

The work presented in this thesis includes a number of contributions to the field of ad hoc networking in general, and personal area networking in particular. These are categorised below.

8.2.1 Enhancing Transmission Reliability

The adaptive retransmission scheme proposed herein allows nodes to increase transmission reliability, without having to increase their transmission power or employ sophisticated coding schemes. By simply retransmitting a subset of the packets, PEC is able to reduce error rates considerably, thus saving overhead, buffer space and power. The proposed PEC scheme can be applied to any ad hoc network that is able to estimate channel error rates. There are many channel estimation techniques found in the literature; however, when nodes share common access codes or channel sequences, two communicating devices can estimate the error rate by simply monitoring this access code as demonstrated herein. This method allows an accurate estimate of the error rates that the channel exhibits in real time and on-line, without the need for training sequences or employing sophisticated predictors. Having an estimate of the channel's error conditions can also be useful in other applications such as adjusting transmission rates or selecting appropriate packet lengths.

8.2.2 Scatternet Formation

The proposed attachment method allows nodes to connect to a random number of neighbours, such that no node connects to more than seven neighbours. Moreover, the network can only admit slave/slave PMPs, as this simplifies the inter-piconet scheduling. The advantage of the proposed network formation algorithm is that it achieves quasi-optimal configuration, without having to rely on sophisticated computations. In addition, random attachment models have proved to be particularly suitable for small networks [152], thus their applicability to scatternet formation.

8.2.3 Network Size Estimation

It is shown in this thesis that it is possible to accurately estimate the size of mobile ad hoc networks through random tour or gossip-based aggregation techniques. In the former, a token is passed from node to node, collecting network information until it reaches the originator. Once the token is received by the initiator, it computes an estimate of the size and other characteristics of the underlying topology. In gossip-based aggregation, on the other hand, the size is estimated from an aggregate average, which is refined each time nodes communicate. In ad hoc networks, having an approximate size of the network can be highly beneficial, as it is not always feasible to maintain an accurate view of the topology, due to

the dynamic nature of such networks. Instead, some applications rely on an estimate of the number of nodes to devise appropriate mechanisms. In the peer-to-peer world, it is important for nodes to know an approximate size of the overlay in order to set query time outs or to approximate the available download bandwidth; this is also true for mobile peer-to-peer networks, wherein nodes may use their knowledge of the size of the network or an approximate count of nodes with desirable characteristics in order to take the appropriate course of action.

8.2.4 Leader Election

A simple but efficient cooperative leader election algorithm is proposed. The algorithm collects and distributes information locally until a unique leader in the network is elected. The advantage of the cooperative leader election algorithm, as opposed to tree-based schemes, is that the former incurs lower overhead and scales well with network size. In this work, leader election is essential for the routing protocol; however, there are many other instances in which leader election can be of use. A network leader could, for example, be used as a data sink, be a coordinator that assigns tasks to other nodes in the network, or carry out security management functions. The ability to elect a leader quickly, even when the topology changes frequently is therefore a particularly desirable feature.

8.2.5 Ad Hoc Routing

The proposed C2HR routing scheme belongs to the family of hybrid routing protocols, in the sense that it combines proactive and reactive routing. C2HR aims at preserving the overall network lifetime by routing through the most energy efficient routes. The novelty of C2HR is that routes can be repaired quickly by searching for an alternative path in the event that the primary proactive route fails. To the best of the author's knowledge, this is the first hybrid protocol that combines proactive and reactive components in this fashion. Although C2HR was designed for personal area networks, it can be used in any scenario in which network survivability and resilience is of concern. Examples of which include battlefield ad hoc networks and sensor networks.

8.3 Future Work

This thesis has contributed to the enhancement of forwarding mechanisms for personal area networks; however, there are certain areas in which further work could be conducted. The following sections highlight the areas that could be investigated.

8.3.1 Improving Transmission Through Network Coding

This thesis demonstrates that transmission can be improved via more efficient error control mechanisms; however, in wireless networks data throughput can be further enhanced by using network coding techniques introduced by the authors of [92]. The advantage of network coding can be illustrated with an example: consider two slaves that are exchanging data within a piconet via the master node. Tradi-

tionally, slave 1 sends its data to the master in an odd-numbered time slot if contacted in the previous even-numbered slot; the master then relays the information to slave 2. Similarly, slave 2 sends its packet to the master, which then forwards it to slave 1. Relying on this scheme means that the transmission takes 4 time slots. Alternatively, if network coding is used, slave 1 and slave 2 send their respective packets to the master, which then broadcasts a linear combination of the data, from which each slave can obtain the required data, thus resulting in 3 transmission slots.

In Bluetooth, however, broadcasts are not acknowledged, which means that in the event of high error rates, the master would send the same information several times in order to ensure successful delivery. In this case, using network coding would deteriorate throughput rather than improve it. In addition, if the data is arranged into single-slot packets, which correspond to the length of poll packets, then network coding does not improve the transmission rate, as data can be transmitted in the poll slot. On the other hand, using multi-slot packets may cause a significant drop in throughput due to retransmissions of lost or corrupted packets. It is, therefore, important to investigate the impact of using network coding on the transmission performance of PANs and to establish the situations in which network coding is beneficial.

8.3.2 Promoting Cooperation Amongst Nodes

It was assumed throughout the thesis that nodes cooperate in forwarding each other's traffic; nonetheless, this cannot always be guaranteed, even in personal area networks, because of the scarce resources available to nodes. An area of improvement is to design incentive mechanisms to promote cooperation amongst the nodes, taking into account the specific constraints of such networks.

There have been many proposals in this direction, which can be classified as either reputation-based or credit-based [71]. In the former, nodes are assigned reputation scores based on observed past behaviour and are served accordingly. Examples of reputation-based incentive mechanisms include *Cooperation Of Nodes: Fairness In Dynamic Ad hoc NeTworks* (CONFIDANT) [26], *COllaborative REputation* (CORE) [110] and *Distributed and Adaptive Reputation mechanism for WIreless ad hoc Networks* (DARWIN) [72]. Credit-based mechanisms, on the other hand, reward cooperative nodes with micro-payments, which can then be used to pay for their own data transfers. Buttyán *et. al.* introduced the concept of Nuglets [27, 28], which are virtual currency units used to reward nodes for being cooperative. Other proposed credit-based incentive schemes include: The *Simple cheat-Proof cRedIT-based systEm* (Sprite) [179], routing based on *Vickery-Clarke-Groves* (VCG) auctioning [10] and the *Hierarchical Routing in Resource Rationed* (HR^3) protocol proposed in [165]

Reputation-based schemes have been criticised for having to rely on broadcasts and eavesdropping which cannot always be achieved, especially if radio signal transmissions are directional. This issue is further accentuated in Bluetooth-based PANs, wherein slaves cannot hear each others transmissions because of the time-division polling approach. On the other hand, in credit-based schemes nodes need to

present some type of proof in the form of a receipt showing that they have accomplished the required task. However, in order to prevent false claims, credit-based mechanisms rely on cryptographic techniques and may require tamper-proof hardware, which maybe impractical in personal area networks.

Incentive mechanisms for personal area networks must account for the specific constraints of such networks, while providing a stimulus for nodes to cooperate. A possible solution is to use reputation credits, which nodes may use to bid for transmission opportunities (time-slots).

Each master in the scatternet is assumed to be a trusted entity and keeps credit accounts of the slaves that have registered with it. Let C_i^m denote the credit of slave i held at master m. A slave gains credit by cooperating in forwarding packets, such that for each forwarded packet the account is incremented by $\gamma \times l_p$ where l_p is the length of the packet in time-slots. This account is decremented each time a master learns of a node's uncooperative behaviour, by the amount $\beta \times l_p$ such that $\gamma < \beta$. Similarly, when a node is given the opportunity to transmit, its credit is also decremented. Slave i is said to have a better reputation than slave j as perceived by master m if $C_i^m > C_j^m$.

For a slave node to obtain reputation credit from its home master the latter would need to have an acknowledgement that it has forwarded the packet. These acknowledgement messages are sent via opportunistic delay-tolerant routing. The home master appends its address to the packet so that a recipient master knows where to send the acknowledgement. If the confirmation does not reach the home master within a predefined timeout, it is assumed that the slave did not forward the message and therefore its reputation account will be decremented as mentioned above. Opportunistic routing relies on the probability of a node meeting another node that will bring the acknowledgement closer to the home master. In this setting, a recipient node chooses a node from a set of candidates with the highest probability of meeting the next node along the reverse path. This means that the reverse path is not necessarily the same as the forward path but rather follows a probabilistic route.

When a slave node moves from one piconet to another, it will be entrusted with its reputation account that it then forwards to the new master. In order to provide security, the account information is encrypted by the current master, which is subsequently read by the foreign master. If slaves fail to provide this information during the registration process, their account will be reset and they can only receive best-effort service until they build a sufficient reputation. In Bluetooth, registration of a new slave in a piconet is done via the inquiry and paging process; then upon completion, the master and slave exchange a packet to test the connection. This slot can, therefore, be used for the slave to provide its account information to the master or send a NULL packet if it is a newly arrived node in the network. Before a node is entrusted with its reputation account, the master needs to detect that a slave is moving. This can easily be achieved by reading the RSSI index and estimating the point at which the slave moves away from its current piconet.

The first step in the realisation of the scheme is to formulate the behaviour of nodes through a game

theoretic analysis of the system and to establish whether the proposed scheme leads to equilibrium, *i.e.,* if nodes would be better off cooperating. The probabilistic delay-tolerant acknowledgement scheme then needs to be developed and analysed. Finally, the entire system must be evaluated through simulation or via a implementation.

In conclusion, this piece of work aimed to improve error control, and thus hop-by-hop transmission, and to design a new routing protocol which conserves energy therefore improving the network's lifetime. The combination of these proposed solutions goes some way to enhancing forwarding mechanisms for personal area networks and thus satisfies the objectives set out herein. The Packet Error Correction (PEC) technique and the Centrally Coordinated Hybrid Routing (C2HR) protocol answer the challenges identified at the outset of the project and make a significant contribution to research in the field of personal area networking.

Appendix A

Acronyms and Abbreviations

ACL Asynchronous Connection-Less

ACLS Adaptive Cyclic-limited Scheduling

AFP Adaptive Flow-based Polling

AM_ADDR Active Member Address

AMP Alternate Mac/Phy

AODV Ad-hoc On-demand Distance Vector

APE Absolute Percentage Error

ARQ Automatic Repeat reQuest

BAN Body Area Network

BB_PDU Baseband Protocol Data Unit

BD_ADDR Bluetooth Device Address

BER Bit Error Rate

BNEP Bluetooth Network Encapsulation Protocol

BLN Bluetooth Location Network

BTCP Bluetooth Topology Construction Protocol

C2HR Centrally Coordinated Hybrid Routing

CAC Channel Access Code

CB Current Buffer

CID Channel Identifier

CL Connection Less

CLKN Native Clock

CO Connection Oriented

CONFIDANT Cooperation Of Nodes: Fairness In Dynamic Ad hoc NeTworks

CORE Collaborative Reputation

CRC Cyclic Redundancy Check

CT Connection Table

DAC Device Access Code

DAG Direct Acyclic Graph

DARWIN Distributed and Adaptive Reputation mechanism for WIreless ad hoc Networks

DCID Destination Channel Identifier

DDR Distributed Dynamic Routing

DH Data High-speed

DIAC Dedicated Inquiry Access Code

DLCI Data Link Connection Identifier

DM Data Medium-speed

DPSK Differential Phase Shift Keying

DQPSK Differential Quadrature Phase Shift Keying

DRR Deficit Round Robin

DSA Dynamic Slot Assignment

DSDV Destination-Sequenced Distance Vector

DSR Dynamic Source Routing

DST Distributed Spanning Tree

DV Data/Voice

EDR Enhanced Data Rates

EDSA Enhanced Dynamic Slot Assignment

EFSA Efficient and Fair Scheduling Algorithm

EPM Exhaustive Pseudo-cyclic Master Queue Length

ERR Exhaustive Round Robin

eSCO enhanced Synchronous Connection-Oriented

FEC Forward Error Correction

FHS Frequency Hopping Sequence

FHSS Frequency Hopping Spread Spectrum

FLC Fuzzy Logic Control

FSR Fisheye State Routing

FTP File Transfer Protocol

GAP Generic Access Profile

GFSK Gaussian Frequency Shift Keying

GIAC General Inquiry Access Code

GN Group ad hoc Network

GPS Global Positioning System

GSR Global State Routing

HCI Host Control Interface

HEC Header Error Check

HFP Hands Free Profile

HR^3 Hierarchical Routing in Resource Rationed

HTF Hybrid Tree Flooding

HV High quality Voice

IAC Inquiry Access Code

IARP IntrA-zone Routing Protocol

IERP IntEr-zone Routing Protocol

IP Internet Protocol

ISM Instrumental Scientific & Medical

Kb Kilo Bits

Kbps Kilo Bits per Second

L2CAP Link Layer Control and Adaptation Protocol

LAA Load Adaptive Algorithm

LAP Lower Address Part

LARP Location Aware Routing Protocol

LIAC Limited Inquiry Access Code

LM Link Manager

LMP Link Manager Protocol

LORP Relay Reduction and Route Construction Protocol

LOS Line Of Sight

LQ Link Quality

LWRR Limited and Weighted Round Robin

Mb Mega Bits

Mbps Mega Bits per Second

MDRP Maximum Distance Rendezvous Point

MFR Most Forward with fixed Radius

MP Max Priority

MQS Maximum Queue Size

MRP Multi Point Relay

ms Millisecond

MTS Max Time Share

MTU Maximum Transmission Unit

NACK Negative ACKnowledgement

NAP Network Access Point

NAP Non-significant Address Part

OLSR Optimised Link State Routing

OUI Organisationally Unique Identifier

OSI Open System Interconnect

PAN Personal Area Network

PANU PAN User

PDA Personal Digital Assistant

PDU Protocol Data Unit

PE Percentage Error

PEC Packet Error Correction

PER Packet Error Rate

PLsWRR Pseudo-Random Cyclic Limited and Slot-Weighted Round Robin

PM_ADDR Parked Member Address

PMP Participant in Multiple Piconets

PRR Pure Round Robin

PSK Phase Shift Keying

QoS Quality of Service

QIPS Qos-aware Inter-Piconet Scheduling

RERR Route Error

RFCOMM Radio Freuency Communications

ROMA Route Maintenance Algorithm

RP Rendezvous Point

RREP Route Reply

RREQ Route Request

RSSI Received Signal Strength Indicator

RT Retransmission Timeout

RTT Round Trip Time

RVM Route Vector Method

RW Rendezvous Window

SCID Source Channel Identifier

SCO Synchronous Connection-Oriented

SDP Service Discovery Protocol

SIG Special Interest Group

SLURP Scalable Location Update-based Routing Protocol

SNR Signal to Noise Ratio

SPRITE Simple cheat-Proof CredIT-based systEm

STAR Source Tree Adaptive Routing

STFT Short-Term Fourier Transform

TCP Transport Control Protocol

TC Time Commitment

TBRPF TopologyvBroadcast Reverse Path Forwarding

TORA Temporally Ordered Routing Algorithm

TDD Time Division Duplex

UAP Upper Address Part

UTF Unicode Transformation Format

UUID Universally Unique Identifier

UWB Ultra Wide Band

VoIP Voice over Internet Protocol

VDP Video Distribution Profile

VCG Vickery-Clarke-Groves

ZID Zone ID

ZHLS Zone-base Hierarchical Link State

ZRP Zone Routing Protocol

Appendix B

Implementation in Matlab

The proposed solutions are evaluated by means of simulation, which emulate the behaviour of personal area networks. This section provides details of the simulator and the way in which its various components interact.

A node is represented by a structure called `blue_node`, in which its corresponding attributes are stored. Each node is indexed by its ID, which is a natural number from 1 to `N`, where `N` is the number of nodes in the network. For an arbitrary node, i, `blue_node(i)` holds the following attributes:

`blue_node(i).id`: The ID of node i

`blue_node(i).location`: The cartesian coordinates of node i.

`blue_node(i).role`: The role of node i in the network. This can be either **master**, **slave** or **PMP**.

`blue_node(i).pico_id`: The ID of the piconet(s) that node i belongs to.

`blue_node(i).mode`: This shows whether node i is in the connected or inquiry mode.

`blue_node(i).neighbours`: Lists the IDs of nodes within transmission range of i.

`blue_node(i).current_neighbours`: Neighbours that i is scheduled to be connected to.

`blue_node(i).next_hop`: Lists the neighbours that node i had established a wireless link with.

`blue_node(i).current_time`: The actual current time. Not to be confused with simulation time.

`blue_node(i).value`: The value used for leader election purposes.

`blue_node(i).residual_energy`: The energy level remaining in i's battery.

`blue_node(i).adjacency_info`: Adjacency information table.

`blue_node(i).parent`: The parent of node i in the tree routed at the leader.

`blue_node(i).children`: List of node i's children.

`blue_node(i).current_leader`: The ID and value of node i's current leader.

`blue_node(i).out_buffer`: The output buffer of i's transceiver.

`blue_node(i).in_buffer`: The input buffer of i's transceiver.

`blue_node(i).adjacency_matrix`: The adjacency matrix distributed by the leader.

In addition to the above attributes, `blue_node` also stores temporary variables such as flags and

counters.

In the simulation, when data is transferred from node i to node j, channels are represented by a free-space loss model as well as forcing errors dictated by the channel bit error rate or the channel packet error rate.

B.1 PEC Implementation

The simulation flowchart for PEC is shown in Figure B.1

The main simulation is run in `pec.m`. The simulation starts by setting the environment variables: `number_of_nodes`, `blue_node.role`, `environment_dimensions` and `distance`, as well as the estimation window, which represents the amount of access code data a node is required to gather before estimating the channel error rate, and the error rate counter, which counts the number of sent packets before a new error rate is enforced. By default the channel error rate is set to 10^{-2}.

The simulator reads an audio data file using the Matlab function `wavread`, encodes it as an 8-bit unsigned stream then stores it in `blue_node(source).data`. The corresponding Link Quality (LQ) is then obtained by mapping the error rate through the function `mapping.m`, from which the buffer size is computed. The source node loads its `out_buffer` then transmits one packet at time after adding the necessary encapsulation. The transmission is modelled by introducing errors into the stream of data under consideration. This is achieved by generating an array of natural numbers of length L, where $L = error\_rate \times data\_size$; each entry in this array represent the Poisson distance between error-free packets. With this in mind, errors are forced into the data. After each transmission, the packet is removed from the source's `out_buffer` and the error rate counter is incremented. If the sent packet happens to be in error, the source retransmits the same packet with the same error probability and increments the retransmission counter. If the packet is received correctly, the source selects the next packet in the buffer to be transmitted until the buffer is exhausted. When the number of sent packets equals the error rate counter, a new channel error rate is selected randomly. The simulation continues until the source has sent all its packets. The delay and overhead are computed by incrementing `blue_node(source).current_time` and `blue_node(source).overhead` respectively. Each time a packet is sent, the time is incremented by the required amount as dictated by the packet length (number of time slots) plus poll delays. `blue_node(source).overhead` is incremented by the amount of redundant data sent, and the resulting overhead is calculated as $\frac{\texttt{blue\_node(source).overhead}}{data\_size}$.

After all the data has been sent, the simulator compares the PER before and after applying the PEC scheme.

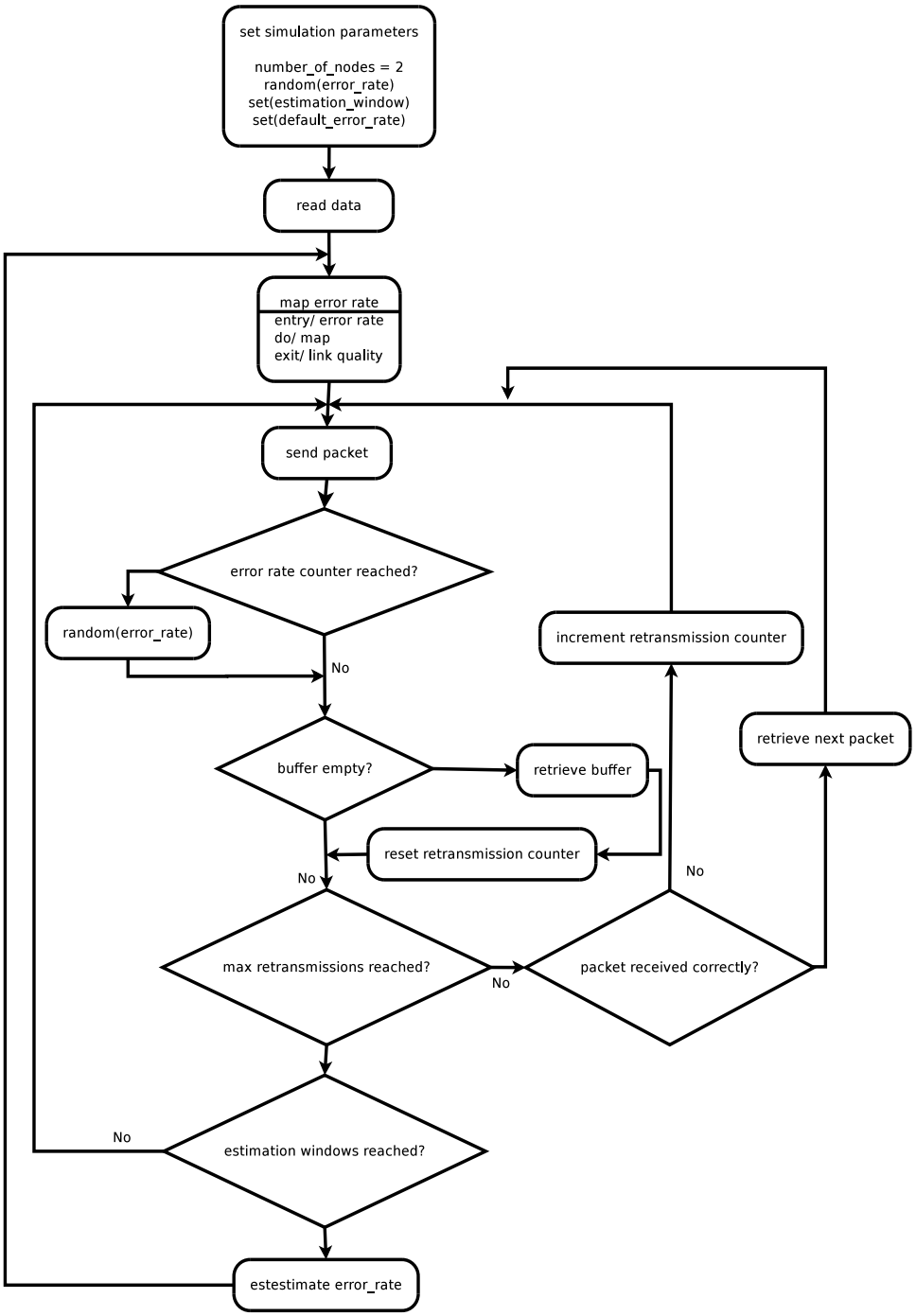


Figure B.1: PEC Simlation Flowchart

B.2 C2HR Implementation

The simulation flowchart for C2HR is shown in Figure B.2

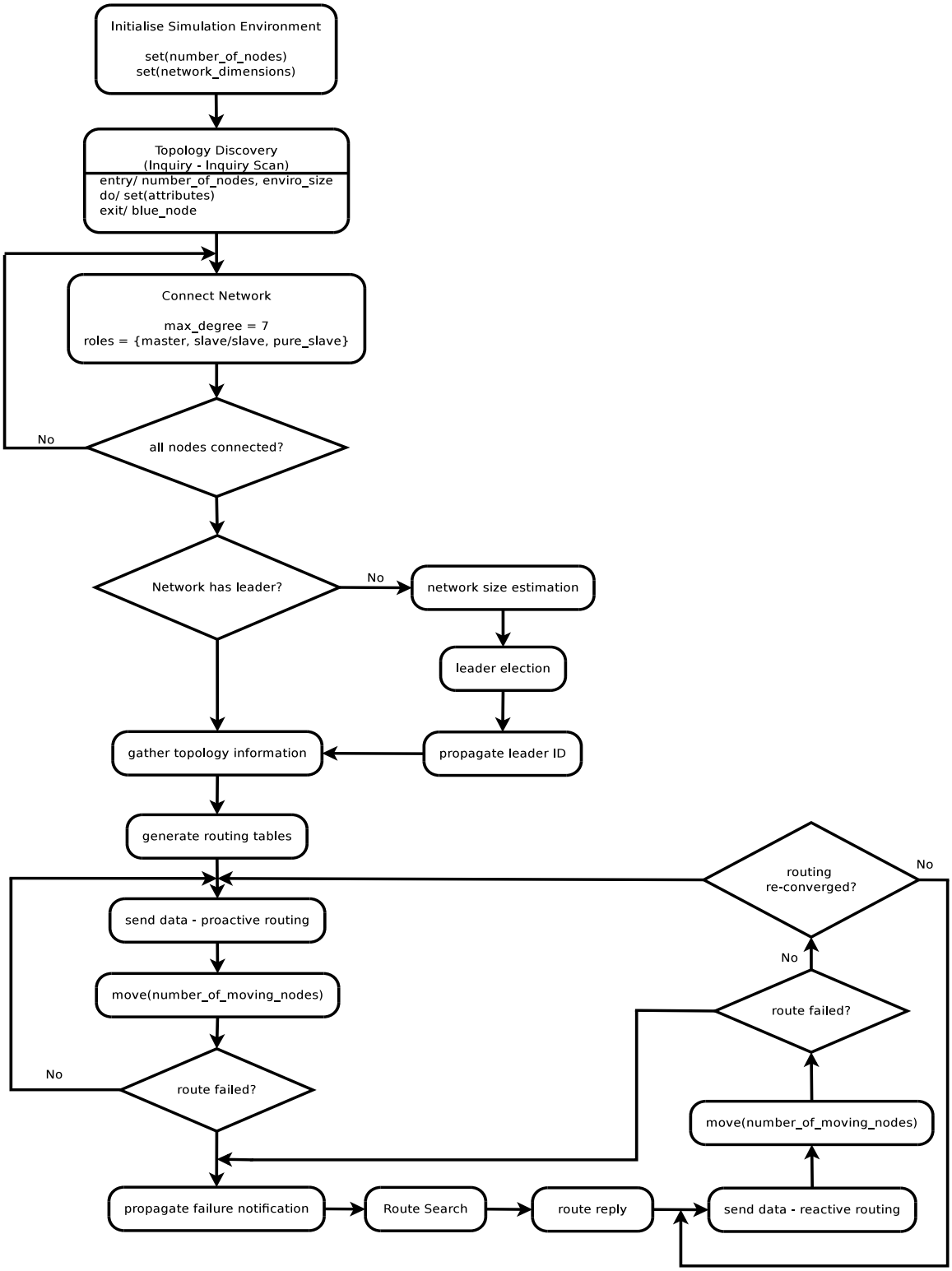


Figure B.2: C2HR Simlation Flowchart

As with the PEC technique, the simulation begins by setting the required variables: `number_of_nodes` and `network_dimensions`. Nodes are randomly scattered in the 3 dimensional space by generating a location matrix that holds the x, y and z coordinates of all the nodes in the network; each node, i then stores its location information in the variable `blue_node(i).location`. The function `topology.m` selects a random number of p-masters, which starts the inquiry and paging processes, to which the remaining p-slaves respond so that every node in the network adopts a role (stored in `blue_node(i).role`. `topology.m` also sets some of the attributes of `blue_node` mentioned earlier.

After the topology has been laid out, nodes connect to one another according the algorithm described in 5.2 via the function `connectivity.m`. Initially the network does not have a leader, thus, a leader is elected by calling the function `leader_election_mob.m`. In turn, `leader_election_mob.m` calls the function `net_size`, through which nodes obtain the estimate of the network size, necessary for the leader election process.

Once the leader has been elected, the latter, builds a tree rooted at itself and sends out topology enquiry messages, to which nodes respond via the function `shrink_tree.m`, the leader then builds the adjacency matrix and calls the function `distibute_adjacency.m` to distribute the matrix to every node in the connected tree.

Nodes run Dijkstra's algorithm on the adjacency matrix so that the routing table can be generated and used to route proactively. Similar to PEC, a random source node reads an audio file and stores the data into `blue_node(source).data` then forwards it to a randomly selected destination. The simulator keeps a counter of the number of packets sent, and when it reaches the value `mob_count`, `number_of_moving_nodes` nodes change location, thereby resulting in several possible failed links. If a link fails, the detecting node propagates the failure notification, which triggers the distribution of the adjacency matrix updates. At the same time the source node looks for an alternative route by calling the function `reactive_update.m`. Once the alternative route is found, the routing resumes on the reactive path, until the nodes finish computing the new routing tables. If the route fails while routing reactively, nodes are notified and another route search/route reply process is run. The simulation exits once the destination node receives the last packet.

Remarks:

- In all actions that require information exchange, the function `sched.m` is called so that nodes alter their memberships according to the inter and intra piconet scheduling.
- Link weights are adjusted after failures only.
- The variable `blue_node(i).current_time` is incremented each time node i sends out a packet. The actual time is then taken as being max(`blue_node.current_time`).

- The overhead is taken to be the ratio of the sum of redundant data generated by all nodes to the actual data generated in the network.

Appendix C

Publications

The research contribution provided in this thesis led to the following publications:

- S. Sae Lor, R. Ali, R. Landa and M. Rio, Recursive Loop-free Alternates for Full Protection Against Transient Link Failures, In *Proceedings of the IEEE 16th Symposium on Computers and Communications (ISCC)*, 2010.

- S. Sae Lor, R. Landa, R. Ali and M. Rio, Handling Transient Link Failures Using Alternate Next Hop Counters, In *Proceedings of the 9th International Conference on Networking (IFIP Networking 2010)*, 2010.

- R. Ali, S. Sae Lor and M. Rio, Two Algorithms for Network Size Estimation for Master/Slave Mobile Ad-hoc Networks, In *Proceedings of the IEEE 3rd International Symposium on Advanced Networking and Telecommunications Systems (ANTS)*, 2009

- R. Ali, S. Sae Lor and M. Rio, Cooperative Leader Election Algorithm for Master/Slave Mobile Ad-hoc Networks, In *Proceedings of the IEEE/IFIP 2nd Wireless Days Conference*, 2009

- R. Ali and J. Pollard, Error Control in Voice over IP over Bluetooth, In *Proceedings of the ECMS 13th International Conference on Analytical and Stochastic Modelling Techniques and Applications (ASMTA)*, 2006

- R. Ali and J. Pollard, Real-Time Voice Over IP over Bluetooth, In *Proceedings of the IEEE 3rd International Workshop on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS)*, 2005

References

[1] Ether Types. `http://www.iana.org/assignments/ethernet-numbers`, 2009.

[2] M. Abolhasan, T. Wysocki, and E. Dutkiewicz. A review of routing protocols for mobile ad hoc networks. *Ad Hoc Networks*, 2(1):1–22, 2004.

[3] M. Abolhasan, T. Wysocki, and E. Dutkiewicz. A Review of Routing Protocols for Mobile Ad Hoc Networks. *Ad Hoc Networks*, 2(1):1–22, 2004.

[4] M. Aguilera, C. Delporte-Gallet, H. Fauconnier, and S. Toueg. On implementing omega with weak reliability and synchrony assumptions. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 306–314, 2003.

[5] I. Akyildiz, T. Melodia, and K. Chowdury. Wireless multimedia sensor networks: A survey. *Wireless Communications, IEEE*, 14(6):32–39, 2007.

[6] O. Al-Jarrah and O. Megdadi. Enhanced AODV routing protocol for Bluetooth scatternet. *Computers and Electrical Engineering*, 35(1):197–208, 2009.

[7] H. Alemdar and C. Ersoy. Wireless sensor networks for healthcare: A survey. *Computer Networks*, 2010.

[8] R. Ali and J. Pollard. Real-time Voice Over IP Over Bluetooth. *IEEE Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2005. IDAACS 2005*, pages 580–583, 2005.

[9] M. Amin, F. Bhuyan, and M. Rahman. Bluetooth Scatternet Formation Protocol: A Comparative Performance Analysis. In *Asia-Pacific Conference on Communications, APCC'06*, pages 1–5, 2006.

[10] L. Anderegg and S. Eidenbenz. Ad hoc-VCG: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents. In *Proceedings of the 9th annual international conference on Mobile computing & networking (MOBICOM)*, pages 245–259, 2003.

[11] T. Arampatzis, J. Lygeros, and S. Manesis. A survey of applications of wireless sensors and wireless sensor networks. In *Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation*, pages 719–724. IEEE, 2006.

[12] S. Baatz, M. Frank, C. Kuhl, P. Martini, and C. Scholz. Bluetooth scatternets: An enhanced adaptive scheduling scheme. In *IEEE INFOCOM*, volume 2, pages 782–790, 2002.

[13] C. Baber and O. Westmancott. Social networks and mobile games: The use of bluetooth for a multiplayer card game. *Lecture Notes in Computer Science*, pages 98–107, 2004.

[14] M. Barbehenn. A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices. *IEEE Transactions on Computers*, 47(2):263, 1998.

[15] S. Basagni, R. Bruno, G. Mambrini, and C. Petrioli. Comparative Performance Evaluation of Scatternet Formation Protocols for Networks of Bluetooth Devices. *Wreless Networks*, 10(2):197–213, 2004.

[16] S. Basagni, M. Nanni, and C. Petrioli. Bluetooth scatternet formation and scheduling: An integrated solution. In *Proceedings of IEEE MILCOM 2006*, pages 23–25, 2006.

[17] P. Bhagwat and A. Segall. A routing vector method (RVM) for routing in Bluetooth scatternets. In *6th IEEE International Workshop on Mobile Multimedia Communications (MOMUC)*, pages 375–379, 1999.

[18] Bluetooth SIG. Bluetooth Specifications Version 1.0. *Available: http://www.bluetooth.com*, 1999.

[19] Bluetooth, SIG. RFCOMM with TS 07.10. *Bluetooth Specification Version 1, Part F*, 1, 2001.

[20] Bluetooth SIG. Bluetooth network encapsulation protocol (BNEP) specification. *Specification of the Bluetooth System, Version 1.0*, 2003.

[21] Bluetooth SIG. Bluetooth Specifications Version 1.2. *Available: http://www.bluetooth.com*, 2003.

[22] Bluetooth SIG. Bluetooth Specifications Version 2.1 + EDR. *Available: http://www.bluetooth.com*, 2007.

[23] Bluetooth SIG. Bluetooth Specifications Version 3.0 + HS. *Available: http://www.bluetooth.com*, 2009.

[24] Bluetooth SIG. Bluetooth Specifications Version 4.0. *Available: http://www.bluetooth.com*, 2009.

[25] R. Bruno, M. Conti, and E. Gregori. Bluetooth: architecture, protocols and scheduling algorithms. *Cluster Computing*, 5(2):117–131, 2002.

[26] S. Buchegger and J. Le Boudec. Performance analysis of the CONFIDANT protocol. In *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing (MOBIHOC)*, pages 226–236, 2002.

[27] L. Buttyán and J. Hubaux. Enforcing service availability in mobile ad-hoc WANs. In *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing (MOBIHOC)*, pages 87–96, 2000.

[28] L. Buttyán and J. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *Mobile Networks and Applications*, 8(5):579–592, 2003.

[29] E. Callaway, P. Gorday, L. Hester, J. Gutierrez, M. Naeve, B. Heile, and V. Bahl. Home networking with IEEE 802. 15. 4: a developing standard for low-rate wireless personal area networks. *IEEE Communications Magazine*, 40(8):70–77, 2002.

[30] J. Cano, J. Cano, E. González, C. Calafate, and P. Manzoni. Evaluation of the energetic impact of bluetooth low-power modes for ubiquitous computing applications. In *Proceedings of the 3rd ACM international workshop on Performance evaluation of wireless ad hoc, sensor and ubiquitous networks*, pages 1–8, 2006.

[31] A. Capone, M. Gerla, and R. Kapoor. Efficient polling schemes for Bluetooth picocells. In *Proc. IEEE ICC*, pages 1990–1994, 2001.

[32] C. Chang, P. Sahoo, and S. Lee. Location-Aware Routing Protocol for the Bluetooth Scatternet. *Wireless Personal Communications*, 40(1):117–135, 2007.

[33] J. Chang and L. Tassiulas. Energy conserving routing in wireless ad-hoc networks. In *Proceedings of IEEE INFOCOM*, pages 22–31, 2000.

[34] L. Chen, R. Kapoor, K. Lee, Y. Sanadidi, and M. Gerla. Audio Streaming over Bluetooth: An Adaptive ARQ Timeout Approach. In *Proceedings of IEEE 24th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 196–201, 2004.

[35] T. Chen and M. Gerla. Global State Routing: A New Routing Scheme for Ad-Hoc Wireless Networks. In *Proceedings of IEEE International Conference Communications (ICC)*, pages 171–175, 1998.

[36] C. F. Chiasserini, M. A. Marsan, E. Baralis, and P. Garza. Towards Feasible Topology Formation Algorithms for Bluetooth-based WPANs. In *Proceedings of the IEEE 36th Hawaii International Conference on System Sciences (HICSS '03)*, pages 313–322, 2003.

[37] Y. Choi, H. Lee, S. Park, B. Hong, S. Lee, and K. Tchah. A unified GFSK, π/4-shifted DQPSK, and 8-DPSK baseband controller for enhanced data rate Bluetooth SoC. *Current Applied Physics*, 6(5):862–872, 2006.

[38] C. Cordeiro, S. Abhyankar, and D. Agrawal. Design and implementation of QoS-driven dynamic slot assignment and piconet partitioning algorithms over bluetooth WPANS. In *IEEE INFOCOM*, volume 2, pages 1252–1263, 2004.

[39] L. Costa and G. Travieso. Exploring complex networks through random walks. *Physical Review E*, 75(1), 2007.

[40] B. Crow, I. Widjaja, L. Kim, and P. Sakai. IEEE 802.11 wireless local area networks. *IEEE Communications magazine*, 35(9):116–126, 1997.

[41] A. Das, A. Ghose, A. Razdan, H. Saran, and R. Shorey. Enhancing performance of asynchronous data traffic over the Bluetooth wireless ad hoc network. In *IEEE INFOCOM*, volume 1, pages 591–600, 2001.

[42] S. Das, C. Perkins, E. Royer, and M. Marina. Performance comparison of two on-demand routing protocols for ad hoc networks. In *IEEE INFOCOM*, pages 3–12, 2000.

[43] A. Derhab and N. Badache. A Self-Stabilizing Leader Election Algorithm in Highly Dynamic Ad Hoc Mobile Networks. *IEEE Transactions on Parallel and Distributed Systems*, 19(7):926–939, 2008.

[44] M. Dianati, X. Ling, K. Naik, and X. Shen. A node-cooperative ARQ scheme for wireless ad hoc networks. *IEEE Transactions on Vehicular Technology*, 55(3):1032–1044, 2006.

[45] T. Do, P. Engelstad, and T. Jønvik. Establishing IP Network Support for a PAN-Based Virtual Device. In *Proceedings of 9th International Conference on Intelligence in service delivery Networks (ICIN 2004), Bordeaux, France*, 2004.

[46] S. Dolev, E. Schiller, and J. Welch. Random walk for self-stabilizing group communication in ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(7):893–905, 2006.

[47] J. Foote. Visualizing music and audio using self-similarity. In *Proceedings of the seventh ACM international conference on Multimedia (Multimedia '99) - Part 1*, pages 77–80, 1999.

[48] M. Frodigh, P. Johansson, and P. Larsson. Wireless ad hoc networking-The art of networking without a network. *Ericsson Review*, 4:248–263, 2000.

[49] J. Garcia-Luna-Aceves and E. Madruga. A multicast routing protocol for ad-hoc networks. In *Proceeding of IEEE INFOCOM*, pages 784–792, 1999.

[50] J. Garcia-Luna-Aceves and M. Spohn. Source-tree routing in wireless networks. pages 273–282, 1999.

[51] C. Gkantsidis, M. Mihail, and A. Saberi. Random Walks in Peer-to-Peer Networks. In *Proceedings of IEEE INFOCOM*, pages 0–0, 2004.

[52] B. Golden. Shortest-path algorithms: A comparison. *Operations Research*, pages 1164–1168, 1976.

[53] J. Gomez and A. Campbell. A Case for Variable-range Transmission Power Control in Wireless Multihop Networks. In *Proceedings of IEEE INFOCOM*, pages 1425–1436, 2004.

[54] F. Gonzalez-Castano and J. Garcia-Reinoso. Bluetooth Location Networks. In *Proceedings of IEEE GLOBECOME*, pages 233–237, 2002.

[55] I. Gupta, R. Van Renesse, and K. Birman. A probabilistically correct leader election protocol for large groups. In *Proceedings of the 14th International Conference on Distributed Computing (DISC)*, pages 89–103, 2000.

[56] J. Haartsen. Bluetooth-The universal radio interface for ad hoc, wireless connectivity. *Ericsson review*, 3(1):110–117, 1998.

[57] J. Haartsen and S. Mattisson. Bluetooth-a new low-power radio interface providing short-range connectivity. *PROCEEDINGS-IEEE*, 88(10):1651–1661, 2000.

[58] J. Haartsen and E. Radio. The Bluetooth radio system. *IEEE Personal Communications*, 2000.

[59] Z. Haas, M. Pearlman, and P. Samar. The zone routing protocol (ZRP) for ad hoc networks. *draft-ietf-manet-zone-zrp-02. txt*, 1999.

[60] L. Har-Shai, R. Kofman, A. Segall, and G. Zussman. Load-Adaptive Inter-Piconet Scheduling in Small-scale Bluetooth Scatternets. *IEEE Communications Magazine*, 42(7):136–142, 2004.

[61] L. Har-Shai, R. Kofman, G. Zussman, and A. Segall. Inter-piconet scheduling in Bluetooth scatternets. In *Proc. OPNETWORK*, volume 2, 2002.

[62] G. Hernando, J. Cabero, J. Jodrá, and S. Pérez. Implementation and Comparison of AODV and OLSR Routing Protocols in an Ad-Hoc Network over Bluetooth. In *Proceedings of the 8th International Conference on Ad-Hoc, Mobile and Wireless Networks*, pages 347–353, 2009.

[63] X. Hong, K. Xu, and M. Gerla. Scalable Routing Protocols for Mobile Ad Hoc Networks. *IEEE Network*, 16(4):11–21, 2002.

[64] M. Hossain and W. Soh. A comprehensive study of bluetooth signal parameters for localization. In *Proceedings of the IEEE 18th International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, pages 1–5, 2007.

[65] I. Howitt. Mutual interference between independent Bluetooth piconets. *IEEE Transactions on Vehicular Technology*, 52(3):708–718, 2003.

[66] C. Hsu and S. Hsu. An Adaptive Interpiconet Scheduling Algorithm Based on HOLD Mode in Bluetooth Scatternets. *IEEE Transactions on Vehicular Technology*, 57(1):475–489, 2008.

[67] S. Hsu. A novel hold-mode-based adaptive inter-piconet scheduling algorithm in bluetooth scatternets. pages 469–474, 2006.

[68] IEEE Computer Society. IEEE Standard for Local and Metropolitan Area Networks - Media Access Control (MAC) Bridges. `http://standards.ieee.org/getieee802/download/802.1D-2004.pdf`, 2004.

[69] L. Isaksson, M. Fiedler, and A. Nilsson. Validation of simulations of bluetooths frequency hopping spread spectrum technique. In *Proceedings of the 2004 Design, Analysis, and Simulation of Distributed Systems, Arlington, Virginia, USA*, pages 156–165, 2004.

[70] P. Jacquet, P. Muhlethaler, A. Qayyum, A. Laouiti, L. Viennot, and T. Claussen. Optimized Link State Routing Protocol. Technical report, Internet Draft, IETF, draft-ietf-manetolsr-00.txt, 1998.

[71] H. Janzadeh, K. Fayazbakhsh, M. Dehghan, and M. Fallah. A secure credit-based cooperation stimulating mechanism for MANETs using hash chains. *Future Generation Computer Systems*, 25(8):926–934, 2009.

[72] J. Jaramillo and R. Srikant. Darwin: Distributed and adaptive reputation mechanism for wireless ad-hoc networks. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking (MOBICOM)*, pages 87–97, 2007.

[73] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems (TOCS)*, 23(3):219–252, 2005.

[74] M. Joa-Ng and I. Lu. A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1415–1425, 1999.

[75] N. Johansson, F. Alriksson, and U. Jönsson. JUMP mode–a dynamic window-based scheduling framework for Bluetooth scatternets. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing (MobiHoc)*, pages 204–211, 2001.

[76] P. Johansson, R. Kapoor, M. Kazantzidis, and M. Gerla. Personal area networks: Bluetooth or IEEE 802.11? *International Journal of Wireless Information Networks*, 9(2):89–103, 2002.

[77] P. Johansson, R. Kapoor, M. Kazantzidis, and M. Gerla. Rendezvous scheduling in Bluetooth scatternets. In *IEEE International Conference on Communications*, pages 318–324, 2002.

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

[79] D. Johnson and D. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *Kluwer International Series in Engineering and Computer Science*, pages 153–179, 1996.

[80] V. Jones, R. Bults, D. Konstantas, and P. Vierhout. Healthcare PANs: Personal Area Networks for trauma care and home care. In *Proceedings Fourth International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pages 9–12, 2001.

[81] M. Joos. Acoustic phonetics. *Language*, 24(2):5–136, 1948.

[82] S. Jung, U. Lee, A. Chang, D. Cho, and M. Gerla. Bluetorrent: Cooperative content sharing for bluetooth users. *Pervasive and Mobile Computing*, 3(6):609–634, 2007.

[83] M. Kalia, S. Garg, and R. Shorey. Scatternet structure and inter-piconet communication in the Bluetooth system. In *IEEE national conference on communications*, 2000.

[84] C. K. Kalló, S. Jung, L. J. Shen, M. Brunato, and M. Gerla. Throughput, Energy and Path Length Tradeoffs in Bluetooth Scatternets. In *Proceedings of IEEE International Conference on Communications (ICC '05)*, pages 3319–3323, 2005.

[85] R. Kapoor, L. Chen, Y. Lee, and M. Gerla. Bluetooth: carrying voice over ACL links. In *Proc. of MWCN*, pages 379–383, 2002.

[86] R. Kapoor and M. Gerla. A zone routing protocol for bluetooth scatternets. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1459–1464, 2003.

[87] R. Kapoor, M. Y. M. Sanadidi, and M. Gerla. An Analysis of Bluetooth Scatternets Topologies. In *Proceedings of IEEE International Conference on Communications (ICC '03)*, pages 266–270, 2003.

[88] R. Kapoor, A. Zanella, and M. Gerla. A fair and traffic dependent scheduling algorithm for Bluetooth scatternets. *Mobile Networks and Applications*, 9(1):9–20, 2004.

[89] M. Karam and F. Tobagi. Analysis of the delay and jitter of voice traffic over the Internet. In *IEEE INFOCOM*, pages 824–833, 2001.

[90] J. Karaoguz. High-rate wireless personal area networks. *IEEE Communications Magazine*, 39(12):96–102, 2001.

[91] O. Karjalainen, S. Rantala, and M. Kivikoski. A comparison of bluetooth low power modes. In *Proceedings of the 7th International Conference on Telecommunications ConTEL 2003*, pages 121–128, 2003.

[92] S. Katti, D. Katabi, W. Hu, H. Rahul, and M. Medard. The importance of being opportunistic: Practical network coding for wireless environments. In *Proc. 43rd Annual Allerton Conference on Communication, Control, and Computing*, 2005.

[93] Y. Ko and N. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. *Wireless Networks*, 35(4):307–321, 2000.

[94] C. Kunze, U. Grossmann, W. Stork, and K. Müller-Glaser. Application of ubiquitous computing in personal health monitoring systems. *Biomedizinische Technik/Biomedical Engineering*, 47(s1a):360–362, 2002.

[95] A. Kvalbein, A. Hansen, T. Cicic, S. Gjessing, and O. Lysne. Fast IP network recovery using multiple routing configurations. In *Proceedings of IEEE INFOCOM*, pages 1–11, 2006.

[96] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica. Achieving convergence-free routing using failure-carrying packets. *ACM SIGCOMM Computer Communication Review*, 37(4):252, 2007.

[97] B. Lavelle, D. Byrne, C. Gurrin, A. Smeaton, and G. Jones. Bluetooth Familiarity: Methods of Calculation, Applications and Limitations. In *Proceedings of the 5th Workshop on Mobile Interaction with the Real World (MIRW)*, pages 55–58, 2007.

[98] C. Law and K. Siu. A Bluetooth Scatternet Formation Algorithm. In *Proceedings of GLOBECOM '01*, pages 2864–2869, 2001.

[99] E. Le Merrer, A. Kermarrec, and L. Massoulie. Peer to peer size estimation in large and dynamic networks: A comparative study. In *Proceedings of the IEEE International Symposium on High Performance Distributed Computing (HPDC)*, pages 0–0, 2006.

[100] S. Lee, R. Muhammad, and C. Kim. A Leader Election Algorithm Within Candidates on Ad Hoc Mobile Networks. *Embedded Software and Systems*, 4523:728–738, 2007.

[101] Y. Lee, R. Kapoor, and M. Gerla. An Efficient and Fair Polling Scheme for Bluetooth. In *Proceedings IEEE MILCOM*, pages 1062–1068, 2002.

[102] P. Lieberman and S. Blumstein. *Speech physiology, speech perception, and acoustic phonetics*. Cambridge University Press, 1988.

[103] T. Lindholm. Setting up a bluetooth packet transport link, 2003.

[104] P. Littieri and M. Srivastava. Adaptive Frame LEngth Control for Improving Wireless Link Throughput, Range, and Energy Efficiency. In *Proceedings of IEEE INFOCOM*, pages 564–571, 1998.

[105] k. Maalaoui and L. Azouz Saidane. Priority Based Intra Piconet Scheduling Scheme for QoS Guaranties in Bluetooth Networks. In *Proceedings of the IEEE Symposium on Computers and Communications (ISCC)*, pages 147–152, 2009.

[106] N. Malpani, J. Welch, and N. Vaidya. Leader election algorithms for mobile ad hoc networks. In *Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 96–103, 2000.

[107] M. A. Marsan, C. F. Chiasserini, A. Nucci, G. Carello, and L. De Giovanni. Optimising the Topology of Bluetooth Wireless Personal Area Networks. In *Proceedings of IEEE INFOCOM*, pages 572–579, 2002.

[108] L. Marzegalli, M. Masa, and M. Vitiello. Adaptive RTP/UDP/IP Header Compression for VoIP over Bluetooth. In *European Wireless: Next Generation Wireless Networks*, 2002.

[109] L. Massoulie, E. Le Merrer, A. Kermarrec, and A. Ganesh. Peer counting and sampling in overlay networks: random walk method. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 123–132, 2006.

[110] P. Michiardi and R. Molva. CORE: A Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad Hoc Networks. In *Proceedings of the 6th IFIF Conference on Communications and Multimedia Security*, pages 107–121, 2002.

[111] G. Miklós, A. Rácz, Z. Turányi, A. Valkó, and P. Johansson. Performance aspects of bluetooth scatternet formation. In *MobiHoc '00: Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, pages 147–148, 2000.

[112] B. Miller and C. Bisdikian. *Bluetooth revealed*. Prentice Hall PTR Upper Saddle River, NJ, USA, 2001.

[113] H. Minn, M. Zeng, A. Annamalai, and V. Bhargava. An efficient ARQ protocol for adaptive error control over time-varying channels. *Wireless Personal Communications*, 17(1):3–20, 2001.

[114] H. Minn, M. Zeng, and V. Bhargava. On ARQ scheme with adaptive error control. *IEEE Transactions on Vehicular Technology*, 50(6):1426–1436, 2001.

[115] D. Miorandi, S. Merlin, A. Trainito, and A. Zanella. On Efficient Configurations for Bluetooth Scatternets. *Ad Hoc Networks*, 4(6):768–787, 2006.

[116] J. Mišic and V. Mišic. Bridges of Bluetooth county: topologies, scheduling, and performance. *IEEE Journal on selected areas in communications*, 21(2):240–258, 2003.

[117] V. Mišić, E. Ko, and J. Mišić. Load and QoS-Adaptive Scheduling in Bluetooth Piconets. In *Proceedings of the IEEE 37th Hawaii International Conference on System Sciences (HICSS '04)*, pages 294–303, 2004.

[118] V. Mišić and J. Mišić. Modeling Bluetooth piconet performance. *IEEE Communications Letters*, 7(1):18–20, 2003.

[119] E. Modiano. An Adaptive Algorithm for Optimizing the Packet Size in Wireless ARQ Protocols. *Wireless Networks*, 5(4):279–286, 1999.

[120] R. Morrow. *Bluetooth operation and use*. McGraw-Hill Professional, 2002.

[121] N. Muller. *Bluetooth demystified*. McGraw-Hill Boston, MA, 2001.

[122] S. Muruganathan, D. Ma, R. Bhasin, and A. Fapojuwo. A centralized energy-efficient routing protocol for wireless sensor networks. *IEEE Communications Magazine*, 43(3):8–13, 2005.

[123] V. Nagarajan. Broadcom: Bluetooth Today and Tomorrow. `http://www.sramanamitra.com/2008/02/24/broadcom-bluetooth-today-and-tomorrow/`, 2008.

[124] K. Naik, D. Wei, and Y. Su. Packet interference in a heterogeneous cluster of Bluetooth piconets. In *Proceedings of the IEEE 58th Vehicular Technology Conference (VTC)*, pages 582–586, 2003.

[125] B. Nazir and K. Zia. QoS aware Interpiconet Scheduling in Bluetooth Scatternet (QIPS). In *Proceedings of the IEEE International Conference on Emerging Technologies (ICET)*, pages 68–73, 2007.

[126] S. Nelakuditi, S. Lee, Y. Yu, Z. Zhang, and C. Chuah. Fast local rerouting for handling transient link failures. *IEEE/ACM Transactions on Networking*, 15(2):359–372, 2007.

[127] P. Networking. Personal Area Networking Profile. 2001.

[128] I. Niemegeers and S. Heemstra de Groot. From personal area networks to personal networks: A user oriented approach. *Wireless Personal Communications*, 22(2):175–186, 2002.

[129] I. Niemegeers and S. Heemstra de Groot. Research issues in ad-hoc distributed personal networking. *Wireless Personal Communications*, 26(2):149–167, 2003.

[130] N. Nikaein, L. H., and B. C. DDR: Distributed Dynamic Routing Algorithm for Mobile Ad Hoc Networks. In *Proceedings of the 1st ACM International Symposium on Mobile Ad Hoc Networking & Computing (MOBIHOC)*, pages 19–27, 2000.

[131] R. Ogier, F. Templin, and M. Lewis. Topology dissemination based on reverse-path forwarding (TBRPF). *Request for Comments*, 3684, 2004.

[132] V. Park and M. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of IEEE INFOCOM*, volume 3, pages 1405–1413, 1997.

[133] G. Pei, M. Gerla, and T. Chen. Fisheye State Routing: A routing Scheme for Ad Hoc Wireless Networks. In *Proceedings of IEEE International Conference Communications (ICC)*, pages 70–74, 2000.

[134] C. Perkins, E. Belding-Royer, and S. Das. Ad-hoc On-demand Distance Vector (AODV) Routing. *Request for Comments*, 2004.

[135] C. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector routing (DSDV) for Mobile Computers. *SIGCOMM Computer Communcations Review*, 24(4):234–244, 1994.

[136] B. Peterson, R. Baldwin, J. Kharoufeh, and R. Raines. Refinements to the packet error rate upper bound for Bluetooth networks. *IEEE Communications Letters*, 7(8):382–384, 2003.

[137] C. Petrioli, S. Basagni, and I. Chlamtac. Configuring BlueStars: Multihop scatternet formation for Bluetooth networks. *IEEE Transactions on Computers*, 52(6):779–790, 2003.

[138] C. Petrioli, S. Basagni, and I. Chlamtac. BlueMesh: degree-constrained multi-hop scatternet formation for Bluetooth networks. *Mobile Networks and Applications*, 9(1):33–47, 2004.

[139] S. Radhakrishnan, G. Racherla, C. Sekharan, N. Rao, and S. Batsell. DST - A Routing Protocol for Ad Hoc Networks Using Distributed Spanning Trees. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1543–1547, 1999.

[140] R. Razavi, M. Fleury, and M. Ghanbari. Fuzzy logic control of adaptive ARQ for video distribution over a Bluetooth wireless link. *Advances in Multimedia*, 2007(1):8–20, 2007.

[141] Road Safety Devision 6, Department for Transport. Mobile Phones and Driving: Regulatory Impact Assessment. `http://www.dft.gov.uk/consultations/aboutia/ria/mobilephonesanddrivingregula5538`, 2003.

[142] E. Royer and C. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE personal communications*, 1999.

[143] E. Royer and C. Toh. A Review of current Routing Protocols for Ad Hoc Mobile Wireless Networks. *IEEE Personal Communications*, 6(2):46–55, 1999.

[144] S. Saha and M. Matsumoto. An Inter-Piconet Scheduling Algorithm for Bluetooth Scatternets. In *Proceedings of the Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services (AICT/ICIW)*, pages 24–28, 2006.

[145] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire. Distributed Topology Construction of Bluetooth Personal Area Networks. In *Proceedings of IEEE INFOCOM*, pages 1577–1586, 2001.

[146] N. Saxena, J. Ekberg, K. Kostiainen, and N. Asokan. Secure device pairing based on a visual channel. In *2006 IEEE Symposium on Security and Privacy*, pages 306–313, 2006.

[147] R. Schmidt, T. Norgall, J. Mörsdorf, J. Bernhard, and T. von der Grün. Body Area Network BAN - A Key Infrastructure Element for Patient-Centered Medical Applications. *Biomedizinische Technik/Biomedical Engineering*, 47(s1a):365–368, 2002.

[148] T. Shafaat, A. Ghodsi, and S. Haridi. A Practical Approach to Network Size Estimation for Structured Overlays. In *Proceedings of the 3rd International Workshop on Self-Organizing Systems*, pages 71–83, 2008.

[149] L. Shek and Y. Kwok. Efficient multi-hop communications in Bluetooth scatternets. In *Proceeding of the 14th IEEE international symposium on personal, indoor, and mobile radio communication (PIMRC)*, pages 755–759, 2003.

[150] R. Shepherd. Bluetooth wireless technology in the home. *Electronics and Communication Engineering Journal*, 13(5):195–203, 2001.

[151] M. Shirali, A. Toroghi, and M. Vojdani. Leader Election Algorithms: History and Novel Schemes. In *Third International Conference on Convergence and Hybrid Information Technology, ICCIT'08*, pages 1001–1006, 2008.

[152] D. M. D. Smith, C. F. Lee, J.-P. Onnela, and N. F. Johnson. Link-space formalism for network analysis. *Physical Review E*, 77(3):1–18, 2008.

[153] Z. Specification. v1. 0: ZigBee Specification (2005). *San Ramon, CA, USA: ZigBee Alliance*.

[154] N. Sriskanthan, F. Tan, and A. Karande. Bluetooth based home automation system. *Microprocessors and Microsystems*, 26(6):281–289, 2002.

[155] C. Strangio. The RS232 standard. *CAMI Research Inc., Lexington, Massachusetts*, 2005, 2003.

[156] The Economist. Bluetooth quiet success. `http://www.ebusinessforum.com/index.asp?layout=rich_story&doc_id=8680&categoryid=&channelid=&search=connecting`, 2006.

[157] TheMathworks, INC. MATLAB R2008b. `http://www.mathworks.co.uk/`.

[158] C. Toh. Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks. *IEEE communications Magazine*, 39(6):138–147, 2001.

[159] UCL Division of Psychology & Language Sciences. SFS/RTGram Version 1.2. `http://www.phon.ucl.ac.uk/resource/sfs/rtgram/`.

[160] M. Valenti, M. Robert, and J. Reed. On the Throughput of Bluetooth Data Transmissions. In *Proceedings of IEEE WCNC*, pages 119–123, 2002.

[161] M. Varghese and M. Shreedhar. Efficient Fair Queuing Using Deficit Round Robin. In *Proceedings ACM SIGCOMM*, pages 231–243, 1996.

[162] U. Varshney. Multicast over wireless networks. *Communications of the ACM*, 45(12):31–37, 2002.

[163] S. Vasudevan, J. Kurose, and D. Towsley. Design and analysis of a leader election algorithm for mobile ad hoc networks. In *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP)*, pages 350–360, 2004.

[164] A. Vlavianos, L. Law, I. Broustis, S. Krishnamurthy, and M. Faloutsos. Assessing link quality in IEEE 802.11 Wireless Networks: Which is the right metric? In *Proceedings of the IEEE 19th International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, pages 1–6, 2008.

[165] W. Wang, M. Chatterjee, K. Kwiat, and N. Rome. An Economic Approach to Hierarchical Routing in Resource Rationed Ad Hoc Networks. In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007*, pages 1–8, 2007.

[166] Z. Wang, R. Thomas, and Z. Haas. Bluenet – A New Scatternet Formation Scheme. In *Proceedings of the IEEE 35th Hawaii International Conference on System Sciences (HICSS '02)*, pages 61–69, 2002.

[167] Z. Wang, R. J. Thomas, and Z. J. Haas. Performance comparison of Bluetooth scatternet formation protocols for multi-hop networks. *Wreless Networks*, 15(2):209–226, 2009.

[168] A. Willig. Polling-based MAC protocols for improving real-time performance in a wireless PROFIBUS. *IEEE Transactions on Industrial Electronics*, 50(4):806–817, 2003.

[169] D. Wolpert and W. Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[170] S. Woo and S. Singh. Scalable Routing Protocol for Ad Hoc Networks. *Wireless Networks*, 7(5):513–529, 2001.

[171] Z. WU, X. ZHONG, W. YU, and Z. LIU. System level simulation modeling of bluetooth voice and its interference. In *Proceedings of the IEEE 7th International Conference on Signal Processing (ICSP)*, pages 29–32, 2004.

[172] D. Yang, G. Nair, B. Sivaramakrishnan, H. Jayakumar, and A. Sen. Round robin with look ahead: a new scheduling algorithm for Bluetooth. In *Proceedings of the International Conference on Parallel Processing Workshops (ICPPW)*, pages 45–50, 2002.

[173] Y. Yang, H. Wu, and W. Zhuang. MESTER: minimum energy spanning tree for efficient routing in wireless sensor networks. In *Proceedings of the 3rd ACM International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks*, pages 8–17, 2006.

[174] C. Yu, K. Yu, and S. Lin. Efficient Scheduling Algorithms for Bluetooth Scatternets. *Wireless Personal Communications*, 48(2):291–309, 2009.

[175] G. Yu, C. Chang, and Shih. Relay Reduction and Disjoint Routes Construction for Scatternet over Bluetooth Radio Systems. *Wireless Personal Communications*, 40(1):117–135, 2007.

[176] A. Zanella and M. Zorzi. Throughput and Energy Efficiency of Bluetooth v2 + EDR in Fading Channels. In *Proceedings of IEEE WCNC*, pages 1661–1666, 2008.

[177] G. Zaruba, S. Basagni, and I. Chlamtac. Bluetrees-scatternet formation to enable Bluetooth-based ad hoc networks. In *Proc. IEEE ICC*, 2001.

[178] C. Zhang, M. Zhou, and M. Yu. Ad Hoc Network Routing and Security: A Review. *International Journal of Communication Systems*, 20(8):909–925, 2007.

[179] S. Zhong, J. Chen, and Y. Yang. Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *IEEE INFOCOM*, pages 1987–1997, 2003.

[180] T. Zimmerman. *Personal area networks (PAN): Near-field intra-body communication*. PhD thesis, Master Thesis - Massachusetts Institute of Technology, 1995.

[181] G. Zussman and A. Segall. Energy efficient routing in ad hoc disaster recovery networks. *Ad Hoc Networks*, 1(4):405–421, 2003.

[182] G. Zussman, A. Segall, and U. Yechiali. Bluetooth time division duplex-analysis as a polling system. In *Proc. 1st IEEE Conference on Sensor and Ad Hoc Communications and Networks*, 2004.