uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Yassine Daadaa

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M. (Computer Science)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Bluetooth Scatternet Formation protocol for wireless devices based on Maximal Independent Sets

TITRE DE LA THÈSE / TITLE OF THESIS

Nejib Zaguia

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Amiya Nayak

Nicola Santoro

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

# Bluetooth Scatternet Formation protocol for wireless devices based on Maximal Independent Sets

by

**Yassine Daadaa**

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For
**Master of Science**
**In**
**Computer Science**

Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering
University Of Ottawa

*Your file Votre référence*
*ISBN: 978-0-494-50869-5*
*Our file Notre référence*
*ISBN: 978-0-494-50869-5*

The undersigned hereby recommend to
The faculty of Graduate Studies and Research
acceptance of this thesis

# Bluetooth Scatternet Formation protocol for wireless devices based on Maximal Independent Sets

Submitted by

**Yassine Daadaa**

In partial fulfillment of
the requirements for the degree of
**Master of Science**
**In**
**Computer Science**

---

Nejib Zaguia, Ph.D.
(Thesis Supervisor)

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Abstract

Bluetooth standard allows the creation of piconets, with one node serving as its master and up to seven nodes serving as slaves. Given a set of Bluetooth nodes that are positioned so that their unit disk graph is connected, the Bluetooth scatternet formation (BSF) problem is to select piconets, and the master and slave roles within each piconet, so that the obtained scatternet is connected, has some desirable properties, and shows good performance with respect to certain metrics.

In this thesis, we propose a new Bluetooth scatternet formation protocol called BlueMis that is based on maximal independent set and guarantees connectivity and degree limitation. Three variations of the protocol are introduced with the basic idea of selecting slaves as the maximal independent set of each Bluetooth device. First, BlueMis as a simple unique iteration protocol is introduced, and then two iteration protocols called BlueMis1 and BlueMis2 are proposed. In the first iteration of BlueMis1 a piconet containing a maximal independent set is constructed for every device, while the second iteration attempts to simplify the scatternet structure and to delete piconets not essential to the connectivity. A major advantage of this novel protocol is its simplicity.

We implemented our proposed protocol and compared it to the well known protocol BlueMesh with respect to certain relevant metrics. Simulations show its advantage over the best competing protocol, especially for moderately dense networks.

# Acknowledgment

First and foremost, I would like to thank that eternal power which exists in this world that we call GOD, who has helped me in a number of ways. I have faced many challenges during this eventful journey and I am grateful to GOD for helping me come out of all of those safe and sound.

I am very thankful to my advisor Prof. Nejib Zaguia, who has always played the role of a lighthouse for me. He has shown me the correct direction for my research and saved me from veering in any wrong direction. He has always been more than just an advisor, being a friend, philosopher, and guide to me. I enjoyed working with him, learning from his past experiences and understanding how to improve myself in different ways. He has always motivated me to work hard and also showed me the importance of accepting others' research opinions while studying. His constant drive for perfection has helped me throughout this work.

I am deeply grateful to Prof. Ivan Stojmenovic for his detailed and constructive comments, his important support throughout this work, and his crucial contribution, which made him a backbone of this research and so to this thesis.

This thesis is dedicated to my family, my father Mohamed Taher, my mother Aziza, my sisters Mouna, Salsabille, and Sana, and my brothers Sofiene and Ahmed. Without their love, support, and motivation, I would not have reached this stage and this work would have been far from complete. My parents have always been more than a friend to me and have provided me with valuable suggestions and shared their experience from time to time. They have shown me a path to walk on with their own footprints. They have given me the inspiration needed to face all the twists and turns of life with courage.

Finally, I would like to thank all those who have helped me make this dream come true.

# Chapter 1

# Introduction

## 1.1. Background

Mobile ad hoc networks (MANETs), or simply ad hoc networks, have received significant attention in recent years due to their potential applications in battlefield, emergency disaster relief, and other application scenarios. In Latin, ad hoc literally means "for this," further meaning "for this purpose only" and thus usually implying temporary nature. Ad hoc networks consist of a collection of geographically distributed nodes that communicate with each other over a wireless medium. Mobile nodes are connected together to form a network on the fly. Unlike wired and cellular networks, no wired backbone infrastructure is installed in ad hoc networks. Mobile ad hoc networks are self organizing, rapidly deployable, and require no fixed infrastructure. They are comprised of wireless nodes, which can be deployed anywhere, and which must cooperate in order to dynamically establish communications using limited network management and administration. Nodes in an ad hoc network may be highly mobile or stationary, and may vary widely in terms of their capabilities and uses. The objective of this type of network is to accomplish increased flexibility, mobility, and ease of management relative to infrastructured wireless networks. Ad hoc networks usually have no fixed network infrastructure for routing traffic and transmission range is limited by power constraints, frequency reuse, and channel effects. Bluetooth and Wi-Fi are two widely adopted technologies for wireless communication between users/devices. They are competing and complementary at the same time, since Bluetooth is more suitable for short communications and provides direct communication between any two neighbouring nodes, while Wi-Fi requires access points and is generally applied to somewhat longer distances.

Bluetooth technology [SIG] is emerging as one of the most promising enabling technologies for ad hoc networks. Bluetooth is a short-range radio Frequency (RF) specification for short-range, point-to-point, and point-to-multi-point voice and data transfers intended to replace the cable(s) connecting portable and/or fixed electronic devices (*Figure 1.1*). Key features are ad hoc connectivity, robustness, low complexity, low power, and low cost [SIG]. It operates in the 2.4 GHz unlicensed Industrial, Scientific and Medical (ISM) band, and the regulatory range of this frequency band is 2.400 – 2.4835 GHz. Bluetooth devices collect themselves into a piconet which is the basic Bluetooth networking unit. A piconet is defined as a Bluetooth network with no more than eight active devices, arranged in a star shape consisting of one master device and up to seven slave devices. The piconet uses a specific frequency hopping pattern that can be determined by observing certain fields in the Bluetooth address and clock of the master. The hopping pattern is of pseudo-random ordering that goes through the 79 frequencies in the ISM band. Bluetooth offers full-duplex communication by employing a Time-Division Duplex (TDD) scheme. Before any two Bluetooth devices can communicate with each other, they must go through the device discovery procedure which consists of two phases: inquiry and paging. During the Inquiry phase defined by Bluetooth standard [SIG] as an asymmetric process, in order for a device to discover other devices, it uses the inquiry channel and acts as a master, while the device to be discovered will have to use the inquiry scan channel and act as a slave. The limit of only 8 active devices in a Bluetooth piconet can be overcome by the formation of scatternets. Scatternet formation describes the joining of piconets for larger networks and the routing of messages through it. The Bluetooth specification describes techniques for device discovery and for the participation of a node in multiples piconets. However, solutions for scatternet formation are not provided. Since scatternet formation is not formally defined in the Bluetooth specification, many issues are still under work and remain open. In this thesis we will study the scatternet formation problem.

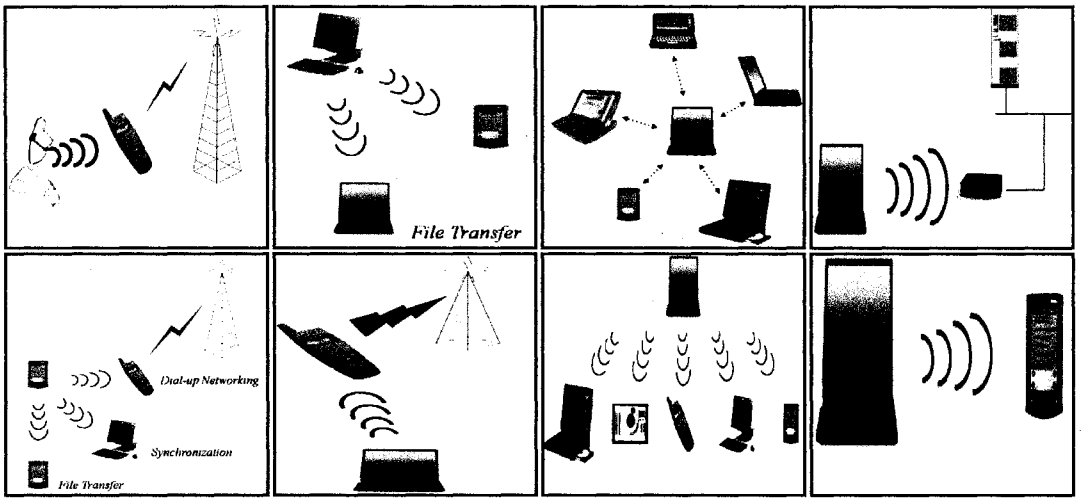


**Figure 1.1:** Various Bluetooth usage scenarios (Adopted from [SIG])

## 1.2. Problem statement

The Bluetooth specification explores techniques related to device discovery and the participation of a node in multiples piconets. Each piconet consists of a master and up to seven active slaves. The network topology resulting from the connection of two or more piconets is called scatternet. However, the Bluetooth specification does not define the structure of the scatternet. In the past few years, scatternet formation has become an area of intense research activity. The problem of scatternet formation concerns the grouping of network devices into piconets (piconet formation), and the joining of the piconets into a connected scatternet (piconet interconnection). Scatternet formation protocols should deal with major issues such as the mobility of the devices and their power efficiency. The network topology is modeled as a unit disks graph (*Figure 1.2*). Nodes are located in the Euclidean plane and are assumed to have an identical (unit) transmission radius. Ideal MAC layer with no collision is also assumed in protocol design. The proposed algorithm is localized, meaning that each node does not know about the global network configuration beyond its one or two hop neighbours. The nodes become aware of their two hop neighbours through the neighbour discovery (or device discovery) phase. The device discovery phase concerns the discovery of each device's one and two hop

neighbouring devices; therefore, each node discovers its one hop neighbours and then it exchanges information about each one hop neighbour with all of its other one hop neighbours. Consequently each node will have two hop knowledge at the end of this phase. Almost all proposed Bluetooth scatternet formation protocols assume that Bluetooth technology is used for both neighbour discovery and data communication in the created scatternet. Each node has to become aware of its neighbours, since the devices initially have no knowledge of their surroundings or other Bluetooth devices; a phase of device discovery has consequently to be performed before the actual scatternet formation process takes place. There are several criteria used in the literature to evaluate scatternet formation protocols. For instance, the following are commonly used criteria: connectivity for all devices, number of piconets in the scatternet, average piconet size, number of roles per device, scatternet formation delay, and the minimum number of hops between any pair of devices in the scatternet.

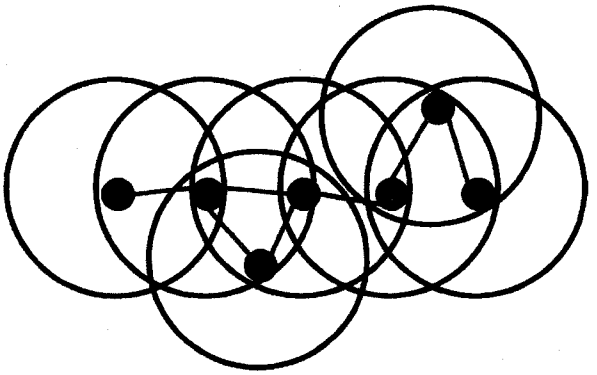

**Figure 1.2:** Unit Disk Graph (UDG)

## 1.3. Existing solutions

Numerous approaches for scatternet formation have been proposed in literature since the Bluetooth specification was published. The Bluetooth scatternet formation algorithms proposed in the literature may be classified into two categories: The first category works properly for single hop networks, since it assumes that all Bluetooth devices are within radio transmission range of each other. The second category functions properly for multi hop networks (and for single hop networks as well), since it assumes that all devices are not necessarily within transmission range

of each other. Single hop solutions such as the ones proposed in [ARKS], [SBTL], [LMS], and [PK] require that all devices are within each other's transmission range. The first known solution is proposed in [SBTL]. The proposed solution is based on using a leader election process to find a coordinator that collects topology information and then assigns roles to devices and forms the topology. Then a centralized algorithm is applied at the leader to assign roles to the devices on the network. To achieve desirable scatternet properties, the centralized scheme executed by the leader requires that the number of devices in the networks is less than or equal to 36. Several papers deal with the more general case of multi hop topology (e.g., [ZBC], [S], [LSW], and [PBC]). In [ZBC] the first multi hop scatternet formation protocol is proposed, where the resulting topology is termed a Bluetree. The number of roles held by each node is limited to two or three, the proposed solution is not time efficient, and the protocol assumes that all nodes start the formation process at the same time. In [S] a three phase dominating set based scatternet formation protocol for Bluetooth networks is proposed. In the first phase, the unit graph is constructed and a localized sparse graph is extracted. In the second phase, a degree reduction technique Yao sub-graph is applied simultaneously to all nodes with excessive degree (more than seven) in order to limit the degree of each node to seven. Finally in the third and last phase, master-slave roles are assigned based on dominating set membership. However no thorough evaluation of this protocol is provided and this technique requires each node to be equipped with additional hardware to provide information on its current location. This hardware may be expensive, and the proposed solution is not feasible in its absence. In [PBC] a Bluetooth scatternet formation protocol called BlueMesh is proposed. BlueMesh is a two phase protocol for scatternet formation that guarantees connectivity and limits the number of slaves in each piconet. The first phase concerns the discovery of each device's one and two hop neighbouring devices. The second phase begins as soon as the first phase is over and proceeds in successive iterations through which connectivity is achieved progressively. This phase thus takes care of scatternet formation and piconets formation and their interconnection to form the scatternet. Slave selection is performed in such way that if a master has more than

seven neighbours, it chooses up to seven slaves among them so that it can reach all of the others via the chosen one. The protocol may show weaknesses in term of the worst-case number of slave roles that a node can assume. For instance, in the case of dense networks (e.g., a complete graph), the second largest node in a neighbourhood may end up serving as slave to all the masters in the same neighbourhood. Also the number of iterations is unlimited. Nevertheless, among all of the methods that do not use position information, BlueMesh currently appears to be the best available method for multi hop networks.

## 1.4. Objective

Due to the limitations of existing solutions, it is important to develop a suitable method for Bluetooth scatternet formation. The goal of this thesis is to propose a new protocol for Bluetooth scatternet formation: the proposed BlueMis protocol attempts to simplify the BlueMesh procedure, without losing the advantages of BlueMesh over other scatternet formation protocols. A scatternet formation algorithm should not depend on a central component. We only consider localized protocols, in which each node makes formation decisions based solely on the information provided from its neighbours (possibly also two hop neighbours). The most relevant questions and challenges that a Bluetooth scatternet formation algorithm must consider include determining how nodes select their role (master or slave), which piconets a slave node should join, how many slaves a master should accept (below the specified number of seven), how many piconets a bridge node should belong to, and whether a master should serve as a slave in other piconets?. The most relevant criteria to be considered when designing a scatternet formation algorithm based on the above questions are as follows: minimizing the bridging overhead between connected piconets, minimizing the number of links in scatternet, maximizing the size of piconets to predetermined capacity, minimizing the number of piconets in scatternet, minimizing the network diameter, minimizing the link active mode, minimizing the average shortest path in the scatternet, maximizing the data flow between various sources and sinks simultaneously, and minimizing the number

of roles played by a Bluetooth device. Protocols can be further evaluated according to their message complexity, time complexity, and memory requirements.

## 1.5. Contributions

This thesis makes two major contributions. First, we develop an extensive and comprehensive survey of what we consider to be the most important existing schemes in the literature. Second, we develop a new protocol for Bluetooth scatternet formation. Our proposed protocol is called "Bluetooth Scatternet formation of wireless devices based on maximal independent sets" (BlueMis). Our approach is based on the idea of constructing piconets in which the slaves represent a maximal independent set (MIS) of the neighbours of the master node, and we implement three different variations of the protocol. Our protocol attempts to simplify the BlueMesh procedure. BlueMis is a two phase protocol, in which the first phase is discovery and the second phase is scatternet formation; it essentially interprets the slave selection as the maximal independent set problem. In the discovery process, two hop neighbours are discovered, as in BlueMesh; this step is required to run the maximal independent set based slave selection. The proposed protocol is based on applying certain key(s) for decisions. The second phase is executed in two iterations (unlike BlueMesh that uses on average 1.8 to 4.7 iterations). In the scatternet formation phase, each device creates its own piconet formed by devices selected in its maximal independent set; all created piconets are preserved if this is possible and meaningful. Two neighbours $A$ and $B$ could select each other as slaves, and thus a resolution to keep only one relationship is based on an arbitrary key. Thus, the node with the higher key remains the master in a symmetric relationship. Two elimination procedures are applied to remove unnecessary piconets: if a master has slaves that are all masters and not slaves in other piconets, then it becomes slave of all of its slaves, on the other hand, if a master that is not slave in any other piconet has only one slave, then it becomes a slave and the slave becomes the master. If, after such applications, a certain node remains without slaves, it does not need to keep its piconet. Several versions of BlueMis will be presented depending on the keys used for decisions and also with the

application of other simplification procedures to the initially formed scatternet. BlueMis has certain desirable properties such as the obtained scatternet being connected, the protocol being executed at each node with information available locally at the node itself, and guarantee degree limitation (no more than seven slaves in each piconet), thus avoiding the overhead of parking and unparking additional slaves. Notice that if a master has more than seven slaves, some slaves will have to be parked. To communicate with a parked slave a master has to unpark it, while possibly parking another slave. In the simulation, we choose a number of wireless nodes between 40 and 200 that are distributed randomly in a square area. Each node is specified by random $X$ and $Y$ coordinate values. Each node has $M$ neighbours on average. In our experiments, we focused on the following parameters: number of iterations, number of piconets, average number of slaves per piconet, average number of roles per device, average shortest-path length. The designed protocol is implemented and compared, in terms of various characteristics, with BlueMesh [PB; PBC], which we consider to be the best competing protocol, This choice is based on the fact that among all methods that do not use position information, BlueMesh has great advantages (No more than seven slaves per piconet, an average of 2.5 roles per node, route lengths not significantly longer than shortest path in the network) and appears to be currently the best available method for multi-hop networks. In our experiments, we tested BlueMis with several options available for selecting keys. Some options for the keys are node *ID*, *-ID*, (*degree*, *ID*), *(-degree*, *ID*), (*slave degree*, *ID*), (*slave degree*, *degree*, *ID*). The experiments show several advantages of our protocol over BlueMesh.

## 1.6. Thesis organization

The remainder of this thesis is organized as follows. We provide background information about Bluetooth technology in Chapter 2. A comprehensive survey on piconet formation, scatternet formation, and performance evaluations of scatternet formation protocols and scatternet formation metrics and issues are discussed in Chapter 3. The Bluetooth scatternet formation of wireless devices based on maximal

independent sets protocol (BlueMis) is presented in Chapter 4. The analysis and simulation of our proposed protocol and the results of its comparison with BlueMesh [PB; PBC] are presented in Chapter 5. Chapter 6 contains a summary of the work done in this thesis and directions for further research work.

# Chapter 2

# Overview of Bluetooth technology

The Bluetooth Special Interest Group (SIG) releases specifications and is composed of over 7000 companies. Bluetooth technology operates within 79 channels in the free radio band (between 2.4 and 2.48 GHz), also called the Industrial, Scientific and Medical (ISM) band. The key features of the technology are robustness, low power, and low cost. Bluetooth is a frequency hopping spread spectrum (FHSS).

## 2.1. Topology

Beyond a single device, the most basic network structure in Bluetooth is the piconet. A piconet is comprised of one master and up to seven active slaves. The master of the piconet polls each of its slaves in a Deficit Round Robin fashion, with only one device being allowed to transmit during a given slot. Slave devices may not transmit unless they receive a transmission from their master, and can only start transmitting during an odd numbered time slot. On the other hand, the master is allowed to start transmitting data in any even numbered slot.

### 2.1.1. Piconet

Bluetooth devices can interact with one or more other Bluetooth devices in several different ways. The simplest scheme is when only two devices are involved. This is referred to as point-to-point. One of the devices acts as the master and the other as a slave. This ad hoc network is referred to as a piconet (*Figure 2.1*). Consequently, a piconet is any such Bluetooth network with one master and one or more slaves. In the case of multiple slaves, the communication topology is referred to as point-to-multipoint. In this case, the channel (and bandwidth) is shared between all the devices in the piconet. There can be up to seven active slaves in a piconet. The

slaves of a piconet are synchronized to the master's frequency hop sequence, which is particular to each piconet. The slaves calculate this particular hop sequence using the master's Bluetooth device address (BD_ADDR) and clock information, which are exchanged during the master-slave connection establishment procedure. To identify each slave, the master assigns a locally unique active member address (AM_ADDR) to the slaves participating in active communications in the piconet.

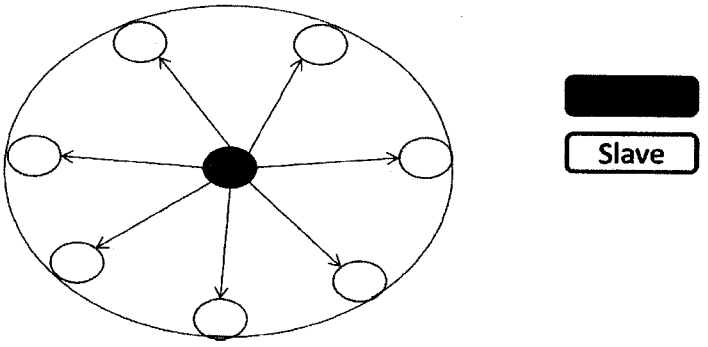


**Figure 2.1:** Piconet

### 2.1.2. Scatternet

Piconets can coexist in time and space since each piconet uses a different frequency hop sequence. The network formed by connecting several piconets via shared nodes is called a scatternet (*Figure 2.2*); slaves in one piconet can participate in another piconet as either a master or slave and are named bridges. This is accomplished through time division multiplexing. In a scatternet, the two (or more) piconets are not synchronized in either time or frequency. Each of the piconets operates in its own frequency hopping channel while any devices in multiple piconets participate at the appropriate time via time division multiplexing. Bridge nodes can be either a slave shared between two piconets, or a node that is a master of one piconet but a slave in another. Bridge nodes can be shared between more than two piconets, but a node can only be the master of one piconet. This restriction exists because the master is the node that dictates the hop sequence of its piconet, and having a node act as a master in two or more piconets would result in those piconets having matching

hop sequences. Bridge nodes are able to participate in multiple piconets by performing time division multiplexing.

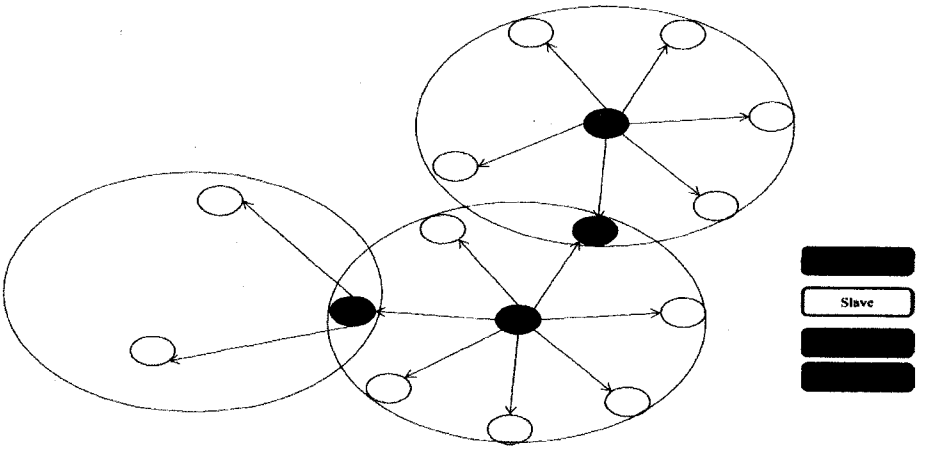


**Figure 2.2:** Scatternet

## 2.2. Device states

A Bluetooth device can be in one of the following states:

**Standby:** The default state and is a low power mode. In this state, the radio transceiver is turned off.

**Inquiry:** During the Inquiry (discovery) phase, a device may discover other Bluetooth devices that are within its transmission range and are operating in the corresponding mode: Inquiry Scan. During the inquiry mode the device is continuously transmitting inquiry messages at different hop frequencies. Between inquiry transmissions, the device listens for inquiry responses. Two inquiry messages are sent in each transmission slot, and the device looks for FHS packets in the reception time. This process is run until another device is contacted and enough responses are received, or until the inquiry timeout has elapsed. Inquiry messages contain no information about the source node, but the FHS packets that are received contain essential information necessary for creating a connection with the sending device. During the inquiry process, a General Inquiry Access Code (GIAC) can be used to look for any device in communication range, or any one of a number of

Dedicated Inquiry Access Codes (DIACs) can be used to look for a specific type of device.

**Inquiry Scan:** A device listens for inquiry messages. When the device receives a valid inquiry packet, it enters the inquiry response sub-state, where it responds to the inquiry packets with its own FHS packet. This packet contains essential connection setup information, such as the address and clock of the device. Both inquiry and inquiry scan utilize a special frequency hopping sequence to reduce the time required for obtaining a frequency match. Although the inquiry frequency hopping sequence is fast, to allow the devices to obtain synchronization the inquiry scan sequence is necessarily slow.

**Page:** The device that will become the master of the connection enters the page state. In this state, the device transmits paging messages to the desired device. When the recipient acknowledges the packet, the master enters the master response sub-state and sends its own FHS packet.

**Page Scan:** The device listens for inquiry messages. The device seeks to become the slave of the connection to the corresponding device in page mode. Once the device receives the page message from the other device, it acknowledges the packet and enters the slave response mode. The device waits for the FHS packet from the soon-to-be master, and upon reception it updates its own timing to match the new master before moving to the connection state.

**Connection:** In the connection state, the slave switches to the master's clock. By doing so, the slave assumes the frequency hopping and timing sequence of the master. After this has occurred, the master and slave can communicate.

## 2.3. Bluetooth protocol stack

The Bluetooth protocol stack (*Figure 2.3*) is composed of layers. The Bluetooth Radio is the lowest defined layer of the Bluetooth specification. It defines the requirements of a Bluetooth transceiver device operating in the 2.4GHz ISM (Industrial Scientific Medical) band and uses a fast (1600 hops/sec), frequency

hopping, spread spectrum (FHSS) technique. Modulation is based on a Gaussian frequency shift keying (GFSK) with a band rate of 1 Msymbol/s which is equivalent to a raw transmission speed of 1Mb/s. The Baseband is the physical layer of Bluetooth. It manages physical channels and links apart from other services like error correction, data whitening, hop selection and Bluetooth security. The Baseband layer lies on top of the Bluetooth radio layer in the Bluetooth stack. The baseband protocol is implemented as a Link Controller, which works with the Link Manager for carrying out link level routines like link connection and power control. The baseband also manages asynchronous and synchronous links, handles packets, and does paging and inquiries to access and inquire about Bluetooth devices in the area. The baseband transceiver applies a time-division duplex (TDD) scheme (alternates transmitting and receiving). Therefore, apart from different hopping frequencies (frequency division), the time is also slotted. The Link Manager carries out link setup, authentication, link configuration, and other protocols. It discovers other remote LMs and communicates with them via the Link Manager Protocol (LMP). To perform its service provider role, the LM uses the services of the underlying Link Controller (the Baseband). The Host Controller Interface provides a command interface to the baseband controller and link manager, and access to hardware status and control registers. This interface provides a uniform method of accessing the Bluetooth baseband's capabilities. The Logical Link Control and Adaptation Protocol is layered over the Baseband Protocol or HCI if available, and resides in the data link layer. L2CAP provides connection-oriented and connectionless data services to upper layer protocols with a protocol multiplexing capability, segmentation and re-assembly operation, and group abstractions (one to many). The RFCOMM protocol provides an emulation of serial ports over the L2CAP protocol. This "cable replacement" protocol emulates the RS-232 control and data signal over the Bluetooth baseband, providing transport capabilities for upper level services that use a serial line as a transport mechanism. The Service Discovery Protocol provides a means for applications to discover which services are available and to determine the characteristics of the available services. This protocol is needed in the Bluetooth environment, as the set of available services

changes dynamically based on the proximity of devices in motion. The SDP defined in the Bluetooth specification is intended to address the unique characteristics of the Bluetooth environment. The Telephony Control Specification defines how a Bluetooth enabled device can be used as a wireless phone and how a Bluetooth enabled mobile phone should switch to Bluetooth enabled wireless phone-function when it comes within reach of a Bluetooth enabled base station.

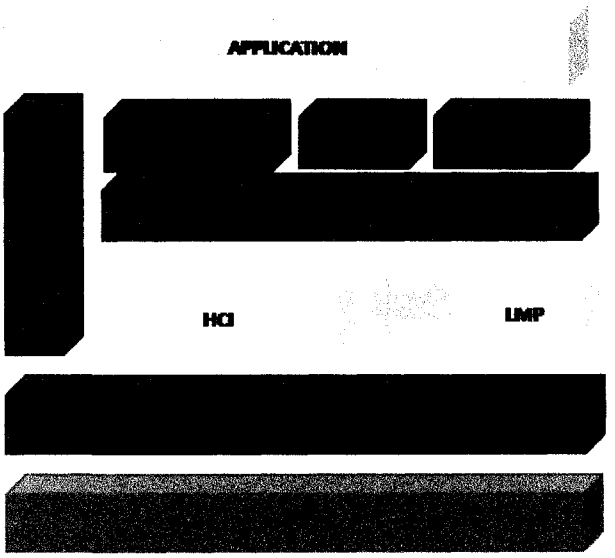


**Figure 2.3:** Bluetooth protocol stack

## 2.4. Bluetooth link establishment

Link establishment (*Figure 2.4*) is specified in the Bluetooth specification [SIG] as a two phase mechanism. The first phase involves devices going into the Inquiry or Inquiry Scan state. The second phase involves devices going into the Page or Page Scan state. Once the devices finish the Page or Page Scan state, a master-slave relationship is formed. The first phase in the piconet formation is to discover neighbouring devices. Each device alternates between the Inquiry and Inquiry Scan states, using 32 dedicated frequencies hopping. A device that wants to discover other Bluetooth units within range enters the inquiry state. It continuously transmits inquiry messages at different hop frequencies. Between inquiry transmissions the unit listens

for responses. If a response is received, it is not acknowledged and the inquiring device continues with the inquiry transmissions. The device leaves the inquiry state either when it has received a predetermined number of responses or when the InquiryTO timer runs out. The inquiry message broadcast by the source does not contain any information about the source. However, it may indicate which class of devices should respond. There is one general inquiry access code (GIAC) to inquire for any Bluetooth device, and a number of dedicated inquiry access codes (DIAC) that only inquire about certain types of devices. A device that wishes to be discovered by other Bluetooth devices in range enters the inquiry scan state. A scanning device listens for an inquiry access code (IAC) for $T_{W\_Inquiry\_Scan}$ seconds. $T_{W\_Inquiry\_Scan}$ should be long enough to scan 16 frequencies. Because scanning 16 frequencies lasts 10ms, $T_{W\_Inquiry\_Scan}$ should also be 10ms. During these 10 ms the receiver of the scanning device listens on a single frequency determined by the inquiry scan hopping sequence and the current value of the device's clock. The scanning device changes its listening frequency, according to the inquiry hopping sequence, every 1.28s. Then the paging phase can commence. When the inquiring Bluetooth device wants to make a connection to an inquiry scanning device, it pages that device. Paging means sending an *ID* packet containing a certain Device Access Code (DAC) over and over again until a response is received. The inquiring device is the master of the communication, while the inquiry scanning device is the slave. The master does not know exactly when the slave will wake up or on which hop frequency; therefore, it transmits a train of identical DACs at different hop frequencies and listens in between for responses. The master uses the slave's BD_ADDR and an estimate of the slave's clock to determine the page hopping sequence. To compensate for the uncertainty in the knowledge of a slave's clock, the master will send its page message during a short time interval on a number of wake-up frequencies. During each transmission slot the master sequentially transmits on two different hopping frequencies. The page hopping sequence of 32 frequencies is divided into two trains of 16 frequencies each and each train is repeated for Npage times or until a response is received. Paging continues until a response is received or timeout value PageTO is exceeded. Page scan works

similarly to inquiry scan; however, in page scan a device listens for its own unique DAC and it alone can respond. There are 32 paging frequencies, which comprise a page hopping sequence determined by the paged unit's BD_ADDR. Every 1.28s, a different listening frequency is selected as in inquiry scan. During a scan window, the unit listens on one frequency. Upon termination of this phase the connection has been established between the master and slave, and they can communicate. The described connection establishment procedure is guaranteed to work. However, the time for it to work can be anywhere from 1.28s to over 80s. This is too large a time range for something as simple and as vital as connection establishment. Moreover, the proposed procedure is complicated – it involves the calculation of two hopping sequences and depends significantly on perfect timing synchronization between the clocks of the two devices involved. The existing procedure is also asymmetrical – it places the burden of solving the uncertainty about timing and phase on the paging unit. This means more power expenditure for the master, although power consumption is of serious concern for a Bluetooth device. However, physically the Bluetooth master and the Bluetooth slave are identical devices. This issue is addressed in [SBTL], where a symmetric protocol for link formation is proposed.

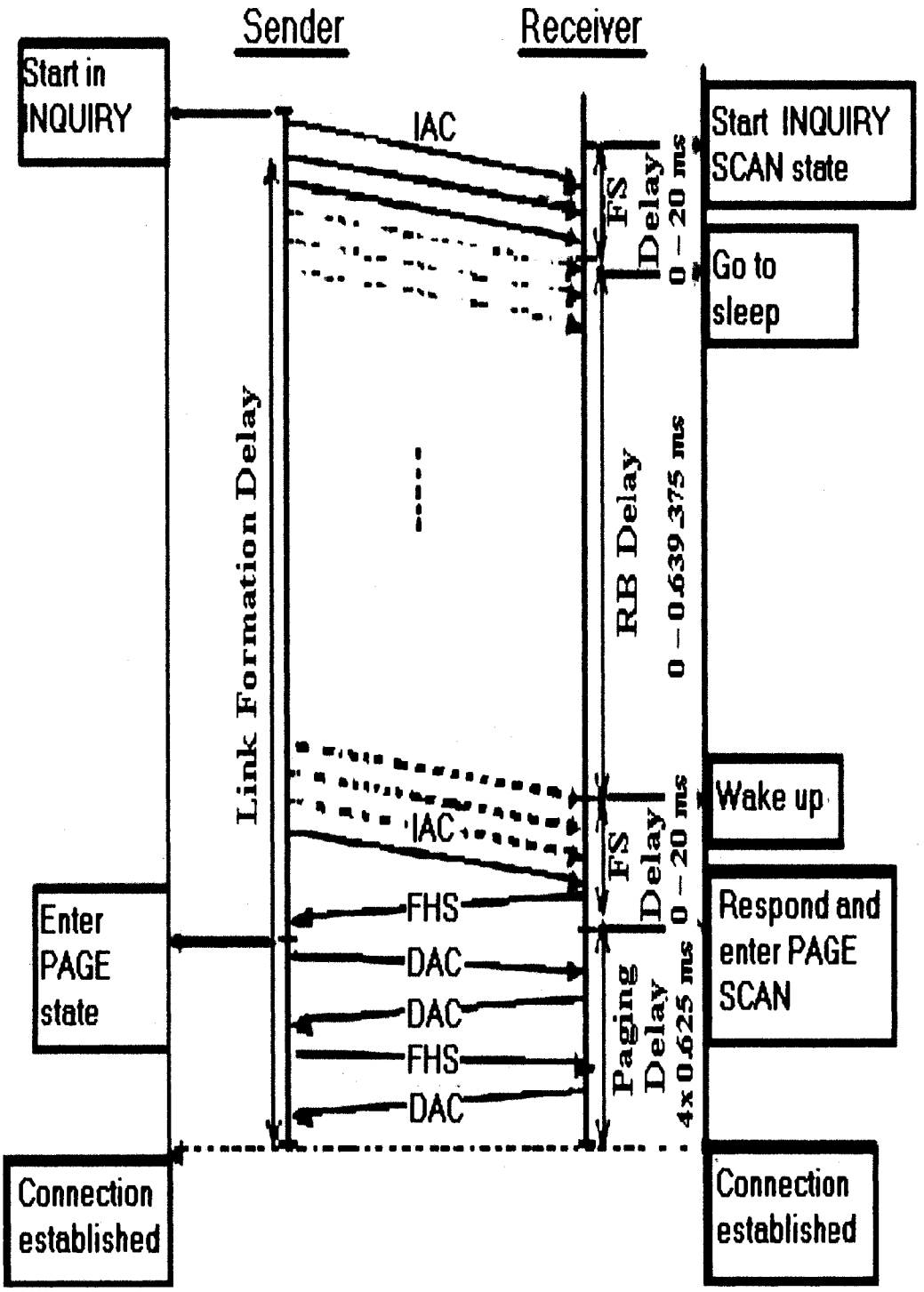


**Figure 2.4:** Bluetooth link formation ([SIG])

# Chapter 3

# Literature review

In this chapter, we review the literature related to the scatternet formation problem and other topics linked to it such as graph theoretical concepts, scatternet formation metrics, and performance evaluations of scatternet formation protocols. The problem of scatternet formation concerns the grouping of network devices into piconets, and the joining of these piconets into a scatternet. This operation requires the devices to be aware of their neighbours. The device discovery process and the formation of point-to-point links have to be performed before the actual scatternet formation phase can take place. In the device discovery phase, each pair of devices learns of each other's identities and synchronizes their hopping sequences to form piconets. The link formation process described by Bluetooth is asymmetric since the proposed protocol distinguishes between the sender and the receiver. In [SBTL], a symmetric link formation protocol is developed, that is symmetric in the sense that no sender or receiver allocation is needed. In [BBFMSK], simulations show that the device discovery process is the most time-consuming operation, independent of the particular Bluetooth scatternet formation protocol to which it is applied.

Many scatternet formation protocols are proposed in the literature; however, the majority of the work has been on simulations and there do not seem to be many implemented protocols. The proposed protocols can be classified in several ways.

We can distinguish between single hop and multi hop solutions. Single hop solutions assume that all devices are within radio transmission range of each other. This assumption simplifies scatternet formation, because local knowledge at each device constitutes global network knowledge; however this is not a realistic scenario. The more realistic multi hop solutions allow devices to join the scatternet if they are within transmission range of at least one participating device.

The next classification is based on whether the formation uses a coordinator [SBTL], [RKSA], [ZBC], and [LTC], or is completely distributed [LMS], [WSL], and [ZHS]. Coordinated topologies are formed by an omnipotent device that has been elected to coordinate the formation, while distributed approaches do not depend on a single device to form the scatternet.

## 3.1. Basic Graph Theoretical concepts used in scatternet formation protocols

For completeness, we review in this section some basic graph theoretical concepts that are used frequently and are the basis of several scatternet formation protocols.

### 3.1.1. Independent Sets

Let $G = (V, E)$ be an undirected graph, where $V$ is the set of vertices and $E$ is the set of edges. A set of vertices $I \subseteq V$ is called an independent or stable set if no two members of $I$ are adjacent in $G$ $(i.e.\ \forall v, w \in I, (v, w) \notin E)$. A maximal independent set (MIS) (*Figure 3.1*) is an independent set for which no proper superset is also an independent set. In other words, a maximal independent set is a set of vertices in a graph such that there is no edge between any pair of vertices, and such that no more vertices can be added and it is still an independent set.

A dual concept to the independent set is a clique. A clique of a graph $G(V, E)$ is a subset $V' \subseteq V$ such that every two vertices of $V'$ are joined by an edge in $E$. Notice that a set of vertices is independent in $G$ if and only if $I$ is a clique in the complement graph of $G$.

Although finding a maximal independent set in a graph is an algorithmically easy problem, it is NP-complete to decide whether there is maximal independent set of size $K$ in a graph. Therefore, the problem of finding a maximum independent set in an arbitrary graph $G$ is also an NP-complete problem. The maximum independent set problem is to find a maximum size set $V'$ inducing an independent set in $G$ (a maximum size independent set).

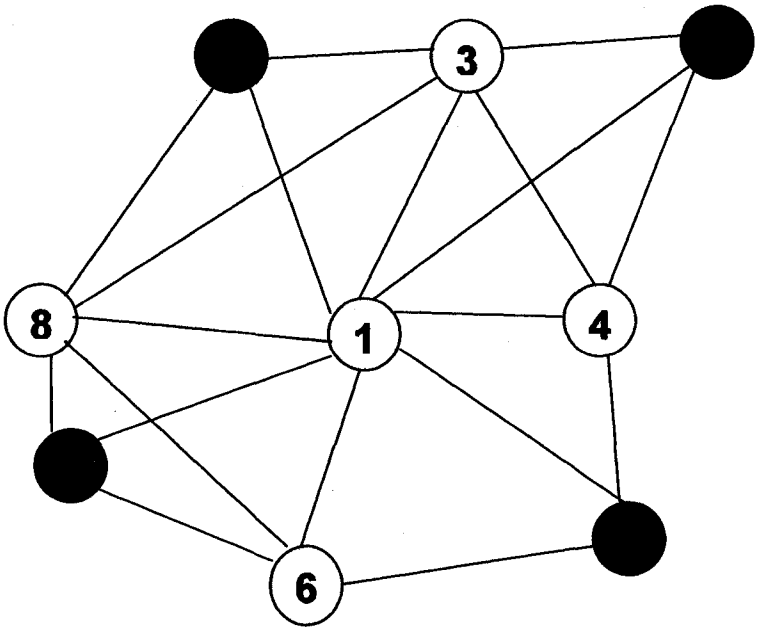


**Figure 3.1:** A Maximal Independent Set (MIS) for the set of neighbours of node 1

### 3.1.2. Dominating Sets

A connected dominating set (CDS) (*Figure 3.2*) in a graph is a subset of vertices such that every vertex is either in the subset or adjacent to a vertex in the subset and the subgraph induced by the subset is connected. The minimum connected dominating set is such a vertex subset with minimum cardinality. The dominating set problem (DS) is to determine whether there is a dominating set of size $K$ or less for $G$. In other words, we want to know if there is a subset $D$ of $V$ of size less than or equal to $K$ such that every vertex not in $D$ is joined to at least one member of $D$ by an edge in $E$. The problem of minimum connected dominating set (MCDS) is to compute a connected dominating set of minimum cardinality.

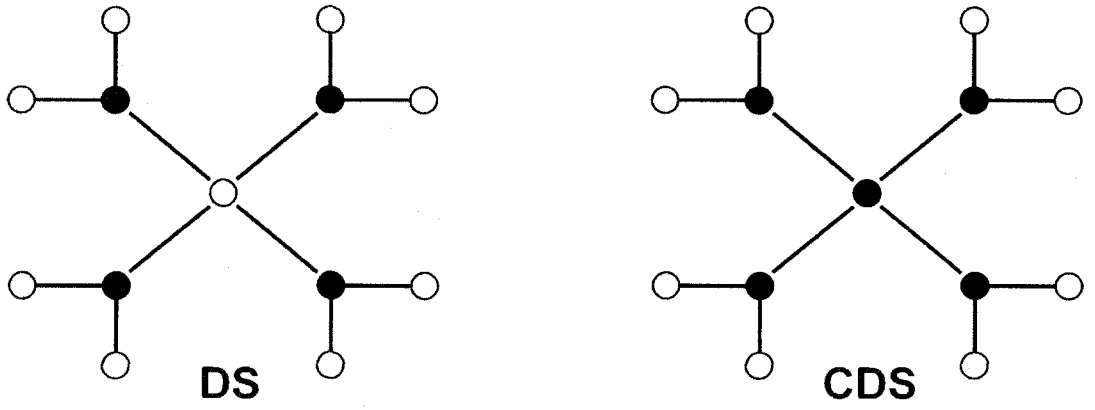


**Figure 3.2:** Dominating Set (DS) and Connected Dominating Set (CDS)

### 3.1.3. Unit Disk Graph

In a unit disk graph (UDG), all edges are less than or equal to a fixed threshold value. Many important and difficult graph optimization problems such as maximal independent set, graph coloring, and minimum dominating set can be approximated efficiently by using the geometric structure of a unit disk graph, in which nodes in the Euclidean plane are considered to define a disk of unit radius. In a unit disk graph [CCJ], there is an edge between two nodes $u$ and $v$ if and only if the Euclidean distance between $u$ and $v$ is at most 1.

Unit disk graphs have proven to be useful in modeling various physical real world problems. The field of wireless networking is a prominent application of unit disk graphs, where a unit disk graph represents an idealized multi hop radio network. Nodes are located in the Euclidean plane and are assumed to have an identical (unit) transmission radius. They can communicate only if they are within mutual transmission range. In a wireless network, a message sent by a node reaches all of its neighbours within its transmission range. If the transmission range of each node is $r$, then we can represent the wireless network by a unit disk graph where an edge connects two nodes if they are within each other's transmission range (i.e. the distance between them is $\leq$ r or within the circle of radius $r$ (*Figure 1.2*)).

### 3.1.4. Yao Graph

The basic idea underlying the Yao graph (YG) (*Figure 3.3*) is to cut the space around each node into sectors of equal angle and to connect each node to the nearest neighbour in each of its sectors. It has been used by Yao in order to construct efficiently a minimum spanning tree (MST) of a set of points in high dimensions and applied in [LSW] to guarantee degree limitation in a scatternet. At a given node $u$, any $k$ equally separated rays originated at $u$ define $k$ cones. In each cone, the closest node $v$ within the transmission range of $u$ is chosen, if there is any, and a directed link is added between $u$ and $v$. Ties are broken arbitrarily, and the remaining edges are deleted from the graph. This construction can be carried out at each node in the graph in several different ways. One choice is to carry it out simultaneously at each node and with the option to keep an edge $(u, v)$, only if $u$ and $v$ mutually select each other. Another option is to keep directional edges as well (i.e., when one node has selected the other but not vice versa). We may also carry this process out first at node $u$, and then at node $v$. In this case, if $u$ has not selected $v$, then edge $(u, v)$ is considered deleted by $v$ and is ignored when $v$ makes its decision afterward.

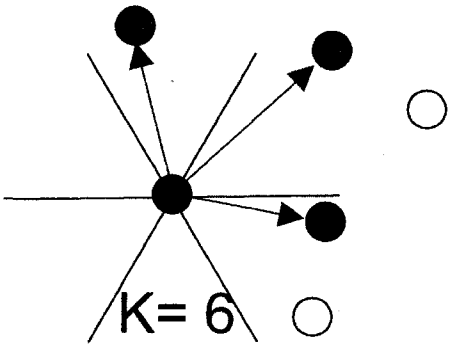


**Figure 3.3:** Yao Graph (YG)

### 3.1.5. Relative Neighbourhood Graph

The Relative Neighbourhood Graph, denoted by RNG (*Figure 3.4*), consists of all edges $(u, v)$ such that the intersection of two circles centered at $u$ and $v$ and with radius $|uv|$ does not contain any vertex $w$ from the set $S$. An edge $(u, v)$ exists between vertices $u$ and $v$ if the distance between them is not strictly the largest side in

any triangle $uvw$ for every common neighbour $w$ of $u, v$. In other words, in the RNG, two points, $u$ and $v$ are adjacent if there is no point that is simultaneously closer to both points than they are to one another. There is also another geometric way of viewing the RNG. For an edge $(u, v)$ to be included, the intersection of two circles with diameter $uv$ and centered at points $u$ and $v$ should not contain any other vertex. RNG is a connected and planar graph that can be constructed locally and contain an MST.

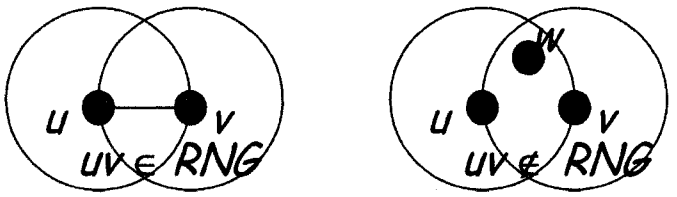


**Figure 3.4:** Relative Neighbourhood Graph (RNG).

### 3.1.6. Gabriel Graph

A Gabriel Graph (GG) (*Figure 3.5*) is a graph that contains an edge between points x and y if a disk with diameter $xy$ contains no other point. Formally, the GG of a set of points $S$ has an edge between $x$ and $y$ if and only if $d(x,y) \leq \sqrt{d^2(x,z) + d^2(y,z)}, \forall z \in S, z \neq x, y$. Notice that a Gabriel Graph is always planar.

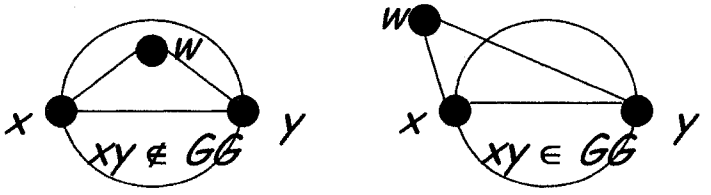


**Figure 3.5:** Gabriel Graph (GG).

### 3.1.7. De Bruijn Graph

The De Bruijn Graph (*Figure 3.6*), denoted by $B(d, k)$, is a directed Graph with $d \times k$ nodes where each node is assigned a unique label of length $k$ of the alphabet $\{0, 1 \ldots, d-1\}$. There is an edge in $B(d, k)$ from a node with label

$x_1x_2 \ldots x_n$ to $x_2x_3 \ldots x_k\, y$, where $y \in \{0, 1 \ldots d-1\}$. It is well known that the de Bruijn Graph enables self-routing inherently. The self-routing path from a source node with label $x_1x_2 \ldots x_k$ to a target node with label $y_1y_2 \ldots y_k$ is $x_1x_2 \ldots x_k \rightarrow x_2x_3 \ldots x_ky_1 \rightarrow x_3x_4 \ldots x_ky_1\, y_2 \rightarrow \cdots x_kx_1 \ldots x_ky_1 \ldots y_{k-1} \rightarrow y_1y_2 \ldots y_k$. Observe that we could find shorter route by looking for the longest sequence that is both a suffix of $x_1x_2 \ldots x_k$ and a prefix of $y_1y_2 \ldots y_k$. Suppose that $x_1x_2 \ldots x_k = y_1\, y_2 \ldots y_k$ is such a longest sequence. The shortest path between the source and the target is $x_1x_2 \ldots x_k \rightarrow x_3 \ldots x_ky_{k-i+2} \rightarrow \cdots \rightarrow x_{i-1} \ldots x_ky_{k-1} \rightarrow y_1y_2 \ldots y_k$. Clearly, the route between any two nodes is at most $k$ hops ($B(d,k)$ has diameter $log_d\, n$, where $n = dk$ is the number of nodes in the Graph). The classical De Bruijn is balanced in the sense that the labels of all nodes have the same length. A generalized de Bruijn Graph is pseudo-balanced if the lengths of the node labels differ by at most one.

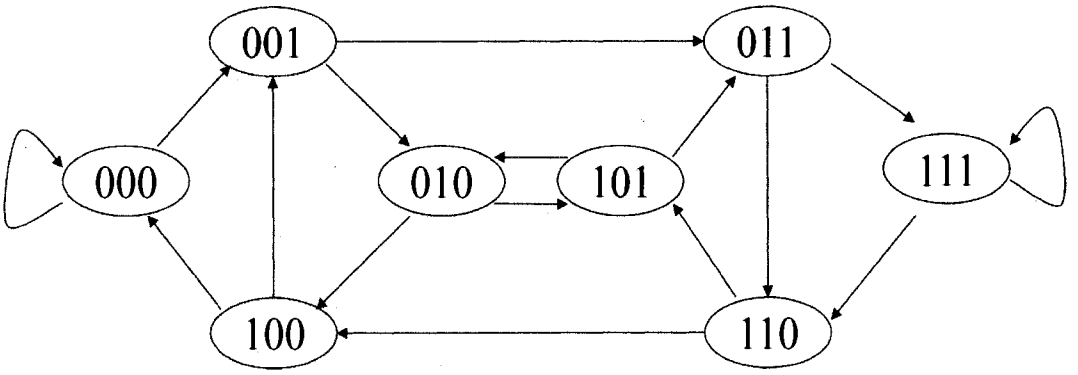


**Figure 3.6:** The de Bruijn Graph *B(2, 3)* [SLWW]

## 3.2. Piconet formation

Piconet link formation is described in Bluetooth specification [SIG] as an asymmetric process, while in [SBTL] a symmetric process for piconet link formation is described. The device discovery is performed by each node randomly entering into an inquiry or an inquiry scan mode (with equal probabilities), and randomly selecting the time length for being in that mode repeatedly until a timeout expires. Inquiry nodes select a repeated pattern of 32 frequencies (out of a total of nearly hundred available frequencies) and send signals on a selected frequency in a given spot.

Inquiry scan nodes also select a frequency at random in each spot and listen to the transmission at the selected frequency. The discovery (and establishment of the master-slave relationship) occurs when both sender and receiver nodes are at the same frequency. The timeout for the overall device discovery protocol (for finding a sufficient number of neighbours) is experimentally determined to have the best value of about 8 seconds. After the discovery phase, the data communication phase is directed by master nodes, which decide the timing and the series of frequencies to be used for frequency hopping based communication between devices. In [DHMPP], a protocol for Bluetooth device discovery is proposed; in this protocol, as soon as a device has discovered $c$ neighbours (i.e. seven), it proceeds to the next phase of piconet formation. Devices with fewer than $c$ neighbours will exit the device discovery phase upon timeout expiration.

## 3.3. Scatternet formation metrics

We must establish a set of metrics that are applicable to the majority of implementations in order to compare diverse scatternet formation protocols. In [MRTVJ] the notion of scatternet efficiency is introduced, and it is shown that efficiency is highly dependent on the characteristics of the network. A variety of metrics tailored to the specific needs of Bluetooth scatternet are presented. Two network characteristics, namely the amount of bridging overhead and the number of Bluetooth links, are found to have a major impact on system performance. Also, other aspects of scatternet formation, namely traffic dependency, the number of piconets, and the standard deviation of device degree, are found to affect performance, but this dependency is less significant compared to the impact of bridging overhead and the number of links. However, there is a variety of other metrics that also have to be considered, particularly the average visibility degree for all devices, the average connectivity for all devices, the number of piconets in the scatternet, the average piconet size, the number of roles per device, scatternet formation delay, the minimum number of hops between any pair of devices in the scatternet, the number of paths that exist between any pair of devices, and the number of devices pairs that use a given

device as a relay for communication. In our simulation some of the listed metrics are considered to test the protocol and compare it with BlueMesh [PB; PBC]. Following are the definitions of some of these metrics.

**Number of piconets in the scatternet:** This can be equated to the number of masters in the scatternet. Minimizing the number of piconets has a number of benefits, including an increase in the scatternet density and a reduction in interference. Also, master devices carry more data in the scatternet and therefore will consume more energy and other resources. Serving as the master of a connection carries a certain amount of overhead beyond the role of being a slave.

**Number of roles per device:** The number of roles that a device serves, which is equal to the number of piconets in which a given device is a member. A device in multiple piconets must split its time between each, increasing the load on that node. However, bridge devices are inherently involved in two or more piconets. As such, this metric can be a more direct indicator of the number of bridge devices.

**Piconet size:** The number of nodes, including the master, participating in a given piconet. Maximizing the piconet size will decrease the piconet density by decreasing the number of piconets in the scatternet.

**Scatternet Formation Delay:** The total time required to create a complete scatternet. In our simulation we consider the number of iterations in the protocol. Intuitively, this should be kept as low as possible to ensure the fastest topology formation, which allows useful communication to occur in the shortest amount of time.

**Average shortest path:** The average minimum number of hops between any pair of devices in the scatternet. Minimizing the shortest path results in fewer devices participating in the route and reduces the overall network load. The shortest path has a direct effect on the end-to-end delay between two devices.

Some other metrics can be used to test and compare protocols for scatternet formation, such as energy efficiency, the number of paths that exist between any pair

of devices, and devices arriving and/or exiting. Each formation algorithm must select carefully the metrics on which to focus.

## 3.4. Scatternet formation protocols

Bluetooth scatternet formation protocols are classified based on whether they are designed for single hop topology or multi hop topology. A second classification is based on whether the protocol is centralized or distributed. Finally we consider the principal contribution of the protocol as another classification.

### 3.4.1. Scatternet formation in single-hop networks

In single hop scatternet formation algorithms, it is assumed that all wireless devices are within transmission range of each other. The Bluetooth scatternet formation protocol designed for single hop scenario can be classified based on whether the formation uses a coordinator or is completely distributed.

#### 3.4.1.1. Coordinated Bluetooth scatternet formation protocols for single hop networks

In coordinated single hop solutions, topologies are formed by an omnipotent device that has been elected to coordinate the scatternet formation process. The elected node gains full knowledge of all other devices in the network and performs scatternet formation, informing each device of its role and the link between them. The coordinated schemes permit the application of traditional graph theoretical techniques to optimize the generated scatternet topology for data traffic. They assume the presence of a special device to gather the topology information, construct the topology graph, and start the scatternet formation process. Typically, such special devices are chosen through a leader election process. This process relies on empirical timeout values to ensure the correctness of the algorithm, and is difficult to auto-tune in real ad hoc scenarios. Furthermore, gathering topology information requires the formation of a large number of temporary piconets, which implies a large scatternet formation delay. Once the initial topology has been determined, it is impractical to re-

run the entire topology discovery phase to account for nodes leaving and joining the network at different times.

The first protocol for the topology creation of Bluetooth networks, called BTCP, has been introduced in [SBTL]. It is based on a leader election process. Leader-election scatternets are formed with the establishment of a leader that decides the role of every other node. For a leader to be selected, global knowledge of all nodes has to be obtained. The obtained topology must satisfy the following properties: degree constraint must be ensured, scatternet must be fully connected and consist of the minimum number of piconets, and two piconets must not share more than one bridge. BTCP is a three phase protocol consisting of the distributed leader election process, role determination, and actual connection. During the first phase of leader election, each device initially has a variable called VOTES which is initialized to 1. Each device alternates between the INQUIRY and the INQUIRY SCAN states to collect information; any two devices that discover each other enter a "one-to-one confrontation" and compare their VOTES variables. The device with the larger VOTES variable is the winner of the confrontation, and if the VOTES variables of the two devices are equal the device with the larger Bluetooth address wins the confrontation. The losing device (slave) gives up all of the FHS inputs it has collected to the winning device (master), tears down the connection it has established with those devices, and enters the PAGE SCAN state. Consequently, it will no longer be able to hear inquiry messages but only page message from devices that page it in the future. The leader elected at the end of the first phase has the FHS input of all of the devices participating. In the second phase, the leader checks whether the number of nodes that it has discovered during the first phase is less than eight. If this is the case, it pages and connects to all of the devices in the PAGE SCAN state and one piconet is formed with the leader as the master and all the other devices as its slaves. In this special case the protocol terminates at this point. If the number of discovered devices is greater than seven, then more than one piconet must be formed and interconnected via bridge devices. The leader, which has a global view of the network, makes decisions about the roles of each device and communicates them to individual

devices. The leader maintains a connectivity list set comprising the SLAVESLIST and the BRIDGELIST of each master elected. The advantage of this process is that any criteria for connection can be taken care of at the leader; these criteria must be communicated to the leader during the first phase in addition to the FHS information, and can help it in determining the role of devices in the final scatternet. The desired topology must be fully connected, have a minimum number of piconets, and satisfy the following properties: each master can have at most seven slaves, and two piconets do not share more than one bridge. With these conditions, the number of piconets satisfies $P = \left\lfloor \frac{17-\sqrt{289-8N}}{2} \right\rfloor$, while $1 \leq N \leq 36$. Due to the desired properties of the scatternet, and as we can observe from the above relation, the described scheme works only for a number of devices less than or equal to 36. During the third phase, each master pages and connects to the slaves and bridge defined in its SLAVELIST and BRIDGELIST respectively. As soon as a bridge receives a notification from its master that it is a bridge it waits to be paged by its second master, then it sends a CONNECTED notification to both masters. When a master receives a CONNECTED notification from all of its assigned bridges, a fully connected scatternet of $P$ piconets is guaranteed to be formed and the protocol terminates. Since the devices alternate between INQUIRY and INQUIRY SCAN states, it must be determined how long they will stay in these states without knowing whether they are a winner or loser. ALT_TIMEOUT is a timer specified for this purpose. The paper analyses the best value for ALT_TIMEOUT, the smallest value of which is determined to be 2527.223ms. BTCP is an excellent protocol for a static environment. If nodes are to enter and leave the network, the leader election process will have to be done repeatedly to ensure optimal routes and to repair potential lost/partitioned routes. BTCP assumes that all nodes are within communication range of each other, that there are at most 36 devices that wish to come together in one scatternet, and that once the scatternet is formed, there will be no additions to or removals from the network. The major limit of this protocol is that due to the full connectivity of the formed scatternet, the maximum number of devices that BTCP can handle is 36. Also, if the coordinator fails, the formation protocol has to be restarted, BTCP's timeout

value can affect the probability that a scatternet is formed, and BTCP is not suitable for dynamic environments where devices can join and leave after the scatternet is formed.

In [RKSA] a two phase scatternet formation protocol that partitions the network into independent piconets is proposed. In the first phase, each node identifies itself as either a master or slave. A master can be viewed as a clusterhead that elects a super master that knows about all the nodes; to construct the scatternet an asymmetric link formation protocol is used, and in the resulting topology the piconets are not interconnected. A reorganization process interconnects the piconets. Like most of the asymmetric link formation scatternet protocols, this protocol is randomized. The approach used here is centralized, which is impractical, and other limitations include that the structure of the linked scatternet has not been explored, and that the scatternet cannot perform communication and computation efficiently. Also, to make sure that there are appropriate numbers of masters and slaves at the end of the first stage, each node needs to know the number of participating nodes; however, in most applications this information is not available in advance. Moreover, too much time is spent on collecting information from the various piconets to decide whether the requirements for being connected have been met.

#### 3.4.1.2. Distributed Bluetooth scatternet formation protocols for single hop networks

In [LMS], a randomized distributed protocol for scatternet formation that assumes that all devices are in radio range of each other is proposed. This protocol has the objective of obtaining a guaranteed minimum number of piconets, low maximum degree of the devices, and low network diameter. The protocol partitions the nodes into components that consist of a single device or a piconet in which the master is the leader of the component. It uses an asymmetric link formation approach similar to the randomized link formation approach used in [SBTL]. Although the protocol uses only one phase, it has many rounds. Each device starts initially as a leader. Leaders alternate probabilistically between Seek and Scan states and take on the role of a Seeker or Scanner. If a device goes into scan mode, it has to listen for

another node (discovery channel); then, if it is contacted, it goes into page scan mode. If a device goes into the seek mode, it has to look for slaves in the discovery channel, and to connect via page. Once two leaders connect, one must retire. Two invariants for partial scatternets must be respected; each leader either has no slave, or has at least one unshared slave, and each leader has fewer than $k$ slaves in its piconet. The protocol requires the constant movement of devices from one piconet to another as the protocol forms a growing tree-like scatternet. This causes a disruption in communications and the tree-like structure limits efficiency and robustness. In terms of the number of messages exchanged to form the network and the time it takes the proposed protocol is an enhancement over [SBTL] in the sense that the former takes only one phase to form a scatternet.

The well known structure of the de Bruijn graph (*Figure 3.6*) is adopted in [SLWW] to form the backbone of the Bluetooth scatternet. The proposed protocol is called dBBlue, and guarantees degree limitation (each master has at most seven slaves), that each node is in at most two piconets, and that a node cannot assume both the roles of master and slave. The dBBlue structure also enjoys a nice routing property: the diameter of the graph is $O(\log n)$ and a path with at most $O(\log n)$ hops for every pair of nodes can be found without any routing table. Moreover, the network congestion is at most $O(\frac{\log n}{n})$, assuming that a unit of total traffic demand is equally distributed among all pairs of nodes. A vigorous method for locally updating the dBBlue structure using at most $O(\log n)$ communication when a node joins or leaves the network is discussed in detail. In most cases, the cost of updating the scatternet is actually $O(1)$ since a node can join and leave the network without affecting the remaining scatternet. The dBBlue scatternet can be constructed incrementally when the nodes join the network one by one. A scalable Mac assignment mechanism is designed to guarantee packet delivery even during scatternet updating, to facilitate self-routing and easy updating. In addition, the formed structure can sustain the faults of two nodes and the network is still guaranteed to be connected. If a node detects a fault of some neighboring master, slave, or bridge device, it can dynamically re-route the packets and the path travelled

by the packet is still at most $O(\log n)$ hops. A novel rule of assigning MAC addresses in a piconet is discussed. The construction of a scatternet for a static n-nodes network is given, and this algorithm works as follows: Given $n$ nodes, a leader initiates construction by selecting master nodes to form the backbone. In a distributed way, each master node invites some slave nodes to form the final topology – dBBlue scatternet. In the dynamic scatternet updating method, each piconet is considered as an abstract node in the de Bruijn graph; some piconets have to be divided in two such that the two new piconets will have some free pure slave slots to hold the newly joined node. The proposed method has some desirable characteristics such as formation and routing without geometric information, low diameter, self routing, load balance, and easy inter-piconet scheduling. However, the proposed protocol does not consider the neighbour discovery process, and assumes that every node already knows the existence of the other nodes.

In [WSL] a scheme that first applies a planar structure, limits degree, and then assigns roles is introduced. The problem of scatternet formation for single hop networks is addressed by adapting the scatternet formation schemes proposed in [LSW]. These schemes are position based and localized, and guarantee degree limitation. This method is applied to single hop networks by showing that position information is not needed. In the neighbour discovery phase, each node selects a virtual position and communicates it to all of its neighbours. The nodes then act according to the virtual position instead of the real position. Since each node has all of the information needed, a Delaunay triangulation instead of the partial Delaunay triangulation proposed in [LSW] is applied. As a planar topology, the authors apply a minimum spanning tree. The experiments prove the good functionality of the created scatternet in addition to its fast creation and straightforward maintenance. However, following the suggestion given in [S], if a minimal spanning tree is used as the scatternet topology, some long edges can be added to provide shorter routes.

In [ZHS], a scatternet formation scheme called loop scatternet formation (LSF) is proposed. The LSF algorithm is composed of two phases. By the end of the first phase, piconets are connected into a ring, i.e., if a piconet is envisioned as a

"node," then these "nodes" form a ring. Moreover, all devices are connected into piconets of at most $(k-1)$ slaves. In the second phase, a slave from each piconet is explicitly selected and shared with another piconet to further reduce the network diameter and the maximum node contention. Note that during the whole process only slave-slave pairs are formed. Initially, consider a set of isolated devices that are within the transmission range of each another. During the course of executing the algorithm, "components" are formed, where a component may be an isolated device, a piconet, or a ring scatternet. Each component has a leader node. The leader of an isolated device is itself. The leader of a piconet is the master of the piconet. The leader of a ring scatternet is the master of one of the piconets in the ring. It is assumed that phase one is complete and that only one leader remains in the network. At this time, the leader will commence the second phase. The second phase starts to further reduce the network diameter and the maximum node contention. Specifically, designate the number of piconets as $P$. By the end of the first phase, each master node has two shared bridge nodes and each bridge is shared by two piconets. The second phase builds a bridge (by sharing a slave), respectively, between piconet $i$ and piconet $\lfloor i \pm \sqrt{P} \rfloor$ to reduce the network diameter to $O(\sqrt{P})$. It is analytically shown that the resulting topologies will be connected with a high probability while the number of piconets and the maximum degree of devices is minimized. The LSF protocol leads to a network with a much smaller diameter as well as a number of relay nodes that is significantly smaller than that in the other types of scatternets. The authors clearly show, theoretically and experimentally, the advantages of this new method, and indicate that to handle the case of multi hop scatternet formation a device can collect the neighbourhood information in the first phase. Then, a set of interconnected cliques can be identified that covers all of the nodes in the network. In each clique, a loop-structured scatternet is built and then interconnected with other loop scatternets.

In [SCH], the authors present an adaptive network formation protocol that adapts scatternet to the needs of data flows. This protocol operates in two distinct phases: in the first phase, an initial network is formed with a view to reducing the latency of network formation; in the second phase, the network continuously changes

according to the traffic needs (continuously optimizes the network topology in order to produce the best suitable topology for the current data stream). To accommodate device mobility the optimization process also takes care of network maintenance. The major contribution of this protocol is the second phase.

### 3.4.2. Scatternet formation in multi hop networks

Multi hop scatternet formation algorithms do not require all devices to be within radio vicinity of each other. Algorithms for general multi hop topologies can also be applied for single hop topologies. This category of algorithms relies on the assumption that each node is aware of its neighbours (i.e., one or two hop neighbours) and that this knowledge is symmetric. This knowledge is therefore gathered before the actual piconet interconnection phase. In a multi hop scenario, care must be taken to ensure that bridge nodes are chosen carefully for both available power and signal strength. It also presents difficulties in reducing the overall traffic of the network, since possible problems arise with packets existing for too long within the network.

#### 3.4.2.1. Coordinated Bluetooth scatternet formation protocols for multi hop networks

Ajmone-Marsan et al. [ACNCG] proposed a solution based on a linear optimization formulation. It is a centralized solution for finding a Bluetooth topology (for multi hop scenarios). Using a min-max formulation, the obtained topology provides full network connectivity, fulfills the traffic requirements and the constraints posed by the system specification, and minimizes the traffic load of the most congested node in the network, or equivalently its energy consumption. Results show that this topology is optimized for certain traffic requirements and is also remarkably robust to changes in the traffic pattern. The proposed solution is based on a linear optimization formulation that leads to an NP-complete problem, and thus is suited only for small and stationary networks.

In Mehta and El Zarki [MeZ] an approach is proposed where the focus is to support a sensor network composed of fixed wireless sensors useful in several applications: health monitoring of highways, bridges and other civil infrastructures.

They proposed a topology formation scheme that not only takes into account the traffic generated by different sensors but also the associated link strengths and buffer capacities. The algorithm makes no particular assumptions as to the placement of nodes, and not all nodes need to be in radio proximity of each other. The output is a tree shaped scatternet rooted at the sensor hub (data logger), that is balanced in terms of traffic carried on each of the links. The protocol is centralized (data logger collects network information and makes all decisions) and is based on a combinatorial optimization formulation followed by a simulated annealing based solution. In this approach, it was assumed that all nodes were present at the execution time of the algorithm.

In [RKSA], both a deterministic and a randomized algorithm are evaluated. The proposed solution is based on growing a tree from the root, where a master node is not always directly connected to its slave node. Their results suggest that the randomized approach produces significantly lower link establishment delay over the deterministic (or symmetric) algorithm. Both approaches first partition the network into independent piconets, and then elect a "super-master" that knows about all of the nodes. However, the resulting network is not a scatternet, because the piconets are not interconnected. A separate phase of reorganization is required. To avoid the role of a device being predetermined statically (which would otherwise cause a topology construction algorithm to lose its generality), a plausible solution is to let each device randomly determine its state before it starts the neighbour discovery process.

#### 3.4.2.2. Distributed Bluetooth scatternet formation protocols for multi hop networks

In [ZBC], Bluetrees, a scatternet formation protocol, is proposed. This protocol has two variants, namely Blueroot Grown Bluetree and Distributed Bluetree. The number of roles taken on by each node is limited to 2 or 3. The first protocol is designed for single hop scenarios, and is initiated by a given single node called the Blueroot, which will be the root of the Bluetree. The root node has the master role and all of its one hop neighbours will be its slaves. The children of the root are now assigned the role of master and all their neighbours that not have any role will become

slaves of the newly created masters. This procedure is repeated recursively until all nodes have been assigned roles. In order to limit the number of slaves per piconet, the authors observe that if a node has more than five neighbours, then there are at least two nodes among these neighbours that are neighbours themselves (*Lemma 4.1*). This observation is used to reconfigure the Bluetree so that each master has at least seven slaves. The second protocol is designed for multi hop scenarios: in the first phase of the protocol, several root nodes, which are nodes with the highest *ID*s in the local neighbourhood, are picked and each of these creates its own piconet. In the second phase, the sub-tree scatternets are merged into one scatternet that spans the entire scatternet. One distinct feature of the Bluetree scheme is that all resulting scatternets assume a spanning tree topology, where the parent node is the master and the children nodes are slaves. While the scheme selects the smallest possible number of links to form a connected scatternet and tries to spend the least amount of network resources on maintaining the scatternet, the resulting scatternet has an inherent deficiency due to its hierarchical structure. Also, the proposed solutions are not time efficient and the protocol assumes that all nodes start the formation process at the same time and that one of the nodes has been designated as the Blueroot before the formation starts.

In [PK], a distributed and energy-efficient Bluetooth scatternet formation algorithm that creates a tree based on Device and Link characteristics (SF-DeviL) and is compatible with Bluetooth specifications is proposed. SF-DeviL handles energy efficiency using classes of devices, battery levels, and the received signal strengths. SF-DeviL forms scatternets with tree topologies that are robust to battery depletions, where devices are arranged in a hierarchical order in terms of battery power and traffic generation rate. SF-DeviL is dynamic in the sense that the topology is reconfigured when battery levels are low, thereby increasing the lifetime of the scatternet. SF-Devil is executed in two phases. During the first phase, each node seeks the 'best master' for itself, i.e. each slave chooses a master. The best master is chosen by making a comparison between the ex-master and newly connected master. During this phase, each node continuously tries to discover other devices, establishing a link to a better master and deleting the older link until a discovery timeout is

reached. By the beginning of the second phase, each device has found a master and connected to it, or has declared itself to be a root of the scatternet. Thus, in the second phase, each node that declares itself to be a root runs the Semiroot procedure. Through the Semiroot procedure the disconnected trees are merged into a single scatternet. The Semiroot procedure ends when all roots are connected. SF-DeviL limits the number of slaves to seven for each node, reserving one link to the master. In cases where the selected master already has the maximum number of slaves, this master deletes its 'worst' slave. The slave with the smallest sum of Received Signal Strength Grade and device grade value is the worst and is requested to look at its neighbour list for a new master. If there are some master candidates for that worst slave, the worst slave link is deleted, otherwise the second-, third-, etc. worst slaves are considered for deletion.

In [LLS], a scatternet-route structure is introduced: the proposed protocol combines the scatternet formation with on-demand routing, thus eliminating unnecessary links and route maintenance. The authors introduce an extended *ID* (*EID*) connectionless broadcast mechanism, where each master node and its slave use the same channel for communication. Compared with the original Bluetooth broadcast mechanism, this protocol provides a practical approach to multi hop using Bluetooth and achieves a much shorter route discovery delay to synchronize the piconet along each scatternet route to remove piconet switch overhead, obtain even better channel utilization, and present a route based scatternet scheduling scheme to enable fair and efficient packet transmission over scatternet routes. Network performance analysis and simulation show that scatternet routes can provide extremely stable throughput for multi hop wireless channels with network utilization, being especially useful in the transmission of large batches of packet and real time data in wireless environment. However the delay incurred in the route discovery process is large. Moreover, the resulting scatternet faces scalability problems.

The clustering based approach has been used extensively since its introduction with the BlueStars algorithm in [PBC2]. This is a protocol for multi hop scatternet formation schemes based on a clustering scheme. The first phase is device

discovery, which makes a node aware of its neighbours' *ID* and weight. Nodes alternate randomly between inquiry and inquiry scan modes, and a temporary piconet is set up (paging) to exchange information. In the second phase, nodes are portioned in clusters to form piconets, and the master is selected using a locally and dynamically computed weight indicating the suitability of a particular node for becoming a master and the knowledge of the weights of its neighbour nodes. The node with the largest weight is selected, and the neighbours (who all have smaller weights) then become slaves. Any node, having been told via paging that one or more of its neighbours has become a master, will become a slave to the first master that paged it. Nodes with the highest weight within their neighbourhood, called init nodes are made masters of new piconets. They invite neighbours to join via paging, and non init nodes decide their roles based on the roles of their bigger neighbours, these nodes can join the first bigger neighbour inviting them to its piconet or, if no bigger neighbour is a master, they become master themselves. No two masters can be direct neighbours. The third phase consists of piconets interconnecting in a connected scatternet. In this phase, connectivity is guaranteed by interconnecting masters at most three hops away, the master selects which nodes are to become bridges to connect with other piconets. The authors contend that it is necessary and sufficient that each master selects bridges to all other masters that are at most three hops away. Information about the other masters is already available from the discovery phase. Thus, the selection of the bridges is based on the weighting of the other masters. BlueStars produces a mesh-like connected scatternet with multiple routes between pairs of nodes. It is a distributed solution, that is, all the nodes participate in the formation of the scatternet. But they do so with minimal, local topology knowledge (nodes only know about their one hop neighbours). Simulations of BlueStars show that up to 20 seconds is required to discover more than 90% of the neighbouring nodes. However, after eight to 10 seconds, enough neighbour nodes have been discovered to guarantee connected topologies. The authors also found that by reducing the back-off interval time of devices in the Inquiry state, connected topologies were formed in four to six seconds. One disadvantage of BlueStars is that the scatternets produced are unbounded, with a

potentially large number of slaves and resultant overhead for parking and unparking. Also, this protocol does not limit the piconet size (degree limitation is not guaranteed).

In [S1], a dominating set based Bluetooth scatternet formation with a localized maintenance protocol for multi hop networks is proposed. The problem of discovering all neighbours within transmission range of all nodes is assumed to be resolved by other Bluetooth protocols. The protocol is composed of three phases. In the first phase, the unit disk graph is constructed and a localized sparse subgraph (such as a Relative Neighbourhood Graph or Gabriel Graph) is extracted. In the second phase, a degree reduction technique Yao subgraph is applied simultaneously on all devices with excessive degree (more than seven) to limit the degree of each device to seven, followed by either the elimination of directed edges or the application of a reverse Yao construct. In the third and last phase, master-slave roles are assigned based on some key comparisons (with dominating set membership as the primary key). The creation and maintenance requires minimal overhead in addition to maintaining accurate location information for single hop neighbours, and the generated scatternet is a mesh with multiple paths between any pairs of devices. However, the Bluetooth scatternet formation protocol to apply is left unspecified and no thorough evaluation of this approach is provided. Also, this technique assumes that each device knows its position and that of its neighbours, which is impractical for Bluetooth applications and requires that each node be equipped with additional hardware (i.e., GPS) to provide the device with its current location information, adding such hardware may be expensive, and the proposed solution is not feasible in its absence.

In [LSW], an algorithm for Bluetooth scatternet formation is presented. Geometric techniques for topology reduction combined with cluster-based scatternet formation are proposed. This algorithm is one of several localized scatternet formation algorithms by the authors that are based on a sparse geometric structure. This algorithm consists of three phases. In the first phase, the neighbour discovery and information exchange are performed. In the second phase, the planar subgraph

construction is optional: each node computes the incident edges that belong to the chosen planar sparse structure, Relative Neighbourhood Graph (RNG), Gabriel Graph (GG), or Partial Delaunay Triangulation, and all others edges that are not a part of the sparse structure are removed. The third and final phase of this algorithm is an iterative phase and consists of limiting the degree of each node to seven by applying the Yao structure and assigning master-slave relations in the created subgraph. But this solution, as in [S1] assumes that each node knows the absolute or relative position of both itself and its neighbours, which is not provided by Bluetooth and requires extra hardware (i.e. GPS).

In [DHMPP], Blue Pleiades, a unified approach to the problem of device discovery and scatternet formation in multi hop Bluetooth network, is introduced. The proposed approach stems from the study of the connectivity feature of the following simple graph model: each node connects at random with uniform probability with $c$ nodes among those at distance $r$ or less, where $r$ is the transmission radius. Extensive simulation shows that the resulting graph will be connected with high probability for $c \geq 4$. One node is selected randomly and then one of its potential neighbours is selected at random both uniformly and independently. If both nodes have a degree of less than $c$, a new link is set up between them, otherwise no link is set up. This action is repeated as long as adding links is still possible. If some nodes have fewer than $c$ potential neighbours, they do not have to execute the device discovery protocol infinitely, but a time limit is placed on the execution of the device discovery protocol. Blue Pleiades does not require location information to sparsify the Bluetooth topology, nor does it need to run any extra messages or time consuming phases before the actual piconet formation and piconet interconnection in a scatternet can start. Blue Pleiades runs the BlueStars [DHMPP; PBC2] algorithm for selecting masters, slaves, and bridges to form a degree-bounded connected scatternet. Extensive simulation shows that $c = 5, 6$ are excellent choices that guarantee connectivity within a relatively short time. A particularly nice feature of the Blue Pleiades protocol is that it avoids completely the expense of parking and unparking procedures as this is granted by the adoption of the new device discovery. The

performance of this scheme depends on the device discovery procedures. Speeding up the device discovery will decrease the time required to set up a connected topology. We remark also that, due to its simplicity, the protocol will not only be easy to implement but also quite efficient in terms of energy and communication.

### 3.4.3. BlueMesh: a distributed Bluetooth scatternet formation protocol for multi hop networks

In this thesis we focus our comparison study on one single protocol, BlueMesh, which is considered to be the best competing protocol. We devote a full section to describing in detail the BlueMesh protocol. It is a two phase protocol introduced by Petrioli et al. in [PB; PBC] that guarantees connectivity and limits the number of slaves in each piconet. Their neat scheme does not require position information, but instead the local information is extended to two hop information, with a two round device discovery phase for obtaining the necessary information. It is a modified clustering process, in which the selection of slaves is performed in such a way that, if a master has more than seven neighbours, it chooses up to seven slaves among them so that it can reach all the others via the chosen ones. Such coverage is always possible with up to five slaves [ZBC].

#### 3.4.3.1. Topology discovery phase

Since we are using the same technique as BlueMesh for topology discovery (*Algorithm 3.1*), we will be detailed in our explanation. In the topology discovery phase, which consists of two rounds for two hop information, each node discovers its one and two hop neighbours (note, that Bluetooth (*2.4*) does not provide functions to achieve this). The inquiry procedure in Bluetooth does not guarantee a symmetric knowledge of neighbours, in the sense that if a node $u$ discovers $v$, node $v$ may not be aware of $u$. To achieve the symmetric knowledge of nodes' neighbours the protocol uses inquiry and paging procedures to set up two-node temporary piconets through which two neighbouring devices exchange their identities. This phase may therefore take several seconds between each node (recall that the recommended time for inquiry is 10.24s). The total time required is not addressed but may be several

hundred seconds even for small networks. After having discovered all of its neighbours, a node exchanges this list with its neighbours, thus obtaining two hops neighbourhood knowledge. Each node performs the topology discovery in two steps. In the first step each node performs the discovery procedure, and sets a timer $T_{disc}$ to the selected length $l_{td}$ of the topology discovery phase. This timer is decremented at each clock tick (namely, $T_{disc}$ keeps track of the remaining time until the end of this phase). Node $v$ then randomly enters either inquiry or inquiry scan mode, and computes the length of the phase it is entering ($T_{inq}$ or $T_{scan}$). While in a given mode, node $v$ performs the inquiry procedures as described by the Bluetooth specification. The procedures that implement the inquiry mode (procedure INQUIRY) or the inquiry scan mode (procedure INQUIRYSCAN) are executed for a time that is randomly computed ($T_{inq}$ and $T_{scan}$, respectively), never exceeding $T_{disc}$. Upon completion of an inquiry (or inquiry scan) phase, and if $T_{disc} > 0$, a node switches to the inquiry scan (or inquiry) mode. To allow each pair of neighbouring nodes to achieve a mutual knowledge of each others' $ID$ and weight, the scheme requires that whenever a node in inquiry (inquiry scan) mode receives (sends) an FHS packet, a temporary piconet is set-up by using the Bluetooth standard page procedures, and nodes exchange their $ID$ and weights, together with the synchronization information required for further communication. As soon as this information has been successfully communicated the piconet is disrupted. As soon as a node exits the discovery procedure, it builds locally a neighbour list $N$ of the discovered neighbours and the second step of the topology discovery phase begins. A node checks whether it has the biggest weight among its neighbours in $N$. If this is the case, this node is called an init node, and executes PECKORDER. Any node that is not the biggest among its neighbours is called a non init node, and executes procedure SUBNODE. The init node goes into PAGEMODE, and pages all of its smaller neighbours, while non init nodes go into PAGESCANMODE and wait for PAGE from bigger neighbours. If a non init node does not receive any PAGE, then it executes PECKORDER, being now the biggest node among those with which it has to

exchange its neighbour list. During this phase, two nodes that want to exchange information must establish a temporary piconet.

```
DISCOVERY(v)
  T_disc ← I_id
  while T_disc > 0
do If    RAND(0,1) < 0.5
   then  INQUIRYMODE
           COMPUTE (T_inq)
         INQUIRY (min (T_inq, T_disc))
         INQUIRYSCANMODE
         COMPUTE (T_scan)
         INQUIRYSCAN (min (T_scan, T_disc))
   else  INQUIRYSCANMODE
         COMPUTE (T_scan)
         INQUIRYSCAN (min (T_scan, T_disc))
         INQUIRYMODE
         COMPUTE (T_inq)
         INQUIRY (min (T_inq, T_disc))
  EXIT
```

```
SUBNODE(u)
  PAGESCANMODE
  for each bigger v in N(u)
      do WAIT PAGE (u, v, N(v), N(u))
PECKORDER (u)
```

```
PECKORDER(v)
   PAGEMODE
   for each smaller u in N(v)
      do PAGE(u, v ,N(v),N(u))
   EXIT
```

**Algorithm 3.1:** BlueMesh algorithm: Discovery phase [PB; PBC]

### 3.4.3.2. Scatternet formation phase

The second phase of scatternet formation consist of two parts, role selection and gateway selection, and proceeds in iterations (*Algorithm 3.2*). Nodes that participate in a given iteration perform the modified clustering process. Initially all nodes are undecided. In each iteration, init nodes (nodes having the largest weight among their undecided neighbours) create piconets, by choosing at most seven neighbours as slaves and deleting the remaining edges. First, an init node executes MASTER, to become master, then executes COMPUTES to choose its neighbours with the biggest $ID$ as slaves and deletes them with all their neighbours from its neighbours list. The master repeats this process until, if there are no more nodes on its neighbours list and the number of its slaves is less than seven, then it adds the node with the largest $ID$ until it reaches seven slaves. At this point, it starts paging all its neighbours to tell them whether it has selected them as slaves. The function $GET\ (m, W)$ returns a set of $m$ nodes from the set $W$ (the $m$ smaller ones, randomly chosen ones, or bigger ones, etc.). Each non init node $v$ executes the NONINIT procedure. In the NONINIT, each non init node starts by waiting for all its bigger neighbours to page it and communicate their role. If a bigger node that has selected is

a master, then it joins its piconet. Once all bigger neighbours have paged, the non init node knows whether it has already joined the piconets of some bigger masters. If it has, the non init node is the slave of the bigger masters that have selected it. If it has not joined any piconets, however, the non init node will become a master itself. In any case, the non init node goes into page mode and starts paging smaller neighbours that may need information about its role to decide their own. It will also page all of the bigger neighbours that are slaves, since they need to know about its role for interconnecting piconets later in the iteration. (Bigger masters have already exited the execution of this part of the iteration). The rest of the procedure depends on whether the node has decided to be master or a slave. In the former case, it acts exactly as if it was an init node. If the node is a slave, after having informed its smaller neighbours and bigger neighbours that are slaves of its decision, it goes back to page scan mode. It then waits for smaller neighbours to communicate the decisions they make on their roles, and for bigger neighbours that are slaves to inform it of their complete list of masters. Once this exchange has been completed, the node knows the roles of all of its neighbours, and which (if any) of its neighbouring masters have invited it to join their piconet. In order for the role selection to be completed, the node's list of masters has to be distributed to all of its slave neighbours, and the node has to become aware of their lists of masters. Therefore, $v$ goes into page mode and communicates its list of masters to all of its slave neighbours, and finally switches again to page scan mode to wait for its smaller neighbours that are slaves to communicate their lists of masters. When all these operations have been completed, each node has a knowledge of its neighbours' *ID*, role, and, in the case that they are slaves, of all the piconets they belong to. The node executing NONINIT then exits the role selection part of this iteration and moves to the gateway selection part. In this part, all slaves communicate to their master(s) information about the roles of their neighbours, their neighbours' list of masters, and whether some of their neighbouring masters have selected them as slaves. Based on this information, each master decides which slaves to select as gateways, to which piconets in order to obtain a connected scatternet. If a pair of masters has selected common slaves, they choose the bigger one between them as the

gateway slave. This is the preferred way to interconnect adjacent piconets. Whenever no gateway slave can be selected to interconnect adjacent piconets, intermediate gateways are selected, again based on their weight (e.g., so that the sum of their weights is maximized, or the minimum weight is maximized, or any other unambiguous selection rule). The iteration stops when all nodes are decided. All created masters, together with the slaves that are not selected for links with slaves from other piconets, withdraw from the next iteration. The intermediate gateways proceed to the iteration $i+1$ to form new piconets that interconnect them. Simulations show that the created scatternets have a low average number of roles per node (about two), with an average path length increase between 20% and 80%. The number of iterations grows slowly with the number of nodes (it is about 1.8 to 4.7). The method may show weaknesses on some other metrics, especially concerning the worst-case number of slave roles a node can assume. For instance, in the case of dense networks (e.g., a complete graph), the second largest node in a neighbourhood may end up serving as slave to all masters in the same neighbourhood. Nevertheless, among all methods that do not use position information, the method appears currently to be the best available method for multi hop networks.

```
NONINIT(v)
  PAGESCANMODE
  for each bigger neighbor w
      do WAIT PAGE(v, w, r, j, NIL)
  if r = master and j = true
     then myRole ← slave
          JOIN(w)
          M ←M ∪ {w}
  if myRole = slave
     then PAGEMODE
  for each w in C(v)
      do PAGE(w, v, slave, false, NIL)
  PAGESCANMODE
  for each w in C(v)
      do WAIT PAGE(v, w, r, j, NIL) from smaller w
         WAIT PAGE(v, w, slave, false, M) from bigger w
  PAGEMODE
  for each neighbor slave w
      do PAGE(w, v, slave, false, M)
  PAGESCANMODE
  for each smaller slave w
      do WAIT PAGE(v, w, slave, false, M)
  EXIT
```

```
COMPUTES(v)
  S(v) ←∅
  U ←C(v)
  while U ≠ ∅
    do x ←bigger in U(v)
    S(v) ←S(v) ∪{x}
    U ←U \N(x)
  S(v) ←S(v) ∪ GET(7 − |S(v)|,C(v) \ S(v))
```

```
MASTER(v)
  myRole← master
  PAGEMODE
  COMPUTES(v)
  for each u in S(v)
      do PAGE(u, v, master, true, NIL)
  for each u in C(v) \ S(v)
      do PAGE(u, v, master, false, NIL)
  EXIT
```

**Algorithm 3.2:** BlueMesh scatternet formation phase algorithm [PB; PBC]

We consider the following topology (*Figure 3.7*).

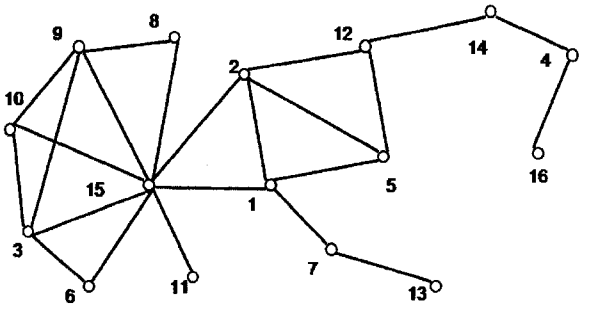


**Figure 3.7:** Initial topology

In the first phase (*Figure 3.8*), being the nodes with the biggest weights in their neighbourhood, devices 16, 14, 13, and 15 are init nodes. They are masters and switch to page mode (procedure MASTER). All the other nodes go into page scan mode (procedure NONINIT). Node 16 has eight neighbours; it selects seven of them as its slaves and drop down the connection with the non selected node, by executing locally the procedure COMPUTES. Node 5 is not selected as a slave by any init nodes; therefore it becomes master and switches to page mode (procedure MASTER).

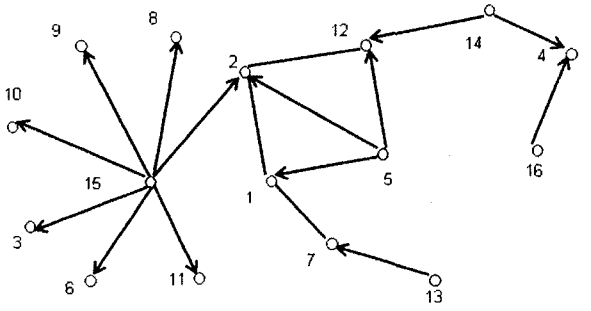


**Figure 3.8:** First iteration of BlueMesh.

In the second iteration (*Figure 3.9*) Nodes 16, 5, 13, and 15 quit the execution of BlueMesh at this time because they are masters and not slaves. Nodes 8, 9, 10, 3, 6, 11, and 4 also exit because they are slaves and have not been selected as

intermediate gateway. Masters 15 and 14 select nodes 12 and 2 as intermediate gateways to connect their piconets; masters 13 and 5 select nodes 7 and 1 to connect their piconets. 7 becomes the master of the piconet formed by itself and node 1, and 12 becomes the master of the piconet formed by itself and node 2.

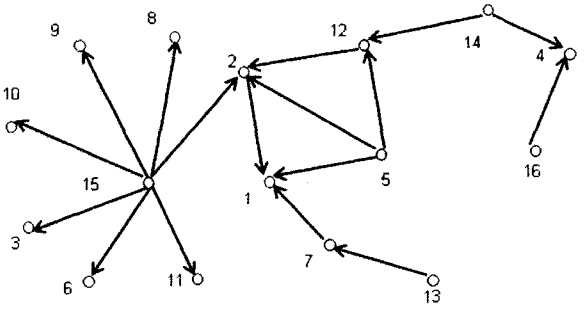


**Figure 3.9:** Second iteration of BlueMesh.

In the third iteration (*Figure 3.10*) nodes 12 and 7 quit the execution of BlueMesh at this time, and select nodes 1 and 2 as intermediate gateways and enter the third phase. Node 2 becomes the master of the piconet formed by itself and node 1. By the end of this phase, all nodes quit the execution of BlueMesh and the algorithm terminates. The scatternet is connected.

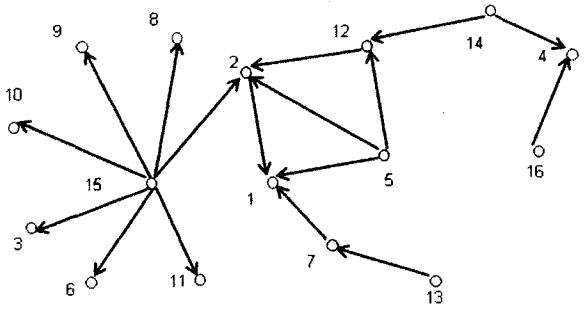


**Figure 3.10:** Third (last) iteration of BlueMesh

## 3.5. Performance evaluation of scatternet formation protocols

In [BBMP] the result of ns2-based comparative performance evaluations among three major solutions for forming multi hop scatternets is described. The three

protocols considered are BlueTrees [ZBC], BlueStars [PBC2], and the "Yao protocol" presented in [LSW]. All operations of the chosen protocols require each node to be aware of its neighbours. For this purpose, device discovery has to be performed before the actual scatternet formation process takes place. All protocols for multi hop scatternet formation rely on the assumption that if a node $u$ knows node $v$, $v$ must also know $u$. This symmetric knowledge is performed during the discovery phase. However, the mechanism provided by the Bluetooth specification for device discovery does not lead to the symmetric neighbour knowledge needed. An inquirer that is trying to discover neighbouring nodes does not transmit its unique Bluetooth identifier, thus remaining unknown to the node that receives the inquiry packet. Furthermore, the Bluetooth discovery mechanisms require nodes to be in opposite inquiry modes in order to be able to communicate, but, no method is described in the specifications as to how to guarantee that two neighbouring devices will be in opposite modes. Therefore, specifications compliant mechanisms must be defined to ensure that, for each pair of neighbouring nodes $v$ and $u$, they are eventually in opposite modes and that, when node $u$ discovers node $v$, $v$ is also made aware of $u$. To overcome this problem, the solution for device discovery in multi hop networks proposed in [SBTL] and described in (*2.4*), is used. Simulations show that in networks with a high density, after 10s of device discovery, only a fraction of a node's one hop neighbours have been discovered. However, device discovery can be kept reasonably short while still ensuring that the resulting discovered topologies are connected, which is the necessary requirement for generating connected scatternets. The price to pay for a short device discovery is that it is not guaranteed in the discovered topology that two nodes that are within each other's transmission range have discovered each other. This can be a problem for those protocols that use this assumption to form connected scatternets with a bounded number of slaves per piconet. The duration of the discovery phase is therefore a critical parameter that should be selected carefully taking all of these trade-offs into account. The BlueTrees protocol designates a node called the Blueroot that initiates the protocol and builds a treelike scatternet rooted at itself. Then, a reconfiguration procedure bounds the

number of slaves per master such that, at the end of this procedure, each master has no more than seven slaves. However, this tree-like topology limits the robustness of the generated scatternet. In addition, BlueTrees depends on the Blueroot to initiate the protocol; this creates a single point of failure, and if the network is not connected after the device discovery phase then this solution will not work. The BlueStars protocol proceeds from the device discovery phase into two phases. In the first phase, piconet formation is initiated by all nodes in a distributed manner wherein each node decides to be a master or a slave based on a locally computed weight. The second phase generates the scatternet by interconnecting the piconets through bridges selected by each master. In the Yao protocol, a geometric technique for topology reduction combined with cluster-based scatternet formation is proposed. The protocol consists of three phases. In the first phase, neighbour discovery and information exchanges are performed. In the second phase, the planar subgraph construction is optional: each node either computes the incident edges that belong to the chosen planar sparse structure, Relative Neighbourhood Graph (RNG) (*3.1.5*), Gabriel Graph (GG) (*3.1.6*), or Partial Delaunay Triangulation, and all others edges that are not a part of the sparse structure are removed. The third and final phase of this algorithm is an iterative phase and consists of limiting the degree of each node to seven by applying the Yao structure and assigning master-slave relations in the created subgraph. But this solution, as in [S1], assumes that each node knows the absolute or relative position of itself and its neighbours, which is not provided by Bluetooth and requires an extra hardware (i.e. GPS). The following desirable properties for a scatternet formation protocol are identified: (1) connectivity, i.e. the produced scatternets should be connected; (2) resilience to disconnections in the network, i.e. the protocol should be able to operate in the connected components of the network; (3) routing robustness, i.e. the scatternets should have multiple routes between any pairs of nodes; (4) piconet size limited to eight nodes, to avoid the overhead associated with parking and unparking slaves; (5) resource-based master selection, which should be driven by devices currently having available resources since the master role is the most resource-consuming; (6) distributed and localized operations,

i.e. a protocol should be executed at each node with information available at the node itself locally (e.g., knowledge of one and two hop neighbours); (7) self-healingness. A protocol should react to changes in the network topology to maintain a scatternet that retains all the properties of the scatternet initially formed. All three protocols considered have distributed and localized operations. However, none of the solutions published so far for scatternet formation in multi hop topologies address the issue of self-healingness. (*Table 3.1*) shows which of the five remaining properties 1–5 is satisfied by each of the three compared protocols.

**Table 3.1**: BlueTrees, BlueStars, and Yao protocol, comparison result [BBMP]

| | BlueTrees | BlueStars | Yao protocol |
|---|---|---|---|
| Connectivity | Y | Y | Y |
| Resilience to disconnection | - | Y | Y |
| Routing robustness | - | Y | Y |
| Degree Limitation | Y | - | Y |
| Resource based master selection | - | Y | Y |

Simulations show that BlueStars is the fastest protocol among the three proposed protocols. BlueStars also generates a lower number of piconets than both the BlueTrees and Yao protocols, which is logical since BlueStars does not guarantee degree limitation whereas Yao protocol and BlueTrees do. This also results in a piconet with a very limited number of slaves for the Yao protocol and BlueTrees, 5 and 4 slaves per piconet respectively, while the number of slave per piconet in BlueStars exceeds the threshold of 7. The average number of roles per node in the BlueStars and Yao protocols increases slightly as the number of nodes increases. In BlueTrees, all internal nodes have two roles, while the leaves only assume the role of slave. Since the percentage of nodes that are masters does not change significantly with the number of nodes, the average number of roles per node remains steady at around 1.5, independently of the number of nodes. A major difference between the protocols is that BlueTrees adopts master-master gateway for piconet interconnection, while BlueStars and the Yao protocol use gateway slaves whenever possible, or intermediate gateways when gateway slaves are not available. The number of master-

slave roles is thus much more limited. At the same time, BlueStars and the Yao protocol have the further advantage of forming scatternets with a higher degree of piconet interconnection (a mesh like topology instead of a tree). Among the three protocols, BlueStars has the shortest routes between any two nodes. The average increase of route length in the mesh-like scatternets formed by BlueStars with respect to the length of the corresponding routes in the discovered topologies is 50%, independently of the increasing number of nodes. This is essentially due to the piconet-based network organization, which may force nodes that are neighbours in the visibility graph to communicate through their common master (if they belong to the same piconet), or through a potentially long inter-piconet route in case they belong to different piconets. Routes in scatternets formed by the Yao protocol are from 18% to 70% longer than the routes in BlueStars scatternets, to reflect the higher number of piconets that has to be crossed (a higher number of piconets is formed by the Yao protocol because of the limit imposed on the number of slave per piconet). Finally, as expected, in the tree-like scatternets formed by BlueTrees, routes are 30% to 94% longer than those in BlueStars scatternets. In this case, two nodes that are possibly close to each other have to communicate through the only route that passes through the first common ancestor. Also, simulations show that for a network of 110 nodes, after 20 seconds of device discovery a node has only discovered about 88% of its neighbors. If connectivity depends on the remaining 12% of the neighbors, then the algorithms will obviously fail to construct a connected topology. Both of those issues are probably not of much significance in small ad hoc networks, i.e., around 10 nodes, where a discovery phase of 20 or even 10 seconds will typically be sufficient to discover all nodes. However, they are likely to result in much more severe problems in large-scale networks. Finally, the enforcement of a fixed discovery period that precedes topology formation is difficult, if not impossible, to implement in a dynamic environment which nodes power on and off or join and leave the network.

# Chapter 4

# BlueMis protocol

## 4.1. Protocol Overview

The proposed protocol is called "Bluetooth Scatternet formation of wireless devices based on maximal independent sets" (BlueMis). It attempts to simplify the BlueMesh procedure presented in (*3.4.3*). BlueMis is a two phase protocol, in which the first phase is the discovery phase and the second is scatternet formation; it essentially interprets the slave selection as the maximal independent set problem. In the discovery process, two hop neighbours are learned, as in BlueMesh; this is required to run the maximal independent set based slave selection. The proposed protocol is based on applying certain key(s) for decisions. The second phase is executed in exactly two iterations (compared to BlueMesh that uses on average 1.8 to 4.7 iterations). In the scatternet formation phase, each device creates its own piconet using devices selected in its maximal independent set; all created piconets are preserved if this is possible and meaningful. Two neighbours $A$ and $B$ could select each other as slaves, and thus make a resolution to keep only one relationship, based on an arbitrary key. Thus, the node with lower key remains master in a symmetric relationship. Two elimination procedures are applied to remove unnecessary piconets; if a master has only slaves that are all masters and not slaves in other piconets and they are not slaves in other piconets, then it becomes the slave of all its slaves; also, if a master which is not a slave in any other piconet has only one slave, then it becomes slave and the slave becomes the master. If after such applications certain node remains without slaves, it does not need to keep its piconet. Several variations of BlueMis will be presented in this chapter that depend on the keys used for decisions as well as on the application of other simplification procedures to the initially formed scatternet.

## 4.2. Protocol description

Given a set of Bluetooth nodes that are positioned such that their unit disk graph is connected, the Bluetooth scatternet formation (BSF) problem is to select piconets, and the master and slave roles within each piconet, so that the obtained scatternet is connected, has certain desirable properties, and shows good performance with respect to certain metrics. In this section we present a detailed description of three variations of the new BlueMis protocol. A major advantage of these new protocols, based on maximal independent sets, is their simplicity. Simulations show the advantages of BlueMis over BlueMesh, the best competing protocol. Our new BSF protocols attempt, to simplify the BlueMesh [PB; PBC] procedure. The protocols essentially interpret the slave selection as the maximal independent set problem, and in contrast with BlueMesh in which the number of iterations can be large, our new protocol reduces the process to either one or two iterations after the initial discovery process. Several criteria are used to evaluate the Bluetooth scatternet formation protocols, although the connectivity and degree limitation of the constructed topologies are the main goals of any protocol. Obviously, there is a tiny probability that two nodes will never find each other, therefore connectivity cannot be guaranteed even for a network consisting of two nearby nodes. Thus, it is a natural assumption that connectivity is judged subject to established neighbours' knowledge. We assume that all nodes have discovered all their neighbours in the initialization phase. We also assume the unit disk graph model of wireless communication, where two nodes can communicate with each other if and only if the distance between them is at most $R$, where $R$ is the transmission radius and is equal for all nodes. We also consider multi hop scenarios, in which some nodes are not within transmission range of each other, but are connected via other nodes. Our proposed protocols do not depend on a central component and each node makes formation decisions based solely on the information from its neighbours (possibly also two hop neighbours). At the end of the discovery phase, which performed at each node, all devices discover their one hop neighbours. During the construction of the maximal independent set for each node, there will be

exchanges of neighbours' information as needed. Pairs of neighbouring devices may exchange information on their one hop neighbours, thus getting to know all the nodes that are at most two hops away. Our procedure is based on applying certain key(s) for decisions. The procedure will be expressed in terms of a general key. Specific options available for selecting keys are as follows:

- $ID$
- $-ID$
- $(degree, ID)$
- $(-degree, ID)$
- $(slave\ degree, ID)$
- $(slave\ degree, degree, ID)$

Note that the degree of a node is the number of its one hop neighbours. Slave degree is the number of slaves selected by the *ComputeMIS* procedure explained below. When the key is a record composed of an ordered list of sub-keys, the comparison is made using lexicographic ordering, i.e. by the first sub-key; if it is equal, then the comparison is made by the second sub-key. If it is also equal, then the third sub-key is used. Keys can also be based on links to neighbours. For example, the number of common neighbours of $A$ and $B$ can be used as the key. This may give priority to selecting further neighbours, thus creating shorter paths.

## 4.3. BlueMis: a single iteration protocol

This is the first and simplest variant of our approach. After learning about one hop neighbours, the first iteration of our protocol will serve to achieve three goals:

1) Each node creates a piconet by selecting the MIS of its neighbours as its potential slaves, based on an arbitrary *key1*.
2) At the same time, if two neighbours $u$ and $v$ select each other as slaves, then we keep only one relationship, based on an arbitrary $key2$. If $key2(u) < key2(v)$, then $v$ deletes $u$ from the

list of its slaves, while $u$ preserves $v$ as its slave. Thus the node with the lower $key2$ remains the master in a symmetric relationship.

3) If a certain node remains without slaves, it does not need to keep its piconet. Our proposed MIS based scheme preserves all created piconets if it this is possible and meaningful.

The BlueMis protocol has two phases. After the discovery phase, each node runs independently the procedure *ComputeMIS* (*Algorithm 4.1*) described below in order to create its own piconet by selecting a maximal independent set of its neighbours as slaves. Each node has two keys, *key1* and *key2*, that are known to all its neighbouring nodes. To find *MIS(u)*, the node $u$ chooses a node $v$ from $Z$, the set of neighbours of $u$, with minimal *Key1(v)*.

```
ComputeMIS (u)
1   MIS(u) ← ∅; Master(u) ← true ;
2   Z ← N(u);
3   while Z!= ∅
4         do v ← Node in Z with smallest key1;
5         Page (v, u, N(v), MIS(v));
6               If u in MIS(v)
7                     if key2(u) < key2(v) then
8                           M(v) ← M(v) ∪{u};
9                           MIS(v) ← MIS(v) – {u};
10                          M(u) ← M(u) - {v};
11                          MIS(u) ← MIS(u) ∪{v};
12              Otherwise
13                    M(v) ← M(v) ∪{u};
14                    MIS(u) ← MIS(u) ∪{v};
15  Z ← Z – (N(v) ∪ {v});
16  If MIS(u) = ∅ then Master(u) ← false
```

**Algorithm 4.1:** BlueMis: Single iteration protocol

Node $v$ is declared a slave of $u$ if the following condition is satisfied: node $u$ receives the value of *key2(v)*, checks whether $u$ is already a slave of $v$ and whether *key2(u) < Key2(v)*, and then decides whether it should declare $v$ as its slave. If $v$ is selected as a slave for $u$, then $v$ is eliminated from $Z$, together with all of $v$'s neighbours. This process is repeated until $Z$ becomes empty.

Notice that in line 16, each node will check independently its set of slaves *MIS(u)*, and if this set is empty then $u$ deletes its own piconet.

During this construction, and in line 7, we will also break any symmetric master/master relations between two nodes, using *key2*. We assume that *key2* contains the unique *ID* for tiebreaking as part of the key.

During the execution of *ComputeMIS (u)*, node $u$ keeps the information *(N, MIS, Master, M)* where:

- *N(u)* is the set of neighbours of $u$,
- *MIS(u)* is the set of slaves of $u$,
- *Master* is a Boolean, i.e. whether $u$ is a master or not a slave, and
- *M(u)* the set of nodes that are masters and have $u$ as a slave.

Our next theorem shows that the BlueMis protocol keeps the scatternet connected.

**Theorem 4.1** For any choices of *key1* and *key2,* and after running the procedure *ComputeMIS(u)*, the scatternet remains connected if the initial graph is connected and arbitrarily defined.

**Proof:** We will prove that the constructed scatternet contains a minimal spanning tree (MST) as its subgraph. Let the 'length' of each edge be 1. To distinguish between edges, we make them different and uniquely sortable by defining *key(B, A)= key(A, B)* = (*key1(A), key1(B)*) where *key1(A) < key1(B).* Note that neighbours $B$ of node $A$ are considered slaves in the ascending order of *key(A, B).* We then follow Kruskal's MST construction algorithm, in which edges are considered for inclusion in MST in ascending order, and included if and only if their addition does not create a cycle. Let *(K2, K3), key1(K2) < key1(K3)* be a pair of nodes that has not selected each other as

slaves. We show then that this edge is not in MST. Since *K3* did not select *K2,* there exists a common neighbour *K1*, selected by *K3* as slave, with $key1(K1) < key1(K2)$ (a node can be eliminated only by a neighbour with a lower key). Then $key(K1, K3) < key(K2, K3)$, and $key\ (K2, K1) < key\ (K2, K3)$. Therefore, edges *(K1, K2)* and *(K1, K3)* have lower keys than edge *(K2, K3)* and have been already considered previously for MST inclusion. Thus the pairs *(K1, K2)* and *(K1, K3)* are already connected within MST (either by being in MST or by the presence of cycles connecting them). But as a result the pair *(K2, K3)* is already connected in MST, and is not included in MST. It follows that all MST edges are selected for master-slave relations, and the scatternet thus remains connected. Regardless of *key2,* the elimination of the master-slave pair *AB* when pair *BA* already exists, and the elimination of piconets left without any slave, obviously does not create partitions. ♦

While the number of slaves for each master is limited, and the scatternet is connected, the number of slave roles for each node is not limited, and in some cases, e.g. a complete graph, one node can be selected as a slave to all other nodes. This is the same problem demonstrated with BlueMesh. The main advantage of the new algorithm is reducing the number of iterations to two and therefore obtaining a faster BSF.

It is easy to show that any MIS will have at most 5 slaves. Let $R$ be the nodes' transmission range and assume that two nodes $u$ and $v$ are neighbours if and only if their Euclidian distance is at most $R$.

**Lemma 4.1** let *G(V, E)* be a unit disk graph; $v$ can reach all its neighbours by selecting at most five among them.

**Proof:** Let $C$ be a circle of radius $R$ and let $S$ be a set of circles of radius $R$ such that every circle in $S$ intersects $C$ and no two circles in $S$ intersect each other. By contradiction, suppose that $|S| \geq 6$. Let $S_i$, $1 \leq i \leq 6$, denote the centers of any six circles in $S$. Let $c$ denote the center of $C$. Denote the ray $\overrightarrow{cs}_i$ by $r_i$ $(1 \leq i \leq 6)$. Since there are six rays emanating from $c$, there must be at least one pair of rays $r_j$ and $r_k$ such that the angle between them is at most $\frac{\pi}{3}$. Since we have six circles, it can be

verified that the distance between $s_j$ and $s_k$ is at most $2r$, otherwise the sum of the angles between them would be larger than 2. Such a contradiction would imply that the circles centered at $s_j$ and $s_k$ intersect, contradicting our assumption. Thus $|S| \leq 5$.♦

Since the MIS of any node will have at most 5 nodes, and during the execution of *ComputeMis* no more than *O(n)* such exchanges of neighbours will happen, where $n$ is the number of devices, there will be no need for symmetric knowledge among neighbours, specifically, if a node $u$ is aware of a node $v$ at most two hops away, then $v$ does not need to have the same information about $u$.

### 4.3.1. An example illustrating BlueMis

The BlueMis protocol is illustrated in (*Figure 4.1*) and (*Figure 4.2*). We will use the node *ID* appearing beside each device as *key1*. A line between any two devices indicates that they have discovered each other during the discovery phase. Devices with a grey color are the masters and the others are the slaves. An arrow from a device $u$ to a device $v$ indicates that $v$ is a slave of the master $u$.

In the first iteration, each node creates its own piconet by selecting its maximal independent set. At the same time, we avoid the situation where two neighbours $A$ and $B$ select each other as slaves. For instance, node 15 selects as a first potential slave a neighbour with a minimal *key1*, which is node 1. Then node 15 pages node 1 to check for any inconsistency with a symmetric Master/Master relation. Node 1 will return the list of its neighbours $N(1)$ and its list of slaves *MIS*(1) chosen so far. Node 1 will select the nodes {2, 7} as its *MIS*. Since 15 is not a part of *MIS*(1), the node 1 is selected in *MIS*(15). Device 1 and all of its neighbours, i.e. {2, 5, 7, 11, 15} are taken out of the list of the potential slaves for node 15. The next node to be selected as a potential slave for 15 is node 3. Then node 15 pages node 3 to check for any inconsistency with a symmetric Master/Master relation. Node 3 will return the list of its neighbours $N(3)$ and its list of slaves *MIS*(3) chosen so far. Since node 3 will select nodes {6, 9} as its *MIS*, node 15 is not a part of *MIS*(3) and thus 3 is selected as a part of *MIS*(15). The same thing will happen with node 8, the next to be selected in *MIS*(15). The next and last node to be selected as a potential slave for 15 is node 11.

Then node 15 pages node 11 to check for any inconsistency with a symmetric Master/Master relation. Node 11 will return the list of its neighbours $N(11)$ and its list of slaves $MIS(11)$ chosen so far. In the case that 11 has already chosen 15 as one of its slaves, and since 15 is larger than 11, node 15 will select 11 as its slave and 11 will take 15 off its list of slaves. Otherwise, node 15 selects 11 as a slave and no changes are made to the lists of node 11. If later in the process, node 11 tries to select 15 as a slave, it will page 15 for its list of slaves; however, since 11 belongs to that list and moreover $11 < 15$, nothing will be done. By the end of this first iteration of our protocol, each device will check whether its constructed *MIS* is empty. For instance, in this example all the devices 6, 9, 13 and 16 should delete their piconets, since they have no slaves (*Figure 4.2*).

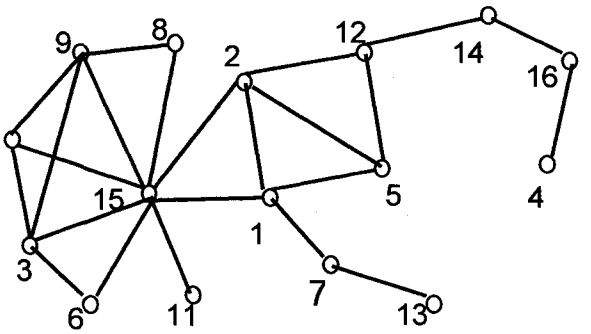


**Figure 4.1:** Initial topology

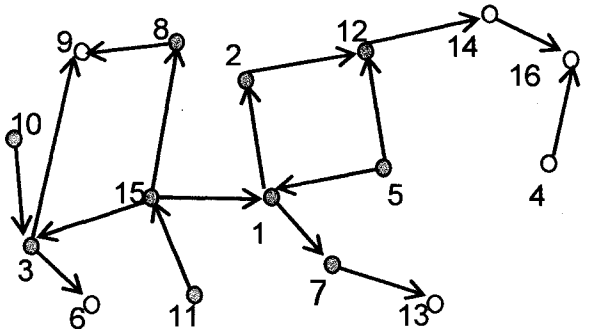


**Figure 4.2:** Scatternet obtained by BlueMis

## 4.4. BlueMis1: a two iteration protocol that guarantees connectivity

The protocol BlueMis presented in (*4.3*) is a fast and simple algorithm; however, the number of piconets created is still high. Our first variation of the BlueMis protocol is called BlueMis1. It has two iterations with the exact same first iteration as that used in BlueMis. In the second iteration we perform independently two simple procedures to further simplify the scatternet structure by deleting piconets that are not essential for the connectivity of the scatternet, therefore lowering the number of piconets. The main idea is that if a master node u is not a slave in any other piconet then we try to eliminate the piconet of u whenever possible:

1) If the master *u* has a unique slave *v*, then *v* becomes a master (if *v* is not already a master) and *u* joins *v* as slave. This is achieved by running a new procedure called *SlaveProc* (*Figure 4.3*).
2) If all the slaves of a master node *u* are themselves master nodes and *u* is not a slave in any other piconet, then *u* loses its piconet and joins the piconets of its slaves.
3) If for all slaves *v* of the master node *u* one of the following conditions holds:
   - Either *v* is a master itself, or
   - There is another master *w* that has *v* as a slave and *w* is a neighbour of *u*,

   Then u joins the piconet of w and v is deleted from the set of the slaves of *u*.

We define two new procedures: *SlaveProc* and *MasterProc*. Each node will run independently either the procedure *MasterProc* if it is a master or the procedure *SlaveProc* if it is a slave. However, we want to ensure that all slave procedures are run before the master procedures. Therefore, if u is a slave, then it runs *SlaveProc*. However, if u is a master, then it waits for the paging of all its slaves that are not masters before running *MasterProc*.

Both procedures will attempt to delete only piconets with a master device $u$ that is not a slave in any other piconet, that is, when $M(u)$ is empty.

The master procedure *MasterProc* is run by every master $u$ and it consists of checking whether $M(u)$ is empty and all slaves of $u$ are themselves masters. In that case the piconet of $u$ will disappear and $u$ joins the piconets of its slaves, as long as they do not exceed the limit of seven slaves.

The Slave procedure *SlaveProc* is run by every slave $u$ and consists of checking all nodes $M(u)$ to see whether $M(v)$ is empty and $S(v) = \{u\}$. That is, if $u$ is the unique slave of a master $v$ and node $v$ is not a slave in any other piconet, then the piconet of $v$ is deleted, $u$ becomes a master (if $u$ is not already a master), and $v$ joins $u$ as slave (*Figure 4.3*). This procedure will stop if the number of slaves of the node $u$ becomes seven.

***SlaveProc(u):*** This is the task of the procedure: if $u$ is the unique slave of a master v and node $v$ is not a slave in any other piconet, then the piconet of $v$ is deleted, u becomes a master (if $u$ is not already a master), and $v$ joins $u$ as slave (*Figure 4.3*). This procedure will stop if the number of slaves of the node $u$ becomes seven (*Algorithm 4.2*).

```
SlaveProc(u)
1   If Master(u) = false then
2         u Pages all nodes v in M (u) to get (MIS(v), M(v))
3           If MIS(v) = {u} and M(v) = ∅ and |MIS(u)| < 7 then
4                 Master(u) ←true,
5                 Master(v) ← false
6                 MIS(u) ← MIS(u) ∪{v}
7                 MIS(v) ← ∅
8                 M(v) ← {u}
9   Page (done) all nodes in M(u)
```

**Algorithm 4.2:** Slave procedure

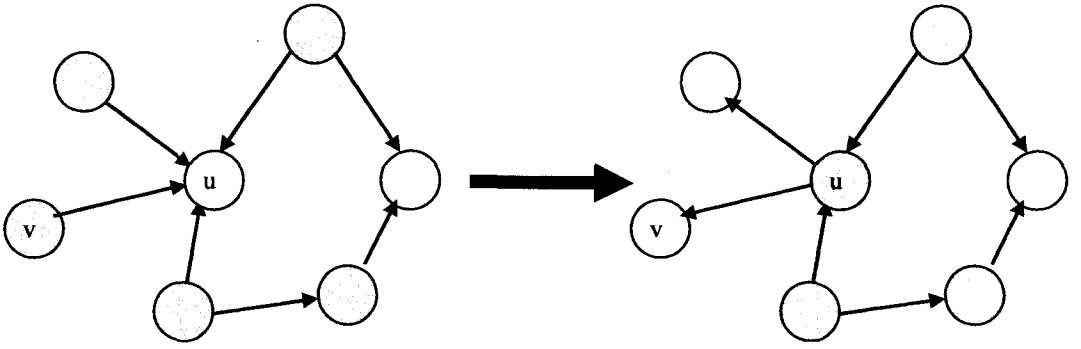


**Figure 4.3:** Slave procedure

For every master node $u$, *MasterProc(u)* checks the properties of the piconet of $u$ and decides whether this piconet can be deleted. This simplification is only considered for masters $u$ that are not slaves of any other piconet, that is when, $M(u)=\varnothing$. In order to be able to delete the piconet of $u$, we try first to transform it into another one where every slave of $u$ is itself a master slave. If that is possible, $u$ will join all the piconets of its slaves (which are masters themselves) whenever they have room, that is, the number of slaves of the piconet will not exceed seven. In the case that the master $u$ is able to join all of its slaves' piconets, that it deletes its own piconet (*Figure 4.4*). This procedure will be finalized assuming that the seven slaves limit is respected.

A master node will check whether all of its slaves $v$ satisfy one the following three conditions:

1. The node $v$ is itself a master node;
2. The node $v$ is not a master node and there is a node $w$ in $N(u) \cap M(v)$.

***MasterProc(u)*:** For the master node $u$, this procedure will check the properties of its piconet and decide whether its piconet can be deleted. This simplification is only considered for masters $u$ that are not slaves in any other piconet, that is when $M(u) = \varnothing$. In order to be able to delete the piconet of $u$, we try first to transform it into another one where every slave of $u$ is itself a master. If that is possible, $u$ will join all the piconets of its slaves (which are masters themselves) and

delete its own piconet. This procedure will be finalized assuming that the seven slaves limit is respected (*Algorithm 4.3*).

```
MasterProc (u):
1  WaitPage(done) from all slave neighbors
2  If Master(u) = true then u Pages all nodes v in MIS(u) to get
   (N(v), MIS(v), Master(v), M(v))
3     If Simplify(u) = true then for every v in MIS(u)
4        If Master(v) = true and |MIS (v)|<7 then
5              M(u) ← M(u) ∪{v};
6              MIS(u) ← MIS(u) – {v};
7              MIS(v) ← MIS(v) ∪{u};
8        If there is w in N (u) ∩ M(v) and w ∉MIS(u) and |MIS(w) |<7 then
9              MIS(w) ← MIS(w) ∪{u};
10             M(u) ← M(u) ∪{w};
11             M(v) ← M(v) – {u};
12 If MIS(u) = ∅ then Master(u) ← false
```

**Algorithm 4.3:** Master procedure for BlueMis1

A master node will check whether all of its slaves $v$ satisfy one the following two conditions:

1) The node $v$ is itself a master node and $|MIS(v)| < 7$

2) The node $v$ is not a master node and there is a node $w$ in $N(u) \cap M(v)$.

If condition 2) is satisfied, *Masterproc* will make $u$ as a slave of $w$ and $v$ is deleted from the set of slaves of $u$, assuming that there is room in the piconet of $w$ (*Figure 4.4*):

- $MIS\ (u) \leftarrow\ (MIS\ (u) \cup \{w\}) - \{v\}$
- $M\ (w) \leftarrow M\ (w) \cup \{u\}$
- $M\ (v) \leftarrow M\ (v) - \{u\}$

If for all slaves of *u*, either one of the conditions 1) and 2) holds, then we perform the transformation to try to remove of the piconet of *u*. All slaves of *u* become master nodes and *u* joins the piconets of its slaves if they have room for another slave and then deletes that slave from its set of slaves. If this transformation is successful for all slaves of u, then *u* deletes its own piconet.

For a master node *u*, we set a Boolean value *Simplify(u)* to be true if *M(u)* =∅ and one of the above two conditions holds for every slave *v* of the piconet of *u*.

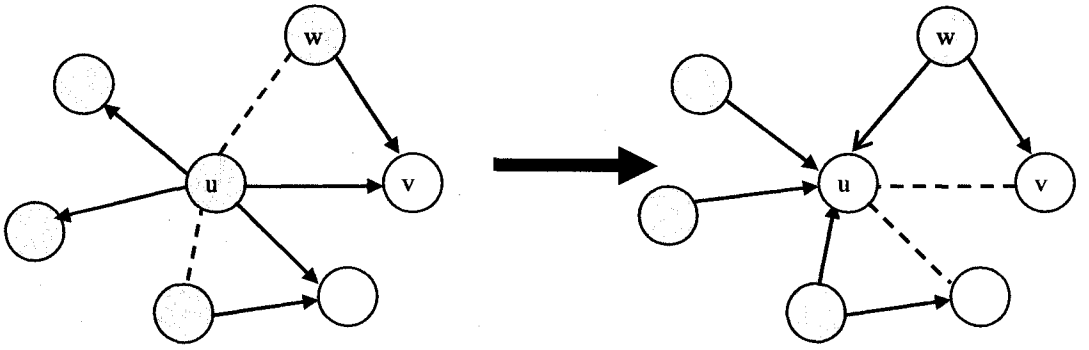


**Figure 4.4:** Master procedure

It may happen that two neighbouring masters try to simplify their piconets at the same time. We thus added the condition $w \notin MIS(u)$ in line 7 of *Masterproc* to avoid both masters using each other at the same time for simplification and thus potentially leading to a disconnected topology. Notice too that *MasterProc* may create piconets where the slaves do not form necessarily an independent set.

The next theorem shows that in some cases the slave procedure is not needed since the procedure *ComputeMIS* will never produce master nodes that are leaves in the scatternet. Therefore, in these cases, there will be no need to run *Slaveproc*.

**Lemma 4.2** If we use the same unique key, that is $key1 = key2$, then after the execution of the procedure *ComputeMIS* there are no master, *u* such that $MIS(u) = \{v\}$ and *u* is not a slave of any piconet, unless *u* and *v* did choose each other as slaves.

**Proof:** Assume that we have a situation where *u* is a master with a unique slave *v* and *u* is not a slave in any piconet. The node *v* became a slave because at one time a

tiebreaking took place with a set of master nodes $w_1, w_2 \ldots, w_n$, and then the piconet $v$ became without slaves and deleted itself. (Notice that the topology is connected $1 \leq n$ and, according to (*Lemma 4.1*), $n \leq 5$). Assume that node $v$ did choose its slaves according to that order. Suppose that $u$ is not a part of the set $\{w_1, w_2 \ldots, w_n\}$, and since $u$ is not a part of *MIS(v)*, thus $u$ must be connected to some of these nodes. Let $i$ be the smallest such that $u$ is connected to $w_i$. Therefore, the three nodes $u$, $v$, and $w_i$ form the configuration shown in (*Figure 4.5*). That is,

1) $u, v$ and $w_i$ are all connected in the original network;
2) $u$ did choose $v$ as a slave, $v$ did choose $w_i$ as a slave, and $w_i$ did choose $v$ as a slave
3) The tiebreaking between $v$ and $w$ went to the advantage of $w_i$.

Since $u$ chooses $v$ instead of $w_i$ then $key1(v) < key1(w_i)$, however the tiebreaking between $v$ and $w_i$ went to the advantage of $w_i$, and thus $key1(w_i) < key1(v)$. This is a contradiction. ♦

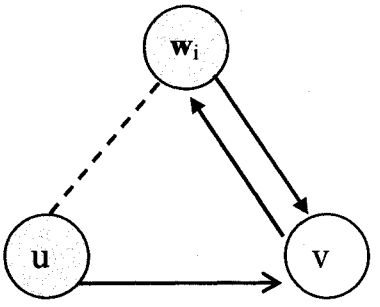


**Figure 4.5:** Tie Breaking

**Lemma 4.3** After the execution of the procedure *ComputeMIS* there is no master $u$ such that $MIS\ (u) = \{v\}$ unless $S\ (u) \subseteq S\ (v)$.

**Proof:** If there exists a node $w$ connected to $u$ but not to $v$, then clearly $MIS(u)$ would contain at least one other node with $v$. ♦

The next theorem shows that if the size of the neighbour set N is used in both keys, then the slave procedure will not be needed since *ComputeMIS* will never

produce master nodes that are leaves in the scatternet. The tiebreaking will go for the node that has the maximum size of the neighbour set in the original network.

**Theorem 4.2** If we use (-|S|, ID) as the same unique key, that is $key1 = key2 = (-|S|, ID)$, then after the execution of the procedure *ComputeMIS* there is no master $u$ such that $MIS(u) = \{v\}$ and $u$ is not the slave of any piconet. Therefore, there is no need to run the slave procedure.

**Proof:** According to Lemmas, 4.2 and 4.3, $u$ and $v$ did choose each other as slaves and $S(u) \subseteq S(v)$. Since u won the tiebreaking with $v$ then $|S(v)| \leq |S(u)|$ and thus $S(u) = S(v)$ and $ID(u) < ID(v)$. Obviously the only possibility is to have $MIS(u) = \{v\}$ and $MIS(v) = \{u\}$. Suppose $S(u)$ contains a node $w$ that is different from $v$. Notice that $u$ and $v$ must have the smallest keys compared to all nodes in $S(u)$, since $v$ did choose $u$ as a slave $key(u) < key(v)$. If a node $w$ is a common neighbor for $u$ and $v$, then during the construction of $MIS(w)$, $w$ should have chosen $u$, which is a contradiction since we are assuming that $u$ is not a slave in any other piconet. Therefore $S(u) = \{v\}$ and $S(v) = \{u\}$, that is, either the whole network consists of the two nodes $u$ and $v$, or it is disconnected. ♦

**Theorem 4.3** For any choices of *key1* and *key2*, the protocol *BlueMis1* creates a connected scatternet topology, if the initial graph is connected and arbitrarily defined.

**Proof:** The proof of Theorem 4.3 is an obvious consequence of Theorem 4.1 and the following two lemmas. ♦

**Lemma 4.4** For any choices of *key1* and *key2*, and after running *Slaveproc* of *BlueMis1*, the scatternet remains connected if the initial graph is connected and arbitrarily defined.

**Proof:** Consider any pair of devices $(u, v)$ and let $u = w_1, w_2 \ldots w_2 = v$ be an alternating master/slave path that connects $u$ to $v$ in the original scatternet topology. Since *Slaveproc* changes only the roles of some devices, the same path remains in the scatternet topology. Moreover, this procedure acts only on masters of degree 1 and their unique slave, thus the only possible change is either to $u = w_1$ that becomes a

slave of $w_2$ or to $w_n = v$ that becomes a slave of $w_{n-1}$, and in both cases we still have an alternating path. Therefore, after running *Slaveproc* of *BlueMis*, the scatternet remains connected. ♦

**Lemma 4.5** For any choice of *key1* and *key2*, and after running *MasterProc* of *BlueMis1*, the scatternet remains connected if the initial graph is connected and arbitrarily defined.

**Proof:** Consider any pair of devices $(u, v)$ and let $u = u_1, u_2 \ldots u_n = v$ be an alternating master/slave path that connects $u$ to $v$ in the original scatternet topology. A problem could occur if one of the nodes $u_i$ in this path has lost its status as a master node. Since the procedure applies only to masters that are not slaves in another piconet, the only possibilities are $u_i = u_1 = u$. or $u_i = u_n = v$. Without loss of generality we may assume that the transformation has affected $u_1 = u$. There are two possibilities to consider:

1) The device $u_2$ is a master: the device $u$ has deleted its own piconet and become a slave in the piconet of $u_2$. In this case the original path between $u$ and $v$ remains an alternating master/slave path.

2) The device $u_2$ is a slave in the piconet of $u$ and can be simplified. That is, there is another master node $w$ in $N(u) \cap M(u_2)$ and $|MIS(w)| < 7$. In this case $u$ has become a slave of $w$ and $u_2$ is no longer a slave of $u$. Thus, the original path is replaced by the new alternating master/slave path: $u = u_1, w, u_2 \ldots u_n = v$.

Notice that *MasterProc* cannot create new devices with the property $M(v) = \varnothing$. Therefore, the distributed procedure *MasterProc* running on different masters will not interfere with other masters, although each master will inquire about the information on its neighbors only once at the beginning of the procedure. This ends the proof of Lemma 4.5 and thus Theorem 4.3. ♦

### 4.4.1. An example illustrating BlueMis1

In the second iteration, each of the slave devices $\{6, 9, 13, 16\}$ will run independently the procedure *Slaveproc*. For instance, node 16 will execute *Slaveproc*. Since its neighbour 4 is a master with unique slave 16, then device 16 becomes a master and 4 becomes its slave. Nothing will be done for the other slaves since they do not have the right property.

When *Slaveproc* finishes for all slave nodes, each of the master devices $\{1, 2, 3, 5, 7, 8, 10, 11, 12, 14, 16\}$ will run independently the procedure *MasterProc*.

The *MasterProc* procedure will have an effect on nodes 5, 10, and 11 since they can be simplified: they are not slaves in another piconet and all of their slaves are masters themselves. Nodes 5, 10, and 11 will join the piconets of their slaves and delete themselves. (*Figure 4.6*) shows the end result of our protocol.

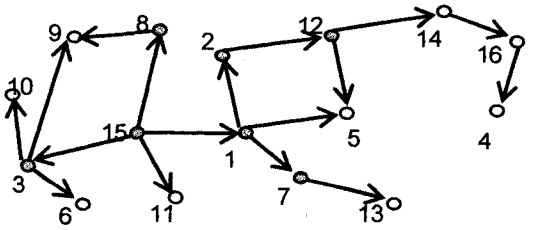


**Figure 4.6:** Second iteration of BlueMis1

## 4.5. BlueMis2: a more efficient two iterations protocol where connectivity is not guaranteed

In this third variation of BlueMis called BlueMis2, we use the same iteration 2 as in BlueMis1 with the addition of another possible simplification of the scatternet. For BlueMis2, the *MasterProc* will allow the deletion of some piconets for a node $u$ even in cases when $u$ is a slave in another piconet. That is, we drop the condition that $M(u) = \varnothing$. Moreover, we introduce a new configuration where we can delete piconets. That is, a slave $v$ of $u$ could be simplified if there is a node $w$

in $N(v) \cap M(u)$. In that case, $v$ becomes a slave of $w$ and $v$ is deleted from $MIS(u)$ (*Figure 4.7*).

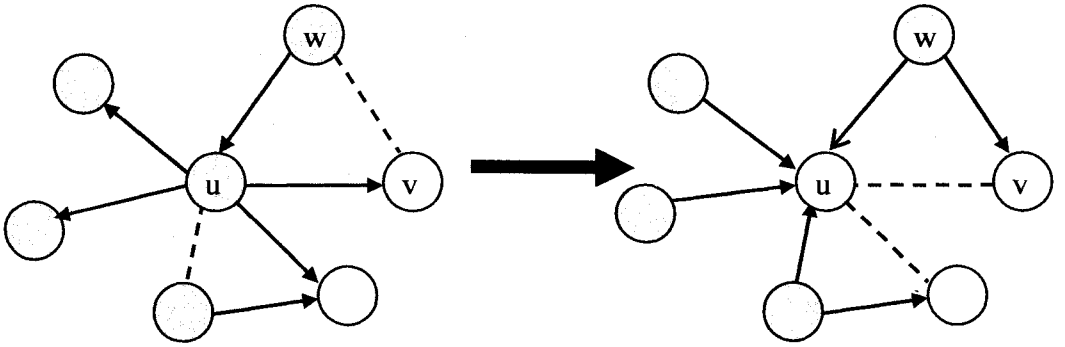


**Figure 4.7:** Second iteration of BlueMis2

The protocol BlueMIS2 uses a more sophisticated, and therefore more costly, simplification procedure in the second iteration. However, it produces a connected scatternet topology where piconets have at most seven slaves.

In BlueMis2, every node will run independently either the procedure: *MasterProc* if it is a master or the procedure *SlaveProc* if it a slave. The *SlaveProc* procedure is exactly the same as that in BlueMis1, and here too we want to make sure that all slave procedures are run before the master procedures. Therefore, if $u$ is a slave, then it runs *SlaveProc*. However, if $u$ is a master, then it waits for the paging of all its slaves that are not masters before running *MasterProc*.

***MasterProc(u)*:** For the master node $u$, this procedure will check the properties of its piconet and decide whether its piconet can be deleted. In order to be able to delete the piconet of $u$, we try first to transform it into another one in which every slave of $u$ is itself a master slave. If this is possible, $u$ will join all the piconets of its slaves (which are masters themselves) and delete its own piconet. This procedure will be finalized assuming that the seven slaves limit is respected (*Algorithm 4.4*).

A master node will check whether all of its slaves $v$ satisfy one the following three conditions:

1) The node $v$ is itself a master node and $|MIS(v)| < 7$

2) The node $v$ is not a master node and there is a node $w$ in $N(u) \cap M(v)$.

If condition 2 is satisfied, *Masterproc* will make $u$ a slave of $w$ and $v$ is deleted from the set of slaves of $u$, assuming that there is room in the piconet of $w$ (*Figure 4.4*):

i. $MIS(u) \leftarrow (MIS(u) \cup \{w\}) - \{v\}$

ii. $M(w) \leftarrow M(w) \cup \{u\}$

iii. $M(v) \leftarrow M(v) - \{u\}$

3) The node $v$ is not a master and there is a node $w$ in $N(v) \cap M(u)$.

If condition 3 is satisfied, *Masterproc* will make $v$ as a slave of $w$ and $v$ is deleted from the set of slaves of $u$, assuming that there is room in the piconet of w (*Figure 4.7*):

i. $MIS(w) \leftarrow MIS(w) \cup \{v\}$

ii. $MIS(u) \leftarrow MIS(u) - \{v\}$

iii. $M(v) \leftarrow (M(v) \cup \{w\}) - \{u\}$

If for all slaves of $u$ one of the conditions 1), 2) or 3) holds, then we perform the transformation to try get rid of the piconet of $u$. All slaves of $u$ become master nodes, and $u$ joins the piconets of its slaves if they have room for another slave and then deletes that slave from its own set of slaves. If this transformation is successful for all slaves of $u$, then $u$ deletes its own piconet.

For a master node $u$, we set a Boolean value *Simplify(u)* to be true if one of the above three conditions holds for every slave $v$ of the piconet of $u$.

**MasterProc (u):**

1 WaitPage(done) from all slave neighbors
2 **If** *Master(u)* = true **then** *u* Pages all nodes v in *MIS(u)* to get *(N(v), MIS(v), Master(v), M(v))*
3 **If** *Simplify(u)* = true **then** for every *v* in *MIS(u)*
4 **If** *Master(v)* = true $|MIS(v)|<7$ **then**
5 $M(u) \leftarrow M(u) \cup \{v\}$;
6 $MIS(u) \leftarrow MIS(u) - \{v\}$;
7 $MIS(v) \leftarrow MIS(v) \cup \{u\}$;
8 $Master(v) \leftarrow$ true
9 **If** there is *w* in $N(u) \cap M(v)$ and $w \notin MIS(u)$ and $|MIS(w)|<7$ **then**
10 $MIS(w) \leftarrow MIS(w) \cup \{u\}$;
11 $M(u) \leftarrow M(u) \cup \{w\}$;
12 $M(v) \leftarrow M(v) - \{u\}$;
13 $Master(w) \leftarrow$ true
14 **If** there *w* in $N(v) \cap M(u)$ and $|MIS(w)|<7$ **then**
15 $MIS(w) \leftarrow MIS(w) \cup \{v\}$;
16 $MIS(u) \leftarrow MIS(u) - \{v\}$;
17 $M(v) \leftarrow (M(v) \cup \{w\}) - \{u\}$;
18 $Master(w) \leftarrow$ true
19 **If** $MIS(u) = \varnothing$ **then** $Master(u) \leftarrow$ false

**Algorithm 4.4:** Masterproc for BlueMis2

**Theorem 4.4** For any choices of *key1* and *key2*, the protocol *BlueMis2* creates a connected scatternet topology, if the initial graph is connected and arbitrarily defined.

**Proof:** The proof of Theorem 4.4 is an obvious consequence of Theorem 4.1, Lemma 4.4 and the following lemma. ♦

**Lemma 4.6** For any choice of *key1* and *key2*, and after running *MasterProc* of *BlueMis2*, the scatternet remains connected if the initial graph is connected and arbitrarily defined.

**Proof:** Consider any pair of devices $(u, v)$ and let $u = u_1, u_2 \ldots u_n = v$ be an alternating master/slave path that connects $u$ to $v$ in the original scatternet topology. A problem could occur if a pair $(u_i, u_{i+1})$ of devices in this path has lost its status as Master/Slave relation. There are three possibilities to consider:

1) The device $u_{i+1}$ is a master: the device $u_i$ has deleted its own piconet and become a slave in the piconet of $u_{i+1}$. In this case the original path between $u_i$ and $u_{i+1}$ remains an alternating master/slave path.

2) The device $u_{i+1}$ is a slave in the piconet of $u_i$ and can be simplified using rule 2. That is, there is another master node $w$ in $N(u_i) \cap M(u_{i+1})$ and $|MIS(w)| < 7$. In this case $u_i$ has become a slave of $w$ and $u_{i+1}$ is no longer a slave of $u_i$. Thus, the original path between $u_i$ and $u_{i+1}$ is replaced by the new alternating master/slave path: $u_i, w, u_{i+1}$.

3) The device $u_{i+1}$ is a slave in the piconet of $u_i$ and can be simplified using rule 3. That is, there is another master node $w$ in $N(u_{i+1}) \cap M(u_i)$ and $|MIS(w)| < 7$. In this case $u_{i+1}$ has become a slave of $w$ and $u_{i+1}$ is no longer a slave of $u_i$. Thus, the original path between $u_i$ and $u_{i+1}$ is replaced by the new alternating master/slave path: $u_i, w, u_{i+1}$.

This ends the proof of Lemma 4.5and thus Theorem 4.3. ♦

### 4.5.1. An example illustrating BlueMis2

In the following example (*Figure 4.8*), in addition to the simplification made in the second iteration by BlueMis1, node 8 deletes its piconet and node 9 joins node 15's piconet (condition 3 satisfied by the piconet of 8 and so it can be deleted). Node 3 also deletes its piconet because of node 15, which is a neighbour of node 3's slaves

(nodes 10 and 6), and nodes 6 and 10 join 15's piconet. Node 1 will not be able to delete its piconet since the piconet of 15 already has seven slaves. Finally, node 2 deletes its piconet and joins 12's piconet. The obtained scatternet is shown in (*Figure 4.8*).

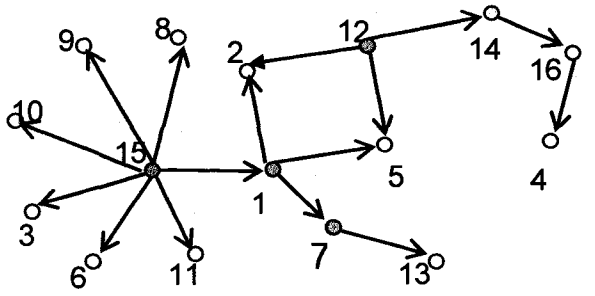


**Figure 4.8:** Second iteration of BlueMis2

# Chapter 5

# Analysis and simulations

In this section, we present our experimental results for BlueMis and a comparison between BlueMesh and the different proposed variances of our protocol in terms of various characteristics. We have simulated BlueMis to demonstrate its effectiveness in generating limited degree and connected scatternet with certain desirable properties. We used a simulator of Bluetooth based ad hoc networks implemented in java. The comparison is made only with BlueMesh since we believe that all other existing schemes have deficiencies as elaborated in [SZ] and Chapter 3. The simulation parameters were chosen to match closely those used by BlueMesh to ensure a fair comparison. We concentrated our evaluation on how the topology was formed from a set of disjoint and discoverable devices, and did not consider the rapid link establishment since frequent device disconnections during the initial formation would not be realistic and would result in an inaccurate formation delay evaluation.

## 5.1. Simulation environment

We have implemented our proposed protocol using java as a programming language. We generate various scenarios randomly based on network size and the dimension of the geographic area. For our simulations, the number of Bluetooth devices has been varied between 40 and 200 devices; each device has a maximum transmission range of 10 meters. The devices are randomly and uniformly scattered in a geographic area which is a square of side $L$. The resulting geographic network topology graph is a unit disk graph; it is assumed that two nodes are in each other's transmission range if and only if their distance in the plane (Euclidean distance) is $\leq$10m. $L$ has been set to either 40 meters or 60 meters to permit the testing and comparison of the protocols of moderately ($L$ = 60 m) to heavily ($L$ = 40 m) dense networks. The simulation was carried out in a Windows XP environment. The

simulation setting parameters are shown in (*Table 5.1*). We implemented the BlueMesh protocol within the same environment in order to compare it properly with our proposed scheme.

**Table 5.1:** graphs parameters

| Description | | Value |
|---|---|---|
| Network size | | Range (40 – 200 nodes) |
| Simulation area model | Heavily dense | 40 m * 40 m |
| | Moderately dense | 60 m * 60 m |
| Transmission radius | | 10 m |

### 5.1.1. Graph generator

We implement a graph generator class to create the network. The procedure works as follows:

1) Generate a set of nodes placed randomly in the simulation area;
2) Calculate the Euclidean distance $d$ between each pair of nodes;
3) If ($d \leq 10$)
4) Create Edge()

At the end, we generate an adjacency matrix for the created graph. This matrix is used later as an input for the simulator. It is important to note that the graph generator does not guarantee connectivity. For this we create a function to check the graph connectivity; once it detects that a graph is disconnected the graph is rejected and replaced by a new one. To better understand the characteristics of the generated graphs, we provide some measurements taken during the simulation for both models. Table 1 shows some characteristics of the randomly generated networks.

**Table 5.2:** Generated heavily and moderately dense networks

| N | | 60 | 80 | 100 | 120 | 140 | 160 | 180 | 200 |
|---|---|---|---|---|---|---|---|---|---|
| Average degree | L=40 | 11.34 | 15.7 | 18.96 | 22.02 | 25.46 | 28.52 | 31.4 | 36.4 |
| | L=60 | 6.2 | 7.13 | 9.56 | 11.22 | 14.52 | 18.24 | 23.9 | 26.4 |
| Maximal degree | L=40 | 19.5 | 26.9 | 30.1 | 34.3 | 39.4 | 45.7 | 51.2 | 56.8 |
| | L=60 | 11.3 | 15.7 | 18.2 | 19.3 | 22.3 | 26.9 | 28.4 | 31.2 |
| Minimal degree | L=40 | 2.5 | 3.2 | 3.6 | 4.6 | 5.8 | 6.1 | 6.7 | 7.1 |
| | L=60 | 1 | 2.1 | 3.5 | 4.7 | 5.4 | 5.9 | 6.3 | 7.8 |
| Average shortest path | L=40 | 3.35 | 2.84 | 2.80 | 2.71 | 2.69 | 2.62 | 2.57 | 2.49 |
| | L=60 | 4.69 | 4.52 | 4.33 | 4.24 | 4.02 | 3.85 | 3.82 | 3.76 |
| % connected graph | L=40 | 43 | 71 | 82 | 85 | 90 | 96 | 99 | 99 |
| | L=60 | 9 | 15 | 22 | 31 | 43 | 57 | 69 | 77 |

*Table 5.2* shows some characteristics of the randomly generated networks. More than 95% of the total generated graphs are disconnected for networks with forty and fifty nodes, thus we omit the class of graph during the simulation.

As expected, our protocol does not perform well when the network is dense. In fact, if all $n$ devices see each other (that is, all devices are within the range of each other) then BlueMis and BlueMis1 will produce ($n$-7) piconets. In the first iteration, device 1 will choose device 2 as a slave and all other devices will choose device 1. Apart from device 2, which will delete its own piconet after losing the Master/Master tiebreaking with device 1, all other devices will keep their own piconets. In iteration 2, only 6 devices will join the piconet of device 1 and all other piconets will stay unchanged.

## 5.2. Performance evaluation

We evaluate our proposed protocol based on five primary metrics: average shortest path ratio, average of master/slave roles, average number of roles, number of created piconets, and finally the average number of slaves per piconet. We investigate also the impact of the network's characteristics on the overall system. In the following sub-sections we analyze the effect of each metric based on the results retrieved from the simulation environment.

A major gain of our proposed protocol over BlueMesh is the number of iterations. It represents our primary contribution. BlueMesh is significantly affected

especially by large network sizes, contrary to our approach where the number of iterations is constant, independent of the network size. Simulation results show that the average number of iterations ranges from 1.8 - 4.7 in BlueMesh. However, in BlueMis, BlueMis1, and BlueMis2 the number of iterations is constant: a unique iteration is needed for BlueMis, and two iterations for BlueMis1 and BlueMis2.

### 5.2.1. Shortest path ratio

We evaluate the variation of the average shortest path in both models when we increase the network size. In (*Figure 5.1*) the different network sizes are shown along the x-axis, while the average shortest path is shown on the y-axis. In the heavily dense model, BlueMesh provides a slightly better result. This is an expected result since the piconet size is less than BlueMesh as shown in (*Figure 5.1*). For both versions BlueMis and BlueMis1 we have exactly the same results; this is due to the fact that the obtained topologies are similar. BlueMis provides the better output among all the implemented protocols since the selected nodes remain unchanged during the entire process.

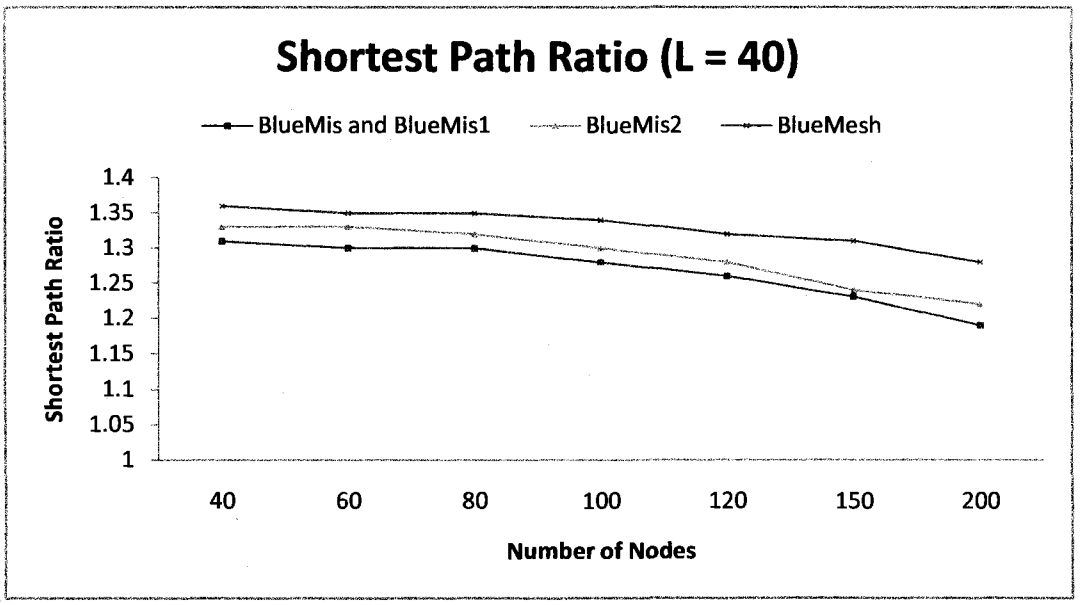


**Figure 5.1:** Shortest path ratio in a heavily dense network

As shown in (*Figure 5.1*), the shortest path metrics is most likely stable among all the implemented protocols. BlueMis2 is highly affected because many connections are dropped.

In the moderately dense model there is a slight improvement, as shown in (*Figure 5.2*). We have a similar variation as in the heavily dense model.

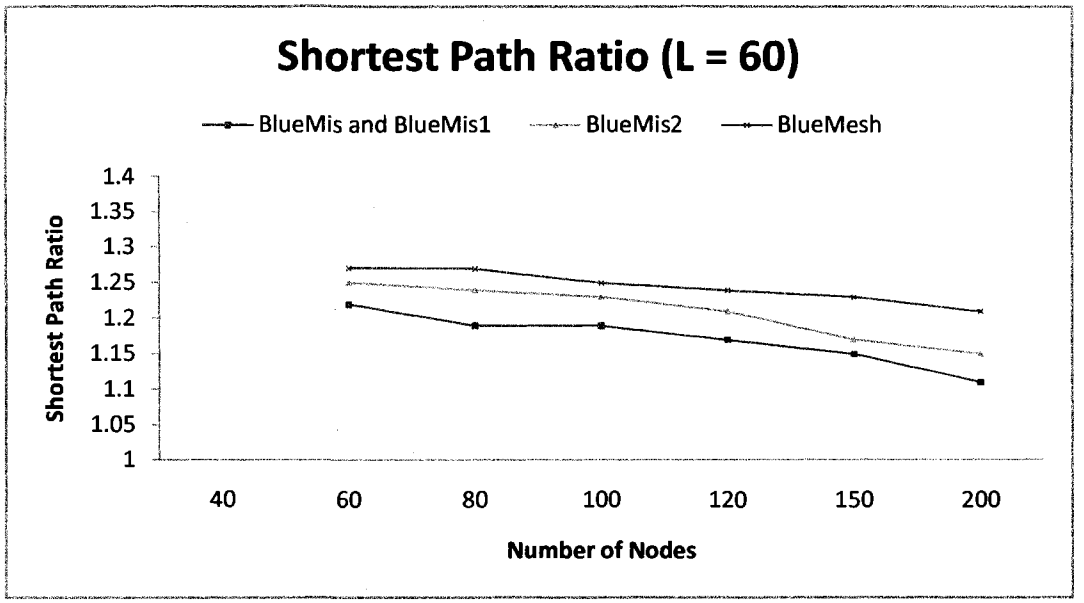


**Figure 5.2:** Shortest path ratio in a moderately dense network

The variation of the ratio is dropped down by 8%, i.e. 1.19 and 1.10 when the network size is equal to 200 nodes. Since we are operating in a moderately dense network, the amount of dropped connections is decreased.

### 5.2.2. Average master/slave roles

In our implemented version, BlueMis2 provides similar results to those given by BlueMesh, as shown in (*Figure 5.3*) and (*Figure 5.4*). There is a slight improvement when we use the moderately dense model because the number of gateway nodes is minimized.

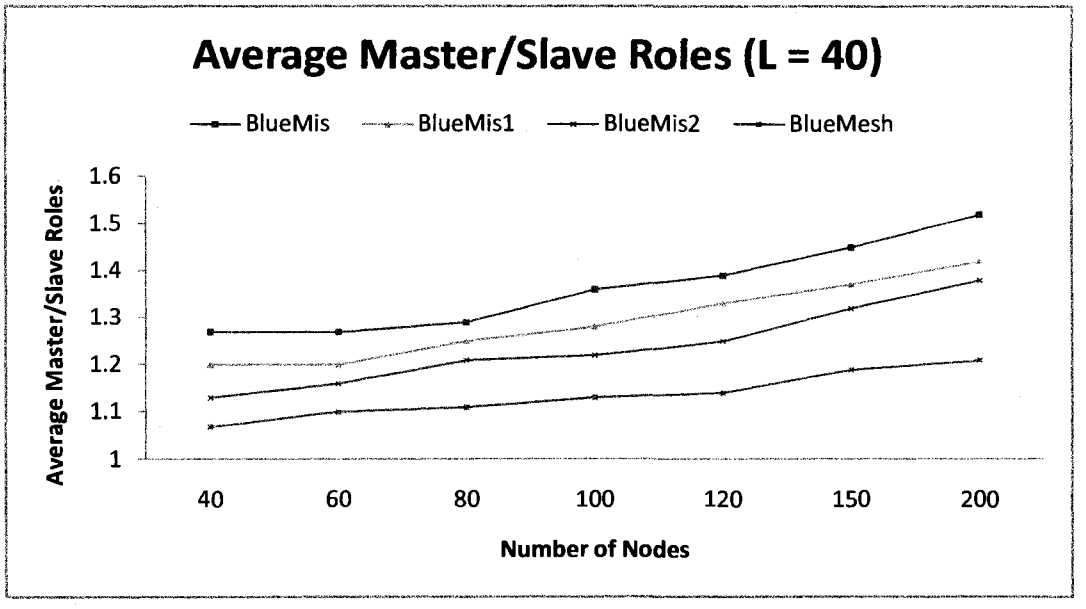


**Figure 5.3:** Average master/slave role in a heavily dense network

In the BlueMis and BlueMis1 versions, the average of master/slave roles is affected due to the increased number of piconets generated.

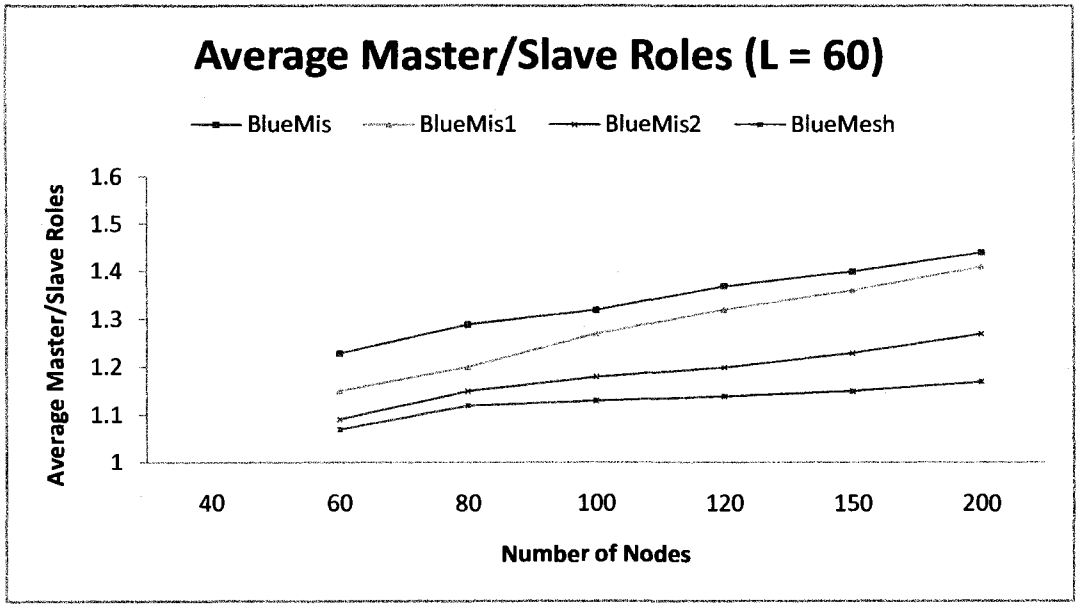


**Figure 5.4:** Average master/slave role in a moderately dense network

### 5.2.3. Average number of roles

In this section we consider the variation in the average number of roles when we increase the network size. (*Figure 5.5*) provides simulation results for the heavily dense model, while (*Figure 5.6*) provides simulation results for the moderately dense model.

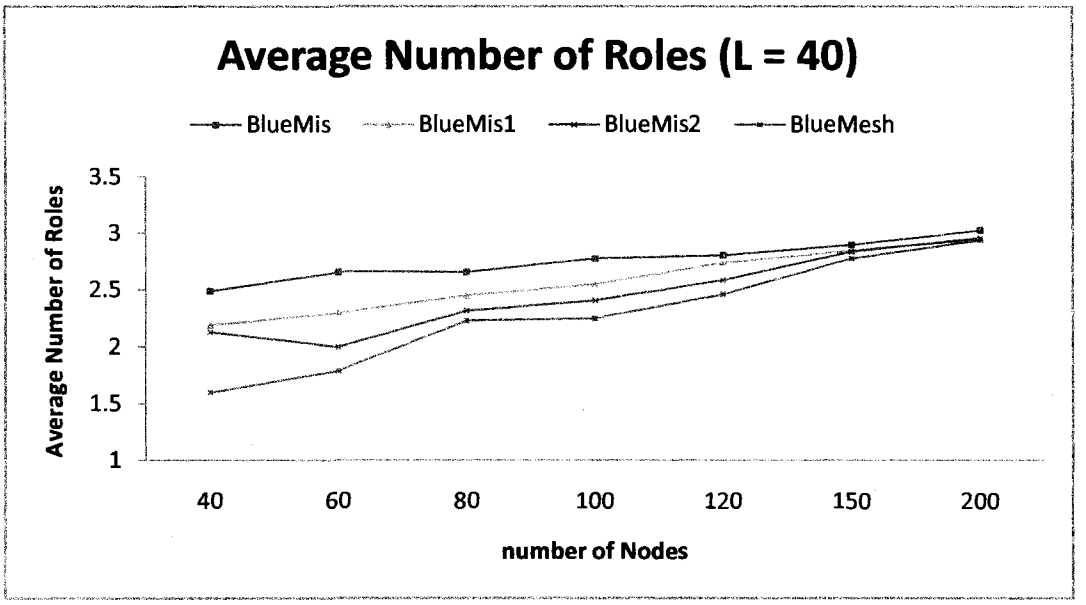


**Figure 5.5:** Average number of roles in a heavily dense network

In the heavily dense model, BlueMesh provides the best results, while BlueMis provides the worst results. Our implementation of the BlueMis2 version is the closest to the BlueMesh protocol. In both protocols, the average number of roles is comparable. This may have an important impact on the network overhead.

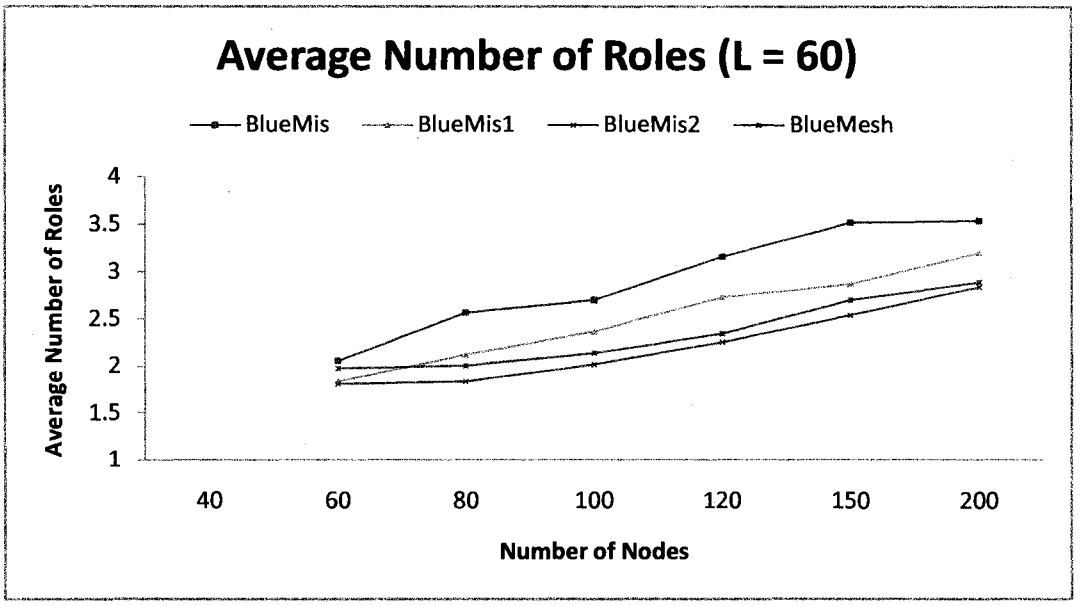


**Figure 5.6:** Average number of roles in a moderately dense network

In (*Figure 5.6*) we have the same result variation as shown in (*Figure 5.5*), but with a slight augmentation since the participating nodes are far away.

### 5.2.4. Average slaves per piconet

All implemented protocols guarantee a degree connection constraint (at most seven). BlueMesh and BluesMis2 provide the better results; this is expected since the selection scheme upper bound is seven, whereas in other protocols the upper bound is five. Thus the number is decreased, as shown in (*Figure 5.7*) and (*Figure 5.8*).

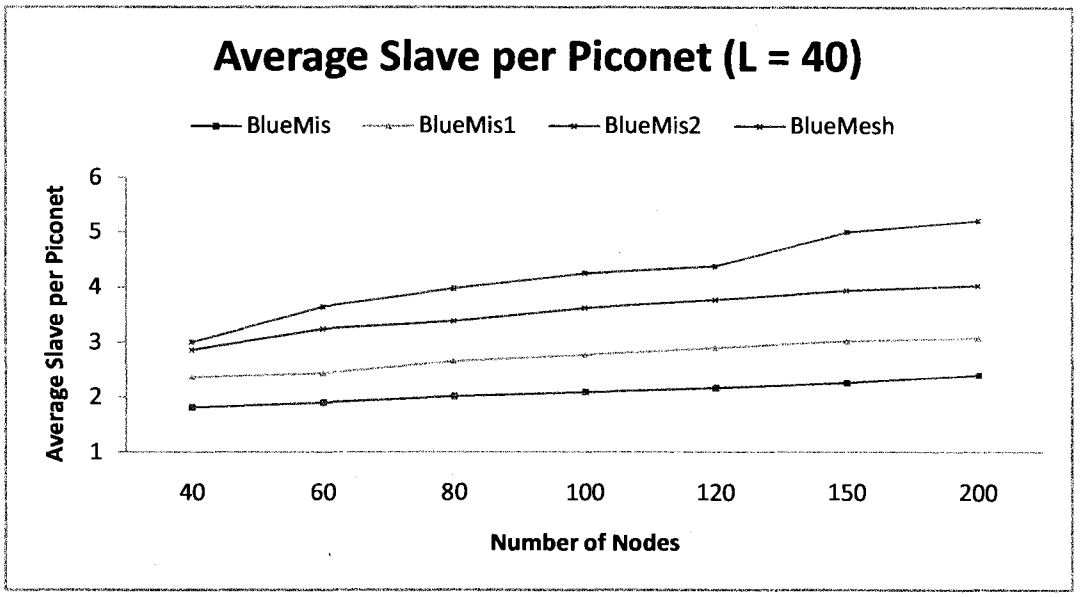


**Figure 5.7:** Average slaves per piconet in a heavily dense network

In the moderately dense model we have almost the same variation, with a reverse case for BlueMis2 and BlueMesh, while the remaining protocols still have similar behaviours as shown in (*Figure 5.8*).

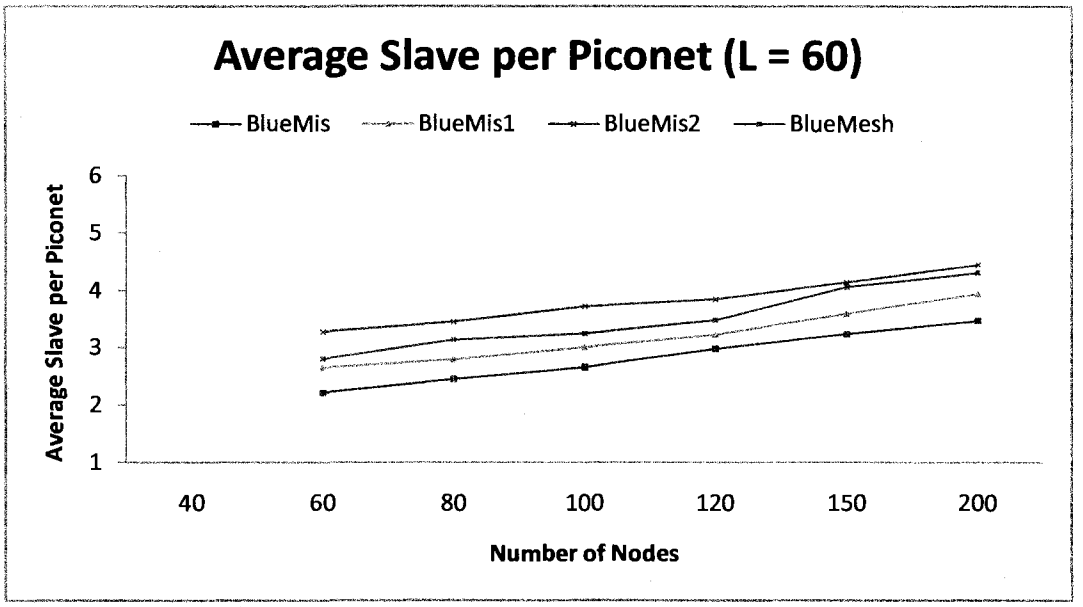


**Figure 5.8:** Average slaves per piconet in a moderately dense network.

In conclusion, as shown in (Figure 5.7) and (Figure 5.8), BlueMis2 offers a better improvement in both models.

### 5.2.5. Average number of piconets

This metric is equivalent to the number of selected master nodes in the scatternet. It is important to keep this metric as low as possible, because this may increase the network overhead. BlueMesh provides better results in a heavily dense network while BluesMis2 provides better result in moderately dense network, as shown in (Figure 5.9) and (*Figure 5.10*).

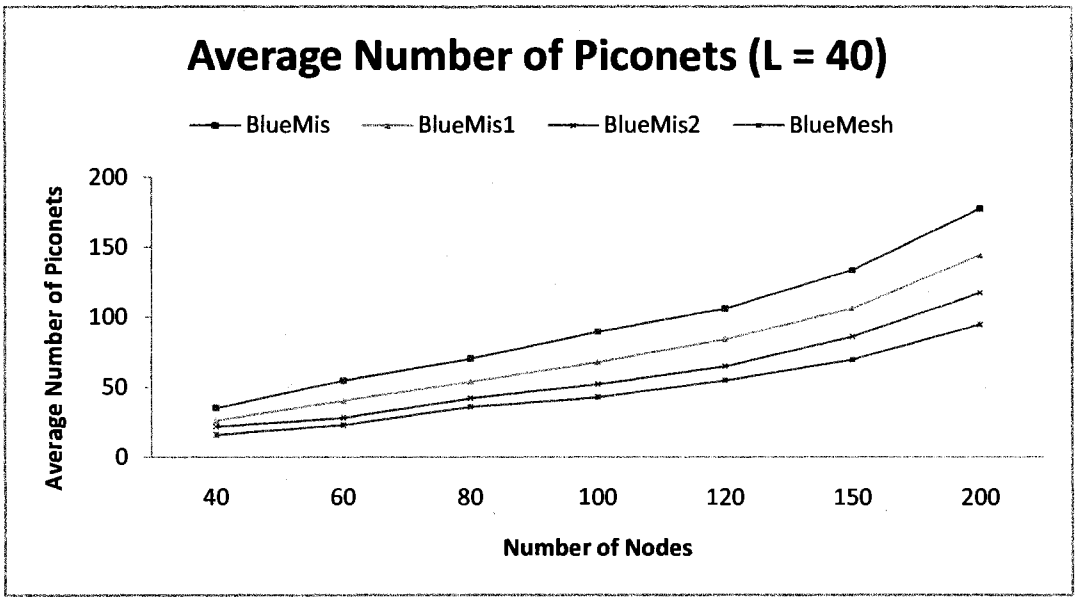


**Figure 5.9:** Average number of piconets in a heavily dense network

The worst case is provided by BlueMis, where almost all of the nodes play a master role, thus the metric is most equal to the network size especially when the network size increases. For this reason we implement BlueMis1, which is an improved version since the second iteration is created to minimize the number of piconets.

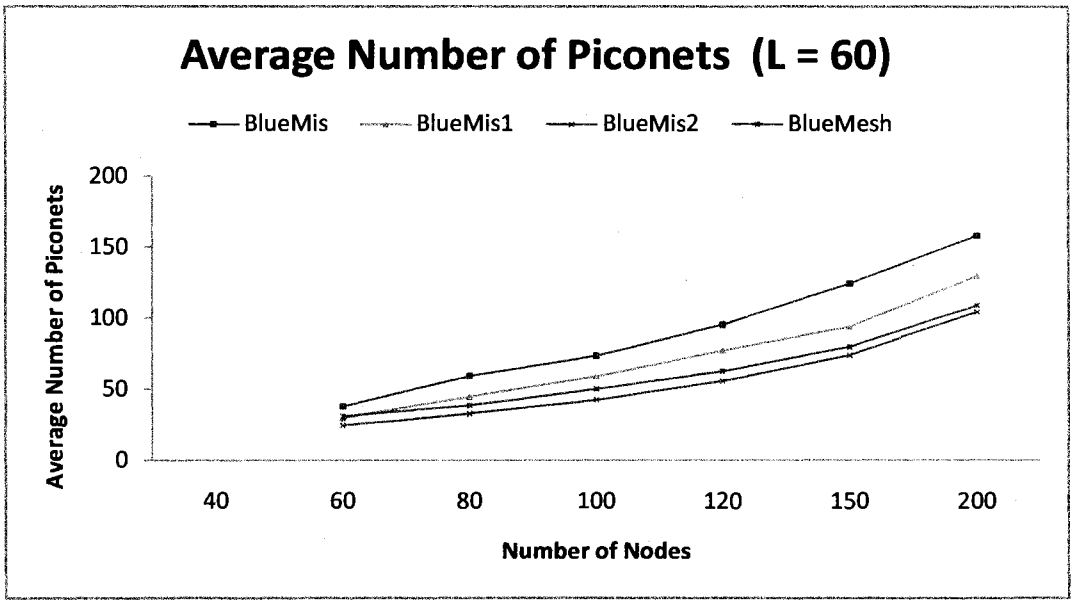


**Figure 5.10:** Average number of piconets in a moderately dense network

When the network becomes less dense, all BlueMis versions improve while BlueMesh deteriorates slightly, since participating nodes are far away and the selection upper bound degree is not always met.

# Chapter 6

# Conclusion and future work

In this thesis, we developed and implemented three different variations of a new protocol for Bluetooth scatternet formation of wireless devices. Our new protocol, called BlueMis, is a two phase protocol, in which the first phase is discovery and the second phase is scatternet formation; it essentially interprets the slave selection as the maximal independent set problem. Our protocol attempts to simplify the BlueMesh procedure. The second phase of BlueMis is executed in two iterations, unlike BlueMesh that uses up to 4.7 iterations. The designed protocol is implemented and compared, in terms of various characteristics, with BlueMesh, which we consider to be the best competing protocol. The experiments show several advantages of our protocol over BlueMesh.

The BlueMis algorithm can be improved by reducing the number of piconets, without sacrificing connectivity. A node can decide to cancel its piconet if it observes that the scatternet remains connected locally after such a decision is made. Local knowledge consists of the awareness of two hop neighbours and existing piconets announced by neighbours. Note that a node may decide to keep its piconet because of local needs, although in reality the connection may exist through the rest of the network. The piconet cancellation procedure may be applied before breaking the mutual slave decisions of neighbouring nodes, so that the proper relation is selected based on the need to perhaps cancel one piconet completely rather than predetermine the outcome and thus potentially keep a piconet with a sole 'wrong' slave.

To apply the piconet cancellation procedure, we need to define an order of decision making among nodes. This can be decided by a *key3*, and can mimic a clustering style approach. Initially all nodes are undecided. The decision is made by a node that has the highest *key3* among all of its original undecided one hop

neighbours, and is communicated to all neighbours. This in turn allows some other nodes to make their decisions.

Note that there are other options to consider about the order of decisions making, which depend on what nodes to wait for before making decisions, and whether to first announce the original decision and then confirm/deny it. For instance, [SZ] has suggested first announcing the original decisions and then considering only the *key3* of selected slaves and masters of neighbouring intended piconets for finalizing decisions.

There is a number of Bluetooth scatternet formation protocols already proposed in the literature. It can be observed that very few of these protocols satisfy most of the characteristics desired. There are relatively few actual implementations and comparisons done in the literature. Device discovery appears to be the most time consuming operation. However forming scatternets is still a formidable task, because of the device discovery and the extra complexity imposed by Bluetooth technology on the implementation of the distributed algorithms.

We anticipate that more Bluetooth scatternet formation schemes will be developed in the near future, and that some modifications to Bluetooth specifications could be made to find solutions that satisfy a number of desired properties and make Bluetooth suitable for commercial applications in multi hop scenarios. An interesting and majorly open problem in the area is to design Bluetooth scatternet formation algorithms that will guarantee connectivity and degree limitation (for both the master and the slave roles) without using the position information.

# Bibliography


[ACNCG] **M. Ajmone-Marsan, C.F. Chiasserini, A. Nucci, G. Carello, and L. de Giovanni**, Optimizing the Topology of Bluetooth Wireless Personal Area Networks. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, Volume 2, pp.572–579, 2002.

[AKRS] **A. Aggrawal, M.Kapoor, L. Ramachandran, and A. Sarkar.** *Clustering algorithm for wireless ad hoc networks*, Proceedings of the fourth International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, pp. 54-63, 2000.

[BBFMSK] **S. Baatz, C. Bieschke, M. Frank, P. Martini, C. Scholz and C. Kueh,** *Building efficient Bluetooth scatternet topologies from 1-factor*, Proceeding IASTED International Conference on Wireless and Optical Communication, WOC 2002, Banff, Alberta, Canada, pp. 300-305, July 2002.

[BBMP] **S. Basagni, R. Bruno, G. Mambrini and C. Petrioli,** *comparative performance Evaluation of Scatternet Formation Protocols for Networks of Bluetooth Devices,* ACM/Kluwer Wireless Networks, Volume 10, Issue 2, pp. 197-213, Mar. 2004.

[CCJ] **B. N. Clark, C. J. Colbourn, and D. S. Johnson,** *Unit Disk Graphs*, Discrete Mathematics, Volume 86, Issue 1-3, pp. 165-177, 1990.

[DHMPP] **D. Dubashi, O. Haggstrom, G. Mambrini, A. Panconesi and C. Petrioli,** *BluePleiades, a new solution for device discovery and scatternet formation in multi-hop Bluetooth networks,* Wireless Networks, Volume 13, Issue 1, pp. 107-125, 2006.

[LLS] **Y. Liu, M.J. Lee and T.N. Saadawi,** *A Bluetooth scatternet route structure for multi-hop ad hoc networks,* 2, IEEE J. Selected Areas in Communications, Volume 21, Issue 2, pp, 229-239, Feb. 2003.

[LMS1] **C. Law, A.K. Mehta and K.Y. Siu**, *A new Bluetooth scatternet formation protocol*, Mobile Networks and Applications, Volume 8, Issue 5, pp. 485-498, Oct. 2003.

[LSW] **X.-Y. Li, I. Stojmenovic and Y. Wang,** *Partial Delaunay triangulation and degree limited localized Bluetooth scatternet formation*, IEEE Trans. Parallel and Distributed Systems Volume 15, Issue 4, pp. 350-361, April 2004.

[LTC] **T.Y. Lin, Y.C and Tseng, K.M. Chang,** *A new BlueRing scatternet topology for Bluetooth with its formation, routing, and maintenance protocols,* Wireless Communications and Mobile Computing, Volume 3, Issue 4, pp. 517-537, June 2003.

[MeZ] **V. Mehta and M. El Zarki,** *A Fixed Sensor Networks for Civil Infrastructure Monitoring..*, Wireless Networks, pp. 401-412, July 2004.

[MRTVJ] **G. Miklós, A. Rácz, Z. Turányi, A. Valkó, and P. Johansson,** *Performance aspects of Bluetooth scatternet formation..*: First Annual Workshop on Mobile and Ad Hoc Networking and Computing, MobiHOC. 2000, Boston, Massachusetts, USA, pp. 147-148, Nov. 2000.

[PB] **C. Petrioli and S. Basagni,** *Degree-constrained multihop scatternet formation for Bluetooth networks,.* IEEE Global Telecommunications Conference, GLOBECOM apos:02, Volume 1, Issue 17-21, pp 222-226, 2002.

[PBC] **Chiara Petroli, Stefano Basagni and Imrich Chlamtac,** *BlueMesh: Degree-Constrained Multi-Hop Scatternet Formation for Bluetooth Networks*, Mobile Networks and Applications, pp. 33-47, 2004.

[PBC2] **C. Petrioli, S. Basagni, and I. Chlamtac,** *Configuring BlueStars: Multihop scatternet formation for Bluetooth networks,* IEEE Trans. Computers, Volume 52, pp. 779-790, 2003.

[PK] **C. Pamuk and E. Karasan,** *SF-Devil: Distributed Bluetooth scatternet formation algorithm based on device and link characteristics*, Proceedings of the Eighth IEEE International Symposium on Computers and Communications, pp 646-651, 2003.

[RKSA] **L. Ramachandran, M. Kapoor, A. Sarkar and A. Aggarwal,** *Clustering algorithms for ad hoc wireless networks,* Workshop on Discrete Algorithms and Methods for MOBILE Computing and Communications, Boston, Massachusetts, United States, pp. 54-63, 2000.

[S] **I. Stojmenovic,** Routing in Bluetooth with geometric structures enhanced with long links. Pending publication.

[S1] **I. Stojmenovic,** *Dominating set based Bluetooth scatternet formation with localized maintenance*, Proceedings of the 16th International Parallel and Distributed Processing Symposium, pp. 148-155, April 2002.

[SBTL] **T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire,** *Distributed topology construction of Bluetooth personal area networks*, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE INFOCOM 2001, Volume 3, pp. 1577–1586, April 2001.

[SCH] **T.V.L.N Sivakumar, H. Chen and L. Huang,** *Adaptive Network Formation Protocol for Bluetooth Scatternet,* The First IEEE and IFIP Int. Conf. on Wireless and Optical Communications Networks (WOCN 2004), Muscat, Oman, June 2004 .

[SIG] **Bluetooth Special Interest Group,** *Bluetooth Specification Version 1.2,* [Online], http://www.bluetooth.com, November 2003.

[SLWW] **Wen-Zhan Song, Xiang-Yang Li, Yu Wang and WeiZhao Wang,** *dBBlue: low diameter and self-routing Bluetooth scatternet* Journal of Parallel and Distributed Computing, Volume 65, Issue 2, pp. 178-190, 2005.

[SZ] **I. Stojmenovic and N. Zaguia.** Bluetooth scatternet formation in ad hoc wireless networks, **J.Misic and V. Misic,** *Performance Modeling and Analysis of Bluetooth Networks: Network Formation, Polling, Scheduling, and Traffic Control.* Boca Raton, Florida, USA : Auerbach, pp. 147–171, 2006.

[WSL] **Yu Wang, Ivan Stojmenovic and Xiang-Yang Li,** *Bluetooth scatternet formation for single-hop ad hoc networks based on virtual positions*, Proceedings ISCC 2004. Ninth International Symposium on Computers and Communications, Volume 1, Issue 28, pp. 170-175, July 2006.

[ZBC] **G.V. Zaruba, S. Basagni and I. Chlamtac,** Bluetrees - scatternet formation to enable Bluetooth based ad hoc networks, Proceedings of the IEEE International Conference on Communications, ICC 2001, Volume 1, Helsinki, Finland, pp. 273-277, June 2001.

[ZHS] **H. Zhang, Jennifer C. Hou and Lui Sha,** *Bluetooth Loop Scatternet Formation Algorithm.* May 2003. IEEE International Conference on Communications, Volume 2, Issue, 11-15, pp. 1174-1180, May 2003.