NEW BLUETOOTH SCATTERNET CONCEPT

A DISSERTATION SUBMITTED TO THE INSTITUTE OF COMMUNICATION AND SIGNAL PROCESSING, DEPARTEMENT OF ELECTRONIC AND ELECTRICAL ENGINEERING, AND THE COMMITTEE FOR POSTGRADUATE STUDIES OF THE UNIVERSITY OF STRATHCLYDE IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

By

Christophe Lafon

May 2005

DECLARATION

I declare that this Thesis embodies my own research work and that it is composed by myself. Where appropriate, I have made acknowledgement to the work of others.

Christophe Lafon

ACKNOWLEDGMENTS

I would like to express my gratitude to Prof. Tariq Salim Durrani, my supervisor, for its excellent guidance, constant encouragement and constructive advice throughout the course of my PhD research. Many thanks for his invaluable suggestions and for reviewing this thesis.

I am truly grateful to all the members of ISLI, based in Livingston, for providing a friendly environment, and Cadence for granting access to their development boards and software. In particular I would like to thank Robert Forbes for offering his knowledge and experience on the Bluetooth development boards environments, and to thank Donald Deasy for his precious help.

Finally, I dedicate the success of this thesis to all my family and to all the friends present around this world.

ABSTRACT

The emergence of Bluetooth as a wireless network solution assists in bringing together multiple technologies in different sectors and provides rapid interconnections to form a network paradigm. Typically, up to 8 Bluetooth devices can form a centralized network, called a *piconet*, controlled by a master node, which allocates transmission slots to all other nodes (slaves) in the piconet. However, the structure of inter-piconet connection: called the *scatternet* is not defined in the Bluetooth Specification. To develop a new scatternet structure, many challenges such as topology formation, intra and inter-piconet scheduling and packet routing are considered. The thesis addresses these critical issues based on the scatternet formation using Bluetooth.

The scatternet, presented in this thesis, employs a tree hierarchy structure formation with a Leader root of three hierarchies. Within the scatternet, the new concept exploits clock and frequency synchronization for all new piconets creation. This synchronization prevents interferences and the need for guard time while switching piconet. Thus enabling a device to switch from one piconet to another at every slot. Furthermore, an innovative intra-piconet design is proposed to improve QoS within Bluetooth piconet. By exploiting the device queue status, the scheme defines a predictable polled sequence, and an adaptive traffic allocation. This offers a better fairness, and a significant power reduction when compared to the conventional Round Robin scheme.

Moreover, with the perfect scatternet synchronization, the devices switch to other piconets to transmit data and, within one slot time, return to the initial piconet before the next predicted poll time exchange occurs. This considerably improves the traffic data transfer, especially for a significant number of devices present within the area.

In addition, a new routing process is developed in this thesis, which facilitates communication within the scatternet. From the scatternet tree hierarchy position, a new addressing node routing is proposed to keep the overhead network low, and

guarantees that any packet forwarded reaches its destination. The performance of the new scatternet is evaluated for each scatternet phase, through a Matlab simulation program, and the significant improvement of Bluetooth QoS achieved through the proposed approach is fully demonstrated.

This thesis also presents a primary implementation of the scatternet concept using a dedicated Bluetooth hardware system.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ACK	Acknowledgement
ACL	Asynchronous Connectionless Link
AFP	Adaptive Flow-based Polling
AM_ADDR	Active Member Address
AMSQP	Adaptive Master Slave Queue Polling
ARQ	Automatic Repeat reQuest
BD_ADDR	Bluetooth Device Address
BTCP	Bluetooth Topology Construction Protocol
Bts	Bluetooth Software
BW	Bandwidth
CAC	Channel Access Code
CBR	Constant Bit Ratio
CDMA	Code Division Multiple Access
CLK	Master Clock
CLKE	Estimated Clock
CLKN	Native Clock
COM	Component Object Model
CRC	Cyclic Redundancy Check
DA	Destination Address
DAC	Device Access Code
DC (offset)	Data Conversion
DH	Data High rate
DIAC	Dedicated Inquiry Access Code
DM	Data Medium rate
DRR	Deficit Round Robin
DS	Downlink Slave
ERR	Exhaustive Round Robin

FEC	Forward Error Correction
FEP	Fair Exhaustive Polling
FF	Forwarding Flag
FHS	Frequency Hopping Sequence
FHSS	Frequency Hopping Spread Spectrum
FPGA	Field Programmable Gate Array.
FTP	Field Transfer Protocol
GFSK	Gaussian Frequency Shift Keying
GIAC	General Inquiry Access Code
GSC	(Grandchild)Slave
GSM	Global System for Mobile communications
HCI	Host Controller Interface
HEC	Header Error Check
HF	Hop Frequency
HTTP	HyperText Transfer Protocol
HV	High Quality Voice
IAC	Inquiry Access Code
ID	Identity
IEEE	Institute of Electronic and Electrical Engineering
IrDA	Infrared Data Association
IP	Internet Protocol
ISLI	Institute for System Level Integration
ISM	Industrial - Scientific – Medicine
L2CAP	Logical Link Control and Adaptation Protocol
LAA	Load Adaptive Algorithm
LAN	Local Area Network
LAP	Lower Address Part
LIAC	Limited Inquiry Access Code
LM	Link Manager
LMP	Link Management Protocol
LSB	Least Significant Bit
LWRR	Limited and Weighted Round Robin

MAC	Media Access Control
MP	Max_Priority
MTU	Maximum Transmission Unit
MS	Master Slave
MSB	Most Significant Bit
OA	Original Address
OBEX	Object EXchange Protocol
PAN	Personal Area Network
PC	Personal Computer
PCSS	Pseudo Random Coordinated Scheduling Algorithm
PDA	Personal Digital Assistant
PFP	Predictive Fair Polling
PLL	Phase-Locked-Loop
PM_ADDR	Park Member Address
PRR	Priority Round Robin
ppm	Parts Per Million
PPP	Point to Point Protocol
QoS	Quality of Service
RFCOMM	Serial Cable Emulation Protocol
RP	Rendezvous Point
RR	Round Robin
RRL	Round Robin List
Rx	Receive
SAR	Segmentation and Reassembly
SCO	Synchronous Connection Oriented
SDP	Service Discovery Protocol
SEQN	Sequential Numbering scheme
SIG	Special Industry Group
SPM	Simple Piconet Model
SR	Slave Ring
SS	Slave Slave
TCP	Transport Control Protocol

TDD	Time Division Duplex
TDMA	Time Division Multiple Access
TH	Tree Hierarchy
TSF	Tree Scatternet Formation
UAP	Upper Address Part
UART	Universal Asynchronous Receiver Transmitter
UDP	User Datagram Protocol
USB	Universal Serial Bus
UMTS	Universal Mobile Telephone System
VHDL	VHSIC Hardware Level Description Language
VHSI	Very High Speed Integrated Circuit Programme
WAP	Wireless Application Protocol
WIFI	Wireless Fidelity
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network

CHAPTER 1
INTRODUCTION

1.1 WIRELESS NETWORK

Wireless networks can be classified into two distinct categories: with or without fixed access points. A cellular network that is used in the mobile phone system represents the most common form of infrastructure with fixed networks. In such a system, base-stations act as bridges that connect the mobile terminals together. As opposed to this, *Ad Hoc Networks* have no fixed routers or access points. Their main characteristic is that all of the nodes are *peers,* and they can be connected to each other dynamically in an arbitrary manner. Nodes in this network function as means of discovering devices and maintaining routes to other nodes in the network (Chakrabarti *et al [2001]*).

1.1.1 The Removal of Cable Connection

Communication within a local area network (LAN) permits computers to talk to each other or to their peripherals such as printers and scanners. The LAN can encompass a coverage area of several rooms, floors in different buildings, and can be either wired or wireless.

Since each node present in the network requires access to the network at a particular location, a wired network obviously needs lots of cable. Such an electrical system has a significant impact in terms of cost and inconvenient cable upgrade especially in old building, while wireless local network (WLAN) can be exploited simply by upgrading new hardware and software into computers.

Nevertheless a wired network: such as the wired Universal Serial Bus (USB) 1.x operates at speeds of about 12 Mb/s, and USB 2.x can send data at rates about 480Mbit/s, while IEEE 1394 works at 100 to 400 Mb/s. Bluetooth operates at a lower

rate of 1Mbit/s, and the well known WLAN institutes of IEEE 802.11b also called Wi-fi works at up to 11 Mb/s. Hence wired network outperform the wireless speed (Bray *et al [2001]*).

1.1.2 Bluetooth Wireless Technology

Bluetooth, standardized in 1999, is a new and promising technology since it carries the WLAN concept to a smaller scale. Bluetooth is designed to be small enough to be included in portable devices such as mobile phones and personal digital assistants (PDAs), but many usage scenarios also involve laptops, desktops, printers, cameras and other types of devices. Along with its low-power and short (10-meter) range, Bluetooth is more suited for connecting devices that are located within the same room, or even on a person. This concept is called the personal area network (PAN). The goal is to accommodate seamless information transfer between different devices without the need for manual configuration or wired infrastructure. This enables changing ability in an *ad hoc* manner (Bray *et al [2001]*).

Bluetooth offers advantages over other competing technology; these include technical features such as non-line-of-sight communication, low power consumption, low cost and usage of frequency hopping.

Besides cable replacement (e.g. between an application running on a PC and a modem), Bluetooth also provides numerous services, such as auto-detection, service browsing (discovering of available services delivered by the devices) and so on. It supports numerous protocols, and allows multiplexing (i.e. numerous links at the same time).

In fact, Bluetooth technology can form a network up to 8 active devices called *piconet* with one node becoming a master and the rest having the role of slaves. The technology promises to enable connectivity not for only a small number of devices in an isolated piconet, but to extend the network by interconnecting multiple piconets. This will form a larger ad hoc network called *scatternet*. A scatternet can consist of more than hundreds devices, with connectivity distances greater than the initial short radio range.

1.2 MOTIVATION

As a contrast to the piconet configuration and procedure, the scatternet formation and operation is not standardised yet. The Bluetooth specification enables communication between two piconets but does not specify how the *bridge* (device part of both piconet) should exactly interact. In fact the scatternet research has been a major area of study in the last few years, and has led to new concept.

The goal of this thesis is to construct a scatternet over new topology formation, new scheduling, new routing, while keeping a superior Quality of Services (*QoS*) through some Bluetooth features such as interferences, power consumption, latency, throughput and fairness.

1.3 THESIS ORGANIZATION

To provide some background knowledge about Bluetooth, the thesis begins with a brief description of Bluetooth history and its protocol stack technology in **CHAPTER 2**. The Bluetooth baseband with packet transmission features are also presented in this chapter.

CHAPTER 3 provides different scatternet topology formation and reviews, and describes a new topology formation using a tree structure. The originality is that all the piconet are synchronised in frequency and in time with one node called *Leader*. The objective is to avoid interferences and guard time, while devices switch between piconets.

One of the most critical factors in designing a new scatternet configuration is to attempt to reduce the power system consumption, which is one of the major purposes of Bluetooth. As portable devices are demanding longer battery life and higher performance, even power reductions on the order of 5mW are crucial to portable system designer and manufacturers. Therefore **CHAPTER 4** evaluates the current Bluetooth *intra-piconet* polling application that utilizes the *Round Robin* (RR) scheme. It is shown that the RR scheme performs weakly, and for that reason a new polling scheme, called *Adaptive Master/Slave Queue Polling* scheme, has been

created with simplified modification. In this chapter, results through extensive simulations compare the new features in terms of fairness, delay and power consumption.

In **CHAPTER 5,** recent researches on bridge scheduling (a node becomes a *bridge* while switching to different piconet) are presented. A new scatternet scheduling piconet is proposed as an extension of the predicted AMSQP scheme. The performance of the proposed predictive inter-piconet are explained and evaluated through simulations. Results show the scheduled efficiency of the system-wide throughput while reducing the end-to end latency.

As a scatternet consists of several piconets there is a pressing need to develop an efficient ad hoc routing algorithm to facilitate communication possible between distant devices. In **CHAPTER 6**, a new forwarding scheme and a new multi-slave connection are presented and compared through simulation. And the simulation performances of the new routing scatternet scheme are analyzed. Then two examples of viable network connection with the new scatternet concept are presented and discussed.

A presentation of the Bluetooth work equipment along with the Bluetooth development boards features are given in **CHAPTER 7**. An implementation using the Bluetooth software of the scatternet is presented in this chapter and using simulation with real condition, the attendant data communications is analyzed.

Finally, a conclusion is provided in **CHAPTER 8** with suggestions of some future research directions.

1.4 ORIGINAL CONTRIBUTIONS

- The first major contribution is the design and development of an innovative tree hierarchy scatternet, which perfectly synchronises a maximum of 400 devices in both time and frequency. As a consequence no guard time and no inter-piconet interference can occur within the scatternet. No other topology, has achieved the resolution of this two metrics.

- The second major contribution is the QoS improvement achieved in a Bluetooth network by applying a novel polling scheme technique. For intra-piconet, the innovative scheme drastically reduces the power consumption and significantly improves the fairness of the Bluetooth system network, when compared to the Round Robin scheme.

- The third major contribution is the combination of both designs. By integrating successfully the new tree structure with the new polling scheme produces an efficient inter-piconet scheduling, which adapts the traffic accurately and allows interconnectivity between piconets within one slot time.

- The fourth contribution is the development of a straightforward routing process. By applying a simple node address system (that keeps the network overhead low) each packet identifies the route path through the scatternet itself within 6 multi-hop sequences. Furthermore, the routing facilitates the establishment of other piconets by forwarding adequate paging information.

1.5 LIST OF PUBLICATIONS

- **C.Lafon**, T.S.Durrani, "New Bluetooth Inter-Piconet Schedule with a Slave to Slave Piconet Formation", 5th IEE Conference EPMCC 2003, pp 81-85 Glasgow, UK.

- **C.Lafon**, T.S.Durrani, "Bluetooth Throughput Improvement Using a Slave to Slave Piconet", 6th IEEE International Conference HSNMC 2003, pp 254-263, Estoril, Portugal, July 23-25.

- **C.Lafon**, T.S.Durrani, "New Routing Strategy For Bluetooth Scatternet", ICSP colloquium 2004, Glasgow, UK.

- **C.Lafon**, T.S.Durrani, "New Bluetooth Scatternet synchronization" submitted to IEEE Journal on Selected areas in communications.

- **C.Lafon**, T.S.Durrani, "New Bluetooth Topology and Intra-Piconet Formation" in preparation for WINET journal.

CHAPTER 2

BLUETOOTH TECHNOLOGY

2.1 BLUETOOTH HISTORY

Bluetooth, named after Denmark's first Christian king Harald Blåtand, is the name of a technology specification for low-cost, short-range radio links between PCs, mobile phones and other computing and electronic devices (Carey *et al [2001]*). Ericsson originally developed the technology. The Ericsson inventors understood that the technology was more likely to be widely accepted, and thus could be more powerful, if it was adopted by an industry group that produce an open, common specification (*FAQ [2004]*). The Bluetooth SIG (Special Interest Group) is an industry group consisting of leaders in the telecommunications and computing industries that are driving development of the Bluetooth technology and bringing it to market. The founding companies are Ericsson, Intel, IBM, Nokia and Toshiba. Today over 2000 companies have signed the Bluetooth adopter agreement and are members of the Bluetooth SIG (Miller *et al [2000]*). There are already a couple of ways to get around using wires. One is to carry information between components via beams of light in the infrared spectrum the second is wireless using radio band. Infrared refers to light waves of a lower frequency than human eyes can receive and interpret. Infrared is used in most television remote control systems, and with a standard called IrDA (Infrared Data Association) it is used to connect some computers with peripheral devices. Bluetooth is intended to get around the problems that come with both infrared and cable synchronizing systems. From the user's point of view, there are three important features of Bluetooth:

- It is wireless. When you travel, you do not have to worry about keeping track of a briefcase full of cables to attach all of your components. Bluetooth devices do not need line of sight as IrDA devices for proper communication while consuming more power.

- You do not have to buy Bluetooth enabled devices from the same manufacturer. Bluetooth guarantee interoperability between different manufacturers.
- It is inexpensive.

2.2 BLUETOOTH SYSTEM CHARACTERISTICS

Devices connected via Bluetooth technology connect in an ad hoc fashion. When only two Bluetooth devices connect they form a network, which in the Bluetooth standard is called a *piconet*. Such as a portable PC and a cellular phone form a piconet network, and may grow up to eight connected devices. All Bluetooth devices are peer units and have identical implementations protocol to allow the communication between two Bluetooth devices.

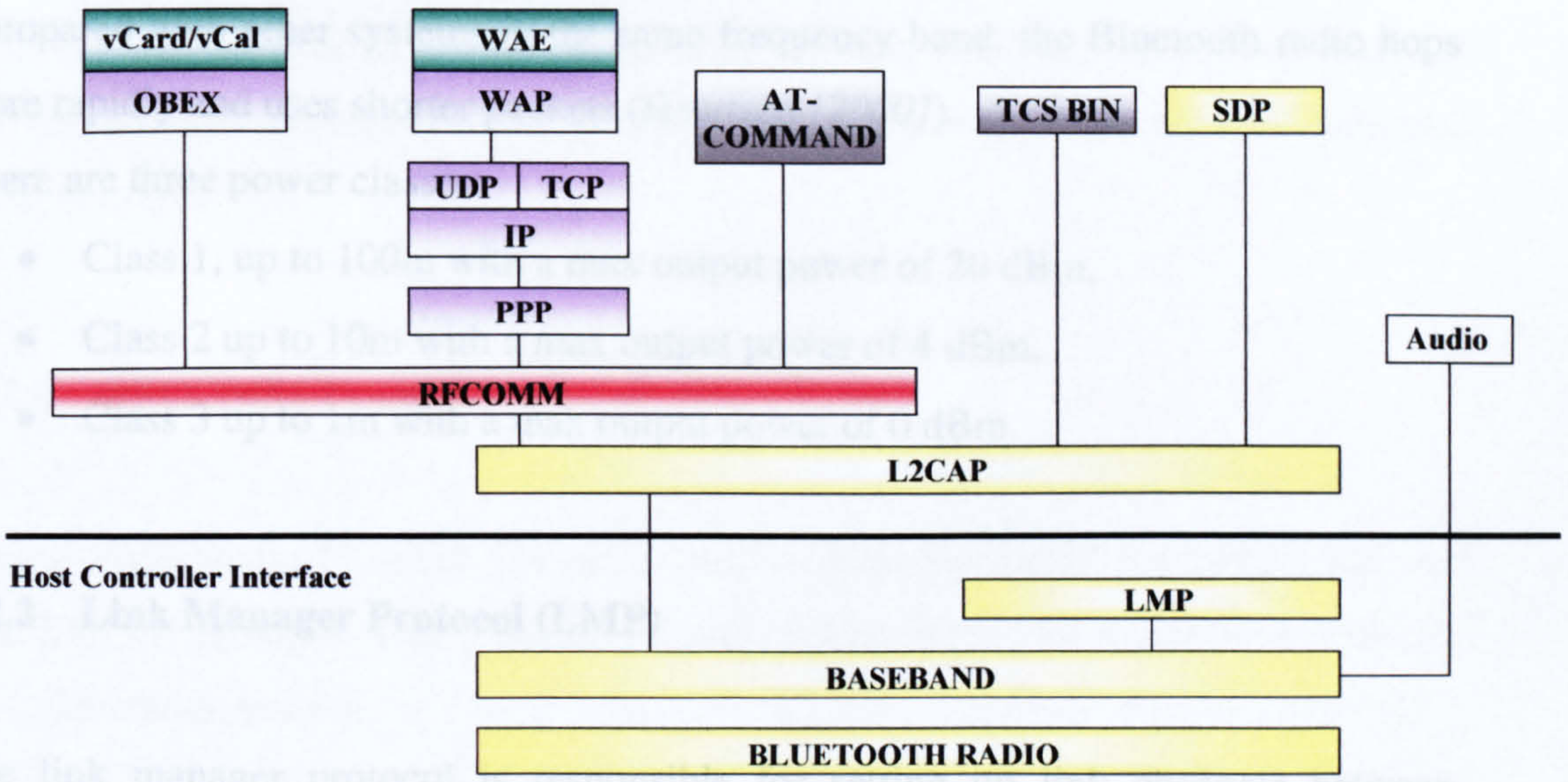


Figure 2.1: Bluetooth Protocol Stack

The complete Bluetooth protocol stack is illustrates by Figure 2.1 (*Technology [2002]*) and has been designed to include the existing protocols as much as possible (like TCP, UDP, OBEX) as well as Bluetooth specific protocols like LMP (Link Manager Protocol) and L2CAP (Logical Link Controller and Adaptation Protocol).

The protocol reuse ensures smooth interoperability between existing applications and hardware. The Specification is also open, thereby allowing vendors to build proprietary applications.

The following sections offer more detail about different subdivision of the protocol (*Nokia [2003]*).

2.2.1 Bluetooth Radio

The Bluetooth radio chip functions at 2.4 gigahertz, which is unlicensed Industrial, Scientific and Medical (ISM) band at 2.402 to 2.4808 GHz (though this bandwidth is narrower in Japan, France, and Spain). The radio uses a spread spectrum, frequency hopping, and full-duplex signal at up to 1600 hops/sec. The signal hops among 79 frequencies at 1 MHz intervals to give a high degree of interference immunity. Compared with other systems in the same frequency band, the Bluetooth radio hops more rapidly and uses shorter packets (Haartsen *[2000]*).

There are three power classes:

- Class 1, up to 100m with a max output power of 20 dBm,
- Class 2 up to 10m with a max output power of 4 dBm,
- Class 3 up to 1m with a max output power of 0 dBm.

2.2.2 Link Manager Protocol (LMP)

The link manager protocol is responsible for setting up link channels between Bluetooth devices, authentication, link configuration and encryption. The LM generates, exchanges and verifies linking and encryption keys and negotiates Baseband packet size. The link manager discovers other link managers and communicates directly with each other through LMP messages. The LM uses the services of the link controller (LC) to perform its service provider role (*Palowireless [2004]*).

2.2.3 Logical Link Control and Adaptation Protocol (L2CAP)

L2CAP packets carry payload, which are carried to the upper layer protocol giving protocol multiplexing capability, segmentation and reassembly operation and group abstractions. The L2CAP packets are defined for ACL link only.

2.2.4 Service Discovery Protocol (SDP)

Using SDP devices information, allows Bluetooth devices to discover what other Bluetooth devices can offer. All information of the Bluetooth services attributes is available on the service record. And each service attribute describes a single characteristic of a service. A service describes any entity that can provide information, perform an action, or control a resource on behalf of another entity. A service may be implemented as software, hardware, or a combination of hardware and software.

2.2.5 RFCOMM Protocol

Cable replacement Protocol is a serial line emulation protocol for serial ports over the L2CAP protocol. The protocol supports 60 simultaneous connections between two Bluetooth devices.

2.2.6 Host Controller Interface (HCI)

The HCI Transport is an interface designed to abstract and simplify physical communication between the Bluetooth stack and the controller. This set of modules implement UART, COM, or USB transports.

The purpose of the HCI software is to make the hardware that comprises the interface transparent to higher-level software in the system (*Palowireless [2004]*).

2.2.7 Adopted Protocols

Bluetooth supports PPP, TCP/UDP/IP, OBEX and WAP protocols to maximize interoperability.

2.3 BLUETOOTH BASEBAND

The Baseband is the physical layer of Bluetooth, which manages physical channels and links. The Baseband protocol is implemented as a Link Controller, which carries out tasks such as link connection and power control in collaboration with the link manager. The Baseband also deals with asynchronous and synchronous links, manages packets and inquiry processes to discover other Bluetooth devices in the area, while it applies a time-division duplex (TDD) scheme to synchronize the channel. All these characteristics are presented in more detail in the next sections (*SIG [2001]*).

2.3.1 Master-Slave Definition

By definition, the master is represented by the Bluetooth unit that initiates the connection (to one or more slave units). Note that the names 'master' and 'slave' only refer to the protocol on the channel: the Bluetooth units themselves are identical. That is, any unit can become a master of a piconet, and once a piconet has been established, master-slave roles can be exchanged (see in more detail *Section 2.3.4.1*). Figure 2.2 shows a typical Bluetooth piconet consists of one master symbolises by a square, surrounded by a large circle that represents its transmission range, while slaves are depicted as small circle. All Bluetooth units participating in the piconet are time and hop synchronised to the channel:

- Time synchronised: The channel is divided into time slots of 625 μs with a nominal hop rate of 1600 hops/sec.

- Hop synchronised: the Bluetooth channel is represented by a pseudo-random hopping sequence hopping through the 79 channels in Europe and US, or 23 RF channels in Japan, Spain and France (Bluetooth devices are compatible and adapt the current hopping sequence among the country they operate). Each slot matches a RF hop frequency, and consecutive hops correspond to different RF hop frequencies.

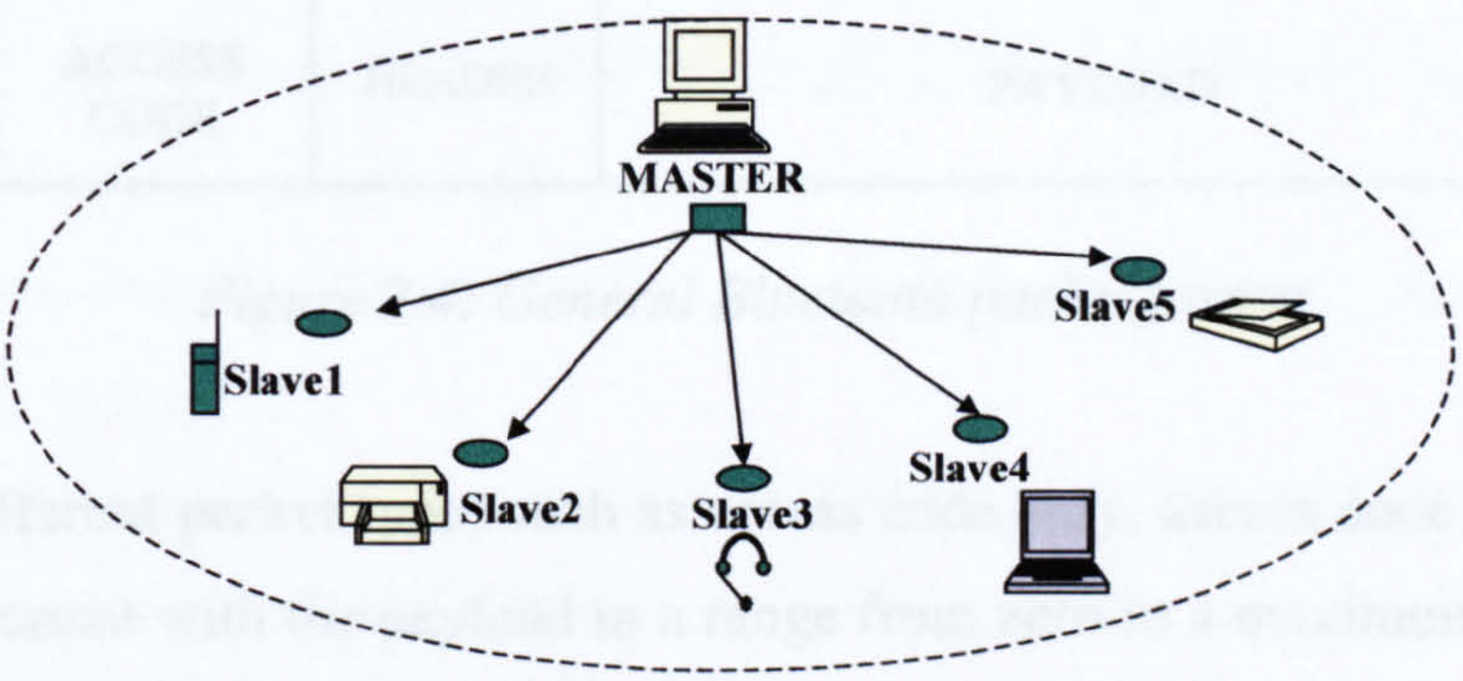


Figure 2.2: Simple Piconet illustration with one master and 5 Slaves.

2.3.1.1 Bluetooth Device Address

Bluetooth device address (BD_ADDR) uniquely identifies each Bluetooth device and it is 48 bits long. The format of the BD_ADDR is as shown by Figure 2.3 (*SIG [2001]*). BD_ADDR is used for calculating various access codes. Frequency hopping sequences are calculated from these access codes. Thus, to know a hopping sequence of a device one needs to know the device's BD_ADDR.

Non-significant address part	**Upper Address Part (UAP)**	**Lower Address Part (LAP)**
16	*8*	*24*
LSB		**MSB**

Figure 2.3: Structure of BD_ADDR

2.3.2 Packet General Format

The data in the piconet channel is conveyed in packets, the format of which is shown by Figure 2.4 (*SIG [2001]*). The packet bit ordering follows the "Little Endian" format: The *least significant bit* (LSB) of the access code is sent first. The packets used on the piconet are related to the physical links they are used in.

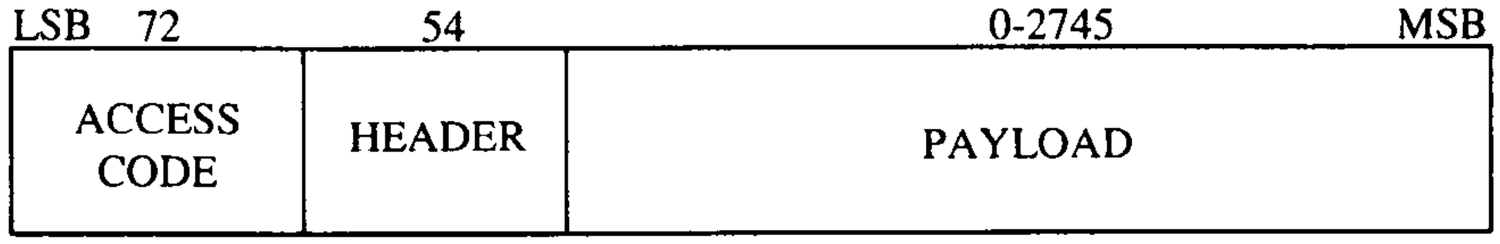


Figure 2.4: General Bluetooth packet format

There are different packet types such as access code only, access code + header, or a full packet format with the payload in a range from zero to a maximum of 2745 bits. Hence different packets could be formed.

2.3.2.1 Access Code

Each packet starts with an access code. The access code is 72 bits long, if a packet header follows; otherwise the access code is 68 bits long. This access code determine the synchronization, DC offset compensation and identification. All packets exchanged within the piconet are preceded by the same channel access code, and identify the piconet. In the receiver of the Bluetooth unit, a sliding correlator correlates against the access code and triggers when a threshold is exceeded. This trigger signal is used to determine the receiver timing. The access code is also used in paging and inquiry procedures (see chapter 4). The access code consists of a preamble, a sync word, and possibly a trailer, illustrates by Figure 2.5 (*SIG [2001]*).

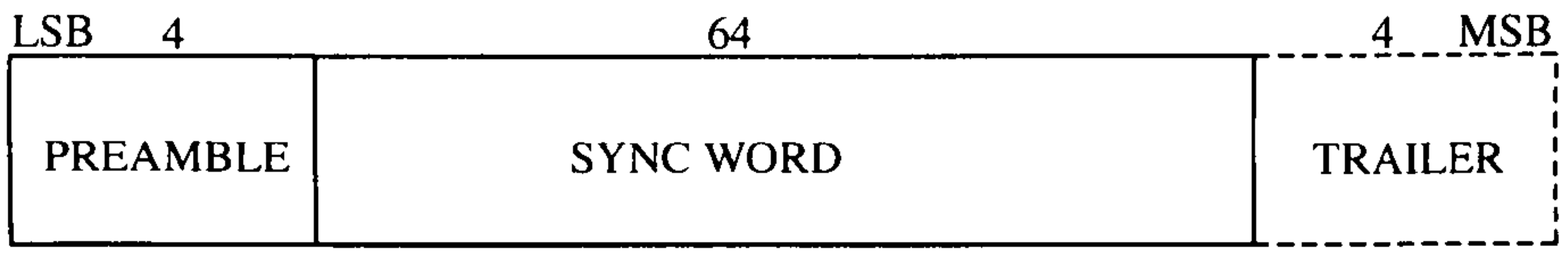


Figure 2.5: Access code format

A Bluetooth device uses three different types of access codes in different operating modes:

- **Channel Access Code (CAC)**: consists of a *preamble*, *sync word*, and *trailer* and its total length is 72 bits.
- **Device Access Code (DAC)**: used for self-contained messages without a header and is 68 bits long.
- **Inquiry Access Code (IAC):** do not include the trailer bits and are of length 68 bits.

For the inquiry access code there are two variations. A General Inquiry Access Code (GIAC) is common to all devices. The GIAC can be used to discover which other Bluetooth units are in range. The Dedicated Inquiry Access Code (DIAC) is common for a dedicated group of Bluetooth units that share a common characteristic. The DIAC can be used to discover only these dedicated Bluetooth units in range.

2.3.2.2 Packet header

The format of the header is as shown in Figure 2.6 (*SIG [2001]*):

AM_ADDR	TYPE	FLOW	ARQN	SEQN	HEC
3	*4*	*1*	*1*	*1*	*8*

LSB **MSB**

Figure 2.6: Packet Header Format

The entire header is encoded with a rate 1/3 Forward Error Correction Code (FEC), resulting in a 54 bit header.

- **AM_ADDR:** 3-bit active member address. The AM_ADDR represents a member address and is used to distinguish between the active members participating on the piconet. It identifies each slave separately and assigns a temporary address of 3 bits (maximum of 7 active slaves) to distinguish active slaves within the Piconet connected to the master. During a packet exchanged between the master and the slave, and the slave to the master, the packets carry the AM_ADDR of the slave. The all-zero AM_ADDR address is reserved for broadcasting packets from the master to the slaves see *Section 2.3.2.6*.

- **TYPE:** 4-bit type code. The 4-bit TYPE code specifies which packet type is used across sixteen different types of packets. The TYPE code also reveals how many slots the current packet will occupy. This allows the non-addressed receivers to refrain from listening to the channel for the duration of the remaining slots.

- **FLOW:** 1-bit flow control. This bit is used for flow control of packets over the ACL link. When the Received (RX) buffer for the ACL link in the recipient is full, a STOP indication (FLOW=0) is returned to stop the transmission of data temporarily. Packets including only link control information (ID, POLL and NULL packets) or SCO packets can still be received.

- **ARQN:** 1-bit acknowledge indication. The 1-bit acknowledgment indication ARQN is used to inform the source of a successful transfer of payload data, and can be positive acknowledge ACK or negative acknowledge NAK.

- **SEQN:** 1-bit sequence number. The SEQN bit provides a sequential numbering scheme to order the data packet stream. If a retransmission occurs due to a failing ACK, the destination receives the same packet twice. By comparing the SEQN of consecutive packets, correctly received retransmissions can be discarded.

- **HEC:** 8-bit header error check. Each header has a Header-Error-Check to check the header integrity. The HEC consists of an 8-bit word generated by the polynomial 647 (octal representation).

2.3.2.3 Common Packets Type

- **ID packet:** The identity or ID packet consists of the device access code (DAC) or inquiry access code (IAC). It has a fixed length of 68 bits

- **NULL packet:** Fixed length of 126 bits, the NULL packet has no payload and used to return link information to the source regarding the success of the previous transmission (ARQN), or the status of the RX buffer (FLOW). The NULL packet itself does not have to be acknowledged.

- **POLL packet:** The POLL packet is very similar to the NULL packet. But it requires a confirmation from the recipient. The POLL packet does not affect the ARQN and SEQN fields. Upon reception of a POLL packet the slave must respond with a packet. This return packet is an implicit acknowledgement of the POLL packet. This packet can be used by the master in a piconet to poll the slaves, which must then respond even if they do not have information to send.

- **FHS packet:** The Frequency Hopping Sequence packet is a special control packet revealing, the Bluetooth device address and the clock of the sender. The payload contains 144 information bits plus a 16-bit CRC (Cyclic Redundancy Check) code. The payload is coded with a rate 2/3 FEC that brings the gross payload length to 240 bits. The FHS packet covers a single time slot.

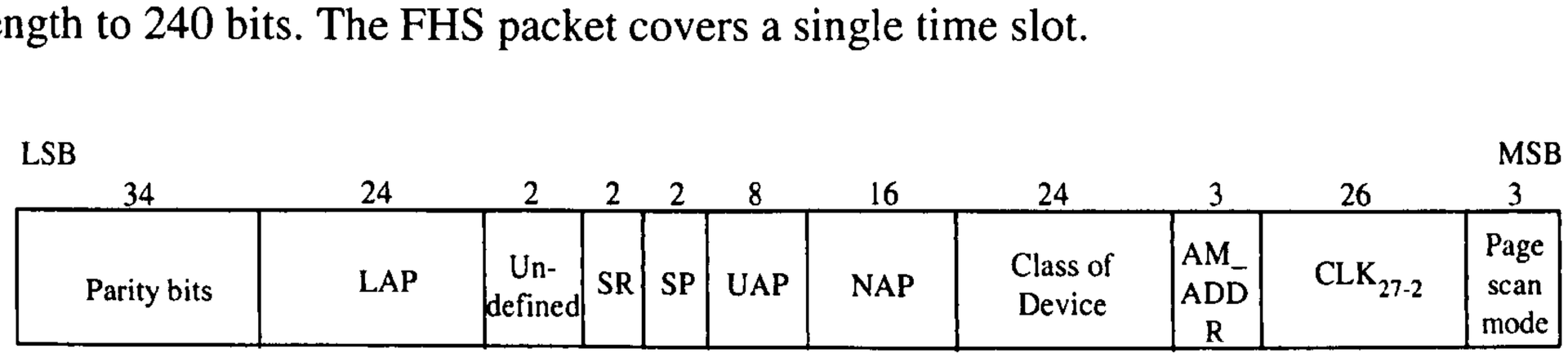


Figure 2.7: FHS packet format.

Figure 2.7 illustrates the format and contents of the FHS payload. The payload consists of eleven fields. The FHS packet is used for frequency hop synchronization before the piconet channel has been established.

The FHS packet contains real-time clock information. Therefore the FHS packets are used in page master response (see Chapter 4), in master slave switch and it is retransmitted until its reception is acknowledged or a timeout has exceeded. Figure 2.7 shows the FHS packet design (*SIG [2001]*).

- LAP: 24-bits field contains the Lower Address Part of the unit that sends the FHS packet.
- UAP field: Upper Address Part consisting of 8 bits
- NAP field: non-significant address part consisting of 16 bits
- Undefined: this 2-bit field is reserved for future use and shall be set to zero
- SR: this 2-bit field is the scan repetition field and indicates the interval between two consecutive page scan windows.
- SP: this 2-bit field is the scan period field and indicates the period in which the mandatory page scan mode is applied after transmission of an inquiry response message.
- Class of device: this 24-bit field contains the class of device of the unit.

2.3.2.4 ACL packets type

ACL packets are used on the asynchronous links. The information carried can be user data or control data. Seven ACL packets have been defined. Six of the ACL packets contain a CRC code and retransmission is applied if no acknowledgement of proper reception is received. The 7th ACL packet, has no CRC and is not retransmitted (*SIG [2001]*).

- **DM packets:** The DM packet is a packet that carries data information only. DM stands for Data-Medium rate. The payload contains up to 18 information bytes (including the 1-byte payload header) plus a 16-bit CRC code. The DM1 packet may cover up to a single time slot, DM3 three slots and DM5 five slots. The information plus CRC bits are coded with a rate 2/3 FEC that adds 5 parity bits

to every 10-bit segment. If necessary, extra zeros are appended after the CRC bits to get the total number of bits (information bits, CRC bits, and tail bits) equal a multiple of 10. The length indicator in the payload header specifies the number of user bytes (excluding payload header and the CRC code).

- **DH packets:** This packet is similar to the DM1 packet, except that the information in the payload is not FEC encoded. As a result, the DH1 frame can carry up to 28 information bytes (including the 1 byte payload header) plus a 16-bit CRC code. DH stands for Data –High rate. The DH1 packet may cover up to a single time slot, DH3 three slots, and DH5 five slots length.
- **AUX1 packet:** This packet resembles a DH1 packet but has no CRC code. The AUX1 packet can carry up to 30 information bytes (including the 1-byte payload header). The AUX1 packet may cover up to a single time slot.

2.3.2.5 Packet Summary:

Table 2.1 represents the twelve different packets types depending on their physical links: SCO and ACL link. Four segments divide the packet type: the first segment is reserved for the four control packets common to all physical link types. The second segment defines packets occupying a single time slot used by both SCO and ACL link. The third segment is reserved for packets occupying three slots and the fourth segment defines packets with five time slots, and both segment are used for ACL link only.

Type	Payload Header (bytes)	User Payload (bytes)	FEC	CRC	Symmetric Max. Rate (kb/s)	Asymmetric Max. Rate (kb/s)	
						Forward	Reverse
DM1	1	0-17	2/3	yes	108.8	108.8	108.8
DH1	**1**	**0-27**	**no**	**yes**	**172.8**	**172.8**	**172.8**
DM3	2	0-121	2/3	yes	258.1	387.2	54.4
DH3	2	0-183	no	yes	390.4	585.6	108.8
DM5	2	0-224	2/3	yes	286.7	477.8	108.8
DH5	2	0-334	no	yes	433.9	723.2	108.8
AUX1	1	0-29	no	no	185.6	185.6	108.8
HV1	na	10	1/3	no	64.0	no	no
HV2	na	20	2/3	no	64.0	no	no
HV3	na	30	no	no	64.0	no	no
DV*	1D	10+(0-9)D	2/3 D	yes D	64+57.6D	no	no

Table 2.1: Different packets Type, with DH1 packet selected for the Scatternet construction (SIG [2001]).

2.3.2.6 Broadcast Packets

Broadcast packets are packets transmitted by the master to all the slaves simultaneously. A broadcast packet is indicated by the all-zero AM_ADDR (note; the FHS packet is the only packet which may have an all-zero address but is not a broadcast packet). Broadcast packets are not acknowledged (at least not at the Link Controller level). Since broadcast messages are not acknowledged, each broadcast packet is repeated for a fixed number of times.

Broadcast packets with a CRC have their own sequence number. The SEQN of the first broadcast packet with a CRC is set to SEQN = 1 by the master and it is inverted

for each new broadcast packet with CRC thereafter. Broadcast packets without a CRC have no influence on the sequence number. The slave accepts the SEQN of the first broadcast packet it receives in a connection and checks for change in SEQN for consequent broadcast packets. Since there is no acknowledgement of broadcast messages and there is no end packet indication, it is important to receive the start packets correctly.

2.3.3 Time Slot

Bluetooth uses a Time Division Duplex (TDD) slot structure for resolving contention over the wireless links. The packet start shall be aligned with the slot start.

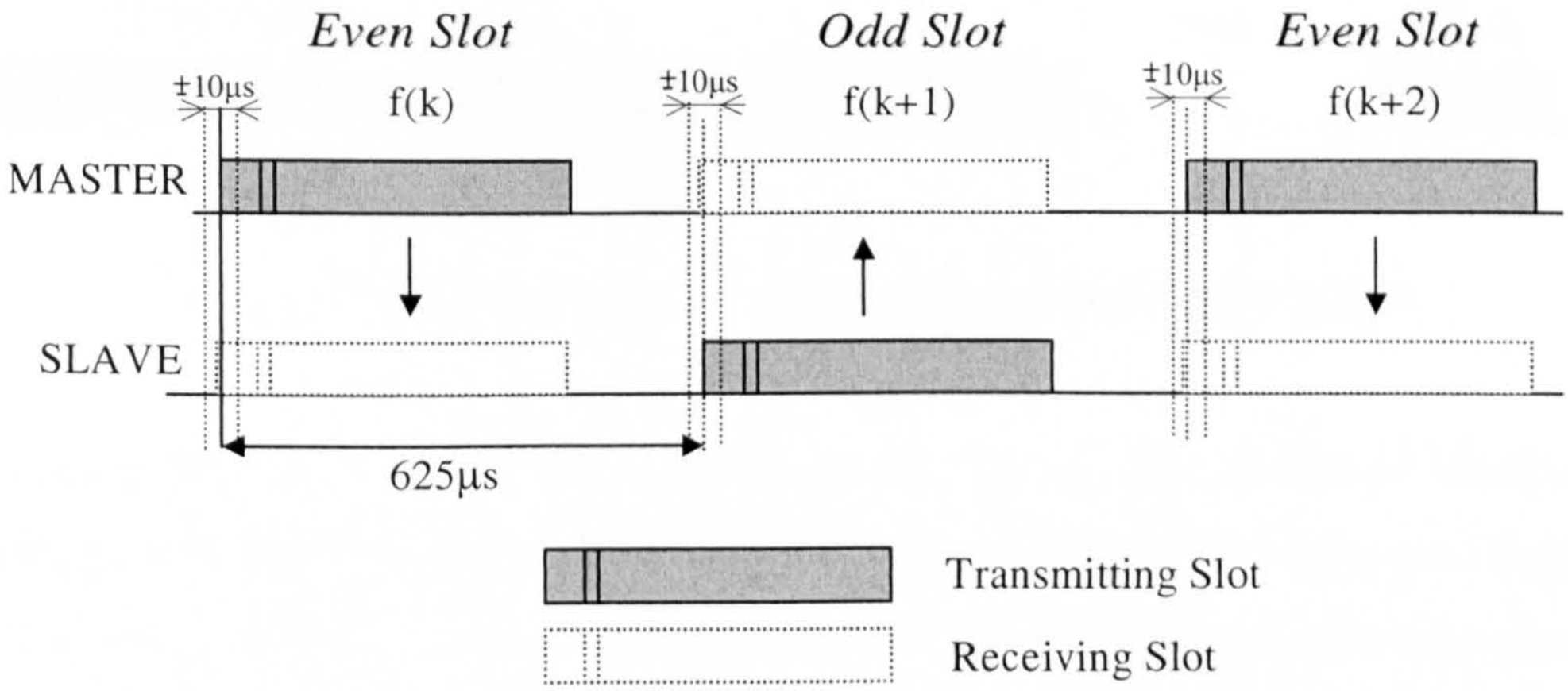


Figure 2.8: Packet exchange process between master and slave.

Figure 2.8 illustrates this TDD scheme; there is a strict alternation of slots between the master and the slaves. The master starts its transmission in even slots, while the slave starts its transmission in odd slots. The time slots are numbered according to the Bluetooth clock of the piconet master. The slot numbering ranges from 0 to 2^{27} -1 and is cyclic with a cycle length of 2^{27}. The RF hop frequency to be used is derived from the current Bluetooth clock value.

This implies that the scheduling occurs in pairs of slots. Usually one time slot of 625 µs is enough to transmit a package but in some cases, when the package is longer, the package is transmitted in 3 or 5 slots. For a multi-slot packet, the RF hop frequency

to be used for the entire packet is derived from the Bluetooth clock value in the first slot of the packet. Figure 2.9 illustrates the hop definition on single- and multi-slot packets. If a packet occupies more than one time slot, the hop frequency applied shall be the hop frequency as applied in the time slot where the packet transmission was started (Haartsen *[2000]*).

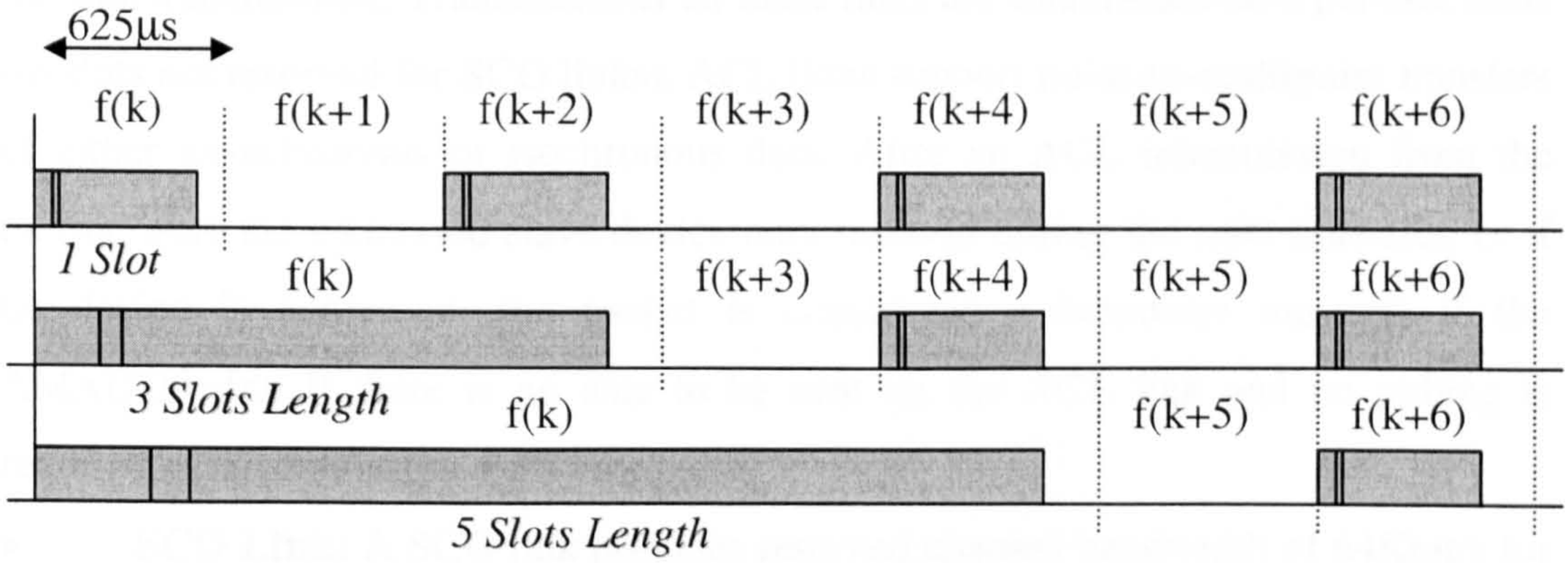


Figure 2.9: Multi-slot packets communication with hop selection.

The connection state starts with a POLL packet sent by the master to verify the switch to the master's timing and channel frequency hopping. The slave can respond with any type of packet. A slave transmits data only if in the previous time slots a message is contained from the master for the slave. If an active slave is not addressed, it may sleep until the next new master transmission. In fact, the packet type indicates the number of slots the master has reserved for its transmission; during this time, the non-addressed slaves do not have to listen on the master-to-slave slots. A periodic master transmission is required to keep the slaves synchronized to the channel. Since the slaves only need the channel access code to synchronize with, any packet type can be used for this purpose.

2.3.3.1 General Link

Bluetooth can support packets that carry Synchronous Connection-Oriented (SCO) link or Asynchronous Connection-Less (ACL) link. Each link type is associated with a specific packet type as shown by Figure 2.10.

- **ACL Link** Asynchronous Connection-Less (ACL) links are typically used for data transmission. Transmissions on these links are established on a per-slot basis (in slots not reserved for SCO links). ACL links support point-to-multipoint transfers of either asynchronous or isochronous data. After an ACL transmission from the master, only the addressed slave device may respond during the next time-slot, or if no device is addressed, the packet is considered a broadcast message if the AMADDR=00. If there is no data to be sent on the ACL link and no polling is required, no transmission shall take place.
- **SCO Link:** A SCO link provides reserved channel bandwidth of 64Kbit/s for communication between a master and a slave, and supports regular, periodic exchange of data with no retransmission of SCO packets. Synchronous connection-oriented (SCO) links are typically used for voice transmission. These are point-to-point symmetric connections that reserve time slots in order to guarantee low jitter transmission. The slave device is always allowed to respond during the time-slot immediately following a SCO transmission from the master. A master can support up to three SCO links to a single or multiple slaves, but a single slave can support only two SCO links to different masters. The master will send SCO packets at regular intervals, the so-called SCO interval T_{SCO} (counted in slots) to the slave in the reserved master-to-slave slots. This time period is equal to 3.75ms, which has to be transmitted every 6 slots. The SCO slave is always allowed to respond with an SCO packet in the following slave-to-master slot unless a different slave was addressed in the previous master-to-slave slot. If the SCO slave fails to decode the slave address in the packet header, it is still allowed to return an SCO packet in the reserved SCO slot. This message will contain timing parameters such as the SCO interval T_{SCO} and the offset D_{SCO} to specify the reserved slots. The slave-to-master SCO slots shall directly follow the reserved master-to-slave SCO slots. After initialization, the clock

value CLK(k+1) for the next master- to-slave SCO slot is found by adding the fixed interval T_{SCO} to the clock value of the current master-to-slave SCO slot:

CLK(k+1) = CLK(k) + T_{SCO}. (*SIG [2001]*)

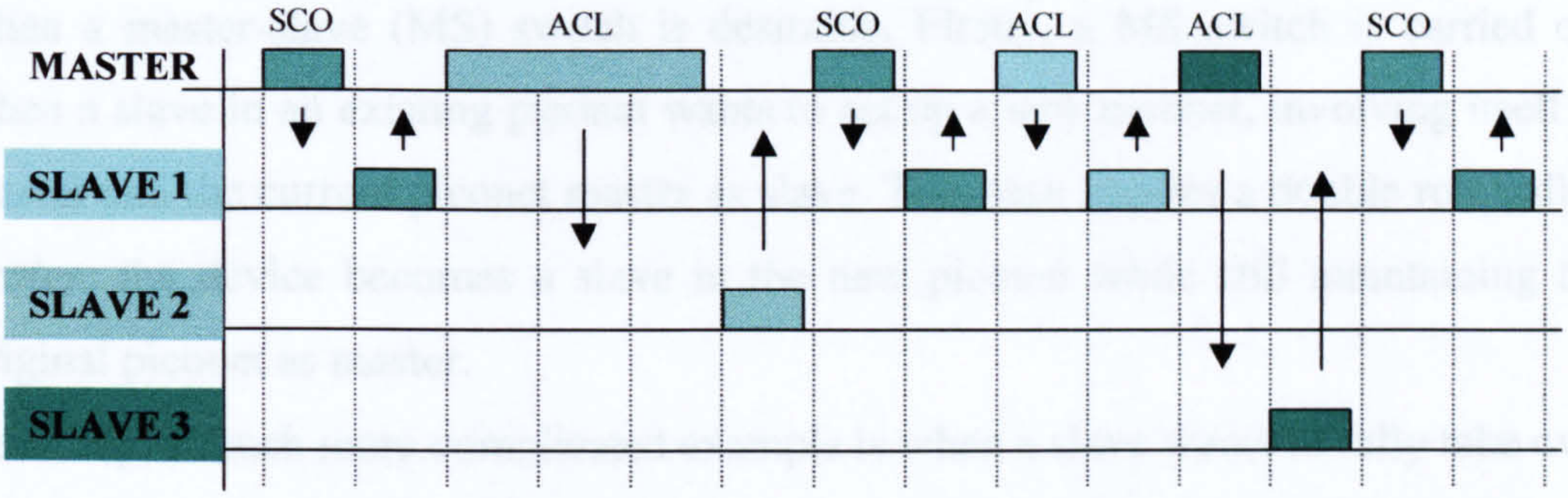


Figure 2.10: SCO and ACL packet transmission between Master and three Slaves.

2.3.4 Connection

The master always has full control over the Piconet. Due to the stringent TDD scheme, slaves can only communicate with the master and not to other slaves. In order to avoid collisions on the ACL link, a slave is only allowed to transmit in the slave-to-master slot when addressed by the AM_ADDR in the packet header in the preceding master-to-slave slot. If the AM_ADDR in the preceding slot does not match, or an AM_ADDR cannot be derived from the preceding slot, the slave is not allowed to transmit. On the SCO links, the polling rule is slightly modified. The slave is allowed to transmit in the slot reserved for his SCO link unless the (valid) AM_ADDR in the preceding slot indicates a different slave. If no valid AM_ADDR can be derived in the preceding slot, the slave is still allowed to transmit in the reserved SCO slot.

But a slave can only communicate with its master in the piconet. There is no slave-to-slave communication as the slave has only the knowledge inside the Piconet of the master device. For that reason, the master controls how the total available bandwidth is divided among the slaves: it decides how often and when to communicate with each slave using a Time Division Multiplexing (TDM) scheme with the round robin policy.

2.3.4.1 Master/Slave Role Switching

Bluetooth supports role switching between master and slave nodes, which means that slave becomes the master and master becomes the slave. There are several occasions when a master-slave (MS) switch is desirable. Firstly, a MS switch is carried out when a slave in an existing piconet wants to set up a new piconet, involving itself as master and the current piconet master as slave. This case implies a double role called *bridge*: the device becomes a slave in the new piconet while still maintaining the original piconet as master.

Secondly, a much more complicated example is when a slave wants to fully take over an existing piconet, i.e., the switch also involves transfer of other slaves of the existing piconet to the new piconet. Clearly, this can be achieved by letting the new master setup a completely new piconet through the conventional paging scheme. However, that would require individual paging of the old slaves, and, thus, take an unnecessarily long time. Instead, letting the new master utilize timing knowledge of the old master is more efficient. As a consequence of the MS switch, the slaves in the piconet have to be transferred to the new piconet, changing their timing and their hopping scheme.

A MS switch carries out an exchange of FHS packets. BD_ADDRs, clock information and AM_ADDRs are exchanged through these FHS packets. The new slave gets the AM_ADDR of the older slave. Moreover, since the piconet parameters are derived from the device address and clock of the master, an MS switch inherently involves a redefinition of the piconet as well as a piconet switch. The new piconet's parameters are derived from the former slave's device address and clock. Finally, for the master and slave involved in the role switch, the MS switch results in a reversal of their transmitting (TX) and receiving (RX) timing: a TDD switch (*SIG [2001]*).

2.4 SUMMARY

This chapter has offered an overview of Bluetooth wireless technology, the brief story of its evolution, the general protocol stack description, with Baseband Layer

backgrounds investigations. And finally, this chapter has a look at packet communication within a piconet to prepare the scatternet formation.

CHAPTER 3

SCATTERNET TOPOLOGY FORMATION

3.1 INTRODUCTION

The problem of organising ad-hoc networks with Bluetooth devices is that even though two nodes may be physically able to receive transmissions from each other, they cannot communicate if they are not in the same piconet at the same time. For small number of devices present in the area, devices create a single piconet but for a significant amount of devices, multiple piconets are created to cover the same area. Therefore to involve connectivity between piconets, a unit called *bridge* or *gateway* must participate in two overlapping piconets. When this occurs a scatternet is created. Though the specification limits the number of slaves in a piconet to seven, the use of scatternet can increase the number of network members and make the coverage area larger. Moreover, if several devices requiring connectivity are not within range of the master, at least one member in their range could interconnect them. An important paper fully exposing the survey of scatternet communication is described by (Whitaker *et al [2004]*).

When a device is present in more than one piconet, it must time-share. The device must spend a few slots on one piconet and a few slots on the other. That means a protocol needs to be established to favor the creation of piconets within the scatternet while minimizing the inter-piconet interference. By grouping nodes accurately into piconet and selecting the proper scatternet topology, there can be a significant impact on the network performance such as the attained throughput or reduced interference.

Knowledge of the scatternet topology that optimises performance would be of prime importance for Bluetooth implementations (Miklos *et al [2000]*). However, the complexity of the problem seems to disallow an analytical approach and the large number of free parameters makes it problematic to exploit numerical optimisation through simulation. Therefore, a presentation of different scatternet topology rules and performances are proposed in *Section 3.2*, intended for the guideline design of an

innovative scatternet topology. *Section 3.3* elaborates a new scatternet formation using a tree topology. A hierarchical approach based on occasional mobility, is addressed. The communication in the proposed scatternet is organised so that the Leader, which is master of one piconet, manages all other piconets. The Leader sends information of its Frequency Hopping Sequence (FHS), and its clock phase as explained in *Section 3.4*, to synchronise all the devices inside the scatternet. Thus the chapter concludes and clarifies that slaves could switch easily from one Piconet to another in one time slot, avoiding guard time and inter-piconet interference.

3.2 SCATTERNET FORMATION REVIEW

Since no scatternet procedure has been defined in the Bluetooth specification, an overview of the different types of scatternet formation follows, as organised by (Persson *[2004]*). Scatternet formations can be differentiated among their coverage and their range area. For instance, in a single hop network, all wireless devices are in radio vicinity of each other, as against the multi hop network, where devices are scattered in an area where some of them cannot directly communicate.

3.2.1 Moving from Piconet to Scatternet

A Simple Piconet Model (SPM), proposed by (Kalia *et al [May 2000]*) is designed to link more than 8 devices, which do not require wireless inter-piconet communication. To support more than seven slaves, the master uses the *park mode* part of the Bluetooth protocol (see *Section 4.3.2.3* for more details of park mode). Figure 3.1 illustrates seven active slaves that participate in communication with the master, while the parked slaves are time stamped and are not part of the communication traffic and enter in sleeping mode. A parked slave with the oldest time stamp is periodically unparked and becomes active within the piconet, while the active slave with oldest time-stamped is parked. This model supported by the Bluetooth specification is very inefficient for large numbers of slaves, and can lead to low throughputs and high delays as the master has to share its bandwidth with several slaves.

Aggarwal *et al. [2000]* introduce a scatternet formation protocol algorithm. Their algorithm first partitions the network into independent piconets, and then elects a Leader aware of all the nodes. However, the resulting network is not a scatternet, because, the piconets are not inter-connected. Thus another phase of re-organisation is required.

To improve the design and to provide a considerable performance improvement, the creation of various piconets is required. To create possible communication between piconets, some nodes are required to act as intermediate devices, participating in multiples piconets and forming a bridge to forward packets, thus allowing piconets to enlarge the network. Hence some topologies use a bridge structure to exchange data between two piconets. The bridge shares its time between the piconets it belongs to, switching from one to another in a cyclical sequence. Data packets with destinations in other piconets are queued by the master piconet, and delivered to the bridge during its residence in the piconet.

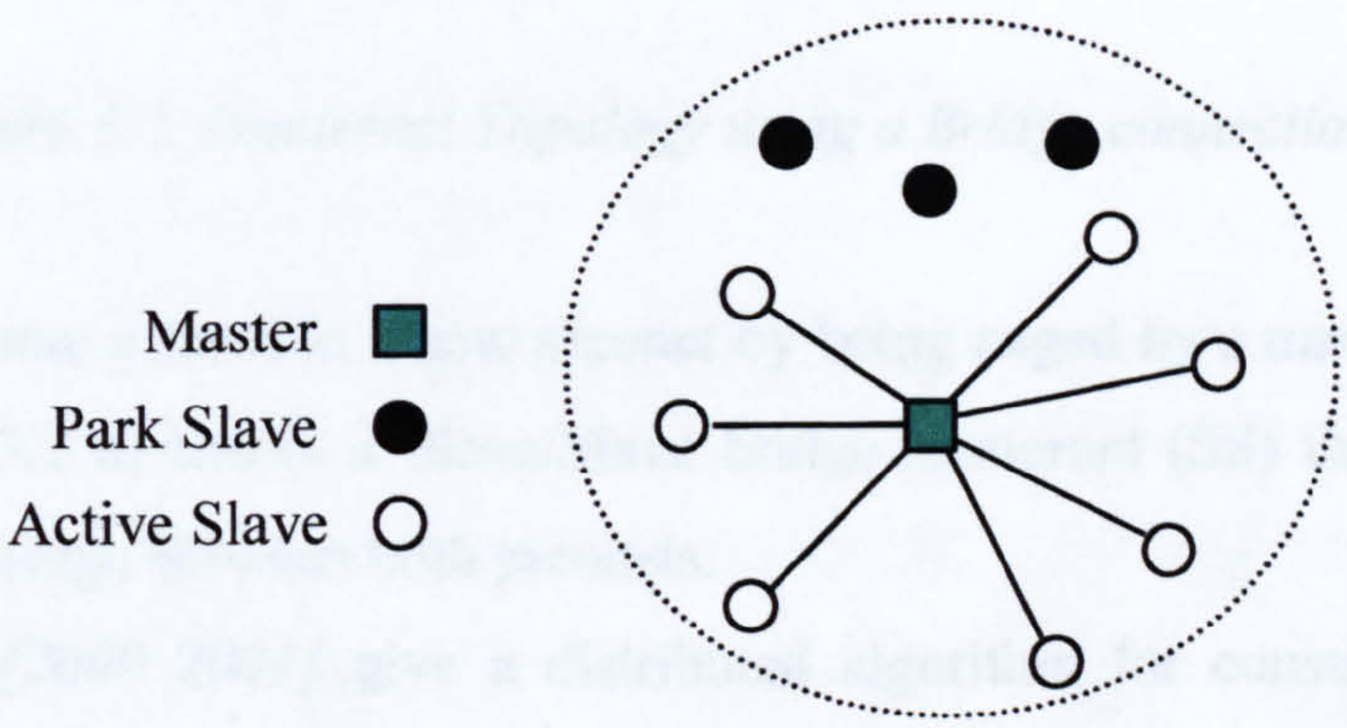


Figure 3.1: Single Piconet Topology using Park Mode to connect more than 8 Devices

3.2.2 Bridge Scatternet Connection

According to the Bluetooth specification v1.1, a Bluetooth unit can act as a slave in two piconets, but only as a master in one piconet. If a device would act as a master in two piconets, these piconets would be synchronized and would use the same hopping

sequence. In other words these two piconets would represent one piconet. The chosen bridge-scheduling algorithm determines the actual duration of bridge residence in each piconet, and the manner in which the switchover times are arranged. Each time the bridge switches from one piconet to another, it has to re-synchronize its clock.

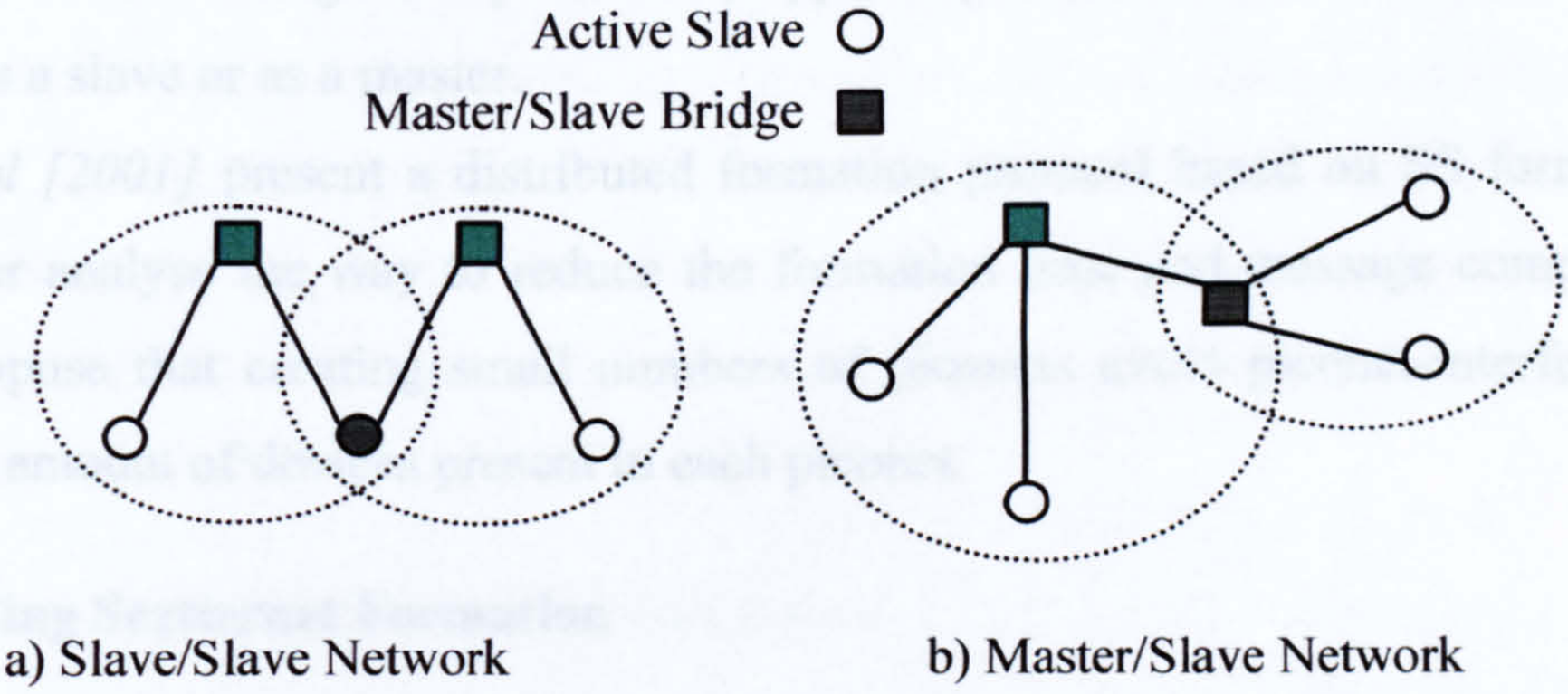


Figure 3.2: Scatternet Topology using a Bridge connection.

A slave can become a slave in a new piconet by being paged by a master of the new piconet. Figure 3.2 a) shows a Slave/Slave bridge scatternet (SS) that uses only a slave to form a bridge between both piconets.

Salonidis *et al [2000 2001]* give a distributed algorithm for constructing a (SS) scatternet structure. They present a distributed Bluetooth Topology Construction Protocol (BTCP), where a leader device assigns roles to the other. Using point-to-point connections a comparison of the vote is made to elect a Leader. The node with the larger vote is elected, and has information about all the other nodes in the network. Then the leader informs other devices on how the scatternet should be formed. However, in Salonidis *et al* concept, all devices are assumed to wake up at the same time to participate in the scatternet formation and are in range of each other. The mathematic co-ordinated algorithm is valid for at most 36 devices.

Haas *et al [2002]* present a Slave/Slave topology called Bluenet. They emphasize that, in comparison to Tree Hierarchy, the average path length is shorter and a

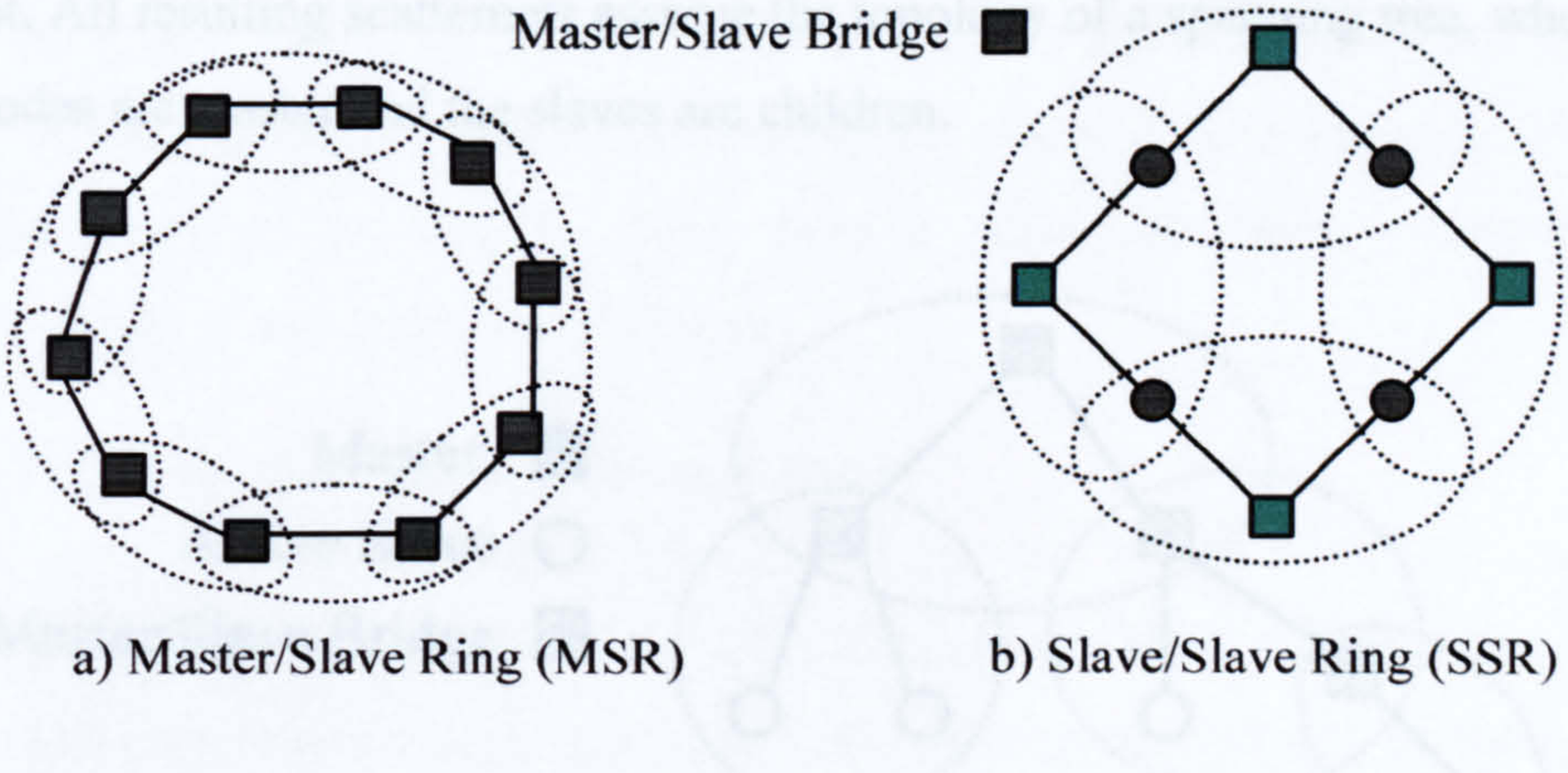
Bluenet can sustain higher traffic flows, since in the heavily connected mesh topology paths are not required to go along the congested and non-optimal paths through the root node. The tradeoffs is that routing in the Bluenet scatternet is much more complex than for Bluetrees. Bluenets are not able to guarantee scatternet wide connectivity.

Figure 3.2 b) shows a Master/Slave bridge (MS) where the master would have a double role: master in one piconet and slave in the other piconet. A MS bridge is similar to the SS Bridge except that any type of piconet interconnection could be formed as a slave or as a master.

Law *et al [2001]* present a distributed formation protocol based on SS formation. The paper analyse the way to reduce the formation time and message complexity. They propose that creating small numbers of piconets avoid piconet-interferences with low amount of devices present in each piconet.

3.2.3 Ring Scatternet Formation

The protocol introduced by Chun-choong *et al [2002]* and Lin *et al [2003$_{(2)}$]* has an entirely different format with a ring structure.



a) Master/Slave Ring (MSR) b) Slave/Slave Ring (SSR)

Figure 3.3: Ring Scatternet Topology.

Figure 3.3 a) and b) illustrate the scatternet ring structure with Slave/Slave Rings (SSR) and Master/Slave Ring (MSR). The main reason behind a ring-structured scatternets is to alleviate the bottleneck problems in TH topologies while maintaining simple scatternet routing. Each node belongs to two piconets. In one, a node acts as a master, in the other as a slave.

They have exactly two links in total to relay packets between each other, and therefore need to be in the same radio range area. In fact a single line consisting of all nodes is closed to form a ring. The ring has the advantage of two paths to any node and has a constant path length. In terms of benefits, this leads to reliability, ease of packet routing and scheduling. The disadvantage of this approach is excessive packet latency for large scatternet formation with long average path (n/2), with n representing the numbers of nodes, and lack of ring maintenance for incremental arrivals and nodes failure.

3.2.4 Tree Hierarchy Scatternet Formation

Figure 3.4, shows a Tree Hierarchy (TH) formation. TH scatternet topologies differ from the preceding models. A single node called a Leader is the root of the tree, and scatternet are built by connecting nodes into a tree structure. The master selects more than one root for tree formation and then the trees are merged that are generated by each root. All resulting scatternets assume the topology of a spanning tree, where the parent nodes are master and the slaves are children.

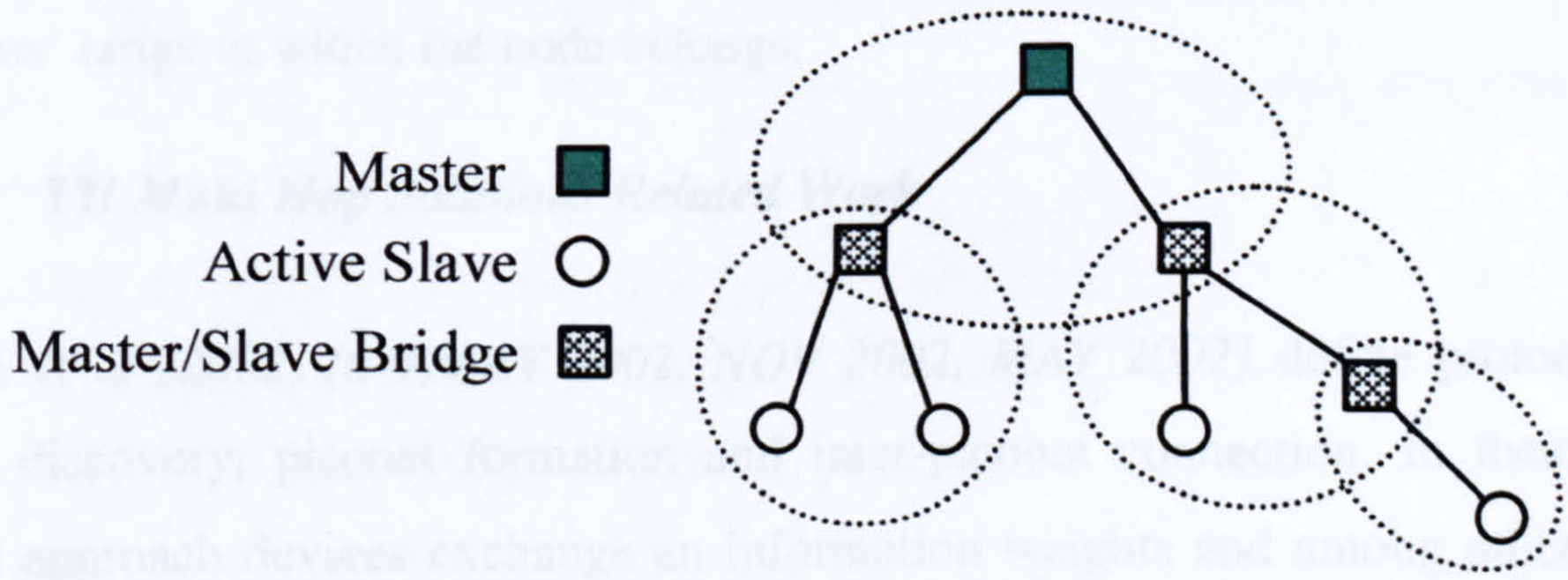


Figure 3.4: Tree Scatternet Topology.

3.2.4.1 TH Single Hop Solution Related work

Zaruba *et al [2001]* proposes protocols that result in a Bluetree topology. First, a node called blueroot starts to page all its neighbours. A paged process is accepted if the device is not part of a scatternet. Once the device becomes a slave, it initiates all the paging of its neighbours and becomes a master in a new piconet. Thus a tree is developed from the root with a parent-child hierarchy. After Bluetree formation, if a master has more than L(≤5) slaves, in order to limit the number of slaves within piconets, at least two slaves (let say S1 and S2) must be directly connected to each other. In this case, S2 becomes the slave of master (S1) and is no more part of the other piconet. Such branch reorganisation is carried out through the network and generates that each master has at most L slaves (where L is 5). The weakness of this algorithm is that both formation and maintenance of scatternet protocol requires communication overhead, which is not measured and presented in the paper.

Tan *et al [2001 2002]* employ a Tree Scatternet Topology. For every new link created one devices is converted into the master role: the root node and the other into a slave: the leaf node. Roots can only contact other roots with one becoming a master. These restrictions prevent loops inside the scatternet. If a connection breaks, the roots become free nodes and attempt to re-connect with other root nodes.

Sun *et al [2002]* proposed a self-routing Bluetree scatternet topology. The BD_ADDR nodes determine the new nodes position in the Bluetree. Once the root finds the correct insertion position, the new node is placed in as a leaf node. Hence a root node receiving a connect request from a node, can easily determine the childrens' range to which the node belongs.

3.2.4.2 TH Multi Hop Solutions Related Work

Petrioli et al *[2002, (6-9)MAY 2002, NOV 2002, MAY 2002]* define protocols for device discovery, piconet formation and inter-piconet connection. In their three-phased approach devices exchange an information weights and among other things the knowledge of their neighbour. Devices with the higher weights become master while the other masters are converted into its slaves and the structure leads to a Bluestar formation. The Bluestar master instructs the slave to page specific

neighbours to facilitate connection with other Bluestar formation where the slaves becomes bridge nodes.

Cuomo *et al [2003]* present the SHAPER algorithm. Their approach employs a TH topology. SHAPER allows both root and non-root nodes of partitioned sub-trees to structure links and initiate tree reconfigurations. The trade-off is that non-root nodes have less time to engage in communication, which reduces the communication efficiency.

Liu *et al [2003]:* propose a solution that builds the scatternet on-demand as element of the route discovery. A route request packet is flooded in the network from the source. Once the destination receives a route request packet, it sends a route reply packet along the reverse path. The route flooding is more difficult in Bluetooth, since a link has to be established before information can be exchanged. The authors note that establishing point-to-point links along all potential paths, in order to broadcast the route request, imposes an excessive amount of overhead. Instead they incorporate the route request packets in the pre-existing inquiry broadcast mechanism. After the destination receives the first route request, the scatternet is formed backwards along the reverse route. To reduce the path latency due to bridge switching overhead, the authors propose that the time slots should be aligned along a route for efficient path traversal. However, when multiple routes are in affect this becomes increasingly difficult. The scheme also requires modification to the Bluetooth specification in order to work.

Stojmenovic *et al [2002]* defines a protocol that limits to 7 the number of slaves per master by applying degree reduction techniques to the network topology graph. The proposed algorithm assumes that each node knows its position and that of its neighbours.

The main advantage of TH topologies over the preceding models is that they simplify scatternet routing and scheduling. On the other hand, if one parent node is lost, the entire family of children and grandchildren connected to the parents are disconnected from the rest of the network. Secondly there is a lack of efficiency in routing because all routing paths have to traverse the tree in the upward and downward directions, which implies a significant traffic for the leader just to forward packets.

3.3 NEW SCATTERNET TOPOLOGY

Following the precedent topology view given above, the solution proposed in this thesis is based on a new scatternet structure, which facilitate the creation of inter-piconet connection within the scatternet. While a slave is exchanging data with its master, other slaves within the piconet does not transmit any data. Therefore these slaves should create new links (piconet) to be connected with more devices, and while the master is not sending data to them they switch to other piconets. This data transfer will increase the overall throughput of the scatternet. However the creation of numerous piconet within a small area will contribute to increase the interference (increase the packet collision) and thus decrease the throughput (Sun *et al [2002]*).

To avoid packet collision, the new topology design could distribute the hopping sequence of adjacent piconet, and apply a different frequency hopping for each slot for all the piconets present within the scatternet. To facilitate the hopping sequence distribution a Tree Hierarchy structure has been chosen.

A designated node (Leader) initialises the protocol and a centralised algorithm is executed to assign roles to the network nodes. A hierarchical approach is developed to coordinate devices and to avoid interference.

3.3.1 New Piconet Creation

Initially all nodes are free nodes when they are turned on. Once two devices get connected, one device acts as a master and the other as a slave. A maximum of eight devices could join the piconet and become slaves as given in the Bluetooth specification. A slave could then try to discover other device present within the area and could create a new piconet. The slave will be assigned the new role of a new master in a new piconet while maintaining its position as a slave in the other piconet with its master becoming Leader.

This creates an innovative hierarchy Leader - Master - and second generation slaves, called (Child)Slave. During the communication with its (Child)Slaves, the new master does not perform another random FHS as defined by the Bluetooth specification; in opposition the new master retains the same phase as its old master (Leader) and apply a derived frequency from the Leader. In fact the new master

forwards the estimated clock of the Leader to be synchronised on time, and operate a frequency unexploited by the leader to avoid frequency interference. Hence, (Child)Slave in order to estimate the master clock, estimate the clock of the Leader by adding an offset to its native clock

Each master child hierarchy forwards its estimated clock (*Section 3.4*) and compute a unique FHS (*Section 3.3.1*) and so on, in a "wave expansion" fashion, till the whole tree is formed.

3.3.2 New Frequency Hopping Sequence for the 1st Tree Hierarchy

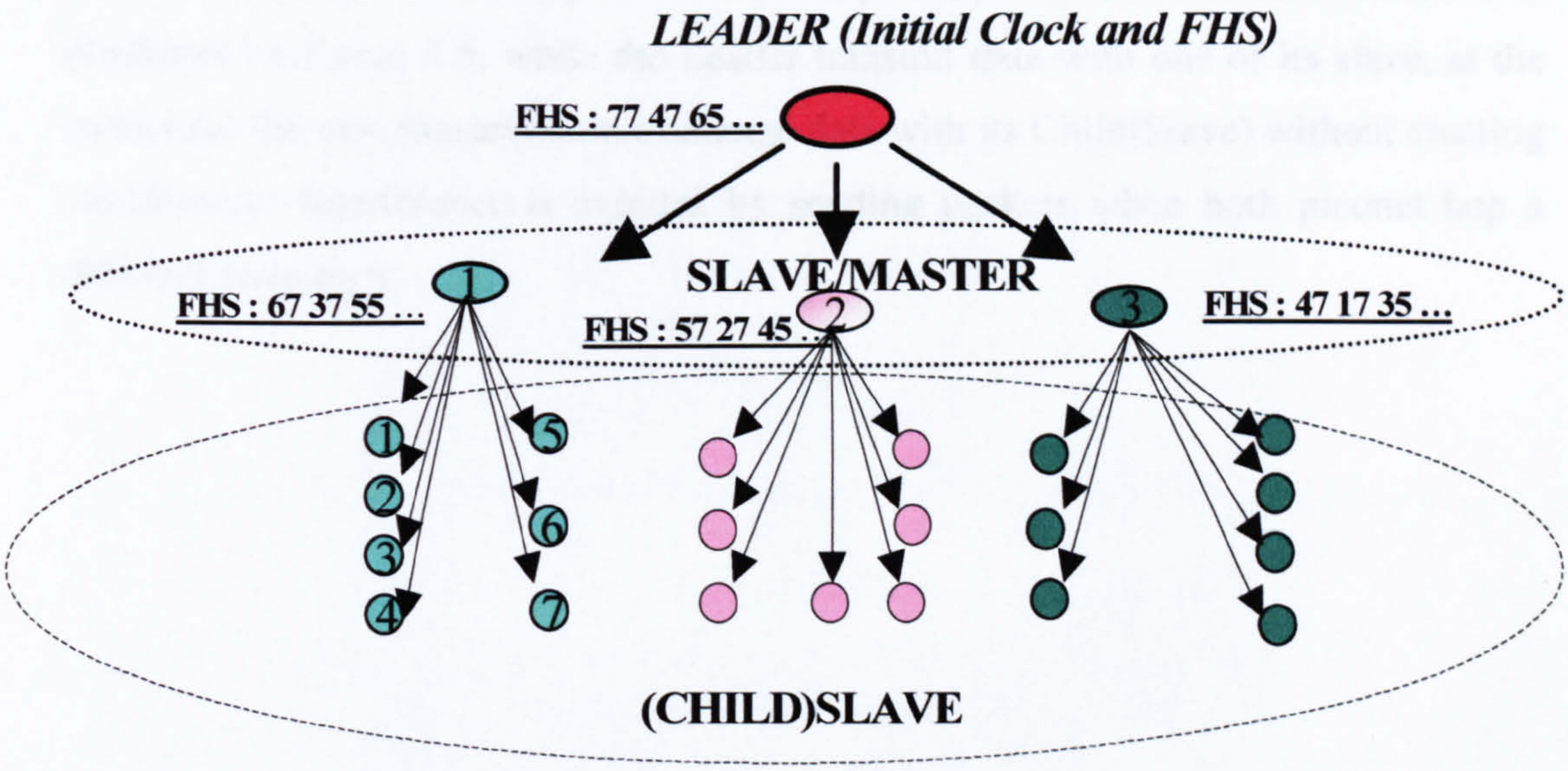


Figure 3.5: Tree hierarchy with FHS synchronisation.

The new hopping frequency sequence, used for new piconet creation, is derived from the current Leader hopping sequence. As shown by figure 3.5, every new hop frequency is decreased by a value, which is 10 × Address of the slave creating a new piconet.

Using standard Bluetooth specifications, with only seven different addresses (called AM_ADDR: Address to distinguish between slave units participating in the Piconet)

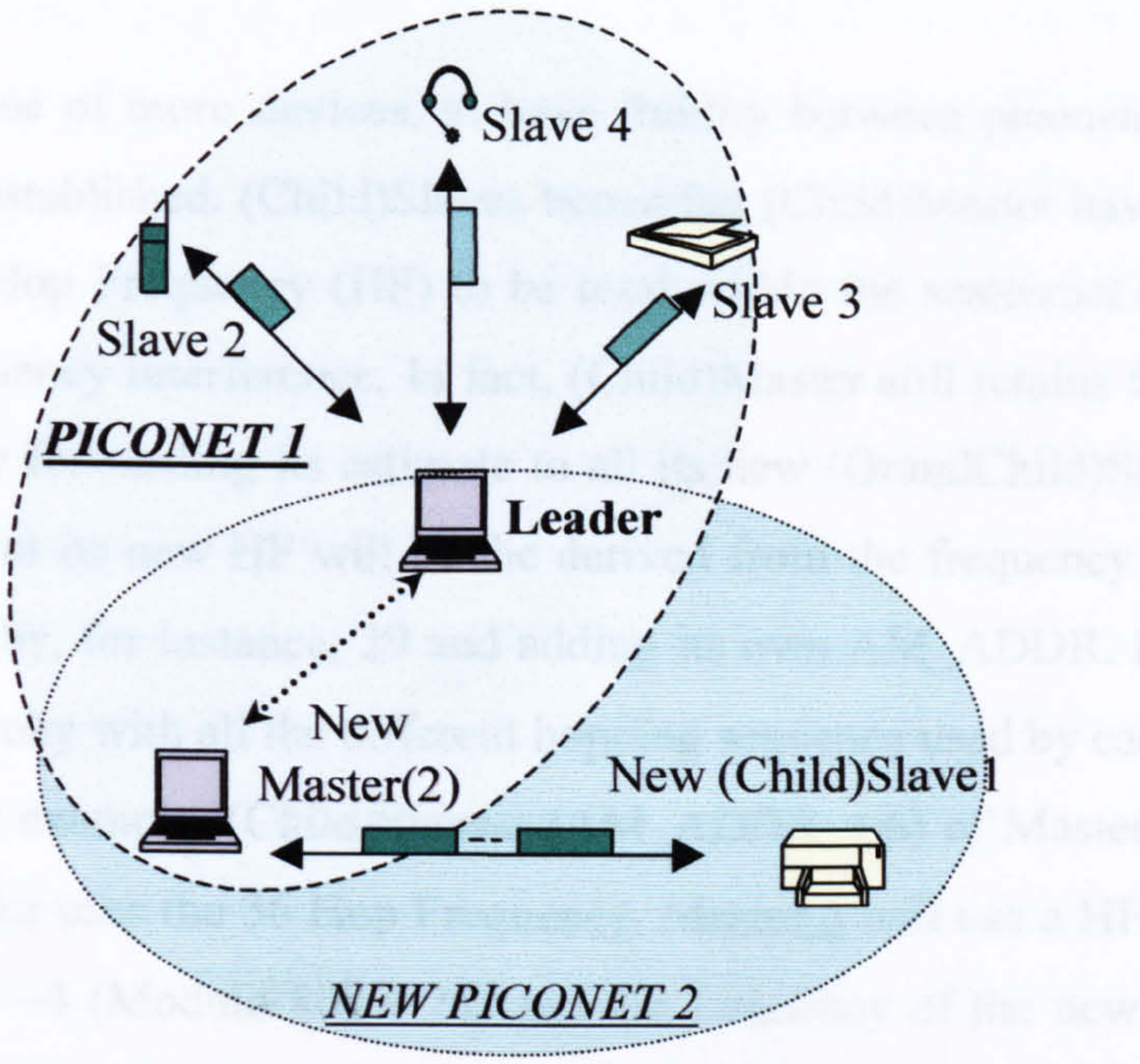
possible inside a piconet and with the communication channel represented by a pseudo random sequence through the 79 (1-MHz) channels; decreasing each FHS by 10 times the AM_ADDR will avoid inter-piconet interference with a minimum of 9MHz difference between piconets.

For example, figure 3.5 shows a predictable hop sequence generated by the Leader for its piconet: (2,402 + k) MHz, with k = 77, 47, 65 ... leading to new piconets hopping sequence depending on the AM_ADDR of slaves:

$Piconet_{(1)}$ (creating by $Slave_{(1)}$): will have a FHS: (77-10*1) and so on such as 67, 37, 55 ...

$Piconet_{(3)}$ ($Slave_{(3)}$): (77-10*3) $\Rightarrow$ 47, 17, 35...

The hop Frequency remains fixed for the duration of the packet; consequently to avoid other interferences only one slot (DH1 packet) is taken into account. Hence as illustrates by figure 3.6, while the Leader transmit data with one of its slave, at the same time the new master could exchange data with its Child(Slave) without creating interference. Interference is avoided by sending packets when both piconet hop a different frequency.



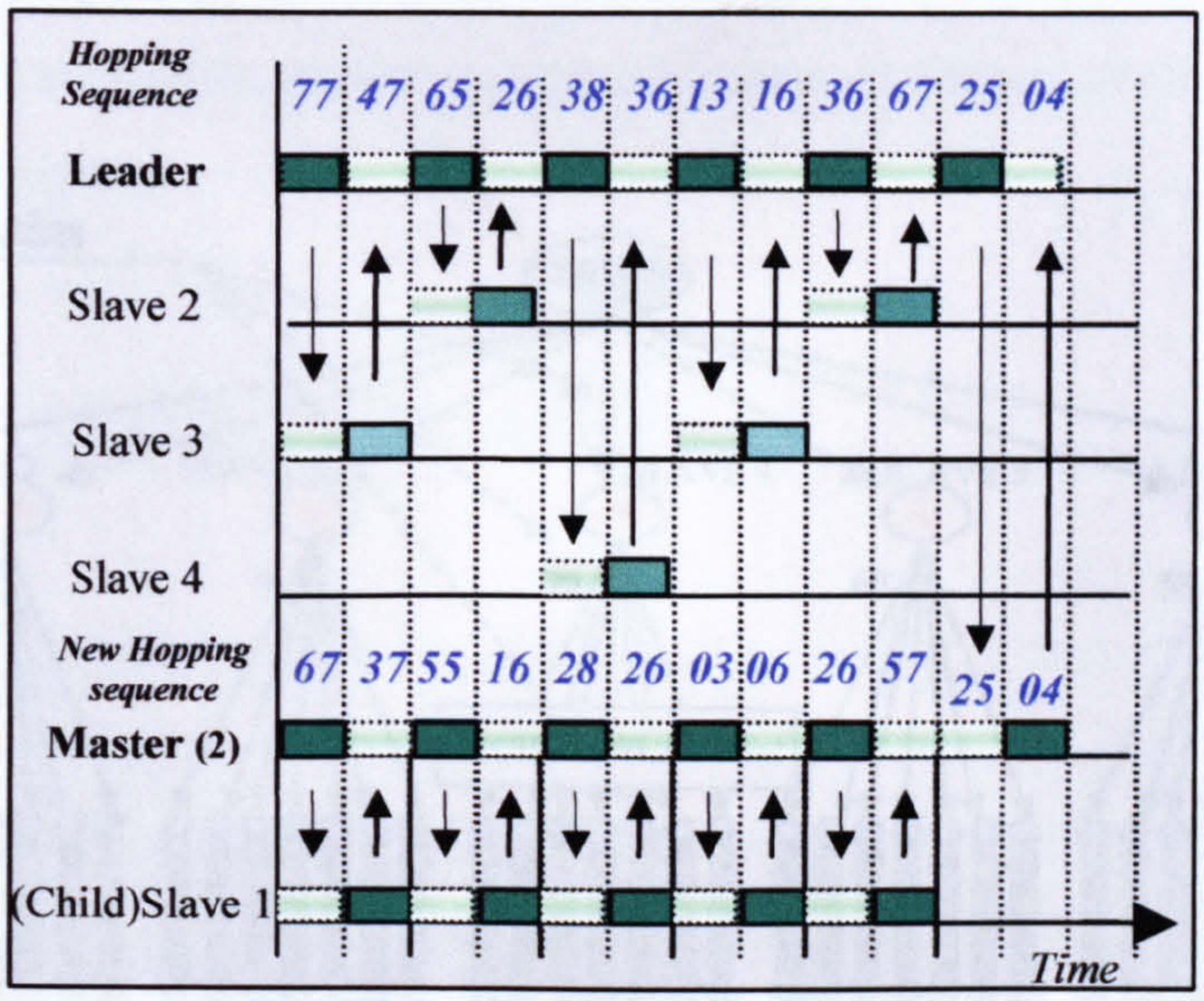


Figure 3.6: Creation of a Scatternet with one slave becoming Master and communicates with a new Slave.

3.3.3 Second hierarchy implementation

In the presence of more devices, to have fluidity between piconets, another master hierarchy is established. (Child)Slaves becoming (Child)Master have to specify their own unique Hop Frequency (HF) to be used within the scatternet and to avoid any hopping frequency interference. In fact, (Child)Master still retains the same clock as the Leader by forwarding its estimate to all its new (GrandChild)Slave, (as done by its master). But its new HF will be the derived from the frequency of its master, by decreasing it by, for instance, 29 and adding its own AM_ADDR. Figure 3.5 shows the tree hierarchy with all the different hopping sequence used by each Piconet at one slot time. For example, (Child)Slave$_{(6)}$ (AM_ADDR = 6) of Master$_{(4)}$ creates a new Piconet. Leader uses the 36 Hop Frequency, Master$_{(4)}$ will use a HF of 75, (which is $36 - 4*10 \Rightarrow -4$ (Modulo 80) = 75) and the frequency of the new (Child)Master$_{(6)}$ becomes $52 = 75 - 29 + 6$.

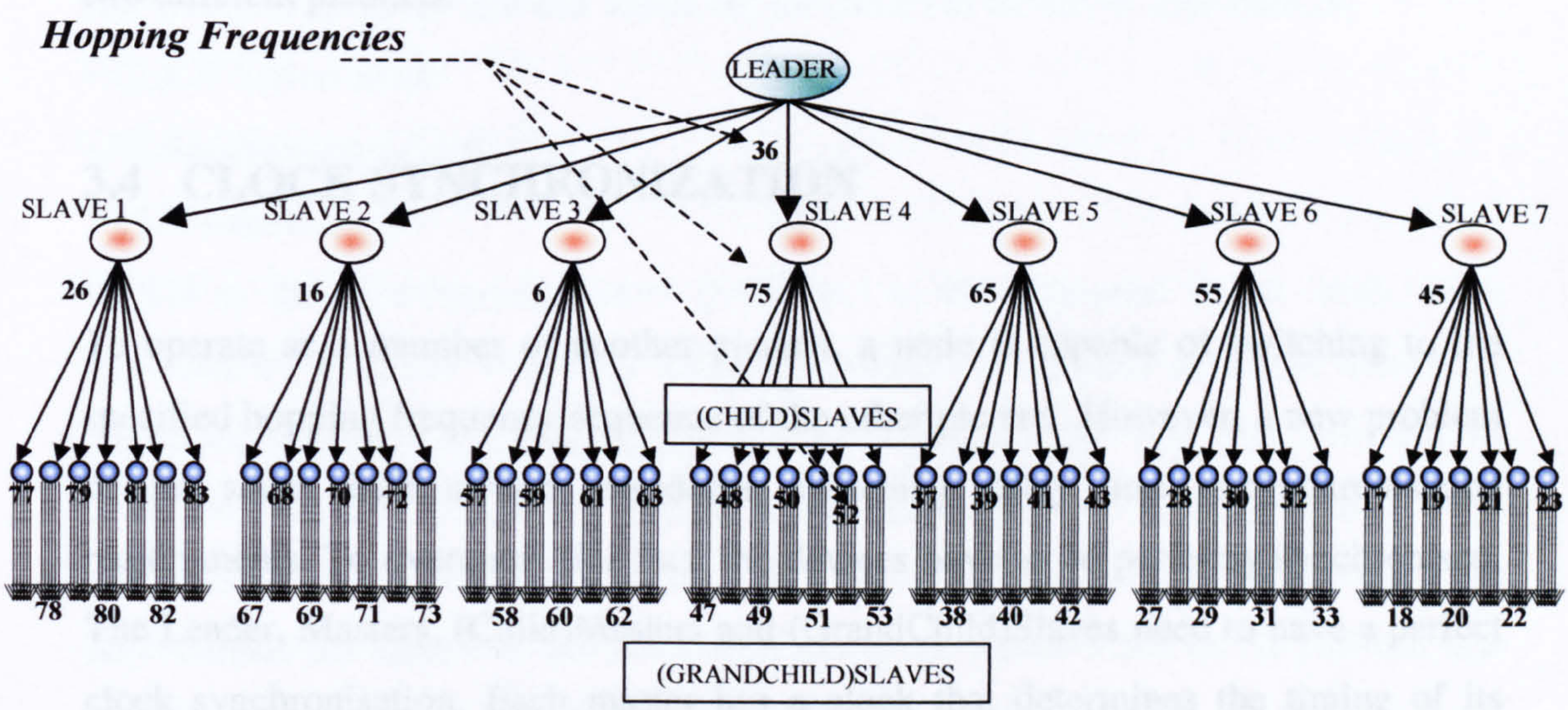


Figure 3.7: Frequency Hopping Synchronisation.

The frequency 52 is then a unique frequency in the scatternet. Hence at the same slot time, Leader communicates to one of its slaves using a hop frequency of 36 (2.438GHz), the Master$_{(4)}$ to one of its (Child)Slaves with HF of 75 (2.477GHz) and

the (Child)Master$_{(6)}$ to one of its (GrandChild)Slaves at HF of 2.454GHz. Since all piconets share the same set of 79 channels, for a small area, the proposed scatternet algorithm could support up to 400 devices: Leader plus seven Masters (= 8) and 7*7 (Child)Masters along with (GrandChild)Slaves give: 8+7*(8*7)=400 devices maximum. There will be a maximum of (8+7*7) 57 piconets sharing the 79 channels without any interference.

As shown in Figure 3.7, the 57 piconets will perform a different hop sequence at every slot leading to non-channel interference even in a small area, which has never been conducted before in any scatternet; this sequence is based on the AM_ADDR, which is known by each device inside its Piconet. (GrandChild)Slave cannot create another Piconet but it can be part of any already existing Piconet see *Section 6.3*. In that case, the (GrandChild)Slave could for example become a slave of the Leader (if the Leader has less than 7 slaves) and still stay (GrandChild)Slave to its (Child)Master by just changing its HF. As a device can switch from one Piconet to another in a slot time with perfect synchronization, all devices will try to be part of two different piconets.

3.4 CLOCK SYNCHRONIZATION

To operate as a member of another piconet, a node is capable of switching to the specified hopping frequency sequence of the other piconet. However, a new problem occurs since each switch introduces switching delay and re-synchronisation requirements. To overcome this fact, the devices have to be perfectly synchronised. The Leader, Masters, (Child)Masters and (GrandChild)Slaves need to have a perfect clock synchronisation. Each master has a clock that determines the timing of its Piconet. As this clock is free running, the clock speed may differ between different masters.

3.4.1 Bluetooth Clock Overview

Every Bluetooth unit has an internal system clock, which determines the timing and hopping of the transceiver. The Bluetooth clock is derived from a free running native clock, which is never adjusted and is never turned off. For synchronization with other units, only offsets are used that, added to the native clock, provide temporary Bluetooth clocks which are mutually synchronized. It should be noted that the Bluetooth clock has no relation to the time of day; it can therefore be initialized at any value. The timing and the frequency hopping on the channel of a piconet is determined by the Bluetooth clock of the master. When the piconet is established, the master clock is communicated to the slaves. Each slave adds an offset to its native clock to be synchronized to the master clock. Since the clocks are free running, the offsets have to be updated regularly. The clock determines critical periods and triggers the events in the Bluetooth receiver. Master-to-slave transmission starts at the even-numbered slots when CLK0 and CLK1 are both zero. In the different modes and states a Bluetooth unit can reside in, the clock has different appearances:

- CLKN-native clock
- CLKE-estimated clock
- CLK-master clock

CLKN is the free-running native clock and is the reference to all other clock appearances. In states with high activity, the reference crystal oscillator drives the native clock with worst-case accuracy of +/-20ppm. CLKE and CLK are derived from the reference CLKN by adding an offset. CLKE is a clock estimate of the native clock of the recipient; i.e. an offset is added to the CLKN of the pager to approximate the CLKN of the recipient. CLK is the master clock of the piconet. It is used for all timing and scheduling activities in the piconet. All Bluetooth devices use the CLK to schedule their transmission and reception. The CLK is derived from the native clock CLKN by adding an offset. The offset is zero for the master since CLK is identical to its own native clock CLKN. Each slave adds an appropriate offset to its CLKN such that the CLK corresponds to the CLKN of the master. Although all CLKNs in the Bluetooth devices run at the same nominal rate, mutual drift causes inaccuracies in CLK (*SIG [2001]*).

3.4.2 Scatternet clock synchronization

The clock offsets in the slaves must be regularly updated such that CLK is approximately CLKN of the Leader. Bluetooth allows a clock drift of max. ± 20 ppm (parts per million) against the ideal timing during activity. The absolute packet transmission timing {Tk} of slot boundary must fulfill the equation [3.1].

$$T_k = \left[\sum_{i=1}^{k} (1+d_i)\mathrm{TN} \right] + j_k + \mathrm{offset} \qquad (3.1)$$

Where T_N is the nominal slot length (625μs), j_k denotes jitter ($|j_k| < 1\mu s$) at slot boundary k, and, d, denotes the drift $|d_k| < 20ppm$ within slot k (*SIG [2001]*). The jitter and drift may vary arbitrarily within the given limits for every slot, while the offset is an arbitrary but fixed constant. The maximum value of the mutual clock drift of Bluetooth devices could be up to 40 ppm (parts per million). From Bluetooth specification, the maximum size of the slave's receive window in active mode is 10 μs, this means that the Leader has to send packets to its slaves at least every 0.25 seconds, in order to keep them synchronised to the piconet. If we assume that Leader sends packet to the Slaves/Masters in the Piconet more frequently than 0.25 seconds, the receive window can be smaller than 10 μs: ($0.25\ s * 40\ ppm = 10\mu s$).

A meeting time is required between the Leader and masters to permit a master to readjust its clock to the Leader. The rendezvous point is a pre-arranged slot agreed between the Leader and the masters. The establishment of rendezvous points allows for the easy development and application of the inter-Piconet scheduling algorithms and is explained in Chapter 5 in more detail. For the meeting time from equation (3.1) the clock drift will increase as the number of hierarchies increase.

When the master creates a new Piconet at this time the clock drift will increase and could be up to 60ppm for its (Child)Slave compare to the Leader's clock. This leads to a meeting point (rendezvous), which will be less than 0.166s where the master needs to exchange clock synchronisation with the (Child)Slave. And for the (GrandChild)Slave to get a perfect clock synchronisation of the scatternet, the

(Child)Master needs to communicate every 0.125s to its (GrandChild)Slave. Figure 3.8 illustrates the clock drift as a function of master hierarchies and numbers of slots.

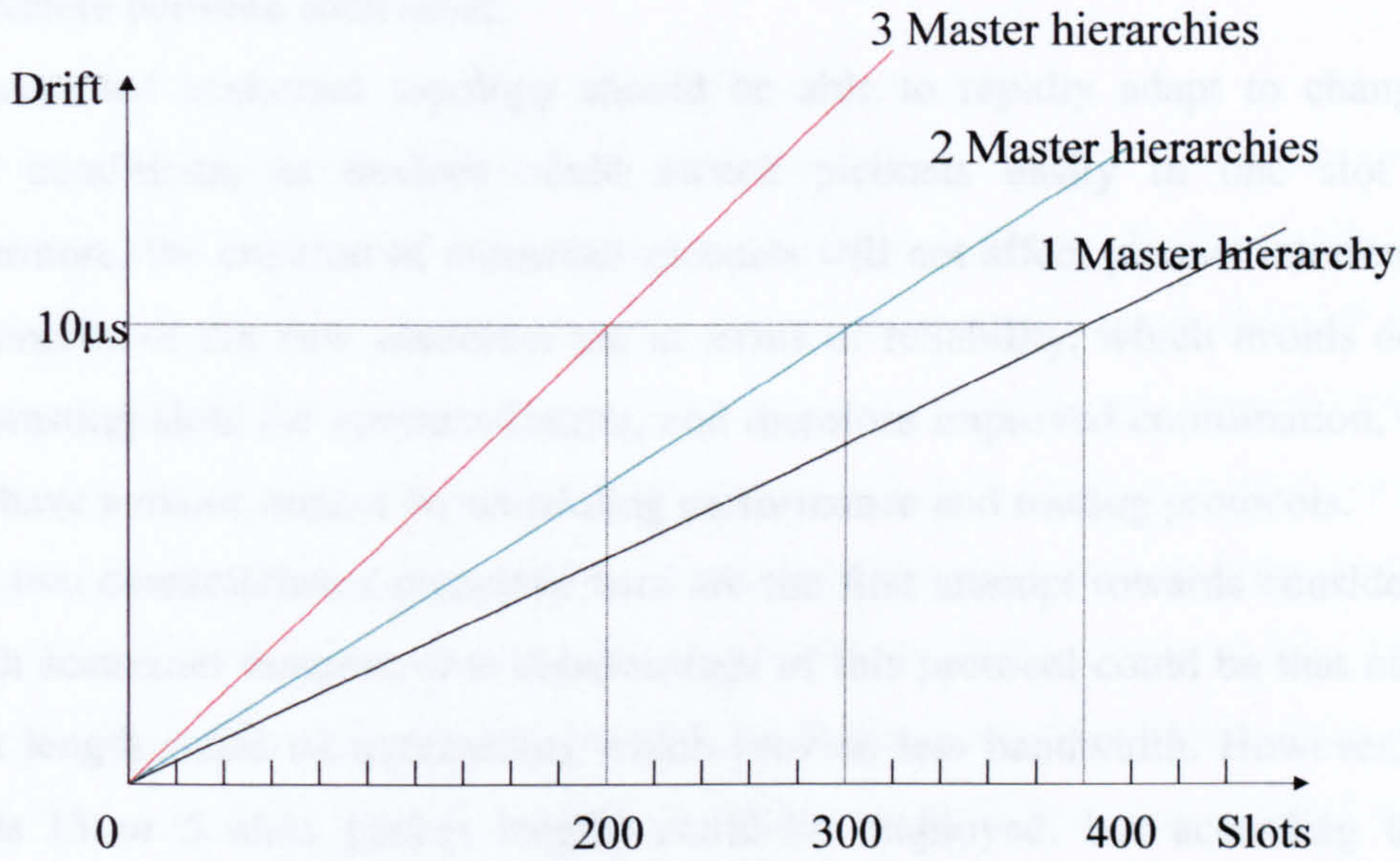


Figure 3.8: Drift time increases in time with master hierarchies, and shows the minimum time in slots for master-slave communication to synchronise.

3.5 CONCLUSION

In this chapter the structure and realization of different scatternet topologies has been investigated; the study shows that for wireless network structure requiring a suitable routing and access delay, the tree hierarchy might be the most appropriate. The new Bluetooth scatternet formation protocol proposed in this thesis is based on a tree structure, with all devices synchronized among one leader. To make this structure effective some of the Bluetooth standard had to be changed. Hence every time a new piconet is created it will exploit the Leader's clock and a derived frequency sequence from the Leader's piconet. All lead to a tree scatternet with three different levels

using a maximum of 400 devices. These modifications have major repercussion on two main characteristics.

First, the fact that piconets are perfectly synchronized avoids the guard time while devices switch from one piconet to another. Second, the fact that all piconets employ a derived frequency hopping sequence from the leader, ensure that the piconets do not interfere between each other.

The presented scatternet topology should be able to rapidly adapt to changes in traffic conditions, as devices could switch piconets easily in one slot time. Furthermore, the creation of numerous piconets will not affect piconet interferences. The benefits of the new scatternet are in terms of reliability, which avoids devices from wasting slots for synchronisation, and therefore improved coordination, which could have a major impact on scheduling performance and routing protocols.

These two characteristics presented here are the first attempt towards consideration of such scatternet features. One disadvantage of this protocol could be that one slot packet length could be transmitting which provide less bandwidth. However, large packets (3 or 5 slots packet length) could be employed, but according to this topology only the same type of packet could to be used within the protocol, which means that for low traffic, 5 or 3 slots packet length could lead to a waste of slot and power consumption once the system has very low rate. On the other hand a high data rate suffer from a higher error probability and may be inappropriate for use due to frequent retransmission.

CHAPTER 4

INTRA PICONET SCHEDULING

4.1 INTRODUCTION

The end-to-end quality-of-services (QoS) delivered to users in Bluetooth networks depend on a large amount of parameters at different levels. In Bluetooth, packet buffering is desirable in numerous protocol stacks, e.g. baseband, L2CAP, IP and TCP. To satisfy the specified delay and throughput constraints it is essential that each protocol has sufficient buffer space and power since power in many "skinny nodes" is limited. Users in corresponding applications, such as video, or data services, in which Bluetooth devices are deployed, demand these QoS constraints. To deal with all these restrictions, Bluetooth uses a Round Robin scheme, which poorly satisfies the QoS, and is the heart of a number of considerations (Chen *et al [2002*], Misic *et al [2004]*).

This chapter is organized as follow. In *Section 4.2*, the Round Robin (RR) scheme is explained in more detail, followed by different studies carry out on new RR design to improve some QoS features within intra-Piconet network. The different powers consumptions within Master-Slave communication are presented in *Section 4.3*. The new RR scheme call AMSQP intra-piconet schedule is explained in *Section 4.4* in combine with simple scenarios. A simulation model is performed with numerous performance analyses over the QoS, such as traffic throughput, delay and power consumption are presented in *Section 4.5*. The last sections are devoted to some open issues and conclusions.

4.2 INTRA-PICONET OVERVIEW

Devices in a Bluetooth piconet communicate using a centralised polling scheme organized by the master in a time-slotted system. A slave is allowed to transmit only

if the master in the preceding time slot has polled it. As a result, the master not only resolves contention but also allocates bandwidth among slaves. The total capacity for communication for a Bluetooth piconet using a single slot transmission is about 172Kbit/s symmetric (Table 2.1). Thus, it is highly undesirable to allocate slots for polling a slave connection only to have nothing to transmit in one or both slots, i.e. one or both slots are wasted.

4.2.1 Round Robin Scheme

The major advantages of the Master/Slave model in Bluetooth are simplicity and low power. The Bluetooth specification suggests a Round Robin (RR) policy to schedule the Master-Slave polling algorithm. Such a basic scheduling policy, equally divides the total number of polls among slaves active in the piconet. A fixed cycle order is defined without taking into account the different requirements of each slave.

A single chance to transmit is given to each Master-Slave queue pair according to the cyclic order. Considering a piconet that consists of a master and seven slaves, each slave gets a seventh of the total number of available polls. As a consequence, some slaves who do not have to send data are polled much more than is necessary, while high-traffic slaves may be polled less than is required. Bandwidth not used by a lightly loaded slave is lost and cannot be used by other slaves. Only if the overall traffic is low, or if the flow rates are similar, this kind of polling scheme can handle traffic well. But, in any other situation, the RR policy is very inefficient.

4.2.2 Related Work on Intra-Piconet Schedule

A number of researchers have addressed the issue of intra-piconet scheduling in Bluetooth, (Rao *et al [2001]*, Lin *et al [2003$_{(1)}$]*, Yang *et al [2004]*, Miorandi *et al [2004]*, Yaiz *et al [2003]*). The fundamental question is the polling discipline used by the piconet master to poll its slaves (harmonize the intra-piconet scheduling schemes). Studies put forward two categories: The master has complete knowledge

of the slave queue status while other studies do not make such assumption and try to estimate the traffic.

Kalia *et al [99, 2000]* were among the first to analyse and compare the behaviour of two new scheduling scheme in the medium access control (MAC) and the Segmentation And Reassembly (SAR) layer. They propose the Priority Polling (PP) and the K-Fairness Policy (K-FP) instead of the Round Robin. Their algorithms take into account the fact that the master has complete and updated knowledge of the queue status of each slave and vice versa. They utilise the queue status information of backlog, traffic arrival rate and the packet size at each master-slave pair to decide on the next slave to be polled. They give priority (PP) to the slaves with two-way traffic over the one-way traffic to improve throughput by avoiding slot wastage, while also guaranteeing a minimum service. K-FP ensures max-min fairness by implementing thresholds and limiting the maximum slots sacrificed by the one-way-traffic slaves to the two-way flows. They also suggest Segmentation and Reassembly techniques to improve performance.

Johansson *et al [2000,2001]* almost at the same time proposed a number of different algorithms: Round Robin (RR), Exhaustive Round Robin (ERR), Priority Round Robin (PRR) and Fair Exhaustive Polling (FEP).

ERR: the scheme is exhaustive and the server does not switch to the next pair (round robin list) until both the master and the slave queues are empty.

Priority Round Robin (PRR) and a modified exhaustive scheduling algorithm named Fair Exhaustive Polling (FEP): Slaves are divided into two groups: a group of active nodes and a group of inactive nodes. Slaves within the group of active nodes are polled in a Round Robin manner. The number of successive useless polls or the average success rate of polls can be used as a measure of activity of a slave. If a slave is found to be inactive, it will no longer be in the active group and will be placed in the inactive group instead. An interval *Tpoll* is defined for each slave and used by the FEP to poll an inactive slave regularly to check whether it has become active or not.

Capone *et al [2001]* proposed a scheme where the master keeps polling the same slave until both the master and slave queues are empty. The next slave to be chosen is the one for which the sum of the master and slave queue lengths is the largest. They propose a practical polling scheme, called Limited and Weighted Round Robin (LWRR). This scheme reduces the rate of visits to queues that have been found empty in previous visits. LWRR gives a weight equal to Max_Priority (MP) to each slave at the beginning. Each time a slave is polled and no data is exchanged between the master and the slave, the weight of the slave is reduced by 1. The lowest value of the weight of a slave is 1, in which case the slave has to wait a maximum of MP-1 cycles to get a chance to be polled.

Das *et al [2001]* study is in a similar line and proposes the schedulers Adaptive Flow-based Polling (AFP), sticky and sticky-AFP to improve the performance of asynchronous data traffic by again using the uplink-downlink queue status. AFP is simply a priority poller based on the traffic flow. It uses the traffic rate information of each slave and polls slaves with less activity more infrequently. Sticky, reduces the mean queue occupancy by polling slaves exhaustively until they have data to send.

Gerla *et al [2002]*, the polling algorithm is based on the master estimating the traffic rate between each slave and itself. It updates the amount of data exchange during the poll phase and gives an estimate of a slave's queue length and of the rate at which traffic is generated. The master polls the slave which has the values above a threshold.

Fabian *[1993]* considered a subset of polling schemes with a fairness constraint: each queue must be visited once per cycle but the visit order can be dynamically changed. The polling policy with symmetric systems is to enforce a decreasing queue length order at the beginning of each cycle.

Yaiz *et al [2001]* proposed the Predictive Fair Poller (PFP), which performs well in the best effort case as well as supporting QoS traffic. To ensure efficiency and

fairness while providing QoS, PFP uses the traffic demand promised to each slave and the previous poll result to select the next slave to be polled.

M. Shreedhar *et al [1996]* proposed The Deficit Round Robin (DRR) algorithm which behaves almost like RR, but gives higher priority to active slaves with backlogged data (in downstream traffic as well as in upstream traffic) and therefore improves the behavior of the RR algorithm. For example, with only upstream traffic, the DRR algorithm polls each slave until it has no more data, which corresponds to reception of a NULL packet from the slave. In this case, DRR stops polling a slave as soon as this slave sends back a NULL packet. Compared to RR, the improvements of the DRR are that it takes into account downstream as well as upstream traffic and that it can handle asymmetric flow rates among the slaves more effectively.

4.2.3 Related work Summary

In the precedent section different intra-piconets scheduling has been exposed for only single piconet configuration. The master uses a polling cycle to determine the order in which the slaves are polled. In the design of an intra-piconet in Bluetooth scheme, all review point out that the RR scheme perform poorly in term of fairness. Thus most innovative scheme provides preference to slaves based on their traffic activity in order to adapt the traffic more efficiently.

4.3 POWER CONSUMPTION IN POLLING SCHEME

Another important factor on polling schemes is the energy conservation, as proposed by (Zhu *et al [2004]*, Perillo *et al [2003]*, Chakraborty *et al [2000][2001]*). Most Bluetooth devices have limited energy for computing and communication because of the limited battery lifetimes, and Bluetooth has been chosen to have a very low consumption. Conserving battery power in Bluetooth devices is an important consideration in designing protocols for networks that include a lot of devices. The constraints of scheduling a device in low power mode are that the end-to-end delays

may increase and thereby violate the Quality of Service (QoS) parameters. If the packet delay is large, there may be re-transmissions due to a higher-level protocol (such as a timeout in TCP) that leads to a further wastage of power. An understanding of the TDD power consumption in Bluetooth is presented.

4.3.1 TDD Power Consumption

To develop a new power saving technique, it is important to understand the energy consumption within a Bluetooth network. Communication between devices could use 14 defined packet types, which encompass 1, 3 or 5 time slots. The Bluetooth specification is defined such that a minimum of 234.5 μs of time is required between subsequent packets including a ± 10 μs uncertain window for reception (clock drift). In the new scatternet topology defined in *Section 2.2*, only DH1 packet are taken into account, however within a time slot, differing levels of current consumption activity may occur. First, the hopping mechanism is rather robust to save power, in that master and slave remain synchronized even if no packet is exchanged over the channel for several hundreds of milliseconds (i.e. the hopping channel is virtually present even when no transmissions are taken). Therefore no dummy data has to be exchanged to keep synchronization between master and slaves. Hence, at the beginning of each time slot, the power consumption of the microprocessor supplies the synthesizer for the Bluetooth radio. It has to be able to provide frequencies from 2402 MHz to 2480 MHz in steps of 1 MHz with frequency tolerance of ± 25 KHz. Later, after the synthesizer has completed tuning, the Bluetooth unit, using the access code correlator, determines the start of the Received (Rx) Packet. A receiver can decide quickly whether a packet is present or not. The receiver correlates the incoming signal in a sliding correlator, which is matched to the access code. Since the access code lasts for a little more than 80 μs, after a scan duration of about 220 μs (to compensate for some timing jitter and drift) the receiver can decide to continue to listen or return to sleep as shown by Figure 4.1 (*Silicon Waves [2004]*).

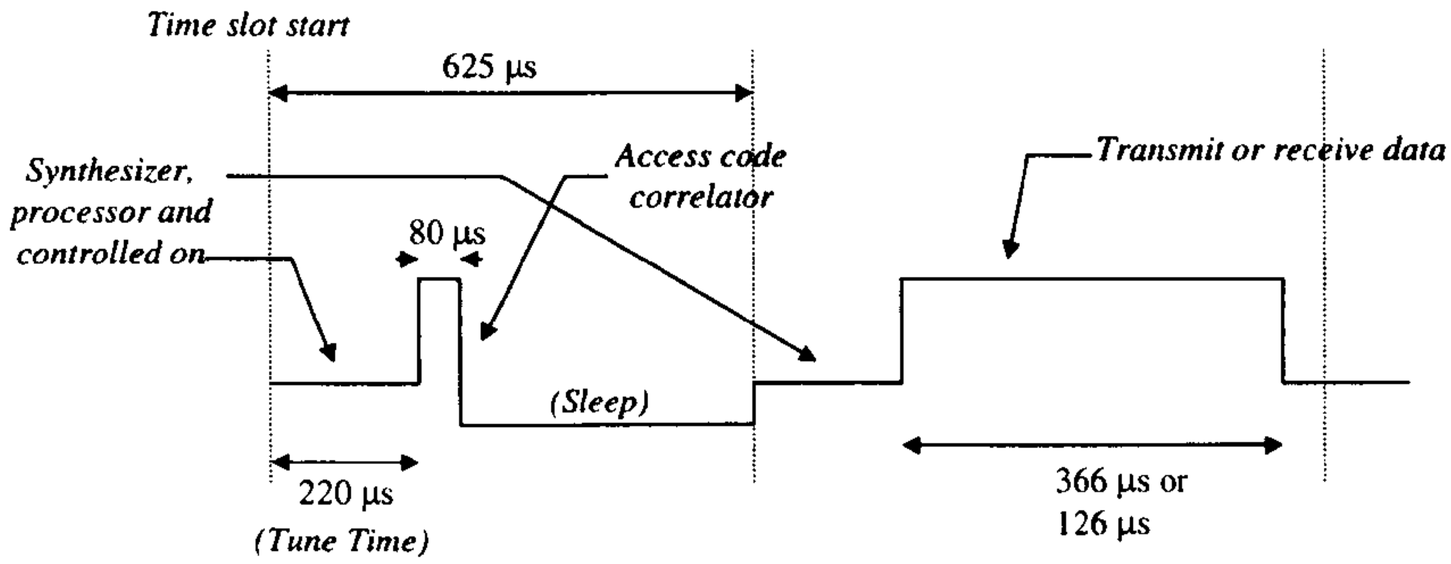


Figure 4.1: Different power phases, for slaves listening packets from the Master.

If the access code is not received (i.e. the correlator does not trigger) within the scan window, the Bluetooth recipient can then sleep for the rest of the receive slot. The packet type indicates the packet length. Thus a slave not addressed in the first slot can go to sleep for the remaining slots that the packet may occupy. This can be read from the TYPE code see Table 4.1 (SIG *[2001]*).

Packet Type	ID	POLL	NULL	FHS	DH1
Length	68µs	126µs	126µs	366µs	366µs

Table 4.1: Types of Packet and their length in the Bluetooth header.

Finally, if the proper access code is received, the receiver will continue to demodulate the packet header. The packet received or transmitted, depends on the payload length. In order to minimize the power consumption, packet handling is minimized both at the TX and RX sides. At the TX side, power is minimized by only sending useful data. This means that if only link control information needs to be exchanged, POLL packets will be used.

From (*Silicon Wave* [2004]), the Power Consumption Estimates for Bluetooth in the active mode depends on the average current consumption. Roughly, four different

currents are provided for a 1-slot period. One for the slave in a sleep mode which is approximately 10 μA, the second one could be measured during the tuning time used for the synthesiser to change the hop selection of the order of 10mA. The third one, receiving a packet is around 40mA, and the fourth one is in the region of 50mA to transmit data; all depend of course on the Bluetooth devices characteristics.

But, since a slave is not aware of the presence of other slaves within the piconet, it will keep listening for each odd slot packet sent (or not) by the master, expecting to receive the packet. By just listening to packets without receiving any data, this method consumes significant power when compared to a sleeping mode with very low consumption; i.e. 10mA vs. 10μA. Therefore the polling based Round Robin scheduling has a significant drawback when considering power consumption.

4.3.2 Low Power Mode

To avoid this important lose of power; Bluetooth standards have mechanisms in place to improve efficiency usage of the available Bandwidth. First, an inactive slave can enter a dormant state for reduced power consumption. Or, a device can participate in two or more piconets and even change its status when moving from one piconet to another. Bluetooth supports three such states: sniff, hold and park mode. A brief introduction to these modes is given below.

4.3.2.1 Sniff mode

A sniff mode has been defined such that the slave does not listen to the master every even slot, but can skip some slots just to save power or to become part of a new piconet. The slave, in the *Sniff* mode, can only receive data from the master at specified time slots called the sniff interval *Tsniff*, which are regularly spaced. As illustrated by Figure 4.2, the slave starts listening at the offset sniff slots for a number of *Nsniff* attempt; the two consecutives receive slots unless a packet with matching AM_ADDR is received (Haartsen *et al [Oct 2000]*).

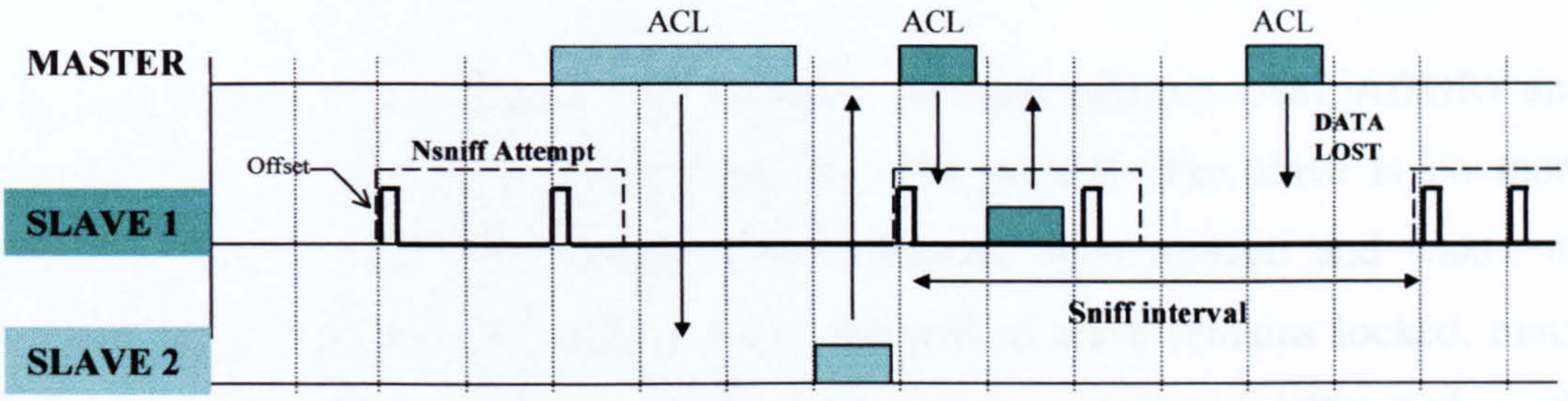


Figure 4.2: Sniff Mode Overview

After reception of a packet with matching AM_ADDR, the slave continues listening at the subsequent Nsniff timeout. If the master sends data between two sniff attempts, the data is lost. A slave can enter the sniff mode upon request made by it or its master after the *sniff* mode parameters are negotiated or the master can force it into sniff mode.

4.3.2.2 Hold mode

If no communication is expected for some time with the master, the slave can be put in *hold* mode. Similar to the sniff mode, the slave does not listen to the master channel for a hold time but is not periodic. Both master and slave agree for the time and duration that the slave stays in hold mode initialized with the *holdTO* value. While in the hold mode, the slave keeps its active member address (AM_ADDR) such that when the timer is expired, the slave will wake up, and could synchronize to the traffic on the channel and will wait for further master instructions. Using the hold mode, the slave can be free to do other things like scanning, paging, inquiring, or attending another piconet. A unit in hold mode can also enter a low-power sleep mode (Maric *[2000]*).

4.3.2.3 Park mode

In the park mode a slave gives up its active member address (AM_ADDR) and receives a new 8 bits park member address: PM_ADDR. The slave is no more participating in the packet exchange, but is staying synchronized and wakes up occasionally to listen to the channel. Since the parked slave remains locked, much lower power consumption can be obtained than in a regular mode. The park mode could be employed when more than seven slaves are present in a single piconet (see Figure 3.1 for example).

4.4 NEW INTRA-PICONET SCHEDULING

Like other intra-piconet environment schedules presented by (Capone *et al [2001]* and Yaiz *et al [2001]*), some slaves may switch to hold, sniff or park mode to save power for a certain number of time-slots, but when these slaves switch back to active mode, many packets may have been accumulated, which need to be sent to the master. Therefore the slave in hold, sniff or park mode, may stay for a longer time in active mode to transfer the data and wait for the link manager to recalculate when the Baseband hardware can be put to sleep during periods of inactivity. It must agree a new time and send a timed sleep request to the Baseband manager to put the Baseband hardware into sleep confirmation. To avoid this time-consuming decision, and mainly to avoid the fact that slaves present in the piconets maintain listening data traffic from the master; a new intra-piconet scheme is presented using a simple table defined by only two bits agreement, whether the slave stays in active mode or goes into sleeping mode.

4.4.1 Adaptive Master Slave Queue Polling Overview

In Bluetooth, a slave has to be polled by a master to be able to transmit. As a result, if the master allocates time-slots according to its own queue size or current load, it

could neglect the uplink traffic. This would result in too many packets being accumulated in the slaves and, therefore, the system performance will be degraded.

To maximize the throughput, the master should poll a slave only if data queued is present at the slave bucker: downlink, or at the master side: uplink, at the same time as other slaves are asleep and wake up only when packets are sent for them, to minimize energy consumption.

This work introduces a new polling algorithm called Adaptive Master Slave Queue Polling (AMSQP). This scheme allows both master and slave to decide the numbers of slots that a slave has to remain "sleepy" within the piconet. It is similar to a hold time calculation between master and every slave with an adaptive calculation of the hold time (or poll time). AMSQP exploits the actual node queue knowledge and uses a simple derived Round Robin polling scheme to control intra-piconet traffic. The new AMSQP polling scheme proposes a flexible intra-piconet scheduling mechanism, which involves different variable-rate polling:

- Prediction of the uplink queue status, the scheduler has to make a knowledgeable prediction of the uplink queue status based on the previous poll results and on the traffic flow information available to it.
- The master-slave connection with little (but not dormant) activity must be polled less frequently. To ensure higher throughput connections with high traffic, the master should give priority by polling the slave more frequently.

In the new scheme, every time a slave enters or leaves the piconet, the master should inform the entire piconet, such that all slaves are aware of the number of devices present in the piconet. In addition, it is assumed that the intra-piconet scheduling is performed using the Round Robin policy, where the master polls each slave with a single packet of one slot burst, and then moves on to the other slave (next slave on the round robin list). Meanwhile, by constantly using two slots in Master-Slave communication, it becomes simple to predict the next polling time. The master communicates to one slave and then goes directly to the next one for no more than two slots (a frame). Bearing in mind that a piconet with seven slaves has a meta-

frame of 2x7=14 time-slots. Therefore the next Master-Slave polling could be predicted and would appear exactly on the 13th slot later on. Until the end of this interval, the slave could be asleep or present in another piconet. As implied in sniff or hold mode, the slave will remain synchronised to the piconet for the sleep mode duration.

Nevertheless within a Bluetooth piconet, the ad-hoc network may exhibit traffic with significant variability. Therefore, to adapt intra-piconet traffic, the new scheduling relies on two bits of information shared between slaves and master. They make a decision of when the next predictable link is going to get polled. The two bits need to be sent in every packet type which will need to be placed in the payload header where free bits are available (see Figure 7.6 for detail).

The two bits of information distinguish four states of the queue, one for the master-slave pair and one for the availability of the next coming slot.

The first bit send from the master to the slave represents the availability of the next coming pair of slot, explicitly the presence or not of the next slave on the round robin list. This may be explained as:

- 1 denotes the state that the next slave, from the round robin list, does not attend the next polling scheme, i.e. the next slave is in a long sleeping time mode, and its two allocated slots are not used and could be utilised by the current slave.
- 0 denotes the non-availability of the two coming slots, i.e. the next slave is present to transfer data with the master.

The second bit information distinguishes whether the master or the slave has data to send or not.

- 1 denotes the state when a master, or a slave, has data in its queue.
- 0 denotes an empty bucket queue.

The two bits information sent from the master to the slaves sets apart four combinations, which are summarized in Table 4.2.

2 bit combination from Master to Slave	*Next Coming Slot*	*Master Queue*
0 - 0	NOT AVAILABLE	NO DATA
0 - 1	NOT AVAILABLE	HAS DATA
1 - 0	AVAILABLE	NO DATA
1 - 1	AVAILABLE	HAS DATA

Table 4.2: Two Bits Combination to aware slaves of master intension on next meeting time.

The master, on the downlink transmission, communicates these 2 bits of information to the slave. By receiving the 2 bits of data, the slave is aware of the availability or not of the next slot (one slot later), and of the queue status of the downlink buffer. Therefore, depending on the 2 bits, the slave modifies the combination on the uplink transmission. It informs the master of its absence (or not) for the next frame with the first bit set to 1 (or 0) and signals its queue status of the uplink buffer via the second bit. The combination could be modified by the slave with the first bit changing from 1 to 0 in case the slave cannot be present on the next slot; the second bit from 0 to 1 in case the slave has data in its queue. With the knowledge of this permutation, both master and slave can determine when the slave should be polled next. The four states are summarized in Table 4.3 and depend on the number of slaves present in the active piconet.

The purpose of the new polling schedule is to predict the next polling time at every end pair polling, to facilitate slaves entering in low power mode for a short period of time. That means slaves are listening to the master only at the polling interval predicted on the last meeting. The slaves will be in an active mode only when they transmits data and so the slaves avoid unnecessarily changing hop frequency to match the access code for every even slot, as is done by the round robin schedule. Hence power conservation is achieved by reducing the number of unnecessary polling slots and unnecessary active periods.

4.4.2 New Intra-Piconet Demonstration

For the purpose of the system demonstration, Figure 4.3 shows a piconet with seven slaves. All slaves are in active mode and slaves have data for the master. Under the Round Robin policy, the master's schedule sequence is $Slave_{(1)}$, then $Slave_{(2)}$, $Slave_{(3)}$, $Slave_{(4)}$, $Slave_{(5)}$, $Slave_{(6)}$ until $Slave_{(7)}$, and then the schedule returns to $Slave_{(1)}$ after the 14^{th} slot sequence (7 frames of two slots needed to end a cycle). Slot 1 defines the slot that the master uses to transmit data to $Slave_{(1)}$, such that the master starts to communicate with $Slave_{(1)}$ on the 1^{st} slot period, while $Slave_{(1)}$ answers back on 2^{nd} slot period. Hence a fair policy is conserved where each slave receives a minimum of one seventh of the Bluetooth bandwidth.

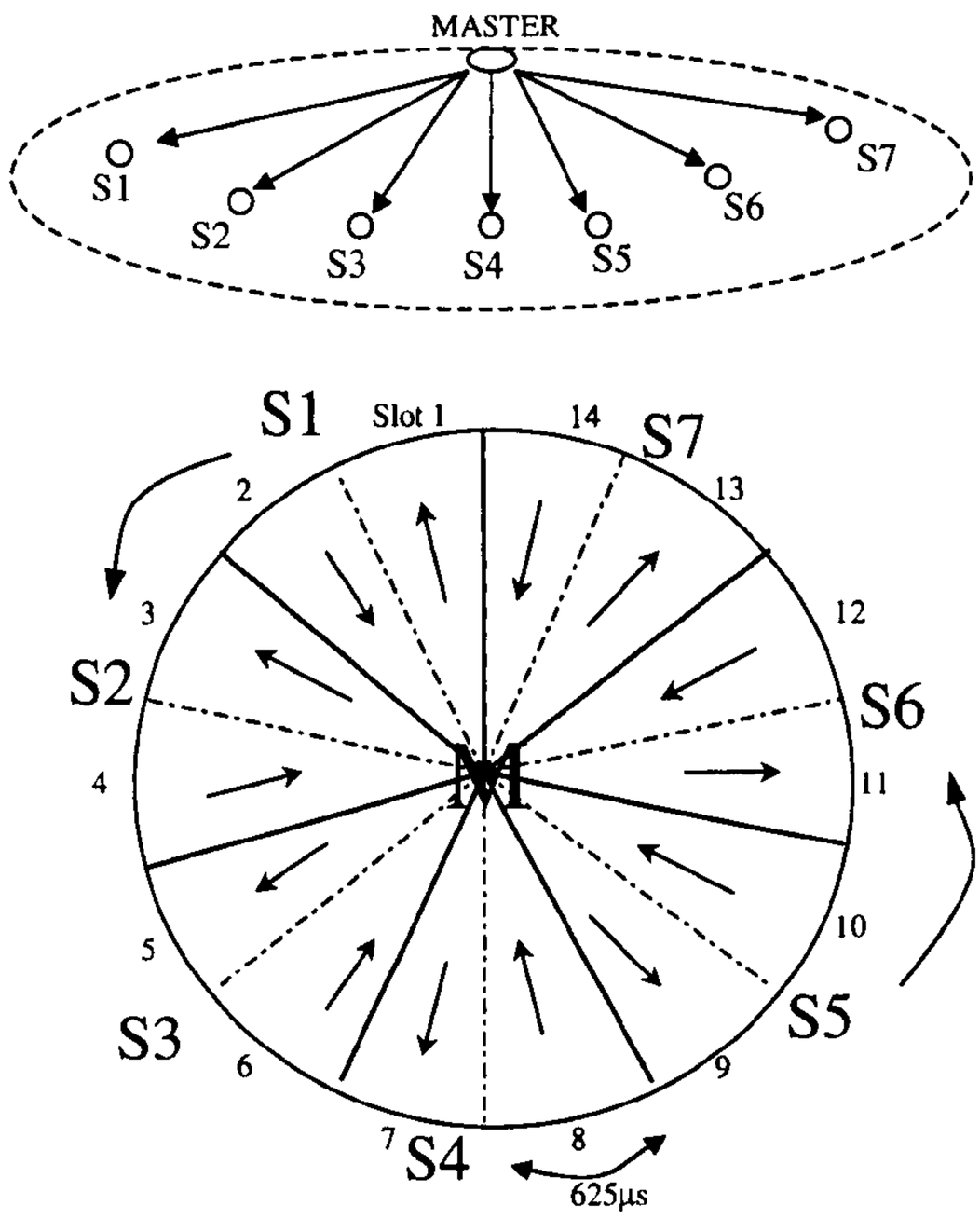


Figure 4.3: Round Robin cycle with seven slaves using 1 slot transmission.

At one instant, when one slave, say $Slave_{(4)}$, detects its uplink and downlink queue data buffer empty, both master and $Slave_{(4)}$ agree a 0-0 combination. $Slave_{(4)}$ will not

participate within the piconet until the 13th Round Robin Cycle end: 195 slots (at the moment this cycle number is arbitrary and it is computed in more detail as shown in *Section 4.5.2*). The slave remains synchronised to the channel, since the master maintains a timer for each slave to poll it again at the exact predicted slot.

The two slots initially distributed for the $Slave_{(4)}$ will be redistributed on the next cycle and given in priority to its predecessor among the round robin list: $Slave_{(3)}$.

Hence twelve slots later, the master indicates to $Slave_{(3)}$ the availability of the next two slots by setting a 1-0 combination (or 1-1 depending on the uplink). This leads to three different choices for the next uplink slot $Slave_{(3)}$-Master:

- If $Slave_{(3)}$ wants to receive or send data to the master on the next two free slots (left by $Slave_{(4)}$), on the uplink it will transmit a 1-1 combination.
- If both $Slave_{(3)}$ and Master queue data are empty, $Slave_{(3)}$ will send a 0-0 bits information, and will enter in a sleeping mode for 195 slots duration. Such that on the next round robin cycle, the master will use the two frames, left by $Slave_{(3)}$ and $Slave_{(4)}$, to transfer data with $Slave_{(2)}$.
- If $Slave_{(3)}$ cannot take the next coming free slot due to scheduling with another piconet, it indicates its availability for the next round robin cycle slot by a 0-1 combination, and returns synchronised to this piconet only after 13 slots.

2 bit combination from Slave to Master	***Next Polling Interval***						
0 - 0	**195** slots						
0 - 1	*Number of Slaves presents in the Piconet*	7	6	5	4	3	2
	Number of slots	**13**	**11**	**9**	**7**	**5**	**3**
1 - 0	Left for future work						
1 - 1	**1** frame						

Table 4.3: Two bits combination to aware Master of slaves' intension on next meeting time.

Master and slave agree on time slot duration uniquely by looking at the current packet queue buffer of both nodes. This defines the interval of 1 frame, 13 slots (for seven slaves) or a multiple of the cycling period called waiting cycling time, in here 13 providing 195 slots. Slaves have priority over master to seek the next meeting slot, which could give them time to operate in other piconets.

In the situation that communication fails between Master-Slave connections, i.e. either uplink or downlink, the next predicted meeting time occurs always on the next cycling sequence. Hence in a piconet of seven slaves, if communication is missed when slave comes back to active mode after 195 slots, the next meeting will be on the next Round robin cycle: after the 13^{th} slot.

Moreover, for slaves' motion within the piconet, i.e. new slave arrives or leaves the piconet, the master updates the new cycling for each slave once slaves come back from a sleeping mode, and modify the slave position on the new Round Robin cycle.

4.5 PERFORMANCE ANALYSIS

4.5.1 Bluetooth System Modelling

To determine the effectiveness of the new AMSQP scheme, a simulation in Matlab C language was conducted with different piconet configurations and transfer rates. The simulation does not take into account the fact that master or slaves enter in inquiry, or scan mode, and relies on the fact that the devices are only focus within their piconet. Within the piconet, the poller does not provide any information about the offered load. This means that neither the L2CAP packet sizes (and thus the number of Baseband packets belonging to the same L2CAP packet) nor the inter-arrival times of these L2CAP packets are known to the poller. Furthermore, the distribution of the inter-arrival times of these L2CAP packets is also unknown. As a result, it has been decided to assume that the load is generated according to a Poisson distribution, i.e., the L2CAP packets are generated with exponential inter-arrival times. Depending on their size, L2CAP packets may be segmented into only one segment size represented by DH1 packet type.

As shown by Figure 4.4, each slave generates packets in 1-slot bursts of 625μs lengths, and the arrival rate follows the Poisson process. This means that two or more consecutive bursts from the same slave could be generated in 1.25ms (or more) bursts but they will be considered one by one and not put in a three-slot or five-slot time as define in Bluetooth specification and it is assumed that there is no packet loss due to transmission error. The traffic rate is the sum of the traffic rates from master to slaves; while in the reverse direction the minimum amount of data is exchanged with Poisson distribution of rate γ (one packet every 195 slots is required to synchronize the slave within the piconet).

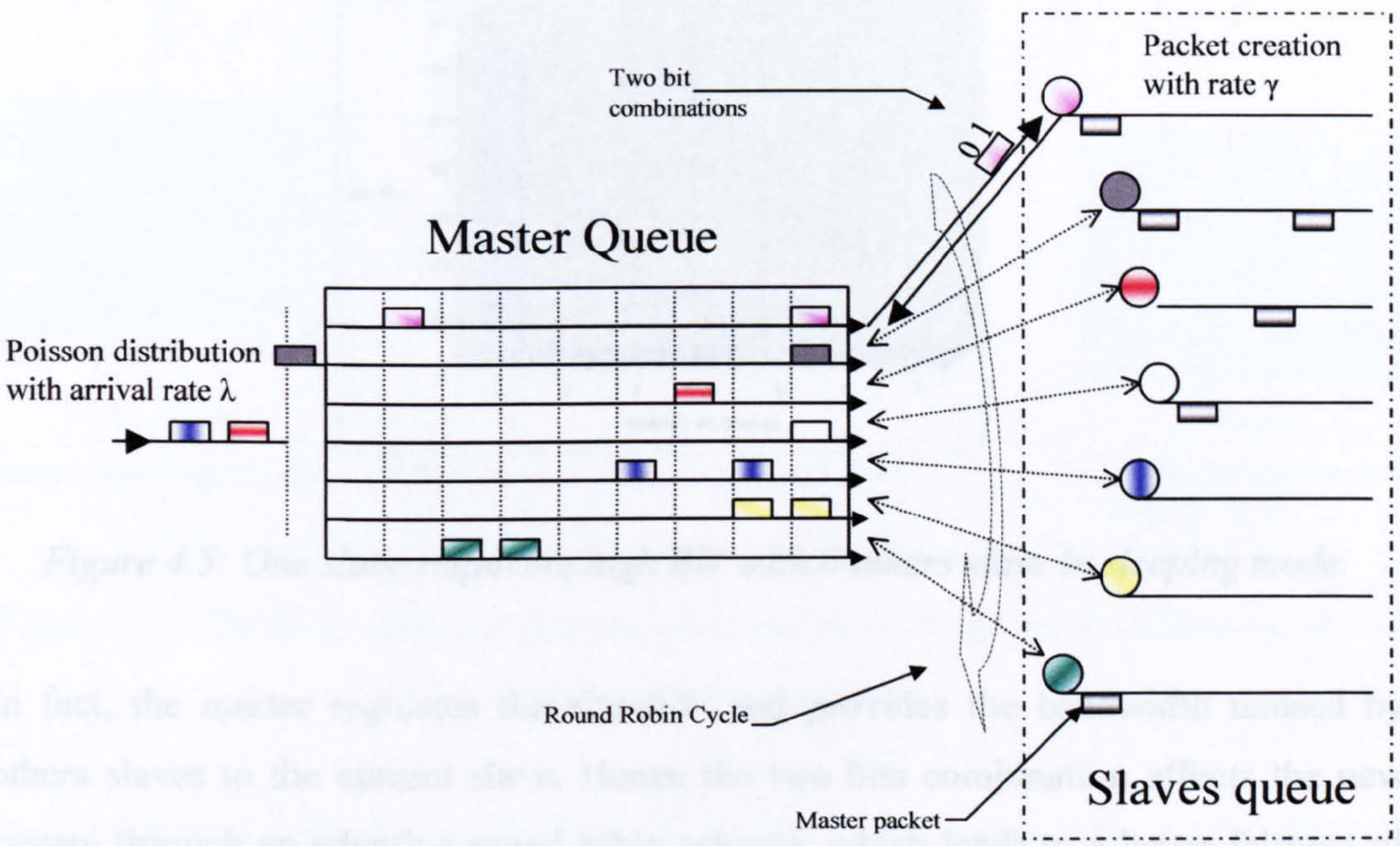


Figure 4.4: A single Piconet with its queuing model.

4.5.2 Throughput Fairness Analysis Performances

The throughput is computed and analysed depending on the number of slaves and traffic flow within the piconet. As shown in Figure 4.3, the formation of the piconet is structured with one master and seven slaves. First of all, only one slave requires a larger bandwidth to transfer data. Figure 4.5 illustrates the AMSQP scheme which establishes a maximum of 129Kbit/s throughput supply for the slave; in which the system adapts to the new restriction within a maximum delay time of 0.12 seconds (195 slots).

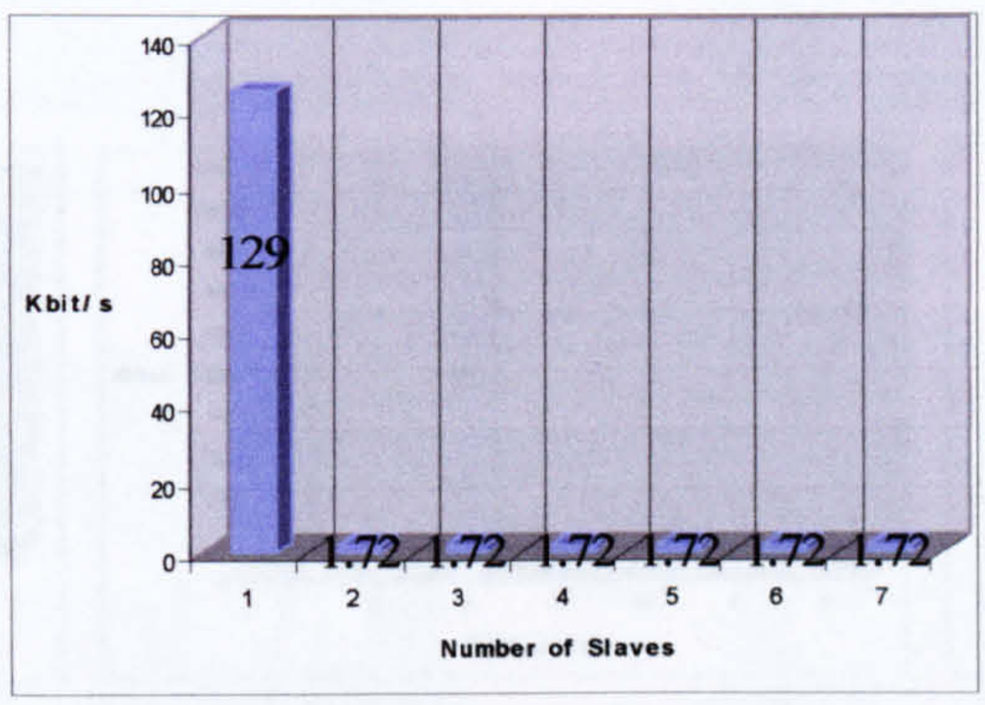
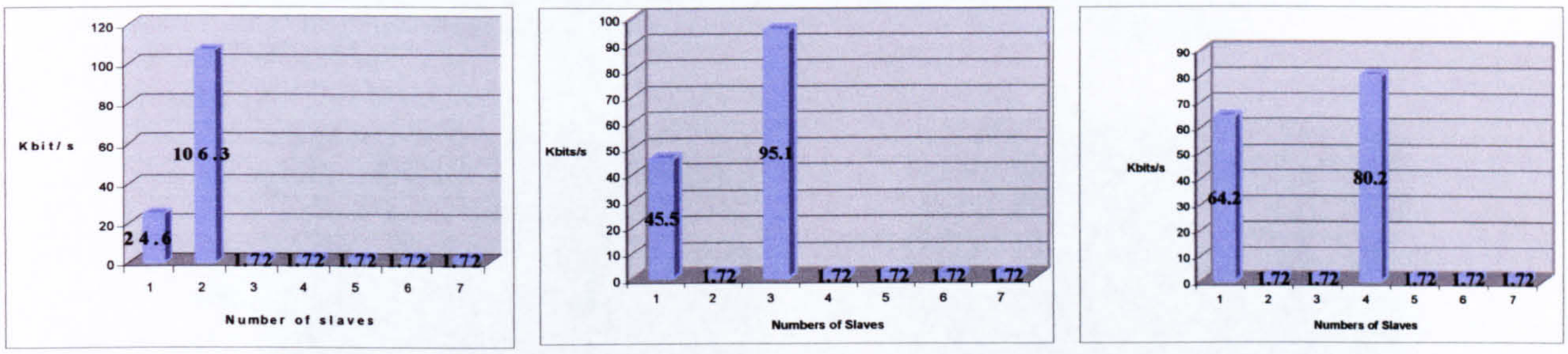


Figure 4.5: One slave requiring high BW with 6 others slave in sleeping mode.

In fact, the master regulates the situation and provides the bandwidth unused by others slaves to the current slave. Hence the two bits combination affects the new system through an adaptive round robin scheme, which leads to a better fairness of the system when compared with a pure Round Robin scheme with slaves receiving a seventh of the total bandwidth (24.5Kbit/s). The fact that a significant number of schemes have been designed (see *Section 4.2.2*); it has been decided to compare the results with the main adopted scheme which is the Round robin method.

The maximum amount throughput intended for only one slave communicating with the master could be expected as 161.7 Kbit/s (172-(1.72*6)). This value cannot be reached due to the fact that others slaves does not enter in long sleeping period within the same cycle, and thus slaves waking up cycle (up to 195 slots) is not

identical. Such that every time a slave comes back to active state, it will receive data from master and stop the intense downlink slave (DS) communication with its master. The DS will go to sleep till its new allocate frame is starting. Hence the bandwidth of the precedent sleeping slave neighbour in the round robin cycle will not be given to the DS and will be lost. This discontinuity could engender a loss up to six frames not allocated for DS. In fact this depends only on the "awake" slave position (AS) towards the DS round robin ordering position. If the AS is just before DS on the Round Robin list, then no slot-pair is unallocated, but in contrast, if AS is next to DS, then the twelve subsequently slots (in this case seven slaves are presents) will be unused by AS and unallocated for DS. Hence around 33 Kbit/s of throughput are lost. However during this lost time neither master nor slave communicate and thus save power.



A)Two downloading slaves side by side B)Two DS separate by one sleeping slave C)Two DS separate by two or three sleeping slaves

Figure 4.6: The three different spacing positions possible of two downloading slaves and fives sleeping slaves.

Figure 4.6 shows the diverse position achievable in the round robin cycle with two DSs and five others slaves in sleeping mode, and whenever two slaves require significant bandwidth at the same time, the system settles into the new situation by providing a minimum of 24.6 Kbit/s for each link transfer. Figure 4.6 shows the throughput dissimilarity is significant, particularly when both DSs are close to each other. The throughput range could reach around 106Kbits/s for one DS (following the slave on the Round Robin List: RRL) and drop to 24.6Kbit/s for the other. In fact the slave preceding the other sleeping slaves uses their left bandwidth, while the other slave does not get this opportunity. Nevertheless, when both are spaced

between two sleeping slaves on the RRL, the gap between both throughputs decreases, i.e. one with 80Kbit/s and second with 64Kbit/s.

Figure 4.7 illustrates the same procedure for three slaves effecting data transfer through the master at the same time. $Slave_{(3)}$ uses the bandwidths left by other slaves and reaches a 94Kbits/s throughput, while $Slave_{(1)\ \&\ (2)}$ do not benefit from other sleeping nodes with 24.6Kbits/s throughput.

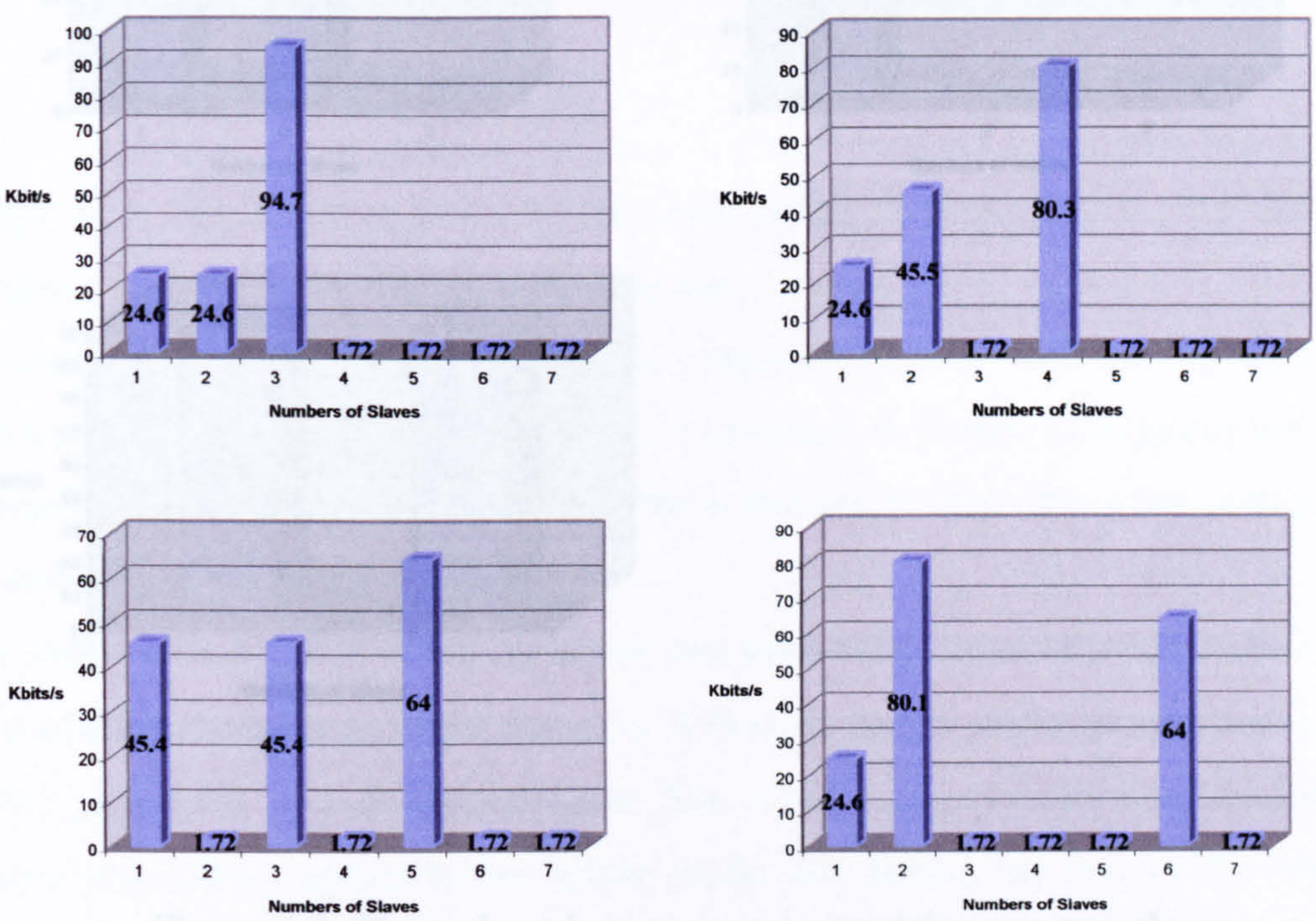


Figure 4.7: Three DS separated by non-downloading slave with different position among the Round robin list.

As mentioned before, the DS position, among the round robin cycle changes the slave throughput and graph 4.7 shows the maximum throughput achieve with piconet of 4 or 3 devices.

In summary, the AMSQP shows some lack of fairness, since some slaves receive more bandwidth than others depending on their position among the Round Robin Cycle. However AMSQP allocate bandwidth to slave requiring higher bandwidth and

obtain an enhanced fairness distribution compared to a seventh of the bandwidth allocated to each slaves in a pure round robin scheme with seven slaves.

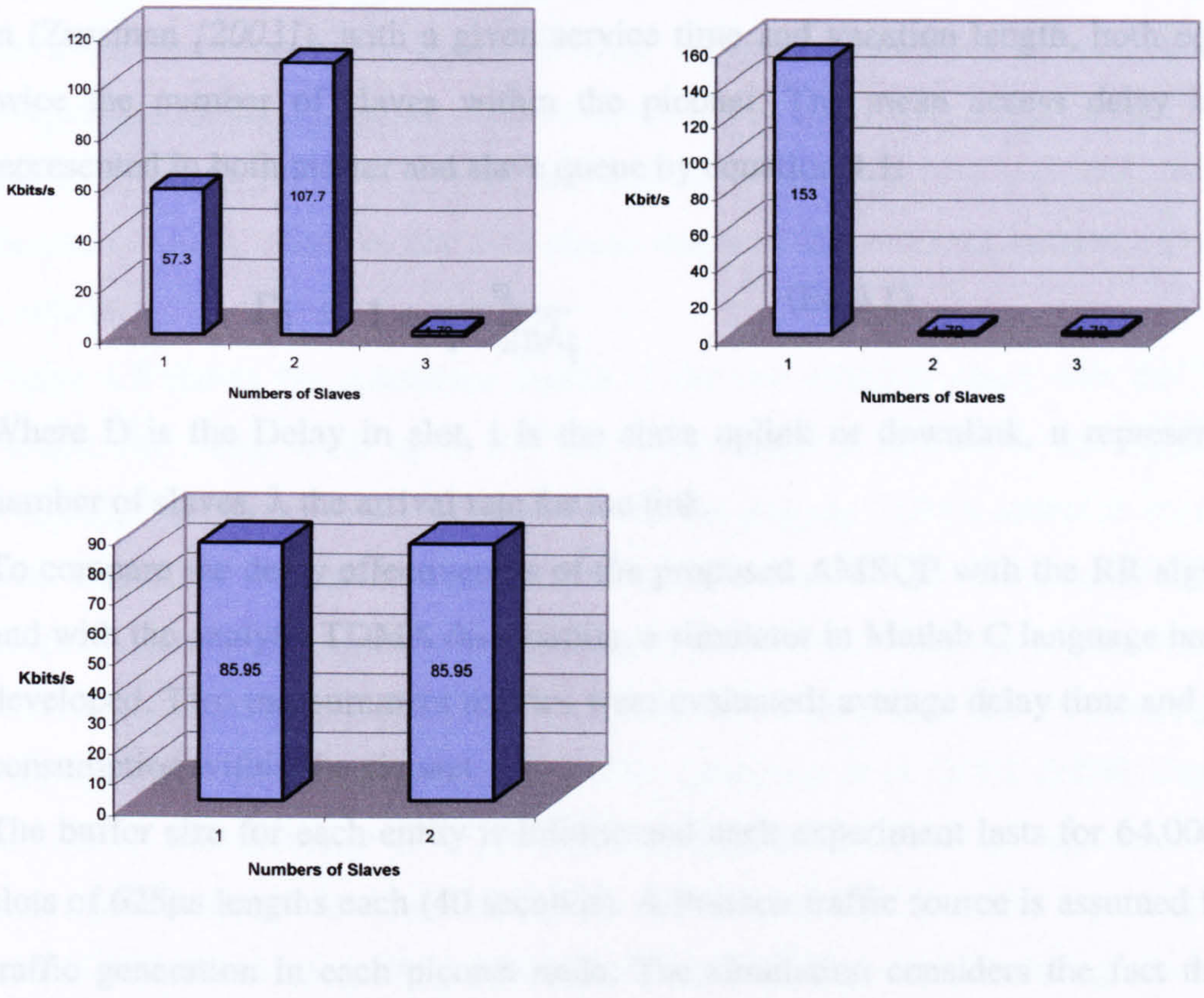


Figure 4.8: Throughput between master and three or two slaves

4.5.3 Packet Delay Simulation

The packet delay in a piconet can be described as a function of arrival burst packets, packets size, and the way these packets are served through the network. Furthermore, the token bucket specification consists on an arrival rate p, a delivery rate d and a bucket size s. To have a minimum delay, the arrival rate factor p must be equal to the delivery rate d, with a bucket size s null. The piconet traffic can be described with the queuing model in which each slave maintains an uplink queue, while the master maintains a number of downlink queues, one per active slave. It is assumed that

buffers have infinite size. Since packet bursts are generated with the same length of a single slot to each uplink and downlink, the distribution of the piconet can be analysed as a TDMA system (Misic *[2003]*), when using Pure Round Robin scheme. The computation of the delay could follow the theory of M/D/1 queue as explained in (Zussman *[2003]*), with a given service time and vacation length, both equal to twice the number of slaves within the piconet. The mean access delay is then represented in both master and slave queue by equation 4.1:

$$D_i = 1 + \frac{n}{1 - 2n\lambda_i} \qquad \text{(Eq 4.1)}$$

Where D is the Delay in slot, i is the slave uplink or downlink, n represents the number of slaves, λ the arrival rate for the link.

To compare the delay effectiveness of the proposed AMSQP with the RR algorithm and with the analytic TDMA distribution, a simulator in Matlab C language has been developed. Two measurement metrics were evaluated: average delay time and power consumption within the piconet.

The buffer size for each entity is infinite and each experiment lasts for 64,000 time slots of 625μs lengths each (40 seconds). A Poisson traffic source is assumed for the traffic generation in each piconet node. The simulation considers the fact that the master and slaves remain in the active mode. No fading, no loss packet and no retransmission are taken into account in this scenario.

In the RR approach, the master transmits data for every even slot to different slaves. If the downlink queue is empty, the master sends an empty (POLL) packet. The slave responds to each downlink packet by sending the data packet from its uplink queue, or an empty (NULL) if the uplink queue is empty. After the end of the exchange, the master moves on, to poll the next slave; and when all slaves have been visited, the sequence is cyclically repeated.

While in the AMSQP approach simulation, two bits are implemented to indicate the empty buffer for both master and slave queue. Here the master could decide to continue polling the current slave or to put it in a sleeping time for m cycles. Where m refers to the polling interval in Table 4.3.

4.5.4 Delay Performance Evaluation.

The simulation consists of seven slaves units, in which all units have exactly the same rate. The polling cycle is simply equal to the number of slaves in the piconet, i.e. seven pair-slot times. Under AMSQP scheme, it has been decided to fix the sleeping period for 13 cycles, which corresponds to 10ms in sleeping mode.

The delay appearing in both schemes is evaluated through load variation, from low flow to full load, which in this case means that each slave-master downlink generates a seventh of the total load.

Figure 4.9 shows the simulation results of the two schemes along with the TDMA system. For traffic loads up to 0.95 the lowest delay is offered by the RR scheme. The difference between the RR scheme and the analytic TDMA curves is negligible, with TDMA curve slightly higher. This confirms that the use of the analytic TDMA proposal could be exploited to simulate a RR scheme for loads less than 0.95, excluding a load of around 1 where the delay in the proposed TDMA theory goes to infinity. Equivalent results are produced by (Zussman *et al [2003, 2004]*, Capone *et al [2001]*, Kleinschmidt *et al [2004]*). As expected the AMSQP approach has the worst performance with small loads, if a data burst arrives just after a polling period of a slave, the current polling cycle of that slave is large with a waiting time of 195 slots, so the access latency of the burst will be on average less than half of the 195 slots. However the unexpected result is that it has the lowest delay for higher loads. The main reason behind this performance is that the amount of slaves in sleeping mode becomes very low due to intensive traffic; and AMSQP applies a flow control and gives the unoccupied slot to other slaves. While in RR the master waste a slot by still sending a POLL packet for the empty message in an empty queue, which increases the delay in other master-slaves queues.

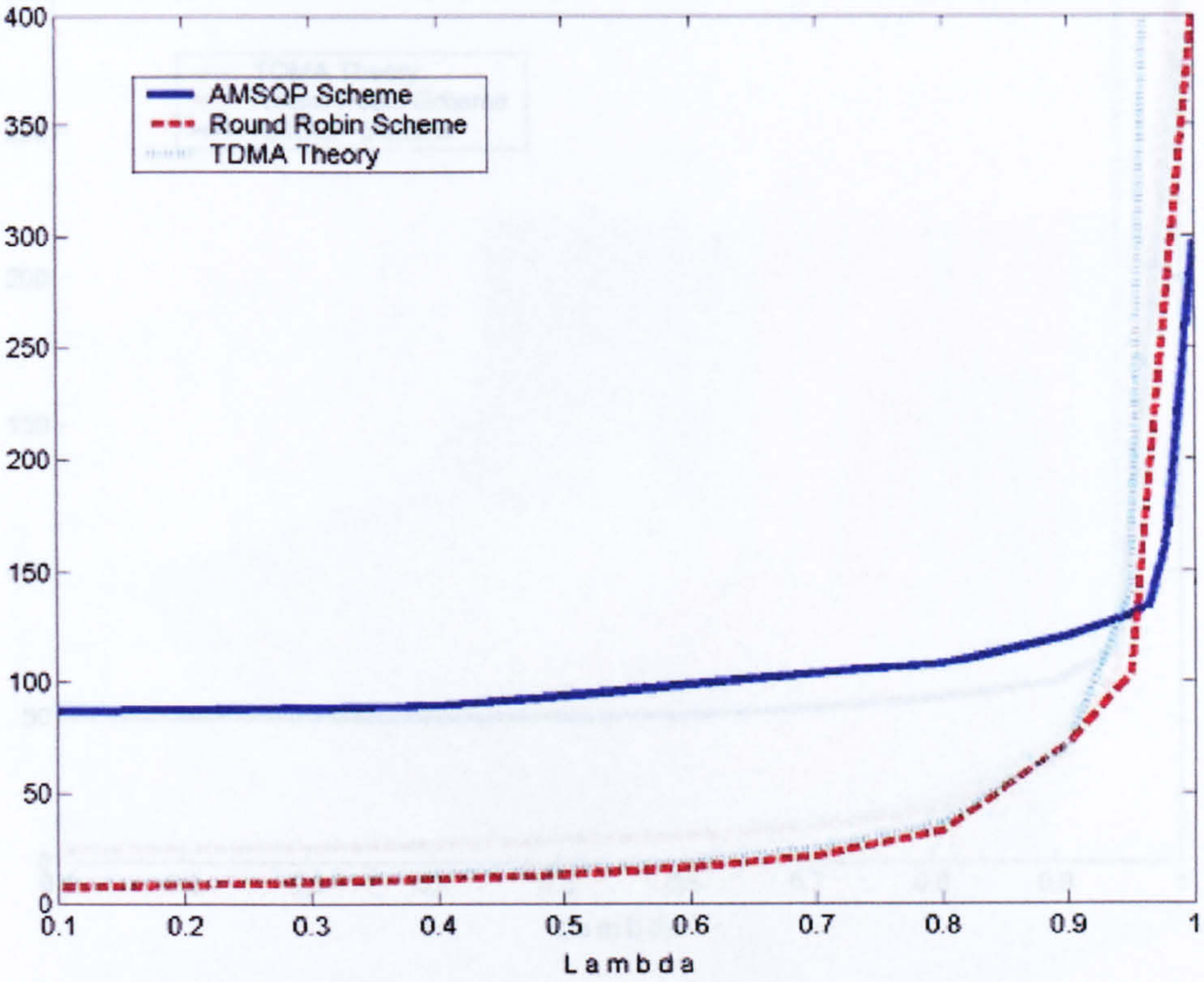


Figure 4.9: Mean access Delay in a piconet with 7 slaves and with same load lambda for each link.

A further simulation approach has been carried out with only four slaves. Figure 4.10 shows the results, moreover the curves confirm the earlier explanations that for a high load, the AMSQP approach reduces the delay, while the RR scheme increases continuously as the flow active time increases. For a lower load the AMSQP performs better than with seven slaves essentially due to the fact that the cycle period is shorter.

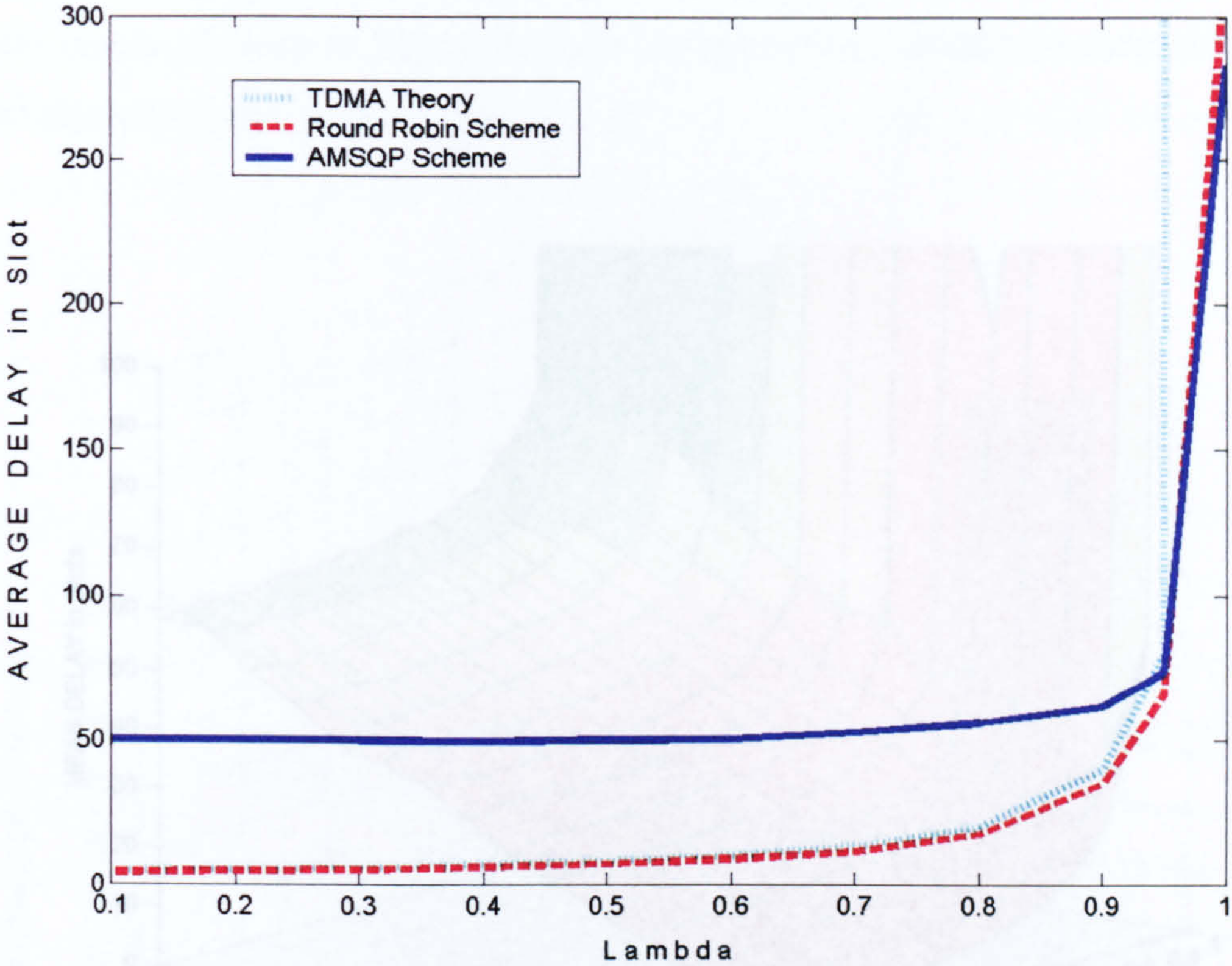


Figure 4.10: Mean access Delay in a piconet with 4 slaves and with same load lambda for each link.

In the next experiment, in addition to the regular (Poisson process) traffic at the master and slaves sides, a variation of the sleeping time is employed. When slaves enter in a sleeping mode (2 bit combination "00"), they stay between 1 to 15 cycles time.

Figure 4.11 shows the impact of the sleeping cycle variation on the average delay of the piconet while applying different load. As expected, when the number of cycles approach zero, AMSQP matches the earlier results obtain by the RR delay for low load in Figure 4.10, (around 5.5 slots delay for low traffic rate). But, while in sleeping mode, slaves have to wait till the next polling time occurs to wake up and sends data. Hence the access delay increases within the piconet as the sleeping time increases. Therefore for a sleeping period of 15 cycles, the average delay increases to 60 slots.

It could be noted that for high load traffic, whether AMSQP uses low or high waiting cycle; the delay increases considerately.

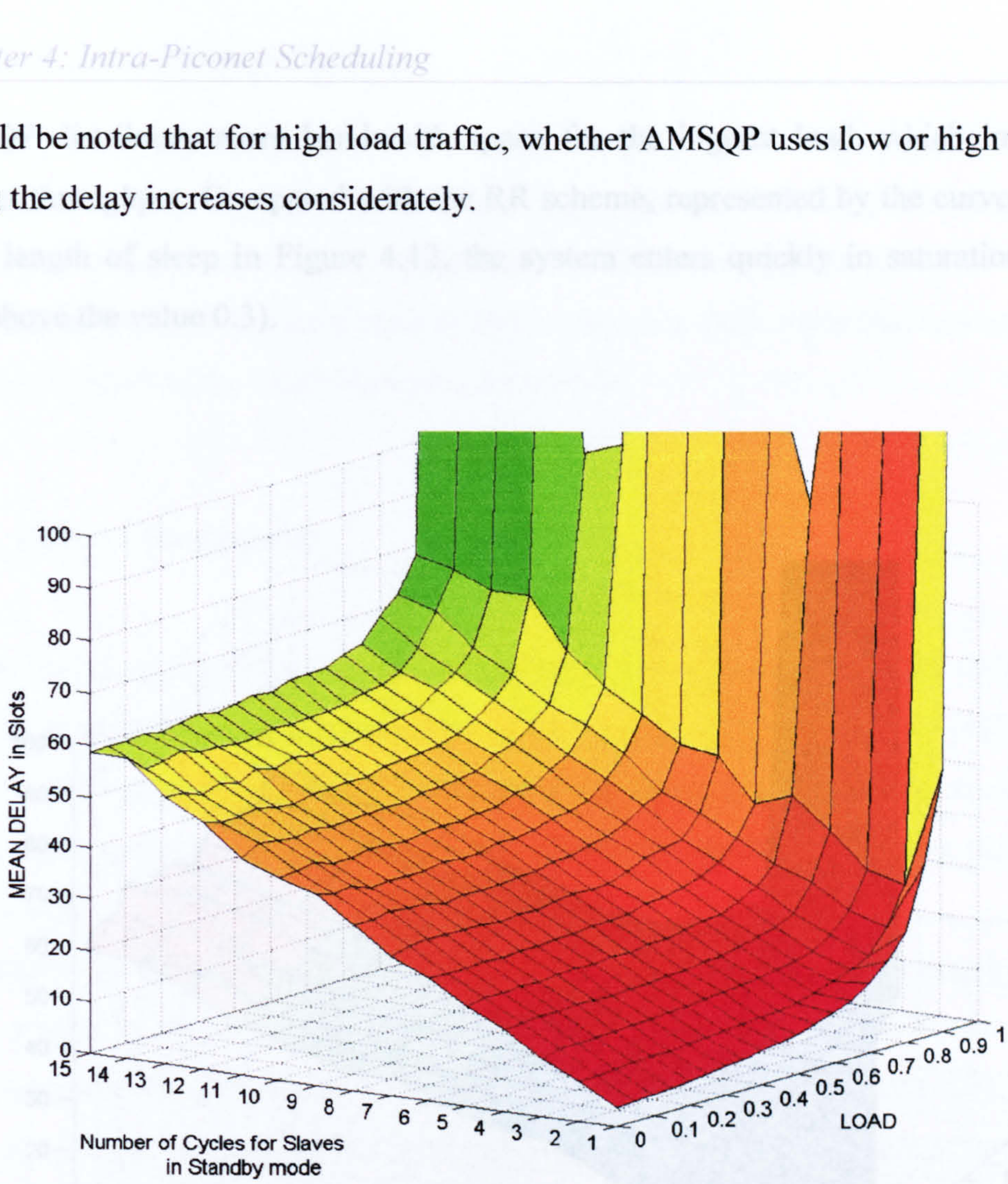


Figure 4.11: Mean access Delay in a 4 slaves piconet with same load for each link, and depending on standby cycle length.

In the second experiment, the comparable piconet with 4 slaves is observed, in which, only one slave varies its traffic rate with the master, while other slaves have a low traffic rate constant and enough to keep the slave synchronised with the master which is 0.015 (1 slot transmission every 130 slot).

Figure 4.12 shows the average delay obtained for all links within the piconet. In contrast to the earlier results presented in Figure 4.11 (slaves having same load), when the piconet applies a single load and uses significant time interval in the sleeping mode (number of cycle above 6), the scheme use the bandwidth more efficiently. By accommodating bandwidth from other slaves' link in sleeping time,

AMSQP distributes more bandwidth space for the biggest load, which improves piconet throughput. Compared with the RR scheme, represented by the curve with 1 cycle length of sleep in Figure 4.12, the system enters quickly in saturation (with load above the value 0.3).

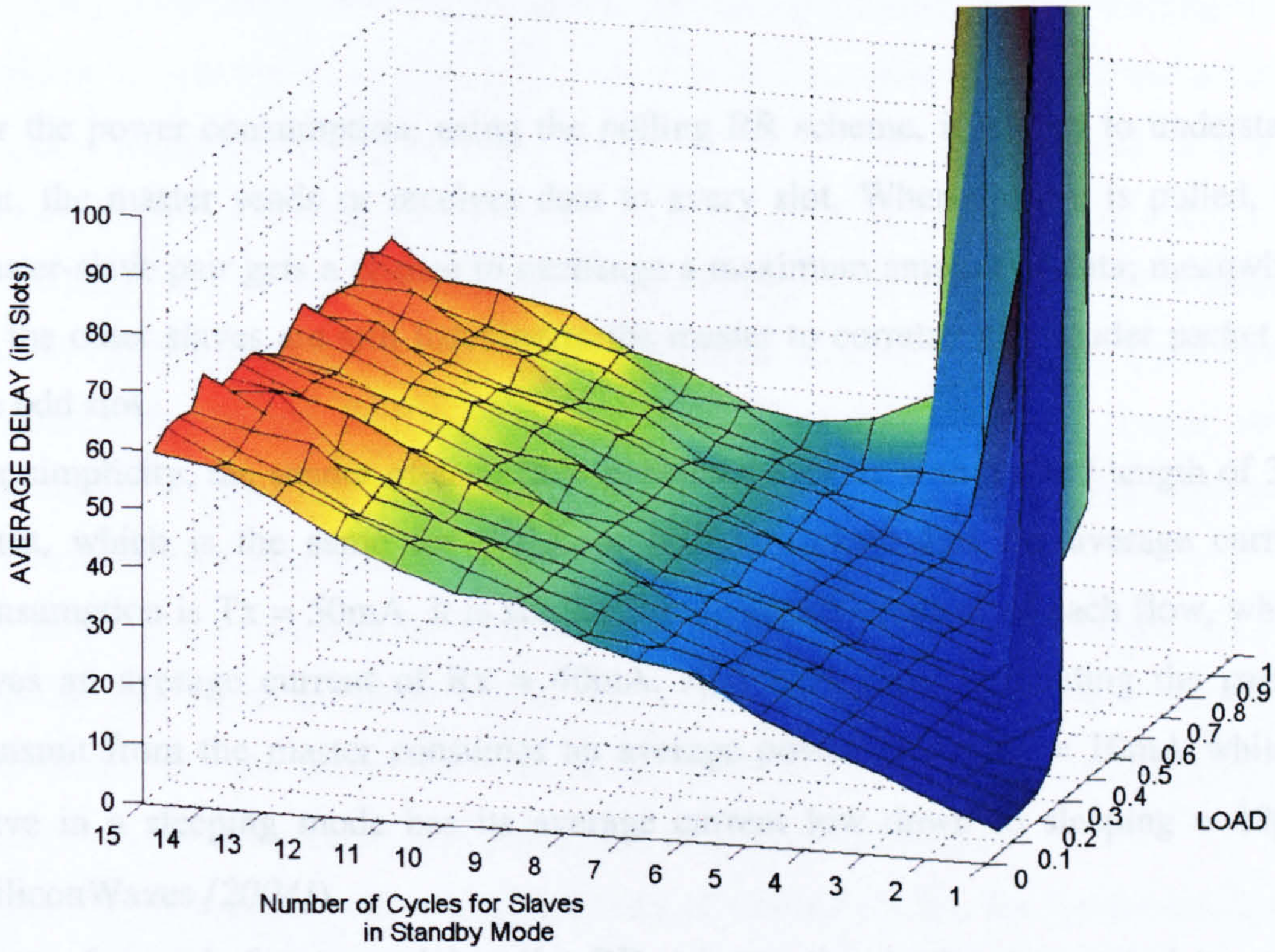


Figure 4.12: Mean access Delay in a 4 slaves piconet depending on only one major load link, and on standby cycle length.

It could be noted that most simulations take into account only 4 slaves to generate the result. These simulations contribute to make the result more consistent with preceding and future results. As shown by Figure 4.10, the overall profile for 7 slaves has identical perspective as Figure 4.11.

The results presented so far suggest that the delay depends predominantly with the number of sleeping cycle employed. For low or similar flow rate, the system adapts

the traffic in excellent manner under short sleeping cycle. While generating an unbalanced traffic, the AMSQP scheme improves the overall performance by applying a number of sleeping cycles above 6. However, the study of power consumption needs to be completed to deliver a better QoS, topic that has not been evaluated in most of the other intra-piconet scheme.

4.5.5 Power Consumption

For the power consumption, using the polling RR scheme, it is easy to understand that, the master sends or receives data to every slot. When a slave is polled, the master-slave pair gets a chance to exchange a maximum amount of data; meanwhile all the other slaves are still listening to the master to correlate the header packet on the odd slot.

For simplicity, the sender of each flow generates packets with a fixed length of 339 bytes, which is the same for POLL or NULL packet. And the average current consumption is Tx = 50mA. It is similar for the packet receiver of each flow, which gives an average current of Rx = 40mA. And each slave correlating the packet transmit from the master consumes an average power of Active = 10mA while a slave in a sleeping mode has its average current low down to sleeping = 10μA (SiliconWaves *[2004]*).

Hence for each frame applying the RR scheme the master transmit data, with Tx=50mA, the slave receives the data, with Rx=40mA, while all other slaves correlate the data, with Active = 10mA. Then the Slave transmits back to the master, with Tx, and master receives the data, with Rx.

So the average current consumption within a piconet of 5 devices with the RR scheme could be expressed as:

For the master: (Tx + Rx) / 2 slots = (50mA + 40mA) / 2 = **45mA**

For each slave with same load:

[(Rx + Tx) + Active *3 slaves] / (4 slaves * 2 slots) = [(40mA + 50mA) + 10mA*3]/8 = **15mA**

Average Current consumption: [45mA + (15mA * 4 slaves)] / 5 devices = **21mA.**

So the average current dissipation of the master and four slaves is 21mA for high or low traffic data.

Figure 4.13 shows the simulation results of the new AMSQP scheme based on the same process as in Figure 4.11 (with 4 slaves applying the same rate). In contrast to Bluetooth RR scheme, the average currents dissipated by the piconet of 4 slaves consume far less than 21mA, even applying different loads and sleeping cycle periods.

For high data traffic, with load above 0.9, independently of the waiting cycle numbers, all curves reach 18mA. In fact, this data transfer saturation make slaves maintain transmitting data to the master, and so a slave never enters in a long sleeping period, that's why the sleeping period of 15 or of just 1 waiting cycle, does not influence the network transmission. The AMSQP simulation performs a reduced amount of 15% less consumption with high traffic than pure RR. This improvement is mainly due to the fact that slaves are not listening when the master sends information to other slaves, such that every odd slot three slaves are not correlating packet from the master and save power.

It is interesting to see that for a slave using zero cycle waiting, the average power is constant and fixed to 18mA. That means the slave is listening in round robin behaviour, but only wakes up every seven slots. Hence delay and throughput are identical as the RR scheme, except that it is consuming 15% less.

Under low data traffic with a high numbers of waiting cycles, the system consumes far less power than the RR with 90% reduction. This energy saved is mainly due, as discuss before, to the efficiency of polling scheme where the master poll only slaves when they have data, and slaves "relax" when data is not present in their queue.

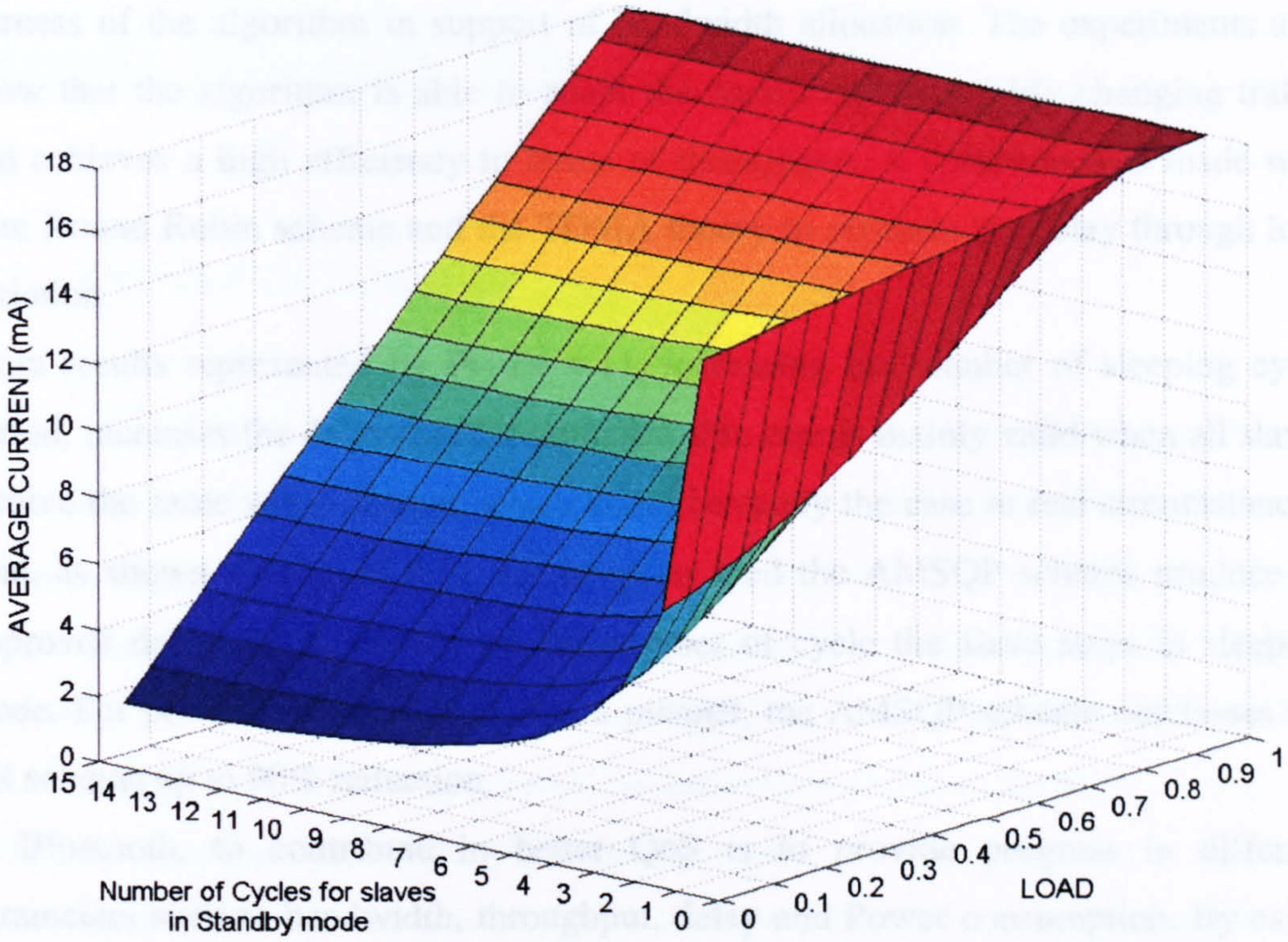


Figure 4.13: Average Current for piconet of 4 slaves depending on slave sleeping cycle and load in the overall piconet.

4.6 CONCLUSION

In this chapter, a flexible adaptive scheme namely AMSQP has been proposed for supporting QoS in Bluetooth intra-piconet. AMSQP defines a predictable polled sequence, and an adaptive traffic allocation to improve pure Round Robin scheme for fairness and power consumption. The master and slaves, supporting AMSQP scheme, need to cooperate and exchange only 2 bits negotiation inside the header packet. No threshold or complex computations are required, which keep the scheme simple. The algorithm follows one slot length (DH1) policy to construct the scatternet topology formation, explained earlier in Chapter 3. In order to evaluate the performance of the algorithm, a piconet model has been designed through Matlab

assuming a Poisson arrival process. Simulation results have demonstrated the fairness of the algorithm in support of bandwidth allocation. The experiments also show that the algorithm is able to adapt the bandwidth to quickly changing traffic and achieves a high efficiency in terms of throughput. A comparison is made with pure Round Robin scheme and the TDMA theory to estimate the delay through load variation.

From results represented by Figure 4.11, increasing the number of sleeping cycle period, increases the delay too. Nevertheless this case is mainly valid when all slaves receive the same value of load, which could be rarely the case in real circumstances. And, as shown by Figure 4.12, for irregular load the AMSQP scheme produce an improved delay independently on the number of cycle the slave stays in sleeping mode. For power consumption within a piconet, the AMSQP scheme outclasses the RR scheme up to 90% reduction.

In Bluetooth, to contribute in better QoS is to provide progress in different parameters such as bandwidth, throughput, delay and Power consumption. By using the AMSQP, a compromise between power consumption and access latency could be estimated while varying the number of polling cycle within the piconet.

CHAPTER 5

INTER-PICONET SCHEDULING

5.1 INTRODUCTION

With its low cost chips, Bluetooth network is expected to be one of the preferred solution to build inexpensive but large ad-hoc networks. Moreover, Bluetooth is well suited for providing ad-hoc networking for the costumer market, with its low power consumption. Nevertheless, user mobility poses additional challenges for connection rerouting and QoS services (Son *et al [2001]*). A method for forming such multi-hop network scenario is still an open issue in Bluetooth so called scatternet.

Bluetooth presents some technical challenges such as scheduling issues. In fact, a scatternet is formed when at least one device participates in two or more piconets, referred to as an inter-connected bridge node or gateway node. Bridges may be attached to only one Bluetooth piconet, preventing them from being active in more than one piconet at the same time. Therefore, a bridge alternates its participation in several piconets. This task is called inter-piconet scheduling. An important issue is that gateways need to coordinate and then schedule their presence among piconets in an efficient manner. It is therefore important to set up new piconets within a scatternet, in which the throughput can be maximized, and packet delay, and power consumption minimised. Furthermore, an efficient scatternet should require tradeoffs between keeping a decent level of connectivity and reserving enough network resources for communication.

In this chapter, *Section 5.2* presents some related work on previous scatternet scheduling. *Section 5.3* puts forward the establishment of a new hierarchy to build up the scatternet topology defined in Chapter 3. Then *Section 5.3* proposes an intra-piconet schedule for the second hierarchy, adjusting the AMSQP intra-piconet scheme (developed for Leader Hierarchy), and extended it to satisfy bridge connectivity within the new scatternet topology. Bridge nodes coordination will adopt an extra bit combination added with the two precedent bits applied within

AMSQP schedule. Various QoS simulations of the new scatternet scheduling conclude the *Section 5.3*.

Section 5.4 proposes scheduling of the third and last hierarchy, among with simulation results to exemplify the new QoS present within the scatternet. While in *Section 5.5* a conclusion concludes this chapter.

5.2 RELATED WORK

The concept of scatternet, with an efficient implementation could reduce the throughput deficiency of Bluetooth with other communication technologies. A presentation of some recent research on these issues is addressed next. The inter-piconet or bridge scheduling could be separated into two different categories: isolated decision or distributed decision.

In the first case, the bridge itself makes the decision concerning its attendance within a piconet and could inform (if the bridge is a slave) its master of its decision. These implementations could be made using some simple modification from sniff or hold mode.

For distributed decision, a presence interval is defined through the establishment of rendezvous points between master and bridges. A rendezvous point is a pre-arranged slot in which both nodes agree to enable the scatternet to develop the best throughput possible. However, the distributed decision approach requires agreement and thus increases the operation complexity and the decision time.

As for the research on isolated decision mechanisms, Racz *et al [2001]* proposed the Pseudo-Random Coordinated Scatternet Scheduling (PCSS) algorithm. The meeting point follows a pseudo random sequence, which will be unique for each pair of nodes. The advantage is that no explicit information needs to be exchanged between the two peer devices. However, this meeting time does not take into account the variation of the traffic within the system and moreover the meeting points might collide more frequently as the number of bridges increase.

Har-Shai *et al [2001]* proposed the Load Adaptive Algorithm (LAA), which determine the duration of the bridge activity within different piconets. This algorithm

adapts to varying values of load regarding the queue size of the different master and of other nodes giving communication priority to the bridge. The disadvantage is that the LAA is applicable to small scatternets, in which bridges connect only two piconets.

Gerla *et al [2000]* initiated the *rendezvous point* concept, which denotes that the master of the piconet order to the gateways node should be present within its piconet to communicate periodically. A *rendezvous window* is also described which represents the time that the gateway has to stay in the piconet.

Among the distributed decision proposals, Johansson *et al [2001]* presented a scatternet scheduling called JUMP mode. A device in a JUMP mode acts as a bridge and divides its time into time windows of a pseudo random length. The node then signals to piconets when it will be present for each of these time windows. The main drawback of this algorithm is that the inclusion of a new mode to the Bluetooth link controller could be difficult to implement.

Kapoor *et al [2000]* presented another distributed decision mechanism based on the *rendezvous point* (RP) concept. At each rendez-vous point the gateway and the master negotiate the next RP between them. The RP is unique for each bridge, and each master maintains a list containing its RPs within the bridges. This information facilitates the optimization for further RP allocation. The negative aspect is that RP demands extra communication for negotiation and to keep master up-to-date.

Summarizing, the performance of the inter-piconet scheduling is based on the efficiency on the complexity of the pre-arranged time slot (or window) agreement. The negotiation of this agreement should be kept simple while adapting on time-to-time the traffic of both piconet, to improve throughput and packet delay metrics. And it should handle the power consumption performance metric, merely presented by Racz *et al [2001]*.

5.3 NEW INTER-PICONET SCHEDULING

5.3.1 Connection Establishment Between Hierarchies

As introduced in the previous chapter, units belonging to a piconet share the piconet capacity according to the polling algorithm AMSQP used by the master. Two bit combinations gives the next predicted polling time between Master/Slave. Waiting for this instance, the slave saves battery life since it does not need to be present in the piconet and instead activate its sleeping mode for the short period.

The slave could utilise this open time to perform periodic INQUIRY/PAGING to sense the existence of other nodes as shown in Figure 5.1. The slave could thus detect other devices in its range and exchange configuration parameters in order to establish direct communication with new devices. According to Bluetooth specification v 1.1 (*SIG* [2001]), within the Inquiry process two opposite states are affiliated to locate devices. One acts as the inquirer (future master), the other acts in a reactive manner that allows it to be discovered in an Inquiry scan (future slave). The inquirer rapidly alternates between transmitting inquiry ID packets asking neighbouring devices to identify themselves and listening for an answer. The ID packet contains an Inquiry Access Code (IAC), which does not identify the sender, but provide synchronisation information. A device in an inquiry scan hops over 32 dedicated frequencies with a very low rate (once every 2048 slots), while the inquirer hops at much faster rate, to facilitate proper frequency synchronisation within a reasonable amount of time.

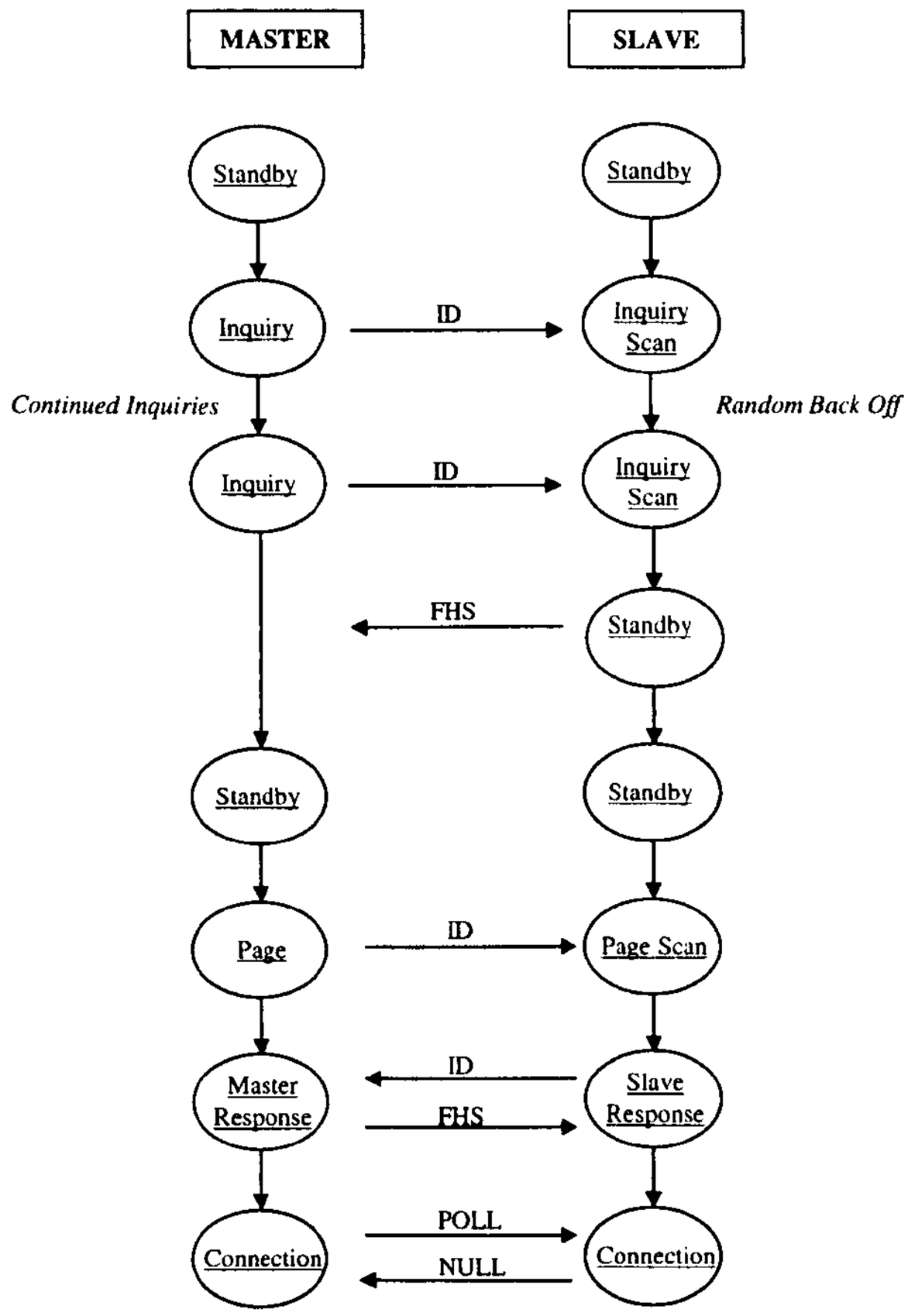


Figure 5.1: State transition involved in establishing a connection

As soon as an inquiry scan device receives an ID packet, (and to avoid contention, as multiple Inquiry scan nodes can receive simultaneously the same Inquirer), the device computes a back off interval and starts listening again between 0 and 1023 time slots. Such that after receiving the ID packet, following the back off interval, the node in inquiry scan will send an FHS (Frequency Hop Synchronisation) packet containing its identity and synchronisation information.

As described in the Bluetooth specification, both devices then enter into a page process: the inquirer in page mode and the inquiry scan device in page scan mode. Within page mode the devices use all information collected by the inquiry process and try to establish contact by sending out trains of ID packets based on the BD_ADDR of the devices and its clock frequency. When a Page device gets an

answer, both proceed to establish the Master-Slave connection and eventually enter in CONNECTION state as illustrates by Figure 5.2.

Once connected, within the new structure elaborated in *Section 3.3* the two devices apply the new scatternet topology with the pager becoming master and the Page scan device its (Child)Slave. Both devices establish a new piconet where the slave, becoming master, applies its Leader's clock, and adjusts a derived FHS (as explain in Chapter 3) to facilitate the synchronisation between both piconets. From now a new hierarchy is formed with a Leader, a Master and a (Child)Slave, along with master becoming a gateway between the Leader and its (Child)Slave.

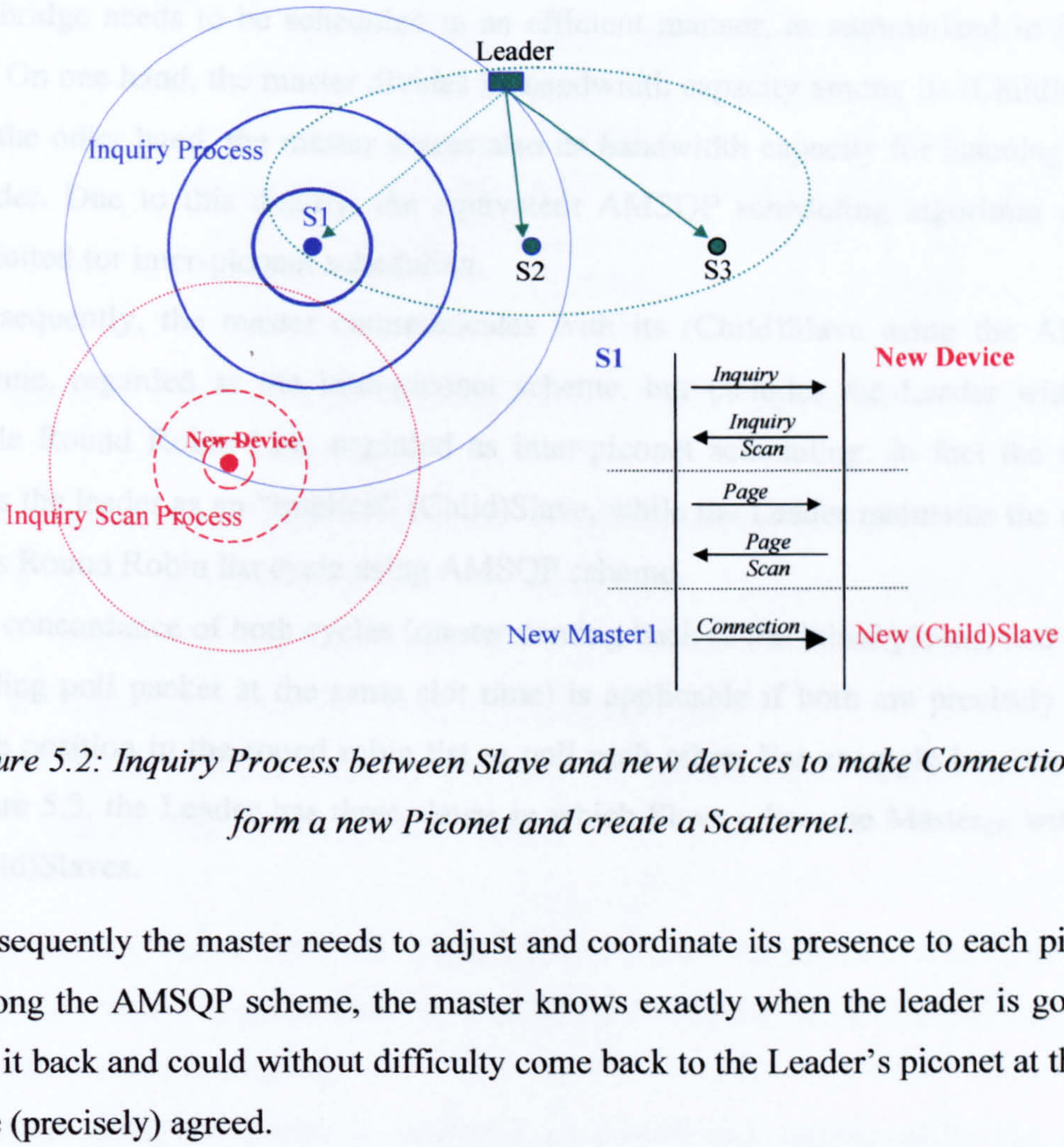


Figure 5.2: Inquiry Process between Slave and new devices to make Connection and form a new Piconet and create a Scatternet.

Consequently the master needs to adjust and coordinate its presence to each piconet. Among the AMSQP scheme, the master knows exactly when the leader is going to poll it back and could without difficulty come back to the Leader's piconet at the slot time (precisely) agreed.

By allowing a master to keep the same phase and the hopping channel of its upper hierarchy, it can contribute to a perfect synchronization between piconets. So slave could be part in one piconet, and for the next slot, be in another piconet as a result of

perfect clock harmonization and of a known Frequency Hopping Sequence. The gateway is able to keep track of the FHS for all piconets it requires to communicate with, without complex computation. This approach eliminates the need for a guard time in the traffic scheduling and prevent interference, which facilitate the fluidity between hierarchies.

5.3.2 Bridge Scheduling Overview

As presented in the previous section, the bridge could act in both piconet. However, the bridge needs to be scheduled in an efficient manner, as summarized in *Section 5.2*. On one hand, the master divides its bandwidth capacity among its (Child)slaves. On the other hand, the master shares also its bandwidth capacity for listening to the Leader. Due to this duality, the equivalent AMSQP scheduling algorithm can be exploited for inter-piconet scheduling.

Consequently, the master communicates with its (Child)Slave using the AMSQP scheme, regarded as the intra-piconet scheme, but includes the Leader within its Cycle Round Robin List, regarded as inter-piconet scheduling. In fact the master takes the leader as an "implicit" (Child)Slave, while the Leader maintains the master in its Round Robin list cycle using AMSQP scheme.

The concordance of both cycles (master coming back to the initial piconet and leader sending poll packet at the same slot time) is applicable if both are precisely at the same position in the round robin list to poll each other. For example has shown by Figure 5.3, the Leader has three slaves in which $Slave_{(3)}$ became $Master_{(3)}$ with two (Child)Slaves.

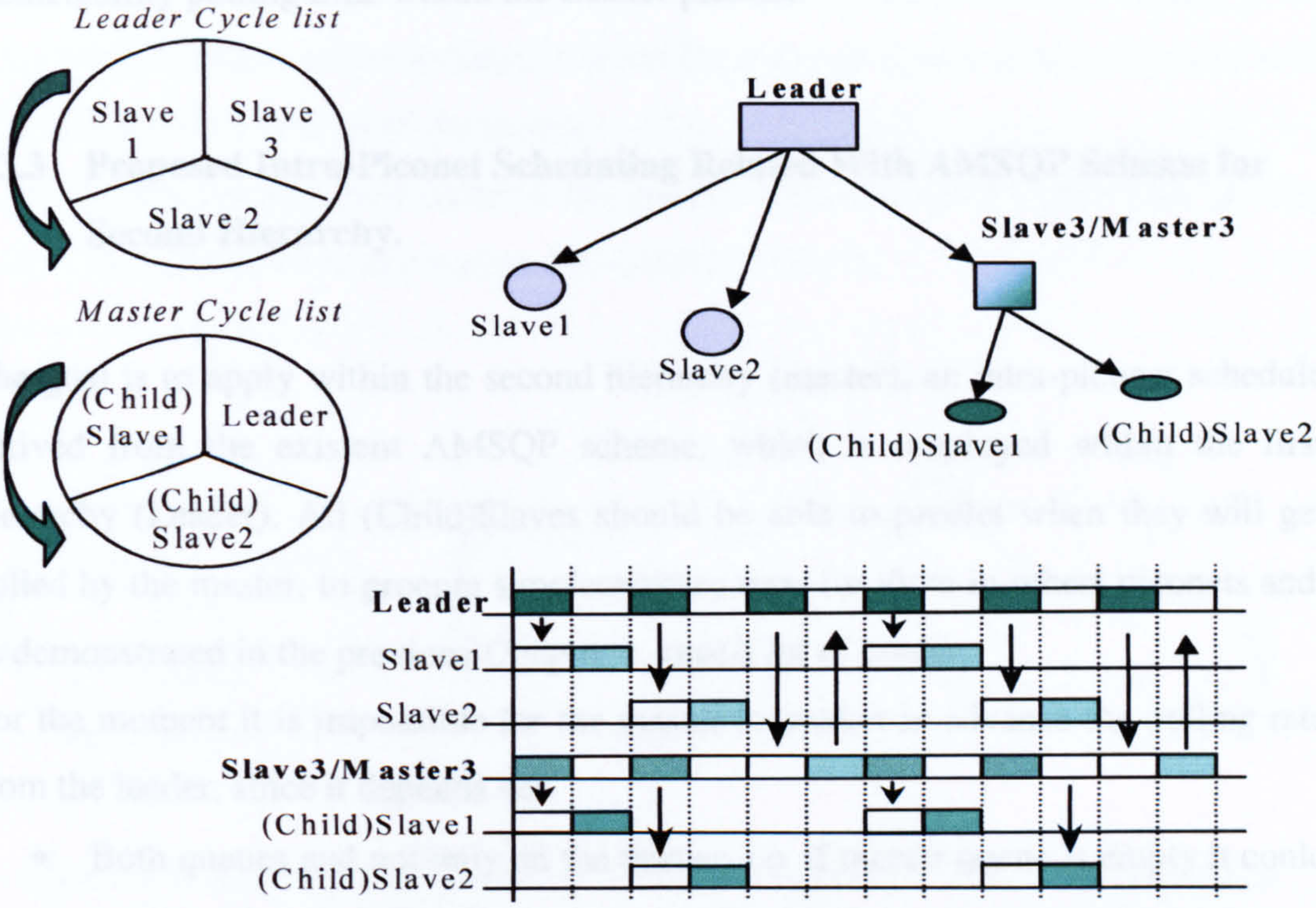


Figure 5.3: Perfect synchronization between Master and Leader meeting point.

The transmission packets in both piconet could be summarised as:

- First pair of slots will be designated for intra-piconet ($slave_{(1)}$ & (Child)$Slave_{(1)}$ at the same time)
- Second pair-slot for $slave_{(2)}$ & (Child)$Slave_{(2)}$
- Third pair-slot concord perfectly for inter-piconet connection between Leader/Master with no slot wasted.

Unfortunately, this coincidence is unusual, and mainly the perfect coordination is not applicable, which requires the evaluation of the AMSQP scheme in another approach for inter-piconet scheduling.

In most cases, the leader is supposed to contact the master, while the master according to its Round Robin list should at the same time, send data to one of its (Child)Slave. These lead to conflict for the master to know on which piconet it must subsist in, either in its own piconet or in the Leader's one. It is decided that the new

schedule attributes leader priority over the (Child)Slave to maintain perfect predictability polling time within the Leader piconet.

5.3.3 Proposed Intra-Piconet Scheduling Related With AMSQP Scheme for Second Hierarchy.

The goal is to apply within the second hierarchy (master), an intra-piconet schedule derived from the existent AMSQP scheme, which is employed within the first hierarchy (Leader). All (Child)Slaves should be able to predict when they will get polled by the master, to procure supplementary time for them in others piconets and, as demonstrated in the previous Chapter 4, save a lot of power.

For the moment it is impossible for the master to predict in advance the polling rate from the leader, since it depends on:

- Both queues and not only on the master, i.e. if master queue is empty it could predict when it will poll a slave next, but if the Leader queue is not empty, then the master should be present in the next Cycling time.
- The next cycling time depends on new slaves arriving or leaving, i.e. modify the RR list polling sequence and thus modify its polling interval.
- Next pair-slot available, i.e. the next slave on the RRL is sleeping, consequently the master could transfer more data in the next free pair-slots.

Hence the master knows the time slot the Leader will poll it, but cannot be aware beforehand for how many slots and when it will happen in the future.

Therefore before the master gets the next poll packet from the leader, it should indicate to its (Child)Slave that its next polling instance is not certain. As a result, the master inform the (Child)Slave of its uncertainty polling attendance by using a single bit detection, in addition of the 2 bits combination used in AMSQP scheme. This could be summarised by:

- 1 denotes that the leader is part of the next Cycle list. This signals that (Child)Slaves predicts the next polling with one more device in the RR list. Hence the (Child)Slave sleeps until the next expected polling slot and wakes

up to collect data. If no data is received, instead of sleeping and waking up on the next polling cycle (as it is achieved by the AMSQP scheme), the (Child)Slave must continue to listen for every even slot after prediction, until it receives data from the master.

- 0 denotes that the Leader is not present within the next cycle. This informs the (Child)Slave to preserve the AMSQP scheme.

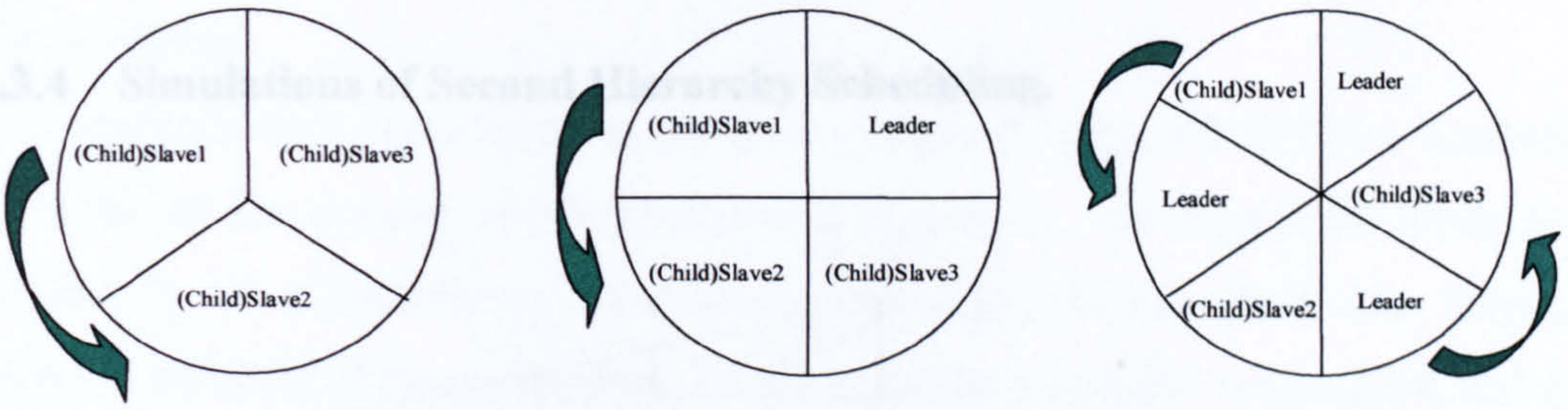


Figure 5.4: Round Robin Master Cycle depending on the third bit combination.

As an example, Figure 5.4 illustrates the different RRL cycle for the master with three (Child)Slaves and with two slaves within the Leader piconet.

Represented on the left of Figure 5.4, is the master cycle for transmitting data to its three (Child)Slave when the third bit combination is equal to 0. This indicates that the next polling Master/Leader does not occur within a cycle, i.e less than 3 frames. The master bandwidth is thus divided by the three (Child)Slaves, alike applying the AMSQP scheme.

In case the third bit is equal to one, all the (Child)Slaves are aware of the master getting connected to the Leader. But they cannot predict for how long and how frequent the master will stay in the leader piconet. Thus all the (Child)Slave will wake up after 4 frames and will keep listening for every even slot after. Represented by the centre of the figure, the master communicates only for one frame with the Leader. Thus the prediction of the (Child)Slave is correct and no extra-power is consumed. But as exposed in the right of the Figure 5.4, the Master stay 3 frames with the Leader and thus after 4 farmes the (Child)Slave wake up and have to listen for 2 more frame before receving data from the master.

It is noticeable, that Slaves take the last decision from the last two bits combination agreement. Hence the Slave/Master could directly notify to the Leader that in the case of significant traffic within its piconet, the master will not take the next bits available but will be present on the next cycle. Instead of sending a "11" combination it could send a "01" or "00" combination and come back to its own piconet to send data to its (Child)Slaves during this short period. At the moment, this feature is not taken into account and the Leader will be given priority as described in *Section 5.3.2.*

5.3.4 Simulations of Second Hierarchy Scheduling.

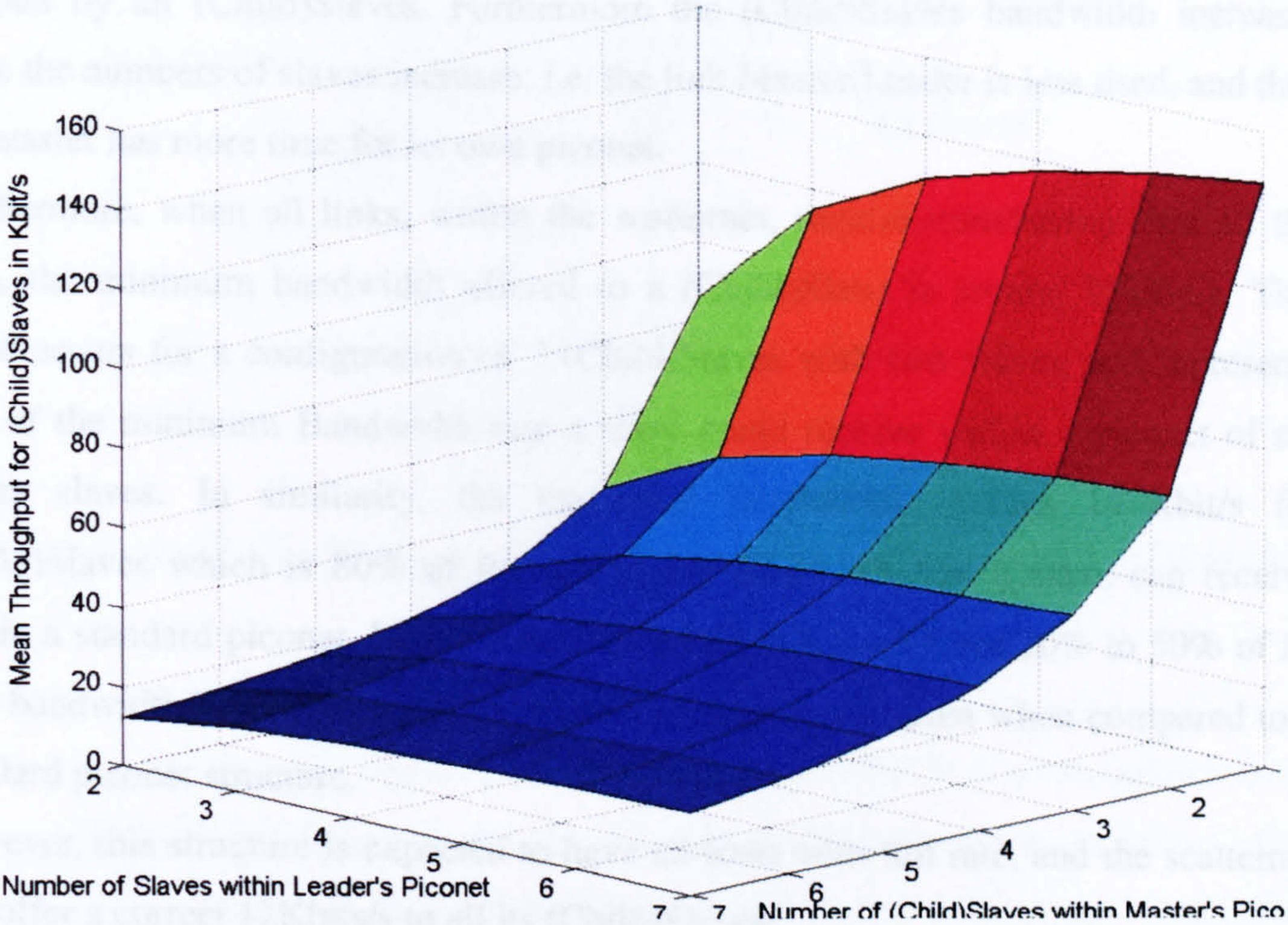


Figure 5.5: Mean Throughput within second hierarchy piconet depending on the number of Slaves and (Child)Slaves.

In order to assess the impact of the bridge node scheduling over the scatternet performance, a simulation has been performed. The model investigates bridge traffic in similar manner to the previous Matlab setting employed in Chapter 4. Leader

applies the AMSQP scheme within its piconet, while (Child)Slaves implements the new extra-bit combination (in complement with AMSQP) to communicate with the master. First an outline of the mean throughput data packet exchanged between all (Childs)Slaves and master is analyzed depending on the number of Slaves within the Leader piconet.

Under the simulation, the Leader has always data to send to its slaves, which means that all slaves receive the equivalent rate but need to be present for every cycle. And using the same process, the master has always data to transmit to all its (Child)Slaves.

In accordance with expectation as shown by Figure 5.5, the (Child)Slave bandwidth decreases as the amount of (Child)Slaves increases, i.e. the bandwidth needs to be divided by all (Child)Slaves. Furthermore the (Child)Slaves bandwidth increases since the numbers of slaves increase; i.e. the link Master/Leader is less used, and thus the master has more time for its own piconet.

Furthermore, when all links, within the scatternet, require transferring data all the time, the minimum bandwidth offered to a (Child)Slave is around 12Kbit/s. This value occurs for a configuration of 7 (Child)Slaves with two slaves, and represents half of the minimum Bandwidth that a slave could receive within a piconet of six others slaves. In similarity, the maximum bandwidth reaches 147Kbit/s for (Child)Slaves which is 80% of the maximum bandwidth that a slave can receive within a standard piconet. Hence a device can be penalized from 20% to 50% of its total bandwidth by being part of the second hierarchy scatternet when compared to a standard piconet structure.

However, this structure is expected to have all links with full rate, and the scatternet still offer a correct 12Kbits/s to all its (Child)Slaves.

On the following section an example of scatternet is simulated and analyzed to evaluate the bridge effect on QoS within the scatternet.

5.3.4.1 QoS in Second Hierarch Scatternet.

To evaluate the QoS (throughput, delay and power consumption) within the second hierarchy, a scatternet model shown by Figure 5.6 is simulated with sevens Slaves in which $Slave_{(4)}$ becomes $Master_{(4)}$ of three (Child)Slaves devices. All slaves receive an identical flow.

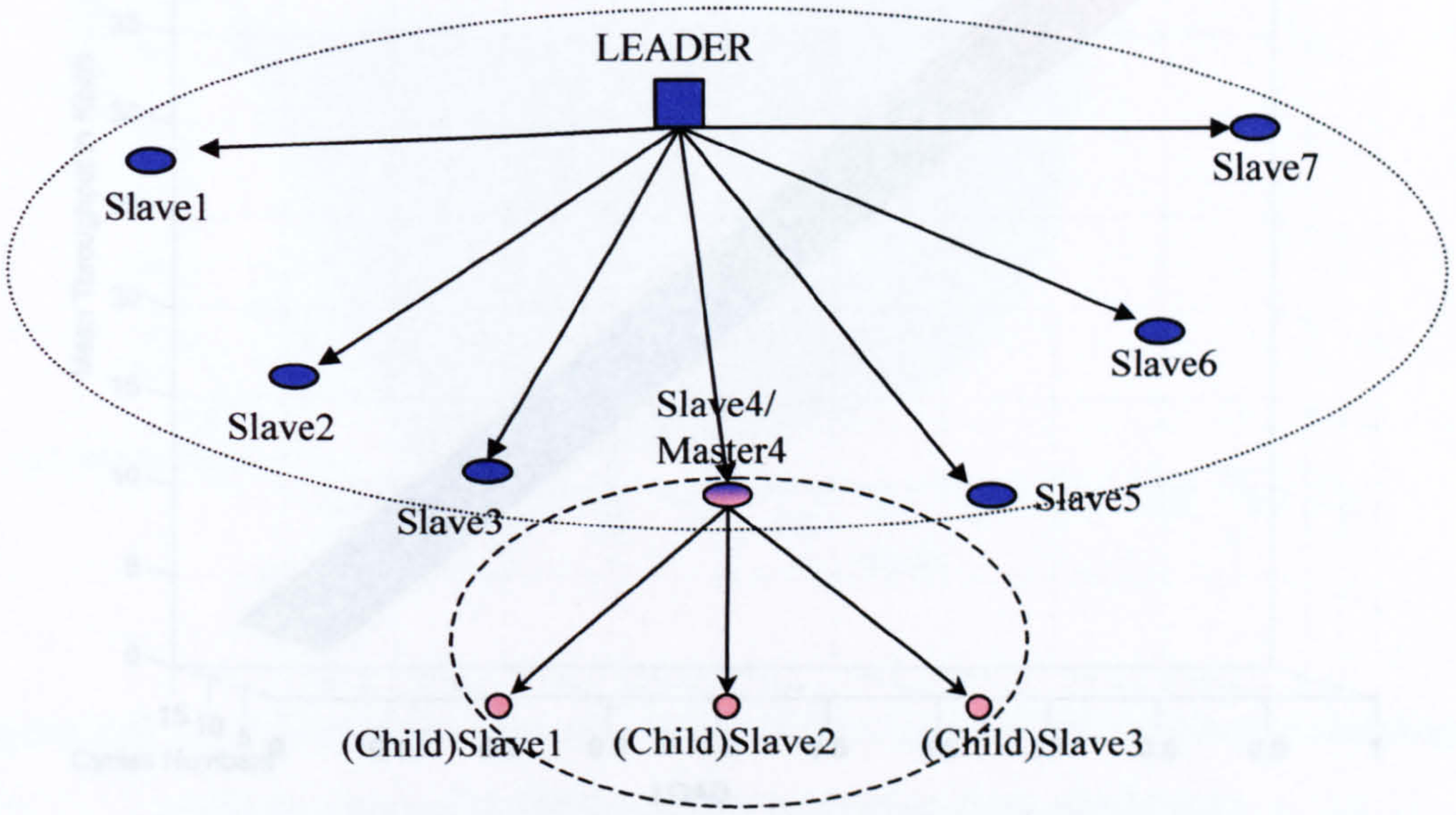


Figure 5.6 Scatternet Topology with two hierarchy of one leader and 7 slaves and one master with 3 (Child)Slaves.

- Throughput: Figure 5.7 shows the three (Child)Slaves throughput results received from the master. It could be noted that the variation of the waiting cycle period does not affect the system throughput. (Child)Slave throughput increases proportionally with the load and reaches a maximum of 48Kbit/s. This maximal value illustrates that the bridge scheduling perform perfectly. Since the Leader needs to share with the master 24 Kbit/s bandwidth (1/7 full load), and at the same time the master distributes the left bandwidth among its three (Child)Slaves which represents a maximum of 49.2 Kbit/s for each (Child)Slave. Hence the bridge scheduling concept only 1Kbit/s bandwidth

demonstrating the absolute harmonization of the master within both piconet.

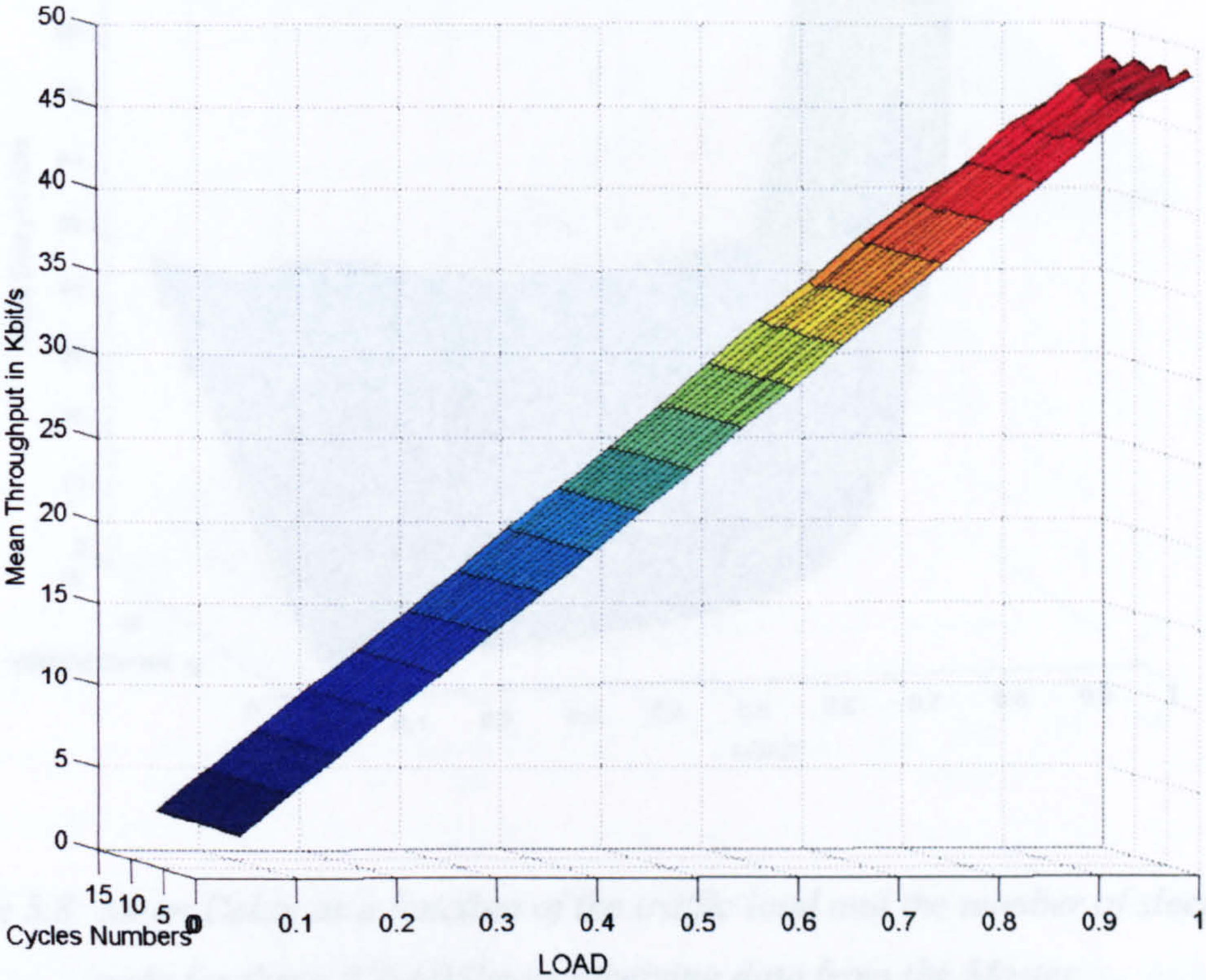


Figure 5.7: Mean Throughput for 3 (Child)Slave devices depending on the Load and waiting cycle number.

- Mean Delay: Figure 5.8 shows the mean delay results for the three (Child)Slaves. In similarity with Figure 4.10 (mean delay within the first hierarchy) the delay increases as the waiting cycle period increases. And above a load of 0.7 the delay rises significantly, while in the first hierarchy the rise occurs for 80% of the full load.

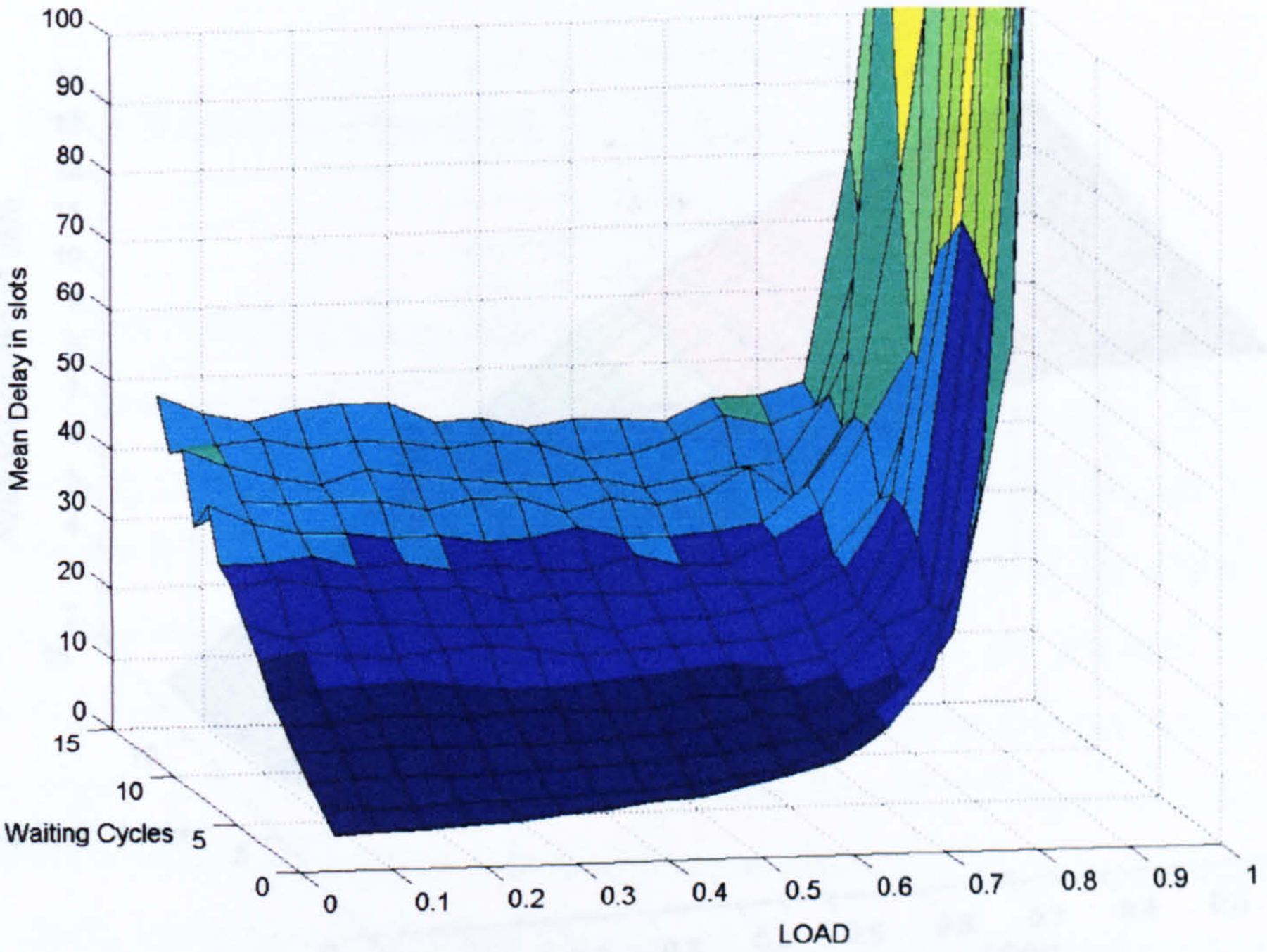


Figure 5.8: Mean Delay as a function of the traffic load and the number of sleeping cycle for three (Child)Slaves receiving data from the Master.

- Power Consumption: The mean current consumption using Pure Round Robin scheme for each slave is around 18.33mA ([(50+40)+10*2]/6). Figure 5.9 compares it with the new scheduling and shows a significant reduction current consumption. The (Child)Slaves experiences a significant reduction in energy consumption especially for low rate traffic. The curve is very comparable to the curves obtained for the first hierarchy in Figure 4.12. By comparing both curves, Figure 5.9 appears more convex while applying long waiting cycle than Figure 4.12. For load between 0.5 and 0.8, this convexity is amplified mainly due to the fact the master stays within the Leader piconet and thus forces its (Child)Slave to listen for data every even slots, which increases the power consumption.

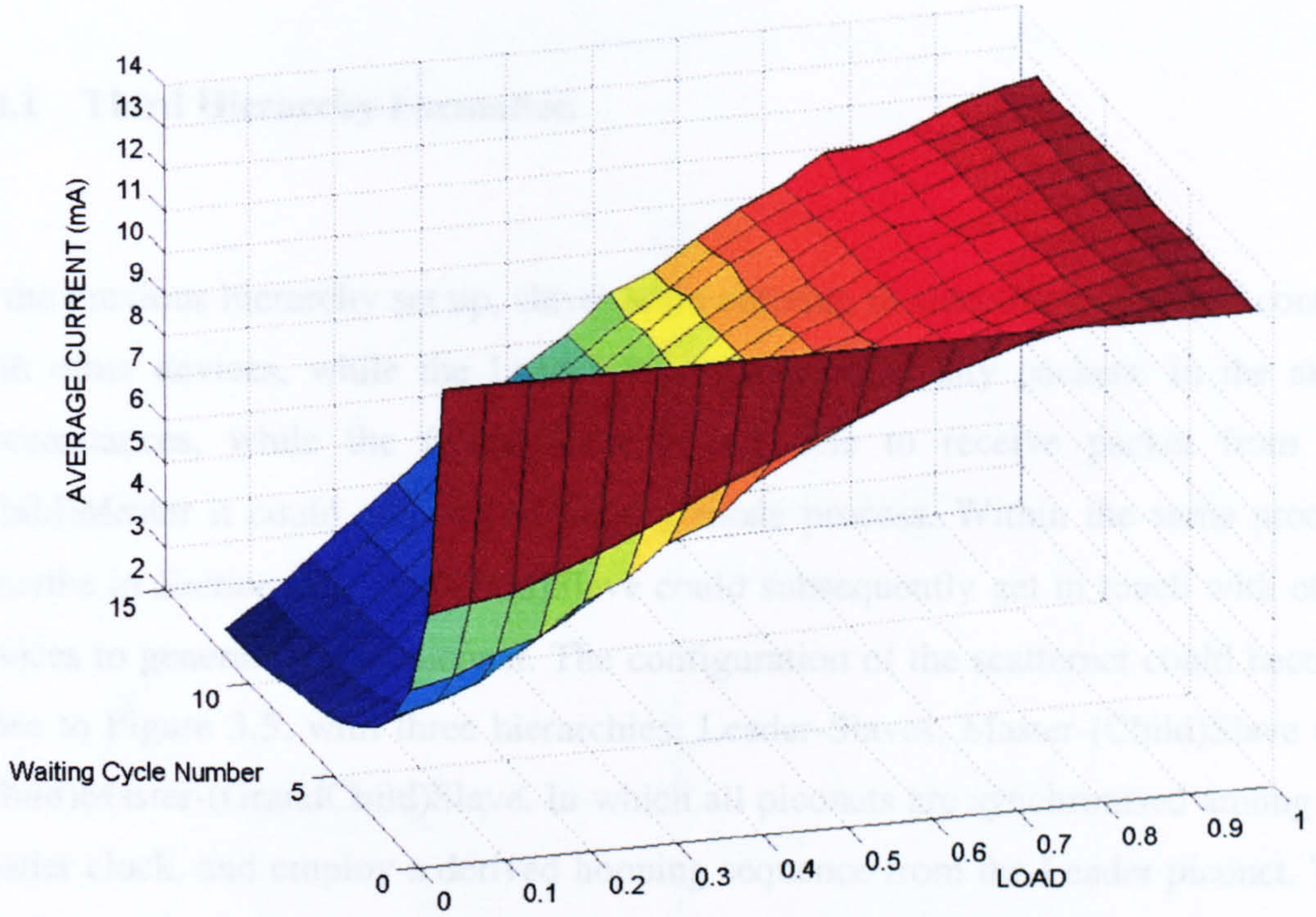


Figure 5.9: Mean Current consumption for the 3 (Child)Slaves within the Master piconet.

From the situations analyzed, some general consideration on the behavior of the studied algorithm and topology could be addressed. As far as fairness is concerned, the network is fair in terms of bandwidth allocated to each node, and reasonable in terms of delays in the packets. The power consumption is reduced and outclasses the normal RR scheme. The implementation of the second hierarchy, and specially the coordination of the bridge, insures a decent Quality of Services within the scatternet.

5.4 THIRD HIERARCHY SIMULATION

5.4.1 Third Hierarchy Formation

In the previous hierarchy set up, slaves were entering in sleeping mode or in contact with other devices, while the Leader is not addressing any packets. In the same circumstances, while the (Child)Slave is not able to receive packet from the (Child)Master it could activate an Inquiry mode process. Within the same process describe in *Section 5.3*, the (Child)Slave could subsequently get in touch with other devices to generate a new piconet. The configuration of the scatternet could become close to Figure 3.5, with three hierarchies: Leader-Slaves, Master-(Child)Slave and (Child)Master-(GrandChild)Slave. In which all piconets are synchronised among the Leader clock, and employ a derived hopping sequence from the Leader piconet. The second and third hierarchies, within the scatternet, apply the AMSQP plus the extra-bit combination to schedule intra-piconet polling and inter-piconet bridge meeting.

The QoS evaluation within (Child)Master piconet has been conducted using the previous simulation model environment, with the third hierarchy expansion. Results are shown and described in the following section.

5.4.2 QoS within the Third Hierarchy

To investigate the performance of the QoS within the third hierarchy, a new scatternet model is shown in Figure 5.10. The Leader, root of the scatternet, sends traffic with identical load λ to its four masters. Each master transmits to its three (Child)Masters a matching amount of load λ, same as the (Child)Master to its three (GrandChild)Slaves.

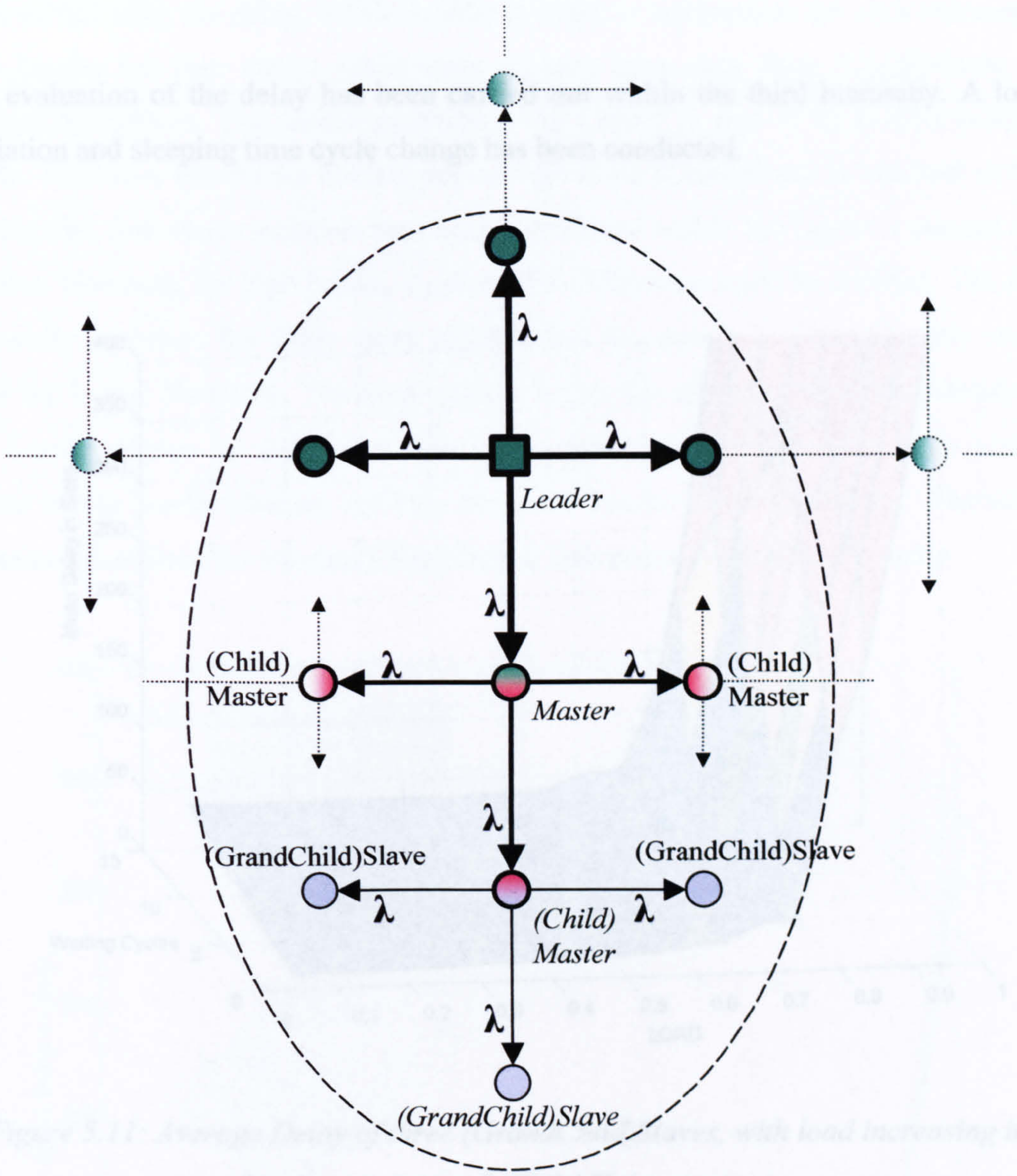


Figure 5.10: Scatternet model all link having load λ, with one Leader, four Masters possessing each three (Child)Masters, themselves having three (GrandChild)Slaves each.

5.4.2.1 *Delay evaluation within third hierarchy*

An evaluation of the delay has been carried out within the third hierarchy. A load variation and sleeping time cycle change has been conducted.

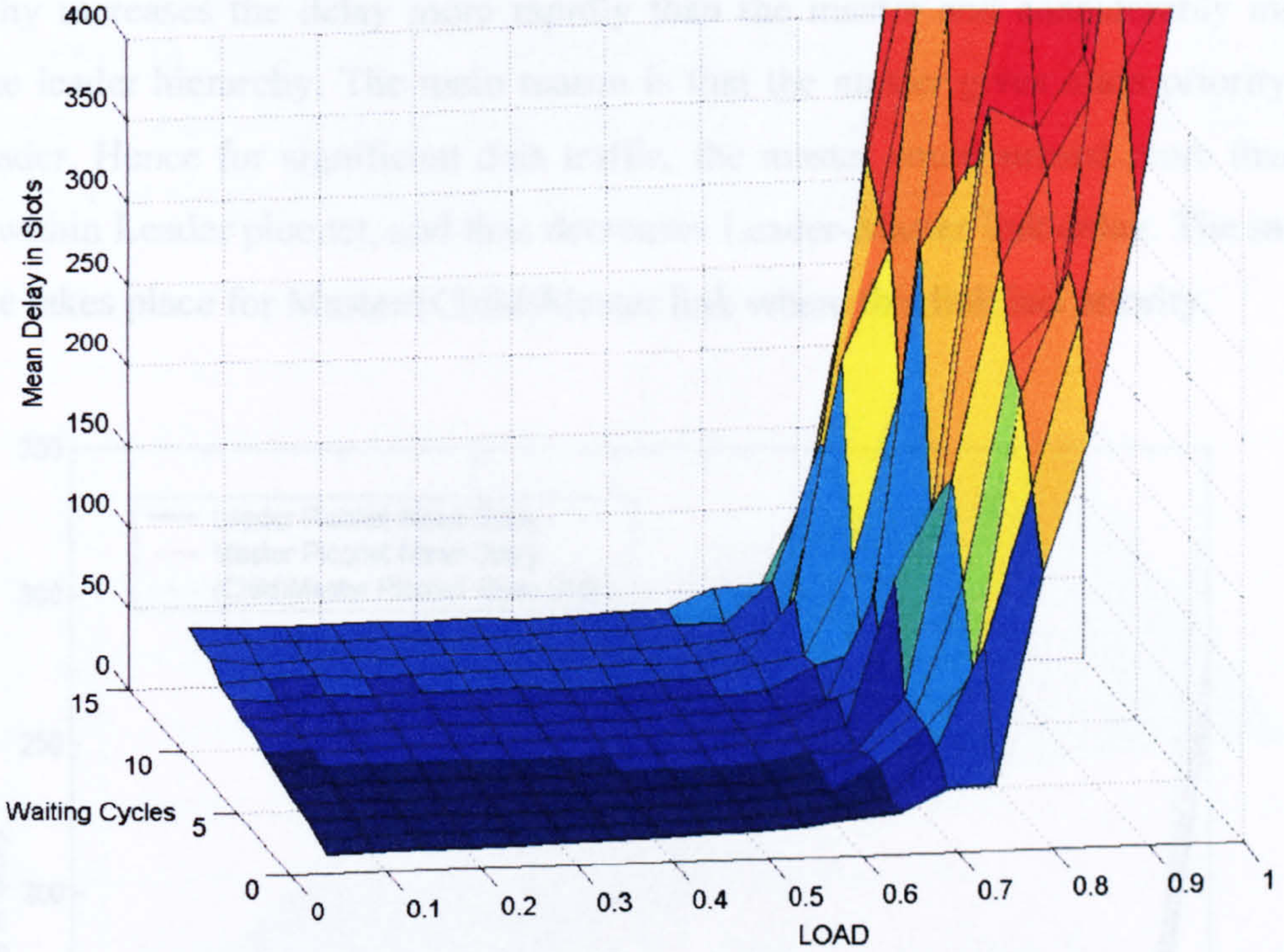


Figure 5.11: Average Delay of three (GrandChild)Slaves, with load increasing in Leader, master and (Child)Slave piconets.

From the results shown in Figure 5.11, it could be noted that the third hierarchy has virtually the equivalent delay characteristic as its two previous hierarchies (see Figures 4.10 and 5.8). This shows that the schedule performs very well in term of consistency. Hence presence of a device within the first, second or third hierarchy has practically no impact on its traffic link with its master.

To compare the three hierarchies delay variation, a simulation, using a sleeping cycle time fixed to 6 cycles, has been conducted in which the load increases at the same rate for all links.

Figure 5.12 shows the results of the three hierarchies delay. From low traffic load to 70% of full load, the delay, within Leader piconet, is comparable for each hierarchy. The Leader has four slaves while other masters have only three (Child)Slaves or (GrandChild)Slave. The master assimilates the Leader as part of its piconet polling cycle. Therefore the master cycling period with three (Child)Slave is identical as the leader with four slaves and thus both delays produced within the Leader's piconet are similar. However, for high traffic, a perceptible difference could be notified. The last hierarchy increases the delay more rapidly than the master and considerably more than the leader hierarchy. The main reason is that the master gives main priority to the Leader. Hence for significant data traffic, the master could subsist more than a frame within Leader piconet, and thus decreases Leader-Master link delay. The same practice takes place for Master/(Child)Master link where this link has priority.

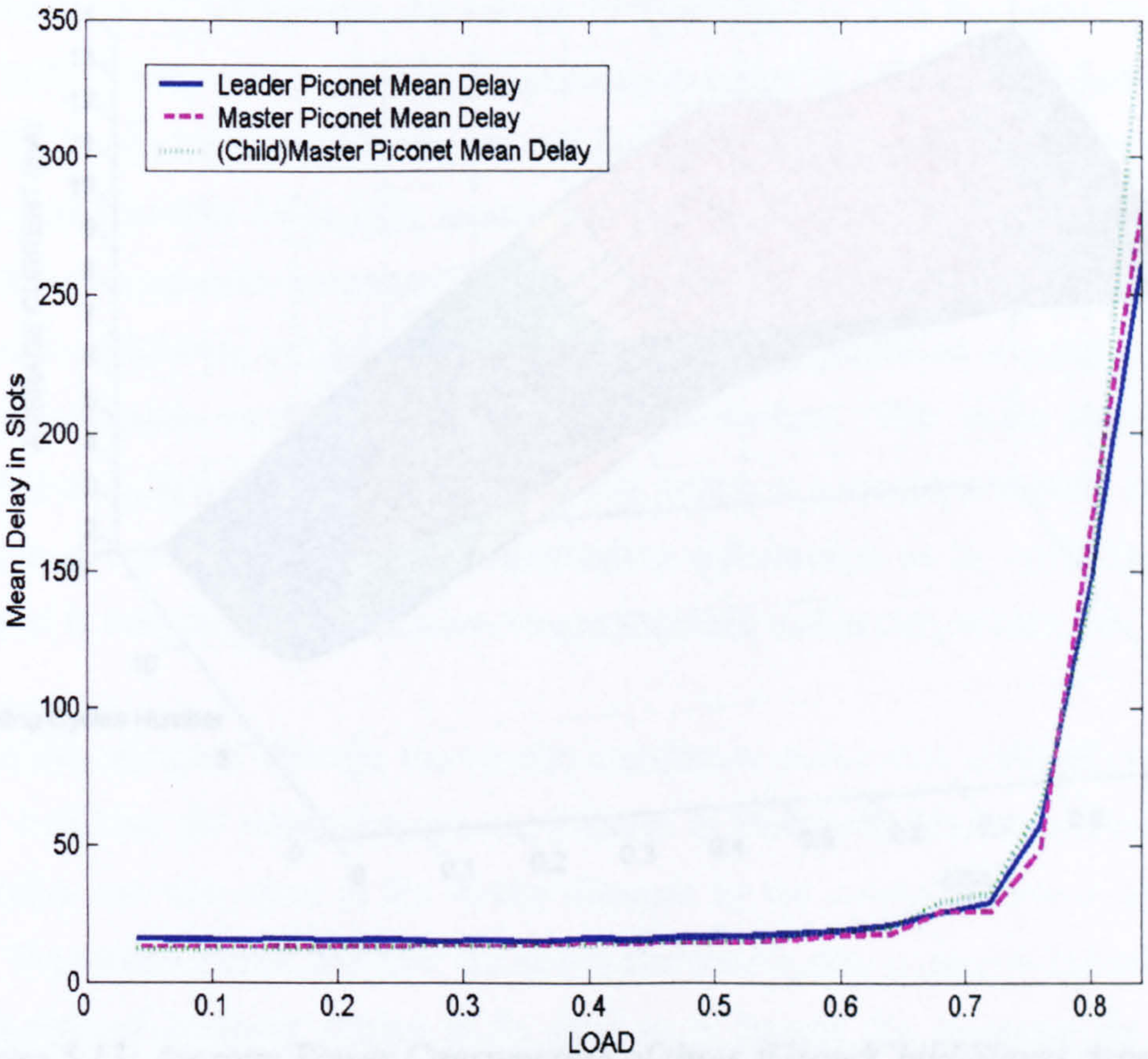


Figure 5.12: Mean Delay of the three Hierarchies applying the same load and sleeping period.

5.4.2.2 Third Hierarchy Power Consumption

The Power consumption within the third hierarchy has been computed applying the equivalent circumstance as described for the delay evaluation in *Section 5.4.2.1*. The results, presented in Figure 5.13, demonstrate that the power consumption of the third hierarchy decreases in an equivalent proportion as its two parents hierarchies. However for high traffic, the power consumption is subject of new reduction. The reason evoked in *Section 5.4.2.1* concerning the priority given to the Leader once again reveals a small decrease for high load traffic. The delay increases, and thus less packets are sent within the link and thus less power is consumed.

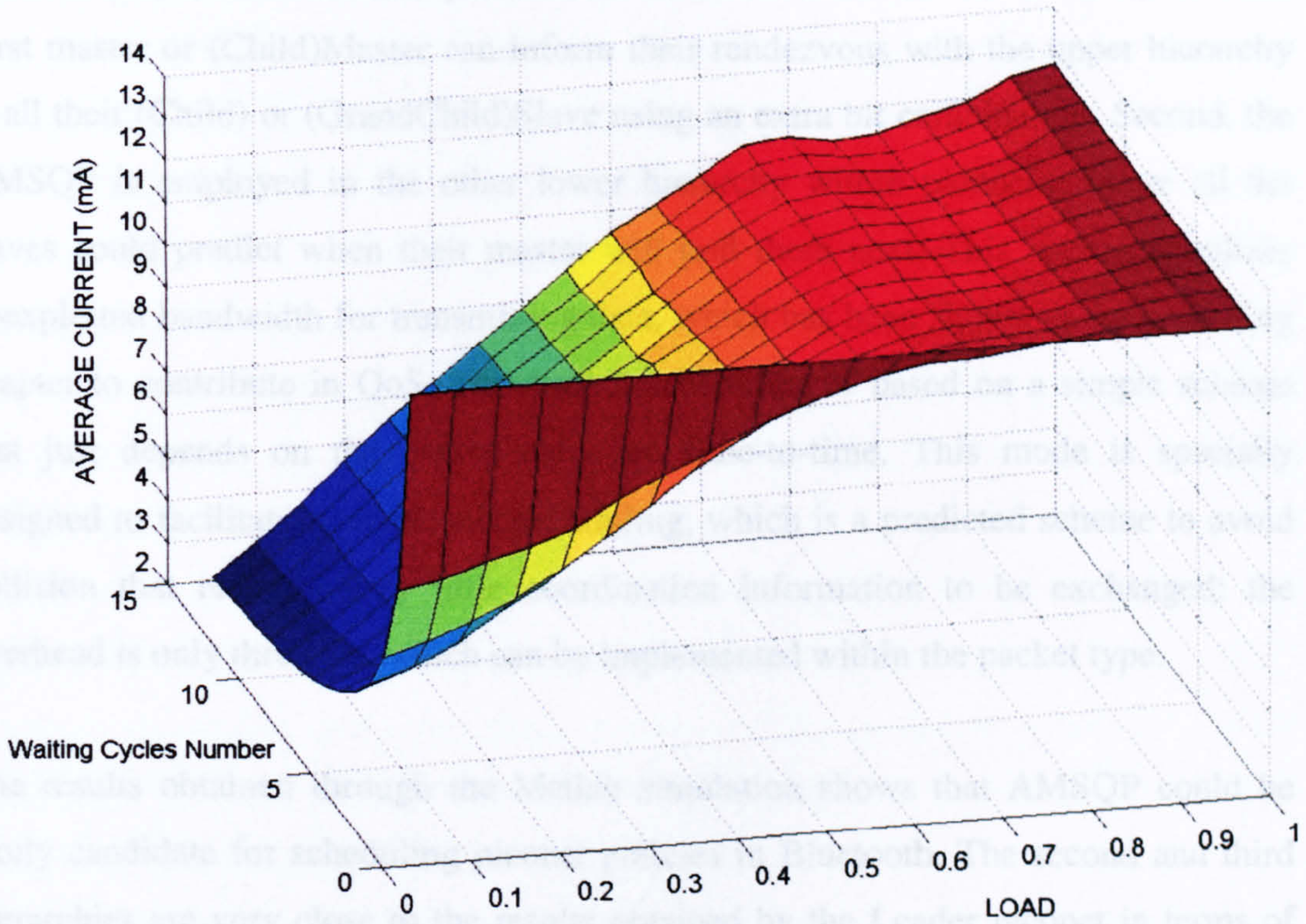


Figure 5.13: Average Power Consumption of three (GrandChild)Slaves depending on the Load and sleeping time period.

5.5 CONCLUSION

The introduction of Bluetooth scatternet enables diverse way of exploiting Bluetooth network. Its connectivity could be wider and more flexible. These characteristics present a technical challenge with respect to network formation and scheduling of traffic within Bluetooth network. A presentation of recent research studies of how devices should schedule their time between successive bridge exchanges has been given, with different technique applying a rendezvous point scheduling issues.

This chapter has presented a new bridge-scheduling scheme as an extension of the precedent intra-piconet scheduling AMSQP scheme. The fact that the AMSQP generates a predicted scheme provides an excellent effect on the lower hierarchies. First master or (Child)Master can inform their rendezvous with the upper hierarchy to all their (Child) or (GrandChild)Slave using an extra bit combination. Second, the AMSQP is employed in the other lower hierarchy within piconets, where all the slaves could predict when their master will poll them next. This approach utilizes unexploited bandwidth for transmitting data, which has been shown in the preceding chapter to contribute in QoS. The overall scheduling is based on a simple scheme that just depends on the queue status on time-to-time. This mode is specially designed to facilitate inter-piconet scheduling, which is a predicted scheme to avoid collision that requires very little coordination information to be exchanged: the overhead is only three bits which can be implemented within the packet type.

The results obtained through the Matlab simulation shows that AMSQP could be likely candidate for scheduling piconet policies in Bluetooth. The second and third hierarchies are very close to the results obtained by the Leader piconet in terms of QoS. The results shows also that the end-to-end packet delays may be minimized if the number of cycles is chosen to be close to 0, though this increases the power consumption. Hence depending on the purpose of the scatternet, a compromise between power and delay must be established. This will be developed in the next chapter using examples of different commercial use.

This chapter deals only with the scatternet architecture, but it becomes obvious that the role of bridge could become more crucial to the overall performance of the scatternet if it could control Bluetooth packet forwarding and hence routing through the overall scatternet. This is also the focus of the next chapter.

CHAPTER 6
SCATTERNET ROUTING AND VIABLE APPLICATION

6.1 INTRODUCTION

In the previous chapter, it has been shown that a Bluetooth piconet could interconnect with other piconets to form a larger network. Therefore, in order to facilitate communication within the scatternet, during formation structure, a routing schedule needs to be established. The scatternet routing should find paths in a fix or dynamic network. The bridge node must arrange to be in active mode not just to receive packet addressed for it, but also to participate in any higher-level routing and control protocols. The requirement of collaboration between power saving and routing protocols is particularly delicate in the case of multi-hop wireless networks, where devices must forward packets for each other and thus consume power.

Before describing a new routing scheme, some observation about Bluetooth characteristics need to be re-visited and clarified.

Firstly, one significant weakness of Bluetooth is that slaves within a piconet do not have the knowledge of their piconet neighbours. Slaves are just aware of master services and address; only the master is aware of the capability specified by other devices. In addition, nodes should be aware of the service offered by other nodes in order to exchange data between each other.

Secondly, as described in (Prabhu *et al [2002]*), Bluetooth devices are expected to be small and resource constraint, which means that complicated processing or large tables of information for routing at the nodes is not suitable.

Thirdly, the scatternet must accept the notion that devices may move from one piconet to another, such that every time a new device enters or leaves a piconet, the master should quickly inform to the entire scatternet of its movement. Therefore devices, aware of the location and of the services offered by other devices within the

scatternet, could exchange communication with other devices by forwarding packets, or by creating a direct link between each other.

This chapter describe a new packet delivery between any pair of devices within the scatternet and is organized as follow: A new forwarding packet method is explained in *Section 6.2* to enable slave communication with another slave through the master. Since the scatternet network could be large, a new packet addressing system differing from Bluetooth standard addressing is presented. The fact that the Leader or bridges (masters) forwards packets for other devices could generate a bottleneck within the system. To elucidate this dilemma, *Section 6.3* explains how both Slaves could be directly in contact with each other and could form a new piconet. A comparison of both methods (forwarding data or creating new piconet) is simulated in *Section 6.3.1*. Once the routing process of the entire scatternet formation has been explained, a numbers of new complete scatternet applications could be envisaged. *Section 6.4* studies two examples of new scatternet access networks. A scatternet lecture room theatre is presented; followed by a scatternet office network. And finally, *Section 6.5* concludes this chapter.

6.2 FORWARDING PACKETS INSIDE THE SCATTERNET

6.2.1 New Layer creation to Allow Forwarding Data

Within the Bluetooth standard, masters are aware of their piconet members' characteristics. The service discovery protocol information of the piconet contains the service of all slaves present within the piconet. Since bridges (master in one piconet and slave in the upper hierarchy) have previously established a link between two piconets, they could forward the memorised piconet information to their upper hierarchy: Master or Leader. Hence the upper master establishes the new memory of all devices services under it, which facilitate the Leader in becoming the head of the scatternet server. Every time a piconet configuration is subject to node variation, the

message is updated when the two hierarchies (Child)Master/Master or Master/Leader meet.

Although knowing the presence of other devices within the piconet, a direction-finding route is required to allow the forwarding of packets. For example a slave cannot use the services offered by another slave. Such communication can be implemented only via the master. Thus, to forward data through the master, a new Layer header in the payload of the packet is needed (Bhagwat *et al [1999]*) (see Figure 6.1).

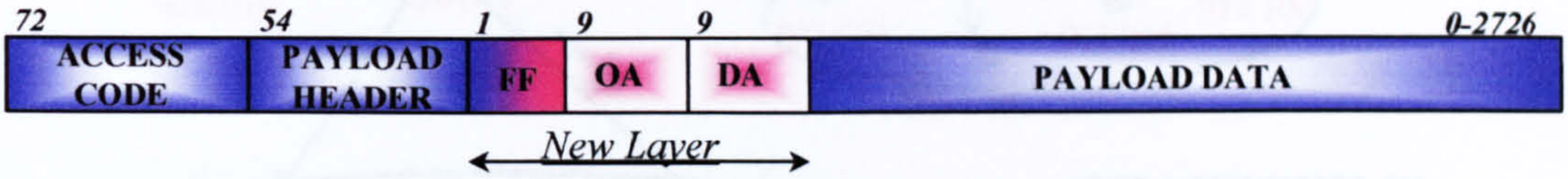


Figure 6.1: Relevant fields in the Bluetooth packet with a new layer.

The general Bluetooth baseband packet format is described in *Section 2.3.2*, with the access code and the packet header. From this format, to develop the new routing, a new layer has been added containing a forwarding flag (FF) of 1 bit long. The value of the flag FF=0 means that the payload of the packet is intended for the master, and it does not request to be forwarded. If FF=1, the packet needs to be forwarded, and the Bluetooth device in question may be either in the piconet or part of the scatternet. Hence within a multi-hop communication, a new layer is added in the Payload for routing process. The device that issue the request is assigned to be the source with Original Address (OA) in the routing connection, and the supplied devices address is assigned to be the Destination Address (DA). Every node is identified by a specific bluetooth address (48bits). Since this address is too large for the routing purpose, a local identifier is utilized. For a full scatternet topology represented by *Figures 3.7*, each device could now be assigned with an innovative 9 bits address. The first 3 bits assigns the seven different masters according to their AM_ADDR, the second set of 3 bits represents the (Child)Masters and the last 3 bits identify the (GrandChild)Slaves. Hence every device has a unique 9 bits address within the scatternet, where Slaves addresses are linked to their parents address.

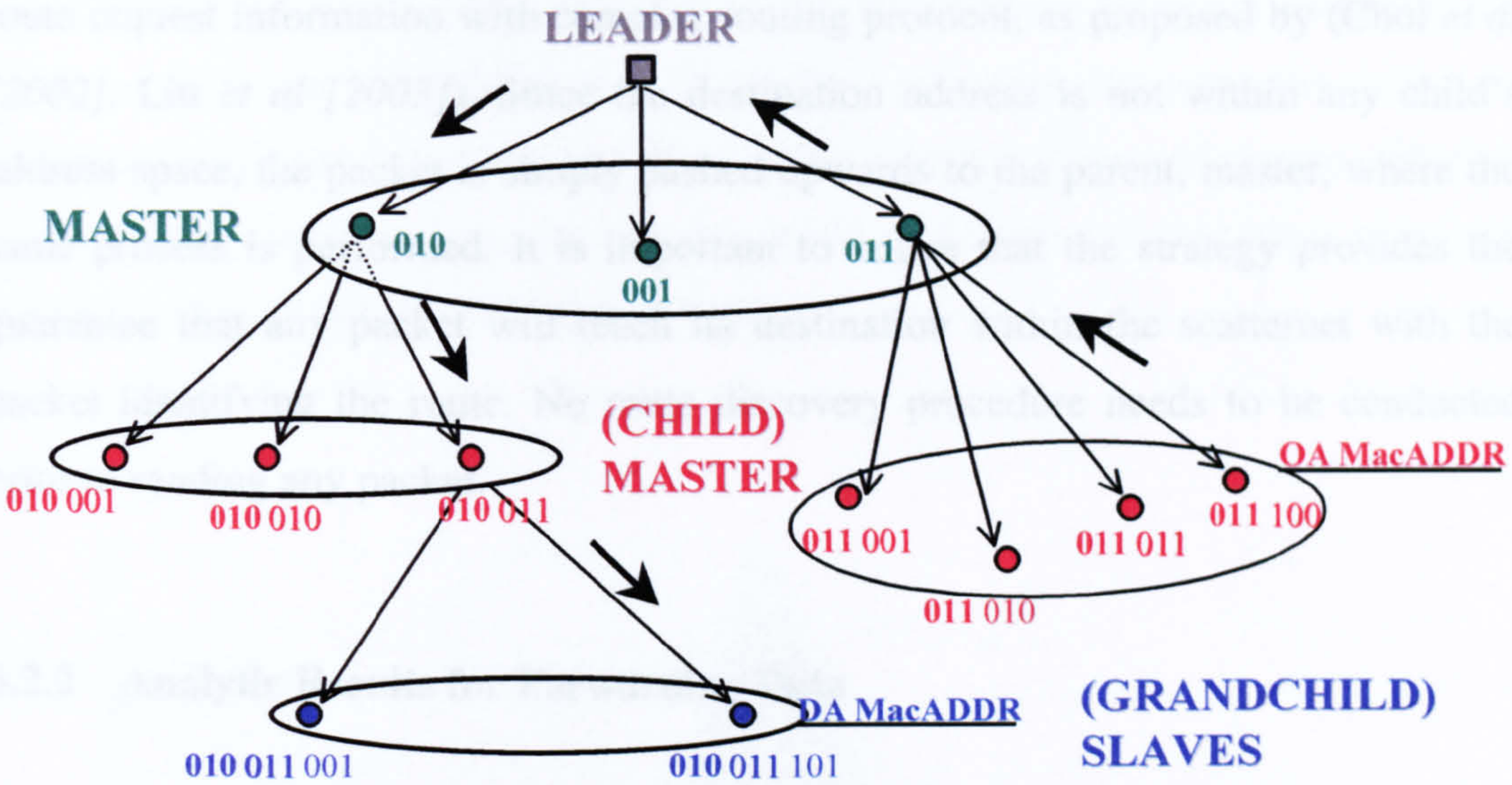


Figure 6.2: Forwarding packets with a new routing schedule using new addressing nodes.

Therefore, when the master receives packets, if FF=1, the two new Layer processor puts the payload in a new packet and sends it to the corresponding slave. If the node Destination Address (DA) packet is inside the piconet the master will identify the DA from one of its children and send the request directly to the specific slave. Else the master will operate as a relay, and will forward upwards the packet to its parent.

As an example, illustrates by Figure 6.2, two devices try to forward packet to each other. At the beginning $Master_{(3)}$ receives a packet with FF=1 and DA=2,3,5 (010,011,101), meaning that the destination of the packet is for (GrandChild)$Slave_{(5)}$ child of (Child)$Master_{(3)}$ and grand child of $Master_{(2)}$; and with OA MacAddr 3,4,0 leading to a packet sent from an original source of (Child)$Master_{(4)}$ child of $Master_{(3)}$.

Consequently $Master_{(3)}$ cannot recognize the destination of the first three bits, and forwards the packet to its parent: the Leader. Since the Leader could correspond to any device, it identifies the destination and forwards the packet to its $Slave_{(2)}$/$Master_{(2)}$; which recognises the packet, and forwards to its (Child)$Master_{(3)}$. The packet finally arrives at (GrandChild)$Slave_{(5)}$ after five hops.

By carefully assigning addresses to corresponding hierarchy position, the topology construction mechanism has simplified packet forwarding and avoided the need for route request information with complex routing protocol, as proposed by (Choi *et al [2002]*, Liu *et al [2003]*). Since the destination address is not within any child's address space, the packet is simply pushed upwards to the parent, master, where the same process is performed. It is important to notice that the strategy provides the guarantee that any packet will reach its destination within the scatternet with the packet identifying the route. No route discovery procedure needs to be conducted prior to sending any packet.

6.2.2 Analytic Results for Forwarding Data

To evaluate the impact of the forwarding process routing, an analytic simulation of the new role of the bridge is considered. In the first case, only the first and second hierarchies are applied. Within the model the master forwards the packets from the Leader to its (Child)Slaves. The Leader sends a full load to all its Slaves (Leader has full load to send to its Slaves/Masters) and the information received by the master is forwarded to its (Child)slaves. Results are shown by Figure 6.3; the maximum throughput forwarded by the master occurs when the leader has only two slaves and the master only one (Child)Slave. The optimal throughput reaching the (Child)Slave from the Leader achieves around 85Kbit/s. Since the bridge needs to divide its time between receiving packets from the Leader, and transmitting them over to its (child)Slave, operating during half of its time in mutual piconets, provides the highest throughput.

Moreover from the results, the evolution of the forwarded throughput from the Leader to the (Child)Slave, reaches its highest traffic when the numbers of (Child)Slaves is one less than the number of slaves within the leader's piconet. As explain in *Section 5.3.2*, this is due to the slot coordination between master and Leader. While the master communicates to every (Child)slaves, at the same time the Leader communicates with all others slaves, and then both communicate to each other.

One deficiency of the forwarding packet scheme is that if the Leader continuously sends data to the master with a high rate, the master does not have the chance to re-communicate the data to its (Child)slave. Hence the master acts as a bottleneck by receiving far more information than it can forward which increases the delay. As illustrated by Figure 6.3; a configuration with a Leader a slave and master with four (Child)Slaves, in that case, the master could only forward 21Kbits/s for each (Child)Slaves instead of 34Kbit/s achievable.

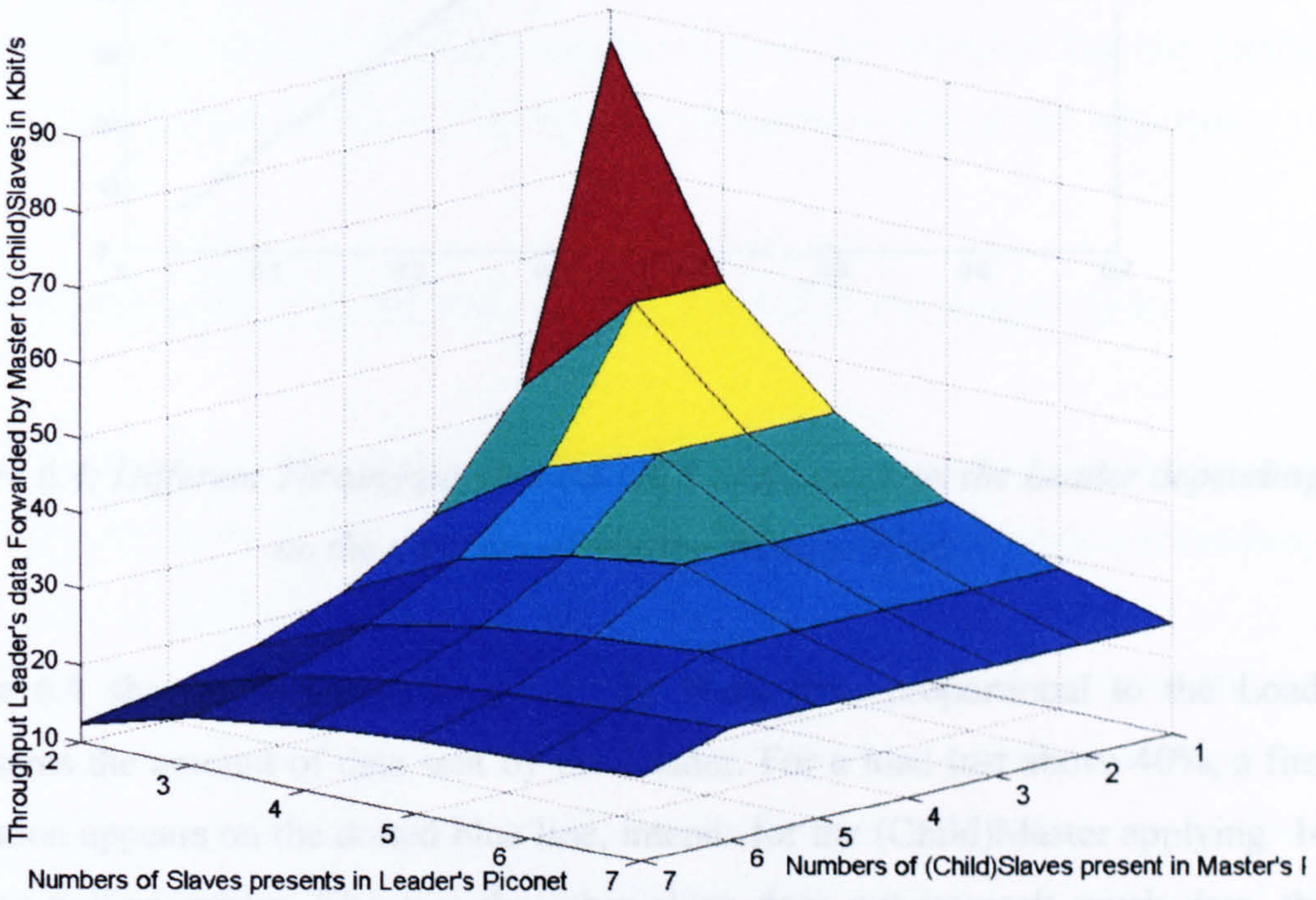


Figure 6.3: Throughput received by (Child)Slaves from the Master forwarding data from the Leader with full rate.

For the third hierarchy forwarding packet evaluation (Leader to (GrandChild)Slave), a configuration of a Leader, two Slaves, a (Child)Master and a (GrandChild)Slave (GSC) is simulated through Matlab. The evaluation calculates the maximum throughput reachable for the GCS from the Leader for varying load and sleeping cycle while low traffic data is sent from the Leader to the other slave.

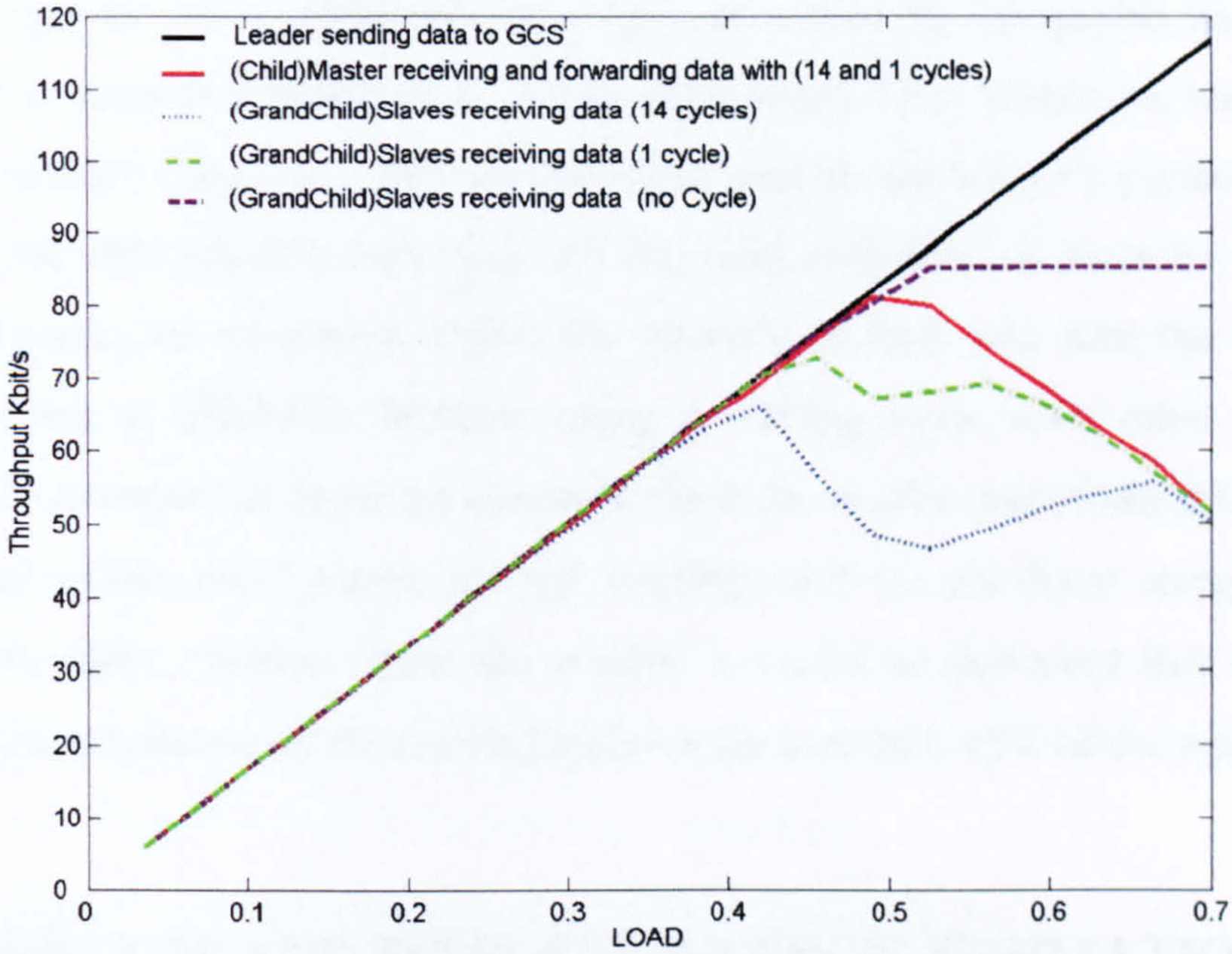


Figure 6.4: Different Throughput that the GCS could get from the Leader depending on the numbers of waiting polling cycles.

Figure 6.4 shows the results in which the black line, proportional to the Load, represents the amount of data sent by the Leader. For a load just above 40%, a first saturation appears on the dotted blue line, intends for the (Child)Master applying 14 waiting polling cycles. The fact the other slave does not transmit much data, the benefit of its sleeping time offers more time for the master to communicate with the Leader. While the Leader sends data, the master stays within the piconet; hence it is difficult for the (Child)Master to predict exactly when the master will be polled back. Thus the (Child)Master has to remain on the master piconet waiting, instead of forwarding data to its own piconet, which produces a loss within the scatternet traffic.

For a very low cycle of 1 polling period represented by the green dash line, the Leader must contact the other slave more often, which gives more time for the master to communicate within its piconet. This is the main reason why the saturation turns up 5% of the load later than using a 14 cycles length at load 0.45.

For a pure round robin combination, i.e. the leader must alternate communication with both slaves (no waiting polling cycle), as shown by the purple dash line the saturation is around 85Kbit/s (half of the total load). This saturation was expected since the master must stay only for two slots within the leader's piconet and thus forwards the data on next two slots. As the load continues to increase, the master remains exactly for two slots within the leader's piconet and thus the throughput stays constant at 85Kbit/s. Whereas using a waiting cycle (two other cases), the throughput decreases as the load increases since the master and (Child)Master spend more time within their parent piconet waiting, and do not have enough time to forward the data. Hence, from the results, it could be deducted that the Leader should transmit data to its (GrandChild)Slaves for less than 45% of the total load.

6.3 DIRECT SLAVE-TO-SLAVE PICONET FORMATION

The traffic between two separated devices could be extensive, increasing the delay at the forwarding node side, in which the Leader and masters act more as a forwarding packet nodes than as an information source.

Therefore, a new strategy is required such that a device requiring communication with another device in its range could create an extra piconet or join an existing one as described in (Cordeiro *et al [March 2004]*). By forwarding information, any slave could know which scan interval T_{scan} and the scan window $T_{w\ page\ scan}$ is employed by the other device. Thus, two devices intending to connect forward their agreement through POLL packets (126 bits packet used on return link for information to the source and required confirmation from the destination) and include the value of the T_{scan} interval along with the frequency used. The fact that the scatternet is synchronised to the Leader, the devices just need to switch their frequency to match the corresponding scan window. By doing so, the connection avoids the long inquiry process (average of 10 second connection) and enters directly in a Page process, which takes much less time (see *Section 5.3.1*). In the case both devices are slaves and a new piconet is created. The new piconet is a priority for the Leader's clock and frequency hopping sequence (as defined previously in S*ection 3.3.2*).

In this manner, the packets do not need to be forwarded by the master anymore.The device could establish any device part of the scatternet. Furthermore this novel communication architecture enables for not only direct slave-to-slave communication but also serves as multi-slaves-masters communication. Leader, Masters, Slaves or (Child)Slaves could directly contact the (GrandChild)Slave, or (Child)Master etc... Hence sharing a group communication within the piconet where every node could speak with any node in its range.

At that point some nodes could act as Slaves in two different piconets, and thus will have to schedule their time between both piconets. Following the same process as bridge nodes, using the AMSQP scheme (see *Section 5.3.3*), devices could predict when they will be polled next and the new gateways called *multi-slaves* will give priority to the highest master hierarchy they belongs to.

Consequently if a (Child)Slave requires direct contact with a Slave, the (Child)Slave should be rehabilitated into (Child)Slave in both piconets, while the other slaves become master. Information of the movement will be brought to the Leader, and the multi-slaves will be affected with two addresses: one from its initial piconet and second address from its new piconet. Reliability is hence achieved within the new scatternet development applying in some cases an alternative path between two nodes (Zhang *et al [2004]*, Cordeiro *et al [June 2004]*).

6.3.1 Comparison between forwarding data and creating a new piconet

To compare the forwarding packet method with the creation of a new piconet process, a simulation of both functionalities is considerate here. As shown in the left side of Figure 6.5, only $Slave_{(1)}$ transmits data to the Master, $Slave_{(2)}$, $Slave_{(3)}$ and $Slave_{(4)}$ with the master forwarding the packets to the relevant slaves. The simulation applies the same load for all uplinks and downlinks Master/Slave connections, except for $Slave_{(1)}$ which has an uploading link four times higher.

In the multi-slaves formation methods, as shown by the right side of Figure 6.5, $Slave_{(1)}$ creates a new piconet to transmit data to other slaves. To create the new piconet faster, as described previously, the master forwards the paging information of

all its slaves to $Slave_{(1)}$, and by applying the page procedure, all other slaves get connected to the new piconet.

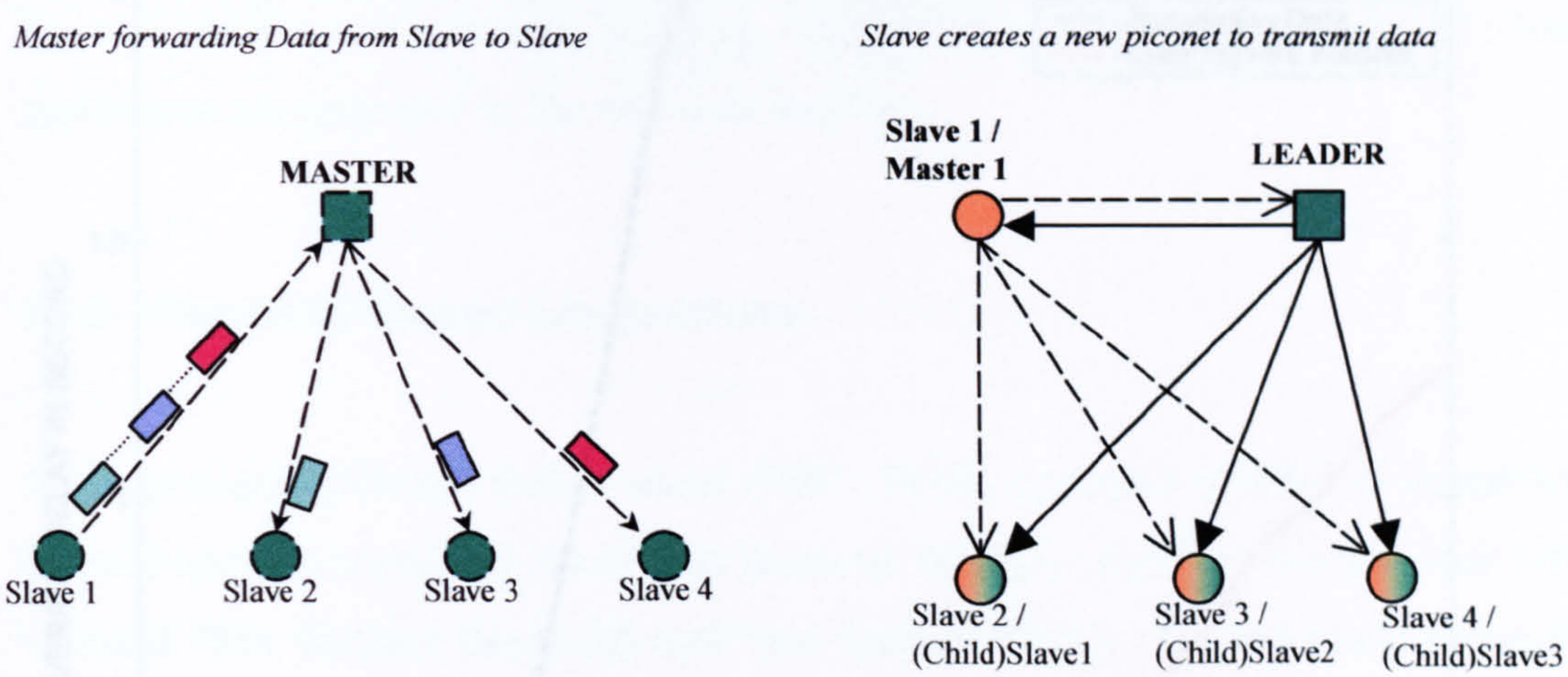


Figure 6.5: Two different ways for a slave to transmit data to others slaves either forwarding data through the master or creating a piconet.

A simulation is proceed for both methods but does not take into consideration the time involve for the creation of the new piconets as it could differ from few slots to few seconds. The simulation computes the delay from the instant a packet is created by $Slave_{(1)}$ to its reception by one of the three slaves or (Child)Slaves.

From results shown by Figure 6.6, after a quarter of the full load, the average delay of the forwarding method, dotted blue line, increases considerably to impractical levels. While $Slave_{(1)}$ increases its link data transfer rate considerably, the master must spend supplementary time with the others slaves, and thus produces a significant delay increase. Represented by the red curve, the extra-piconet formation delay starts rising notably (but not unacceptable) after half load traffic, in proportionality with the Load. As shown by Figure 5.7 in *Section 5.3.4*, the delay rising point, within Leader piconet of 4 slaves, is reached just after the load value of 0.7 and not after 0.5 as shown by Figure 6.6. Hence for identical configuration a noticeable variation of the delay is shown. This difference is explained from the fact that (Child)Slaves allocate priority schedule to the upper hierarchy: the Leader; i.e. a

slave, with two identical poll slot schedule time from two different piconets, favours its Leader.

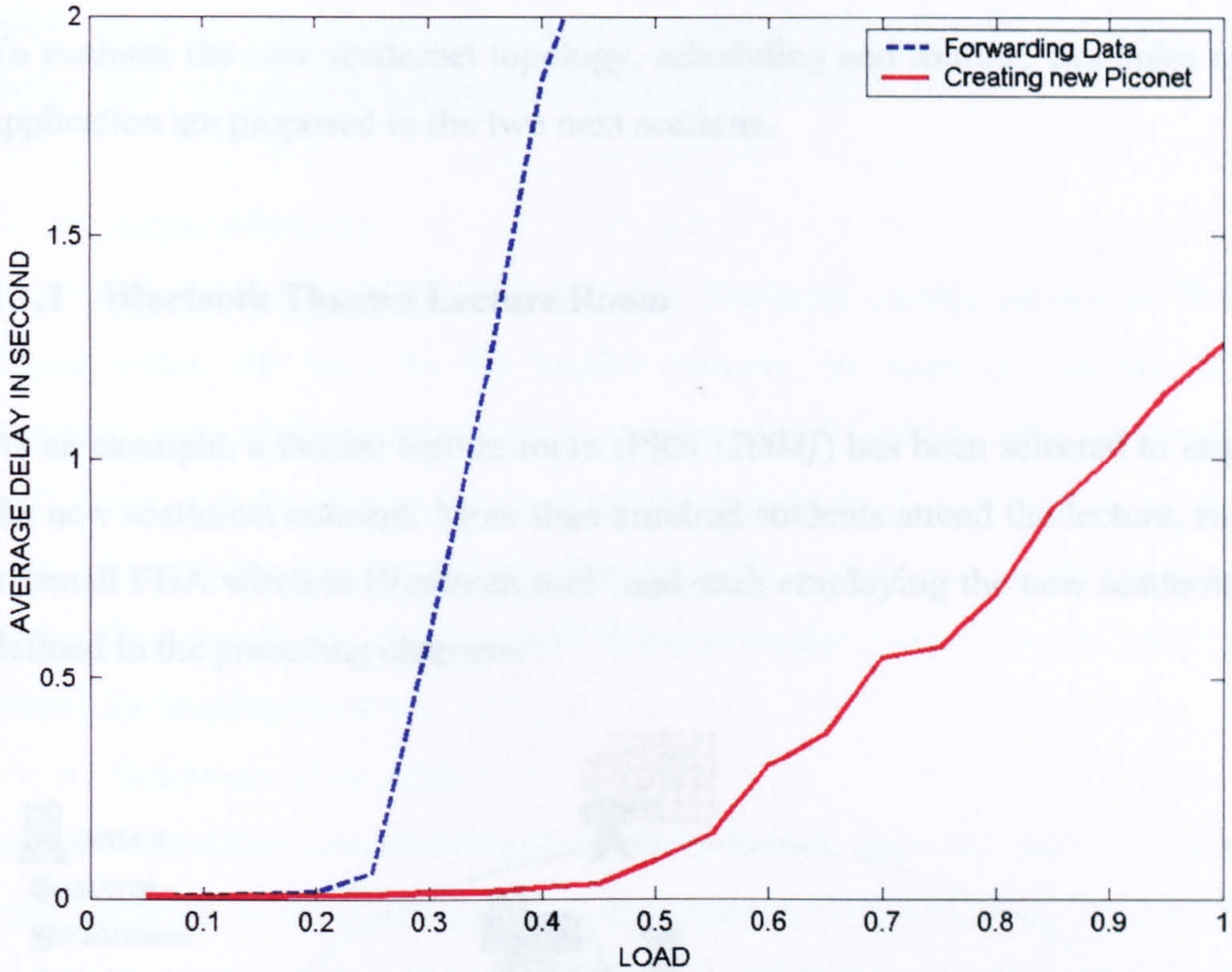


Figure 6.6: Average Delay in second for transmitting the same amount of Load using forwarding transfer or creating a new piconet.

This comparison shows that the system fairness favours the Leader throughput as proposed in *Section 5.3.3*. However the creation of the new piconet improves significantly the delay compared to the forwarding method. Hence the Leader should keep forwarding data information for a load below 0.25 (43Kbit/s) and should recommend to the slave to switch to a new piconet for a higher data load transfer.

1.1 ANALYSIS OF THE NEW ENTIRE SCATTERNET PROTOCOL

To evaluate the new scatternet topology, scheduling and routing, examples of viable application are proposed in the two next sections.

1.1.1 Bluetooth Theatre Lecture Room

As an example, a theatre lecture room (PRS *[2004]*) has been selected to implement the new scatternet concept. More than hundred students attend the lecture, each with a "small PDA wireless Bluetooth tool" and each employing the new scatternet mode defined in the preceding chapters.

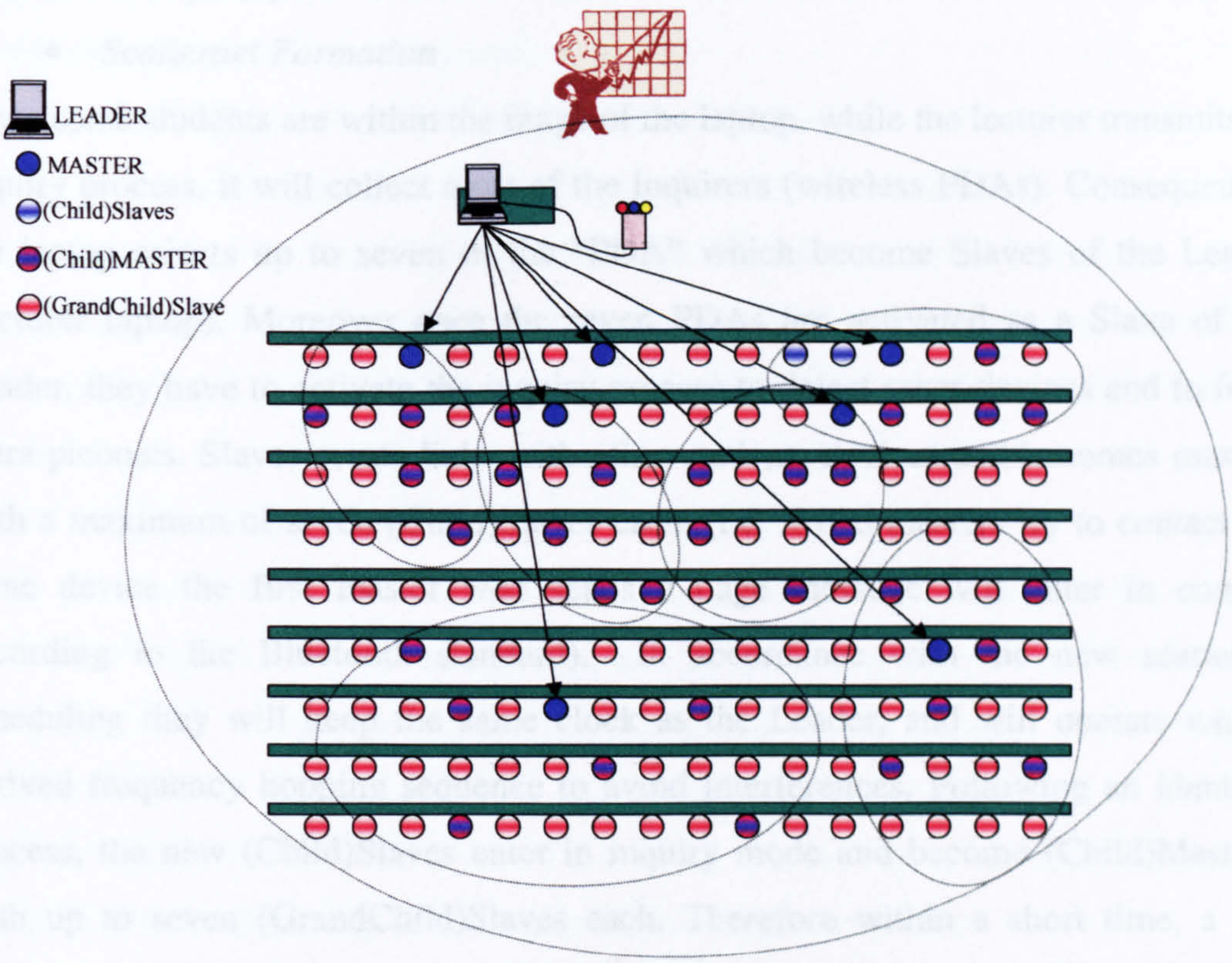


Figure 6.7: Theatre Lecture room with students PDAs acting as slaves of the Leader lecturer Laptop.

Figure 6.7 shows this typical theatre room, small circles representing the students listening to the lecture with their *PDAs* connected together to form a scatternet. The lecturer represents the Leader of the scatternet via its Laptop. Some arrows show the earliest students connected to the network that represents the Slaves/Masters of the Leader. The process of the scatternet establishment is following:

- *Inquiry Process*

At the start, the lecturer, with help from his Bluetooth laptop, queries all Bluetooth devices within the room by the Inquiry process. As soon as students enter the classroom, they activate their special "wireless PDA", which turns on the "inquiry scan" mode. As described in more detail in *Section 5.3.1*, a device in inquiry scan could only become slave within a piconet, while a device in inquiry becomes the master. Hence only the laptop could become master, since it is the only devices initiates the inquiry process.

- *Scatternet Formation*

Since some students are within the range of the laptop, while the lecturer transmits an inquiry process, it will collect most of the inquirers (wireless PDAs). Consequently, the laptop selects up to seven of the "PDA" which become Slaves of the Leader (lecturer laptop). Moreover once the seven PDAs are activated as a Slave of the Leader, they have to activate the inquiry process to detect other devices and to form extra piconets. Slaves create links with other students devices and becomes masters with a maximum of seven (Child)Slaves each, (i.e. if two masters try to contact the same device the first master who sends a page message will enter in contact according to the Bluetooth standard). In accordance with the new scatternet scheduling they will keep the same clock as the Leader, and will operate with a derived frequency hopping sequence to avoid interferences. Following an identical process, the new (Child)Slaves enter in inquiry mode and become (Child)Masters, with up to seven (GrandChild)Slaves each. Therefore within a short time, a full scatternet is established with one Leader, the lecturer laptop, synchronising more than 100 devices without any interferences between piconets. The time taken to establish such a scatternet requires three different piconet construction phases. This time could vary from seconds to a few minutes, but it should be noted that this is

more a question of the student attendance time, rather than the scatternet formation process time.

- Sleeping time

Once connected to the scatternet, devices spend most of their time sleeping, as students are listening to the lecturer. In addition the latency will not be extremely important, so the number of waiting polling cycle in sleeping mode should be very high (around 15) to save more battery life up to 90% less power consumption, which is a significant feature in this circumstance.

- *Leader Sending Information to all devices*

During the lecture, the Leader could forward information such as slides up to 21kbit/s to all devices, as shown by Figure 6.3, where students download the source and get it directly onto their small PDA screen. For example during a power point presentation, all students receive the document in their PDA within few seconds, and save it directly on their memory card.

- *All Slaves Hierarchies send data to the Leader*

At one point, the lecturer could ask questions to all students to evaluate their comprehension of the lecture through Multiple Choice Questions… Students receive the questionnaire within a short time and could send back the answer at approximately 1.5 Kbit/s (see Figure 6.6) , which is sufficient for small data transfer. The lecturer is then aware of every student's results and attendance.

- *Only one slave communicates with the Leader.*

Since the leader is aware of the answers transmitted from all devices, the lecturer could then request that one student explains the results in more detail. The student using his PDA as a slave could draw a graph on its screen, which will be retransmitted onto the Lecture room screen. Since the student is the only one who requires high bandwidth, his (slave) PDA could transmit up to 60 kbit/s (shown by Figure 6.6) instantaneously to the leader (directly connected to the screen).

- *Summary*

Two crucial advantages behind the concept of the new scatternet Topology: First the creation of multiple piconets in such a small area will not be affected by interferences since all piconet are applying a derived frequency from the Leader (as

explain in *Section 3.3.2*). Hence any slave part of the scatternet could quickly search for other devices and create new piconet without any conflict.

The second point is that the devices save a lot of power once they are connected; by using a 15 waiting polling cycle, this guarantees a high conservation. Figure 4.12, shows more than 90% consumption saved.

In addition, the fact that the bandwidth is adapted to the traffic, the fairness of the schedule is respected. Such that any time when one student requires high bandwidth, he could get around 60Kbit/s, or when all students need the bandwidth, it is shared among all devices.

The Leader is aware of all PDAs present in the room, along with their time response (maximum a few second supplementary), which could be used and conducted for assessment.

The drawback is that since slaves are not listening during all even slots, it is not possible to use the "Bluetooth broadcast features" (as explain in *Section 2.3.2.6.*), which could have improved the throughput for Leader communicating to its slaves. However the broadcast process does not have an acknowledgment system for the slave reception, and if one device, close to the roots, does not receive the entire information, the forwarding of the incomplete information could lead to some problems, which would amplify from hierarchy to hierarchy.

1.1.2 Bluetooth Scatternet Base Station Access

Bluetooth was designed to supply ad-hoc network intended for connecting all sorts of portable devices from laptops, PDAs, 3rd generation wireless phone, digital cameras, video projectors, etc… In an environment in which the wireless Internet Access service of low cost is more important than high performance, Bluetooth with the scatternet concept could become the normal system architecture. Bluetooth Internet base stations could be distributed strategically in a given area to provide access to users in order to connect to a single Internet Access point. The proposed scatternet protocol with leader hierarchy could be developed in a fixed area (Lilakiatsakun *et al*

[2001]), and any new Bluetooth device (PDA, laptop…) entering in contact within the scatternet will could have access to their services.

1.1.2.1 Typical Traffic over Bluetooth

The Internet traffic files transfer is mostly affiliated with continuous bit rate traffic. Hence the Poisson Process approach is not valid to study the performance of such a network. Therefore others models need to be employed in order to analyse bursty traffics.

Data applications running over Bluetooth such as HTTP, FTP and CBR will need a transport layer to ensure a reliable end-to-end performance such as the transmission control protocol (TCP) and UDP, both widely used for delivery packets. In the Bluetooth technology, the point-to-point protocol (PPP) is designed to run over RFCOMMM.

The L2CAP layer provides connection-oriented data to upper layer protocol. Which means that L2CAP reassembles the Bluetooth packets into IP segments and sends them to the IP layer, which could be directly configured on the L2CAP layer (Das *et al [2001]*).

The maximum *Ethernet* packet size is roughly around 1500 bytes. Since L2CAP uses segmentation and reassembly (SAR) mechanisms to improve efficiency by supporting a maximum transmission unit (MTU) size larger than the largest baseband packet, the Ethernet packet could fit into 55 DH1 packets (27 bytes payload long) or in 8 DH3 (183 bytes payload) packets.

From sources generate traffic, an UDP header has been preferred to transmit the transport layer. The L2CAP layer adds another header and divides the Ethernet packets into 55 (or 8) packets and place them in the queue into the baseband in order to transmit them. Consequently some delay occurs during the application layer, for creation and reception of data. In (Johansson *et al [2000]*), this delay could be estimated to approximately 2ms, a value adopted for the following simulation.

1.1.2.2 Network Access Point Scenario

Figure 6.8 illustrates a scenario of different Bluetooth Personal Area Network (PAN) in a room area. The topology adopted reflects a real Bluetooth configuration of an office environment with one Bluetooth Internet Based Station, which has to share the bandwidth with 8 Bluetooth computers.

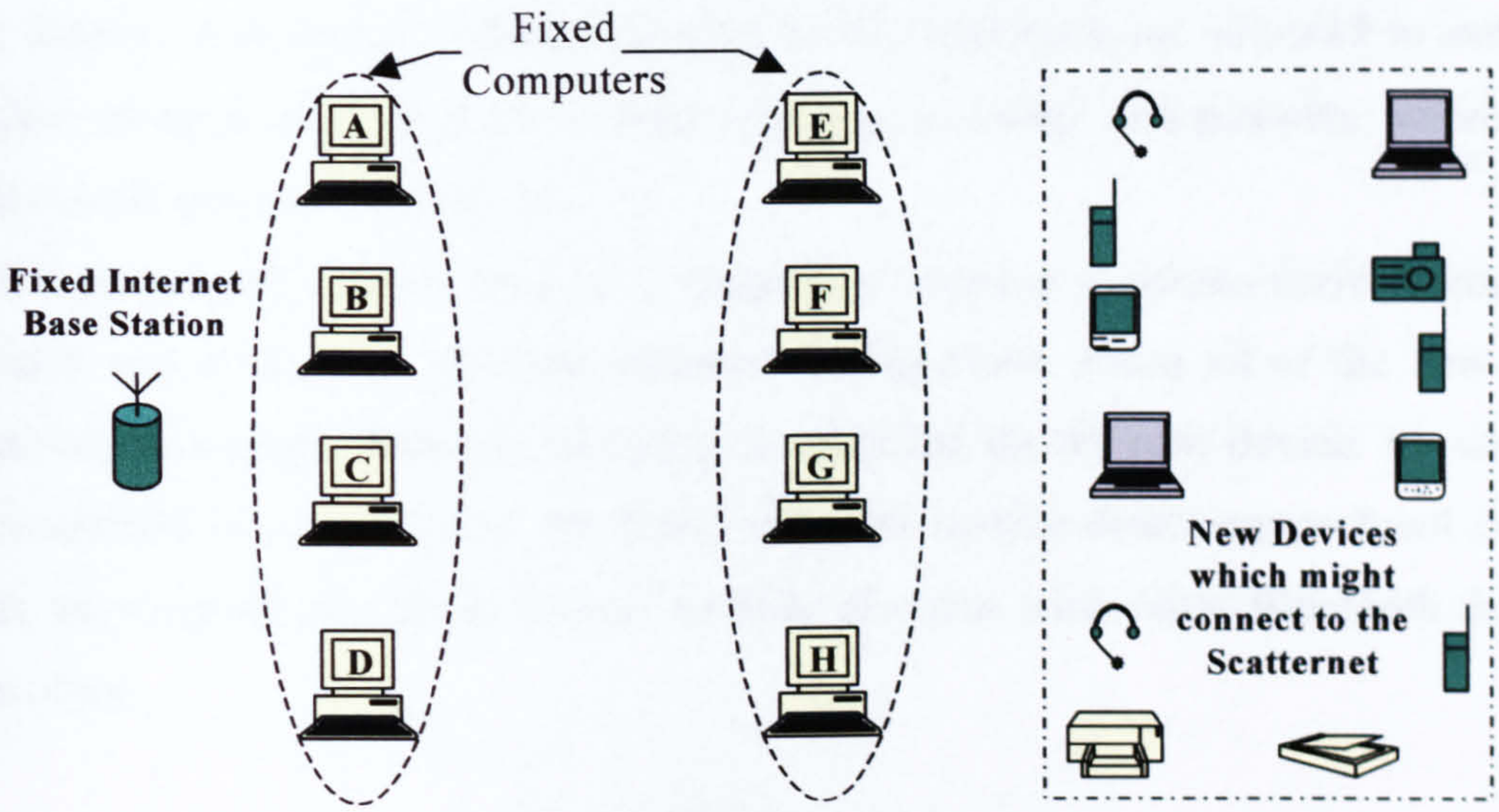


Figure 6.8: Office using Bluetooth for Internet connection and all data transmissions.

The scatternet must be coordinated in such way that the traffic can flow within and between the piconets as efficiently as possible, maintaining fairness while integrating new devices.

1.1.2.3 Base Station Topology Formation

Every device is primarily assumed to be disconnected and in a standby mode. Since the Leader is responsible for synchronisation as well as controlling the traffic on the channel, the Leader needs to be robust and have significant power lifetime. It is natural

to allocate the Internet base Station as the initial Leader of the new scatternet with other devices acting as its Slaves using an inquiry scan process.

Devices that become part of the scatternet, periodically execute a service discovery inquiry, applying uniquely the Inquiry Scan process. Since only the Leader applies the opposed inquiry process, devices could uniquely make contact with the Leader. And after the paging process a connection could be established between the Leader and new devices (its slave). The intra-piconet schedule, between the Leader and PCs, is performed using the AMSPQ scheme and the piconet bandwidth is divided among the Slaves. Subsequently slaves, as part of the scatternet are allowed to enter in inquiry mode in order to discover other devices and create new piconets, where each child could generate piconet too.

The new piconet created, employs a frequency hopping sequence derived from the Leader, and it operates with the identical timing clock. From all of the new links established, a new addressing hierarchy is specified for the new device. Its services characteristic is forwarded to the leader from the service discovery protocol (SDP), thus allowing all Bluetooth devices to fully discover what other Bluetooth devices can offer.

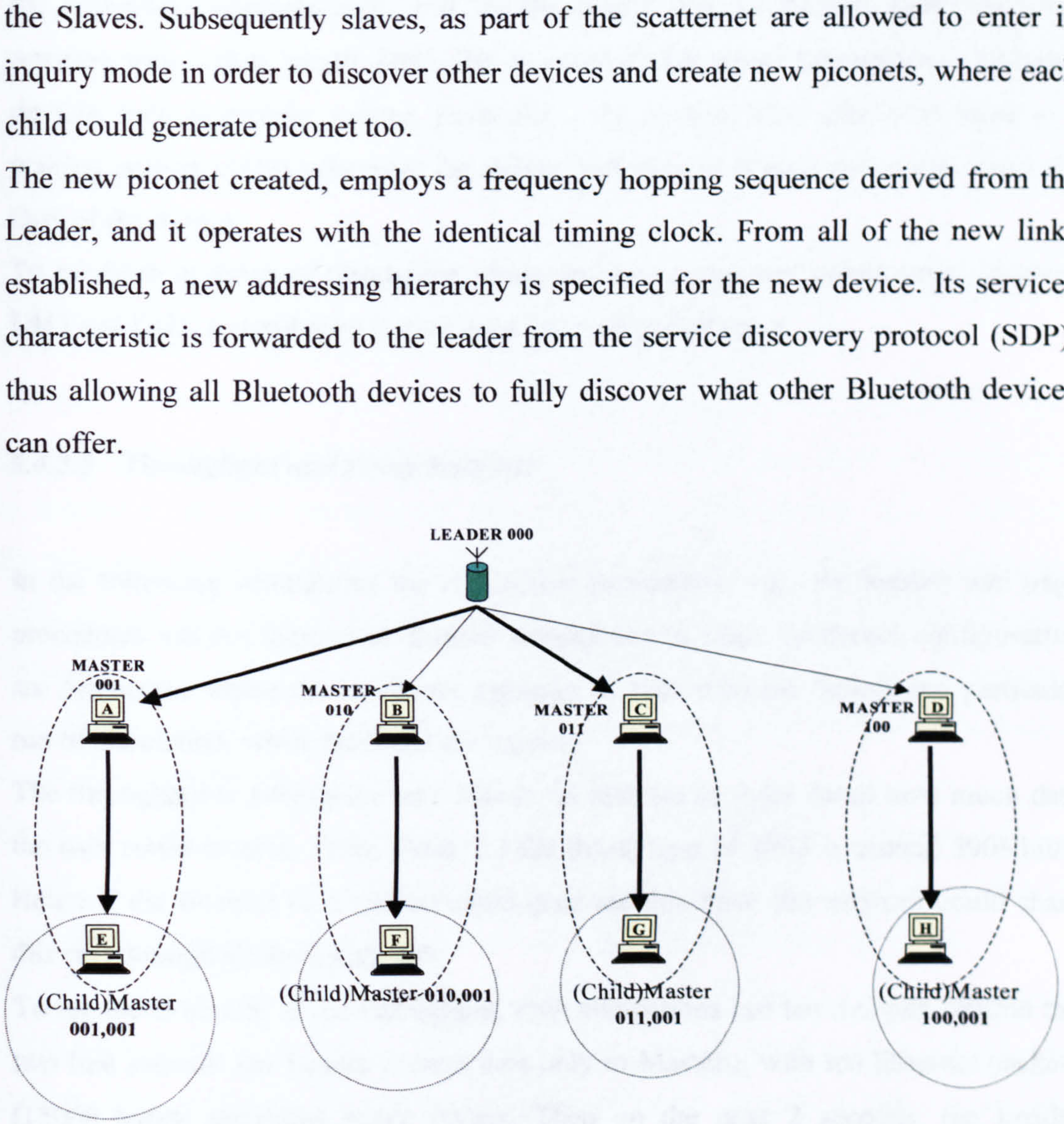


Figure 6.9: New Scatternet topology Office evolution.

In the following example, during scatternet formation, four computers are becoming slaves, while fours others computer their child slaves, as illustrates by Figure 6.9.

6.4.2.4 Scatternet parameters fixed by the Leader

As the bandwidth is a significant feature for Internet connection, the higher the throughput the better is the system. Hence in this particular case, the Leader within the scatternet communication will fix the metric that all packets have the same duration with 3 slots length: DH3. But to avoid a high power consumption for small devices such as mobile, mouse, keyboard… the cycling time should be fixed to 4 waiting polling cycles otherwise the delays will exceed 100ms and could affect the QoS of the mouse.

To establish in terms of throughput, delay and power the best compromise between DH3 and DH1, a simulation is evaluated for both packet types.

6.4.2.5 Throughput and Delay Analysis

In the following simulations the connection procedures, e.g., the inquiry and page procedures are not simulated. Instead simulations in static scatternet configuration are performed where the scatternet topology is kept constant during one particular run of simulation, while the loads are varying.

The throughput is averaged every 100ms, to indicate in more detail how much data the user could receive. From *Table 2.1* the throughput of DH3 is around 390Kbit/s. Hence if the Internet base station could generate this flow; the network could share this rate through all the computers.

To see the evolution of the throughput, each simulations last ten seconds. Within the two first seconds the Leader creates data only to $Master_{(4)}$ with ten Ethernet packets (15000 bytes) separated every 100ms. Then on the next 2 seconds, the Leader produces data for $Master_{(1)}$ applying the same CBR rate, and continues for the next two seconds for $Master_{(2)}$ and then $Master_{(3)}$. At the 8^{th} second the leader generates the same CBR rate (15000 bytes) to all the masters but for only 1 second.

Figure 6.10 shows the throughput results including the time of the Ethernet packet reception by the four different masters, starting from the Leader applying the DH3 packets link. The overall results shows that the AMSQP scheme adapts the CBR files very well with an average of 150 Kbit/s. In fact the bandwidth is predominantly given to the $Master_{(4)}$ for the first two seconds, and changes for the other master when they necessitate supplementary bandwidth. From the 8^{th} second, time when all the devices required a maximum bandwidth, the throughput shows that all master receive an equivalent rate of around 93kbit/s (between 8 to 10 sec), which proves the absolute fairness of the scheme. Since the interval (when packets are generated) allows some slots between emissions, the link handles the traffic very well. This is not too much for the case of links operating with DH1 packets, as indicated by Figure 6.11. The equivalent amount of data generated from the Leader with the same rate does not progress at the same throughput intensity. This is mainly due to the fact that the throughput is higher for DH3 links, while an Ethernet packet waits for the 9^{th} Bluetooth packets to be completed, using DH1 it required 55 packets.

Nevertheless the Leader makes use of most of the bandwidth to communicate with the master. As shown, between the 2 and 6 seconds, the $Master_{(1)}$ has the opportunity to acquire in the first instance the BW of the two other masters since both are sleeping; and the $Master_{(1)}$ is just ahead of them, in the RR cycle. While the $Master_{(4)}$ could not get this chance and must be satisfied with a ¼ of the total bandwidth. Consequently it could be observed, an increase from $Master_{(4)}$ before the 8^{th} seconds, once $Master_{(1)}$ terminates its communication, then $Master_{(4)}$ could utilize the left over bandwidth to increase its transmission throughput.

Hence as expected and confirmed by Figure 6.13 the Ethernet packet delay on $Master_{(4)}$ is significantly higher than the delay for $Master_{(1)}$, and it attains a value of 8000 delay slots (5 seconds), which in some applications could be significant.

From Figure 6.12, the average delay is fairly close to 110 slots (70ms), which is mainly due to the fact that the Ethernet Packet is an assembly of nine Bluetooth packets. This increases the typical delay by a factor of nine. Since all masters required a high throughput, it is reasonable to find an increase, which does not exceed the 0.75 seconds.

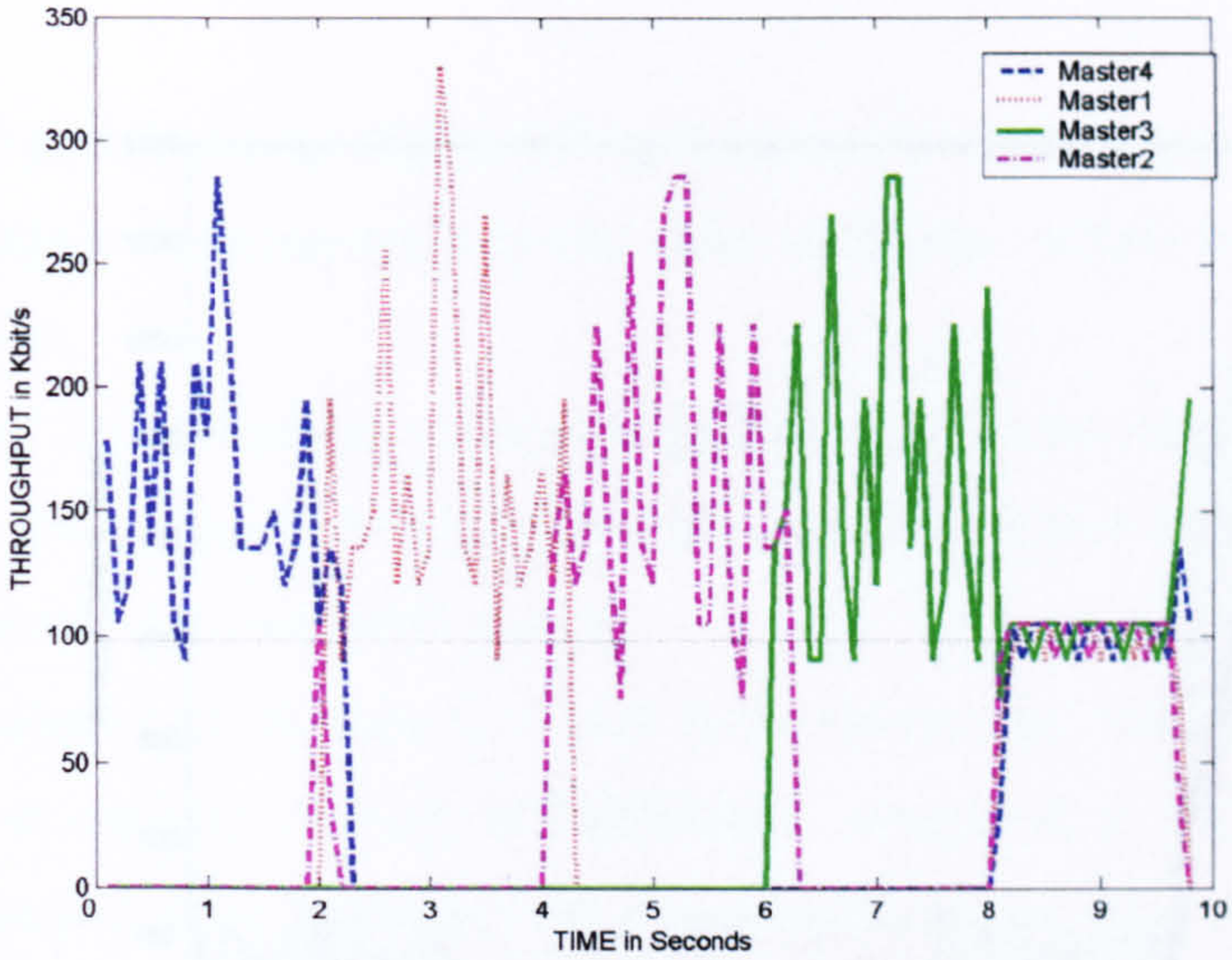


Figure 6.10: Throughput of the four masters from the Leader sending Ethernet at 0.1sec frequency, and using DH3 packets.

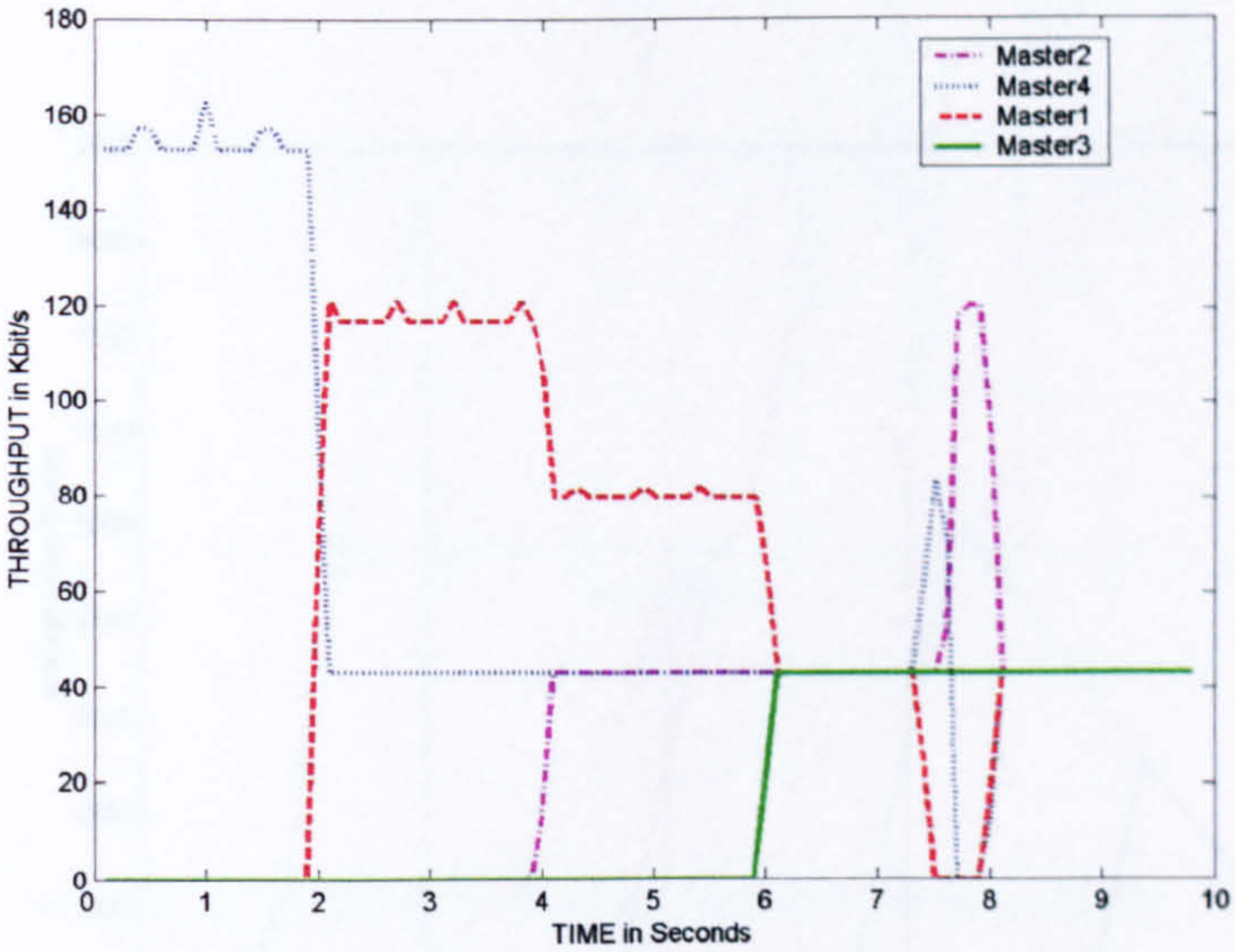


Figure 6.11: Throughput of the four masters from the Leader sending Ethernet at 0.1sec frequency, and using DH1 packets.

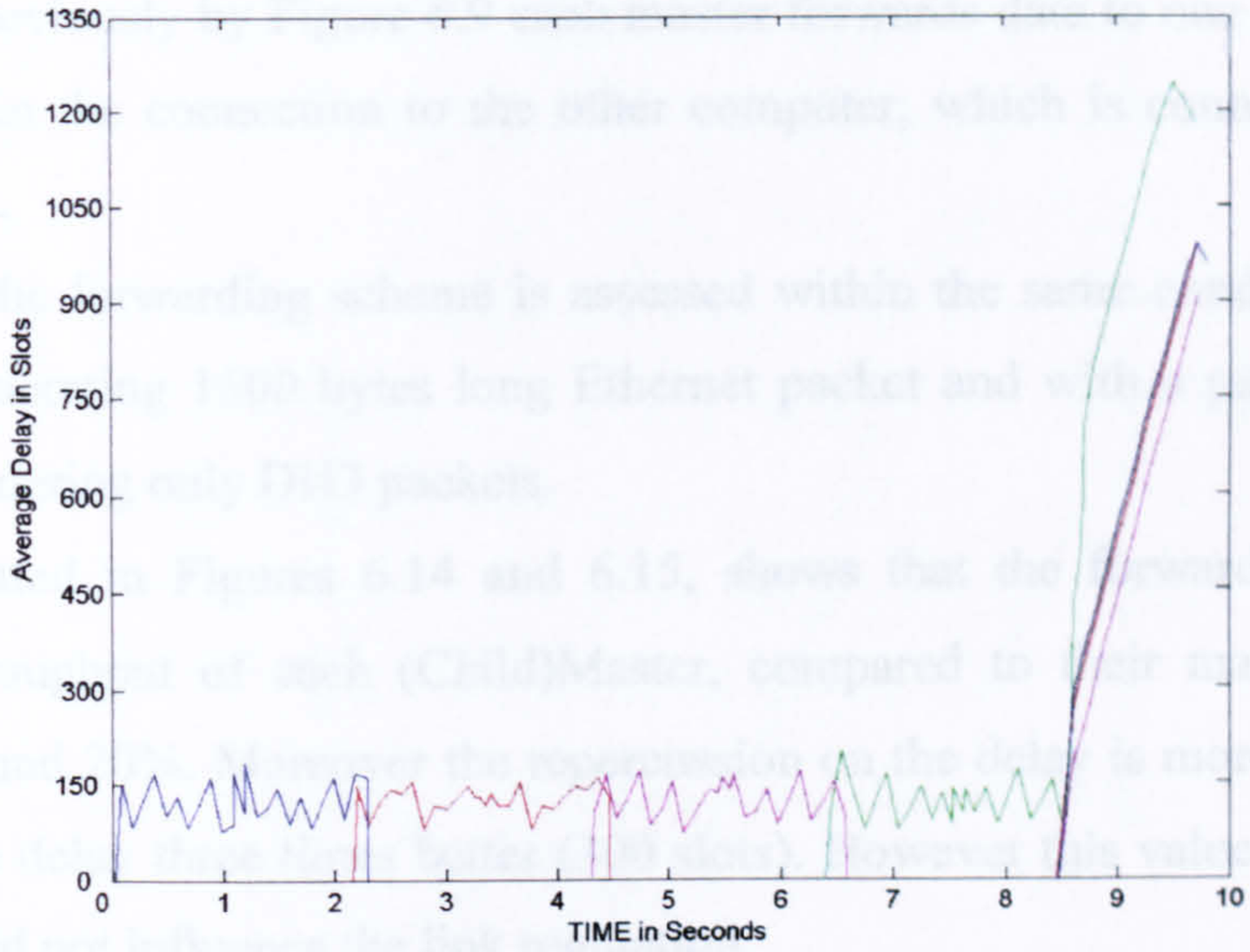


Figure 6.12: Ethernet packet Delay of the four masters using DH3 packets.

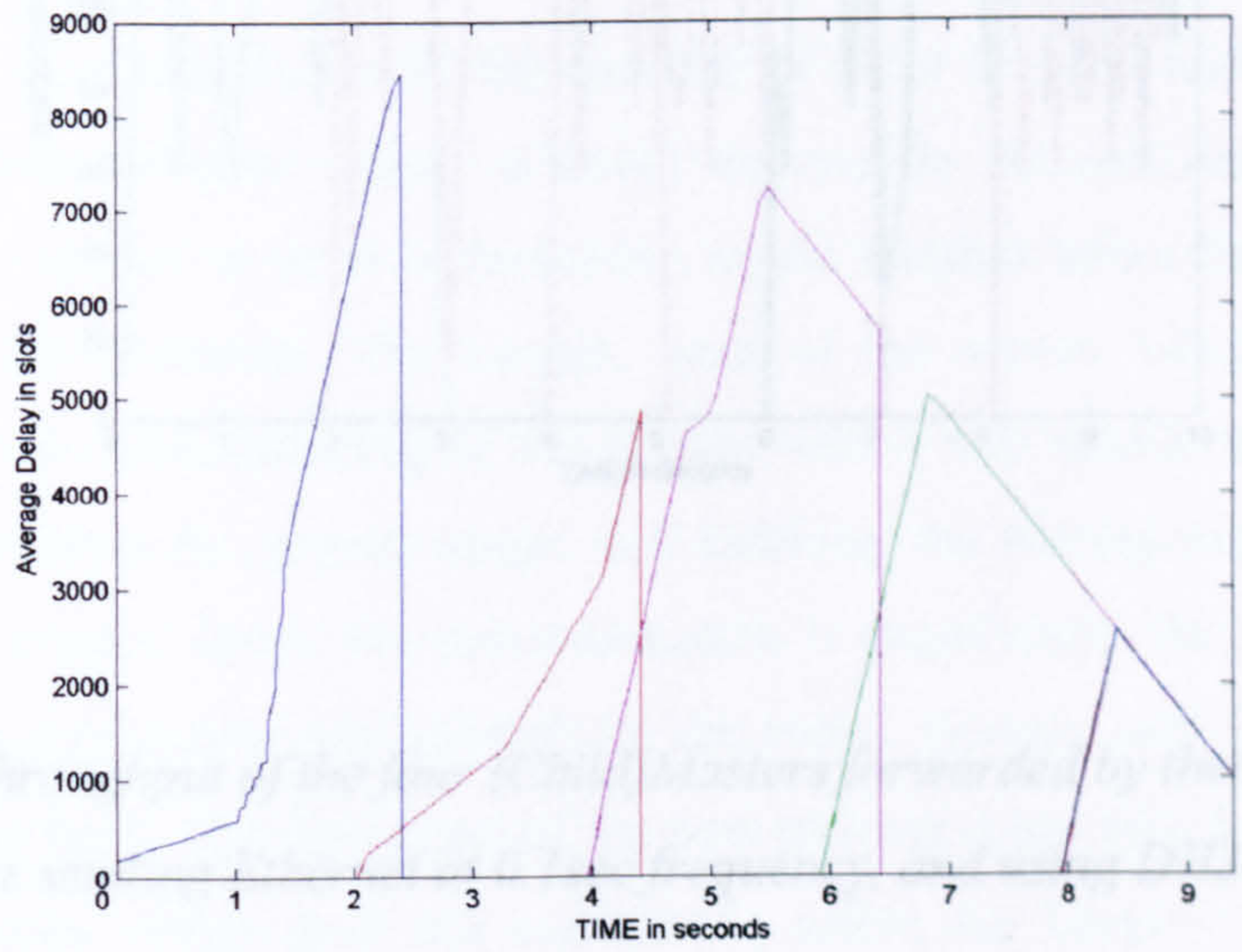


Figure 6.13: Ethernet packet Delay of the four masters using DH1 packets.

6.4.2.6 Second Hierarchy Simulation

As illustrated previously by Figure 6.9 each master forwards data to one computer in order to maintain the connection to the other computer, which is connected to the outside network.

The impact of the forwarding scheme is assessed within the same condition with a CBR source generating 1500 bytes long Ethernet packet and with a period of 0.1s length but considering only DH3 packets.

The results plotted in Figures 6.14 and 6.15, shows that the forwarding packets reduces the throughput of each (Child)Master, compared to their master, with a decrease of around 20%. Moreover the repercussion on the delay is more significant with an average delay three times better (300 slots). However this value is still very low and it should not influence the link regulation.

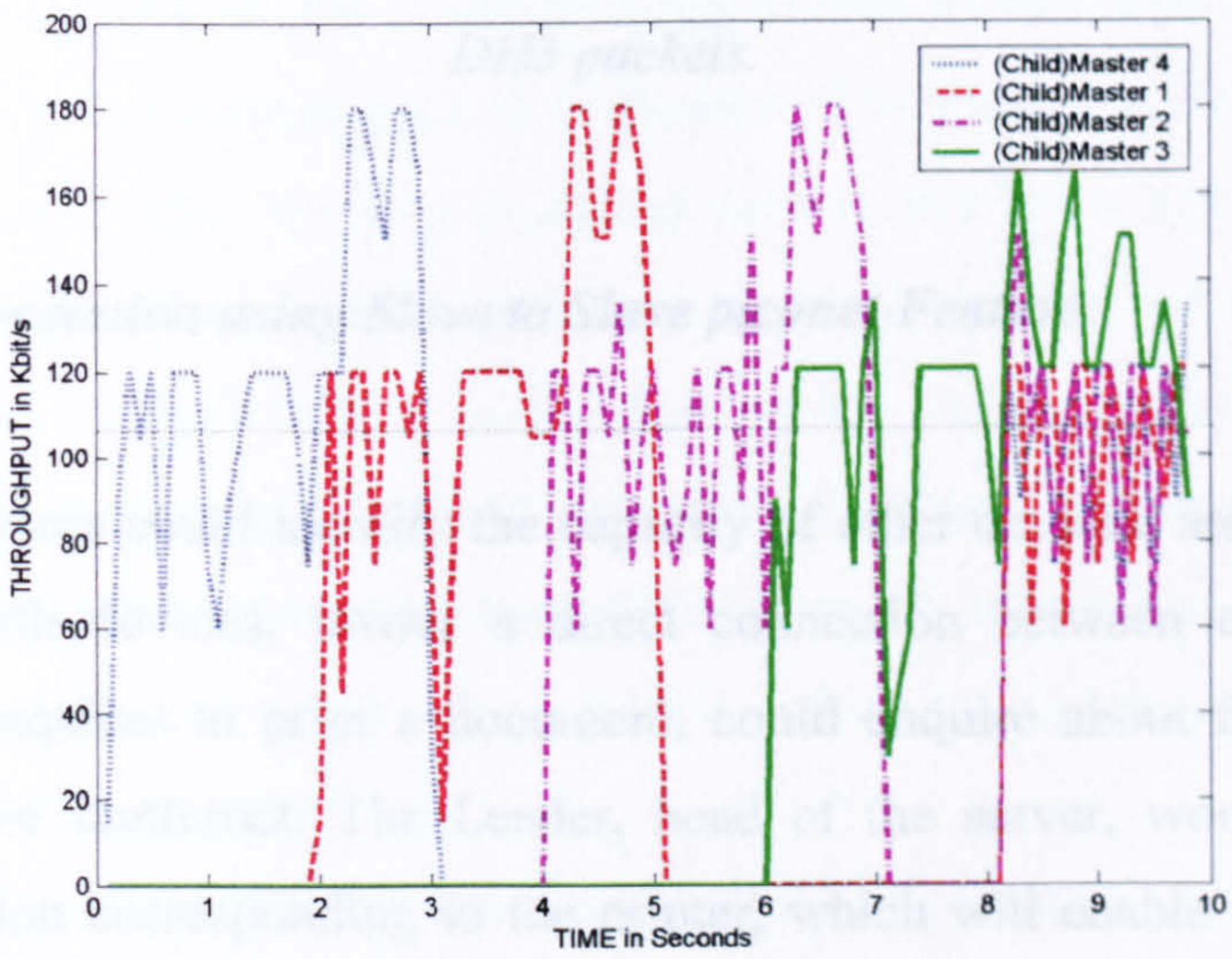


Figure 6.14: Throughput of the four (Child)Masters forwarded by their Master from the Leader sending Ethernet at 0.1sec frequency, and using DH3 packets.

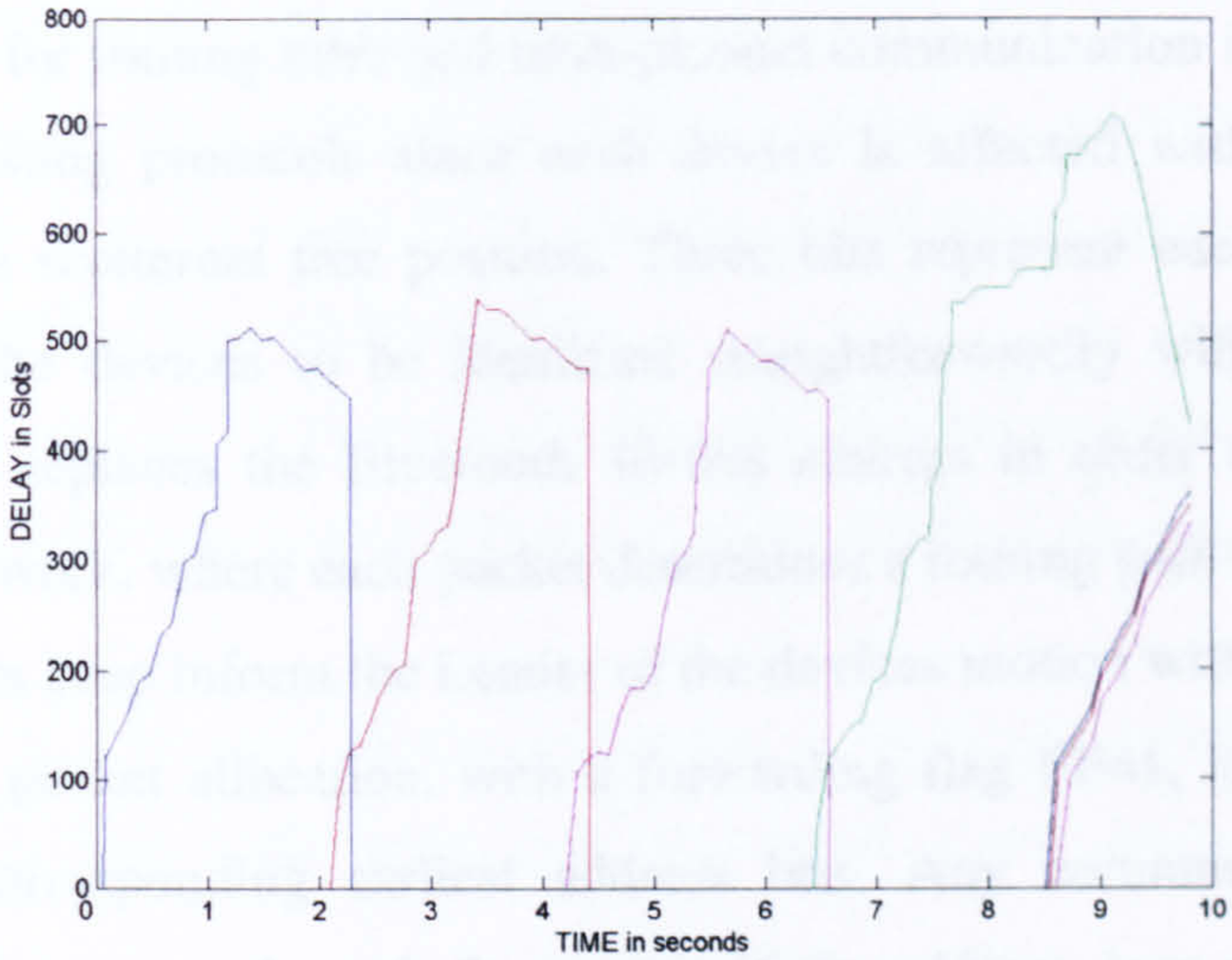


Figure 6.15: Ethernet packet Delay of the four (Child)Masters from the Leader using DH3 packets.

6.4.2.7 New Formation using Slave to Slave piconet Feature

The fact that devices could identify the capacity of other devices, and the proximity of each Bluetooth devices, favour a direct connection between each other. For example a PC requires to print a document, could enquire about the existence of printer within the scatternet. The Leader, head of the server, would forward the paging information corresponding to the printer, which will enable the PC to enter quickly by bypassing the inquiry stage, and catching the corresponding page scan window of the printer. Once the communication is established, the printer will be assigned with a new 9 bits AM_ADDR, in the initial piconet, and, it will keep its previous address there. The formation of the new piconet could take from a few slots to several seconds, which does not excessively affect the latency of the printing process itself.

6.5 CONCLUSION

A new approach for routing inter-and intra-piconet communication is described. This differs from existing protocols since each device is affected with a new address depending on its scatternet tree position. Three bits represent each tree hierarchy, which enables the devices to be identified straightforwardly within the network. Hence nine bits replaces the Bluetooth 48-bits address in order to further reduce overheads in network, where each packet determines a routing path to the destination itself. All masters keep inform the Leader of the devices motion within the scatternet. The forwarding packet allocation, with a forwarding flag FF=1, is simply done by checking the corresponding earliest address bits. Any communication between slaves' devices has to go through the master, if the address does not correspond to the initial piconet; the master pushes the packet upward to the parent, where it is allocated to the correct slave, to reach its destination. Some simulations confirm the effectiveness of this process, the AMSQP scheme share its bandwidth fairly for intra-piconet and also for forwarding packets as illustrated by Figures 6.3 and 6.4.

However when the traffic between separated devices reaches a particular threshold amount (50 Kbit/s in the example given), an impractical delay occurs as shown by Figure 6.6. Hence to optimise the packet forwarding a new concept of multi-slave communication has been presented. A slave could get into direct contact with any device in range within the scatternet. Simulation of this novel communication architecture, in Figure 6.6, shows a drastic enhancement in current Bluetooth performance in terms of delay.

Some examples are selected to illustrate different circumstances on which this new routing scatternet could be implemented. A lecture room and an office with different scatternet requirement have been described. No overhead such as time lost due to node synchronisation, and predicted polling sequence; permit more frequent piconet switching and creation, which increases the reliability of the scatternet. The creation of further piconets will not affect the piconet interferences within the scatternet as against any other scatternet schedule presented up till now.

Furthermore, any device could communicate to any device within the scatternet, which offers supplementary traffic alternative. This improves the reliability of the

scatternet: each node could contain more than one link using the predicted AMSQP scheme, and in case of a Master/Slave link breaks, the presence of the alternative path could maintain the device synchronise to the scatternet.

The fact that multi-slave belong to multiple piconets improve the scatternet connectivity. And thus, by reducing the number of communication hops to transfer data, improve overall delay and throughput.

At the same time slot up to 57 links *(Section 3.3.2)* could transmit data within the scatternet, improving the overall scatternet fluidity, while a safe power management is organized.

To increase the effectiveness of the Scatternet, a realistic implementation on the Bluetooth chip has been under-taken, and its first part establishment follows in the next chapter.

CHAPTER 7

AMSQP SCHEME IMPLEMENTATION ON BLUETOOTH DEVELOPMENT BOARD

7.1 INTRODUCTION

Fabricating millions of transistors on a single chip is feasible with the current silicon technology. That means electronic systems, which are built today from large numbers of chips could in future be integrated onto just one. The problem is that it takes an enormous amount of time and effort to design chips using today's technology, which companies want new products rapidly so that design costs can be recovered before the next product has to be commissioned. Something needs to be done to speed up the design process (SLI *[2004]*).

The adoption of reuse and improved design tools will leave the design team to concentrate on the system application and its high-level design. The design team will be relatively small and will consist of software, hardware and systems engineers working together to get the product designed in as short a time as possible.

This chapter describes the design and work carried out to implement the new Bluetooth intra-piconet scheduling scheme, explained in *Chapter 4*, on real hardware using Bluetooth Protocol Stack Software. The scheme is implemented employing the Bluetooth Intellectual Property (IP) simulated on the ARM-based Gorgon$_2$ Development Boards along with Ericsson Bluetooth development kits. The work materiel was provided by Cadence consisting in the Bluetooth IP developed by Tality.

All the hardware devices for the implementation of the new scheme are described in *Section 7.2*. *Section 7.3* explains how the devices and software interact with each other followed by the test strategy employed. *Section 7.4* shows results of the new scheme implemented through the software. And *Section 7.5* provides the conclusions to this chapter.

7.2 HARDWARE RESOURCES

7.2.1 Gorgon$_2$ Development Board

The Gorgon2 is a Bluetooth Development board, which was provided by Cadence within the Institute for System Level Integration (ISLI). The Board is illustrated in Figure 7.1. The board was set up in a piconet as a Slave device. This co-simulation test environment operated in real time, and was utilized to validate the hardware/software interface for the Bluetooth baseband drivers and lower layers of the protocol stack in conjunction with a VHDL model.

The board uses two FPGAs (Altera and Xilinx) to implement the baseband controller hardware and requires the use of a co-simulator such as the ARMulator.

NB: There were four of these boards available at ISLI, but once testing had commenced on the boards it was found that two of them did not function correctly. This presented a problem for simulation of the new software.

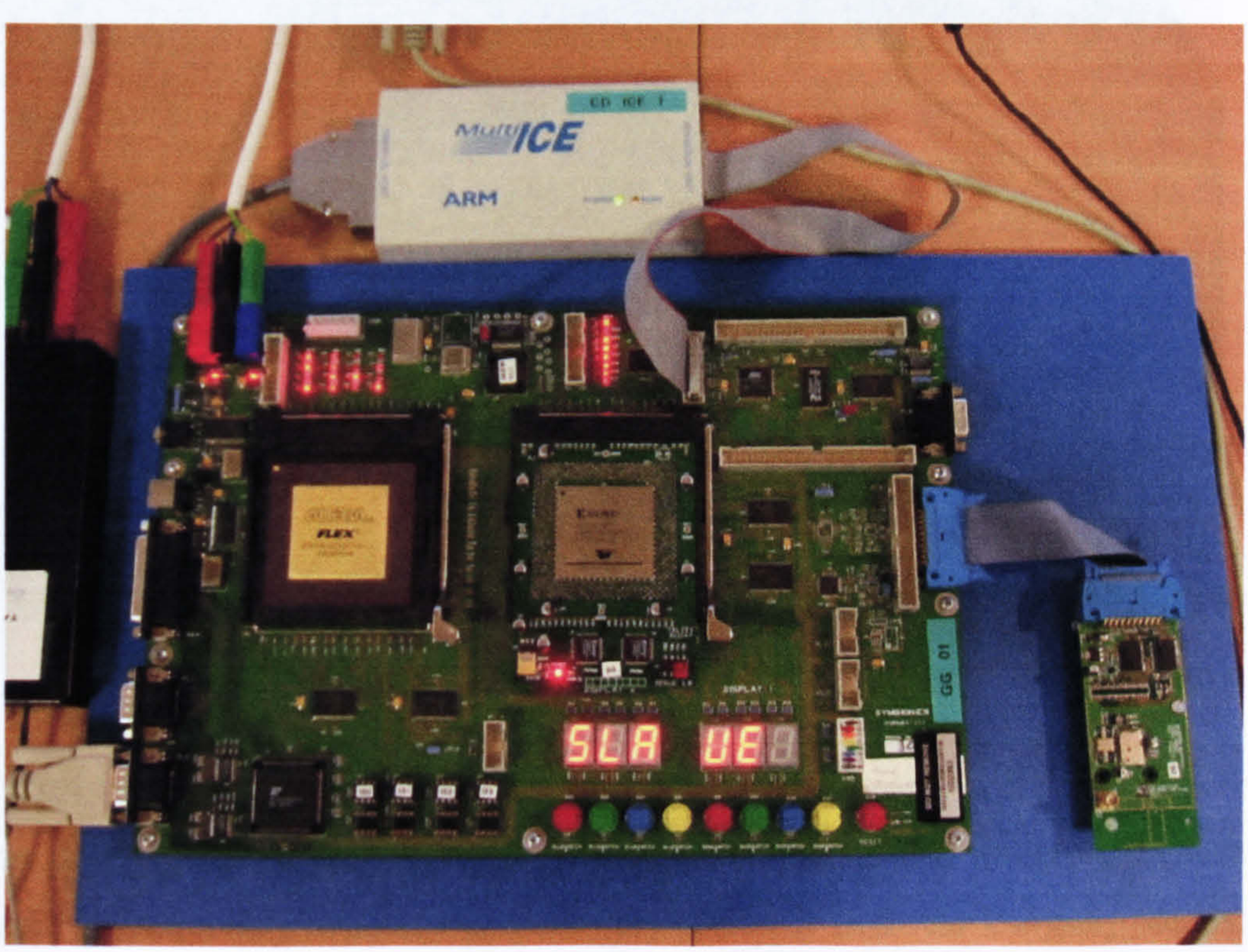


Figure 7.1: Gorgon$_2$ Development Board.

7.2.2 ARM Multi-ICE Protocol Converter

The modified executable image of the BlueSoft project (C code) was uploaded onto the target board by use of an ARM Multi-ICE Protocol Converter.
This piece of hardware was used for low-level debugging of modified software that was running on the ARM7TDMI processor core that was available on the $Gorgon_2$ boards. Software used for communication with the Multi-ICE was provided for use on Windows 2000. The hardware is given in Figure 7.1 at the center top of the picture.

7.2.3 Ericsson Bluetooth Development Board

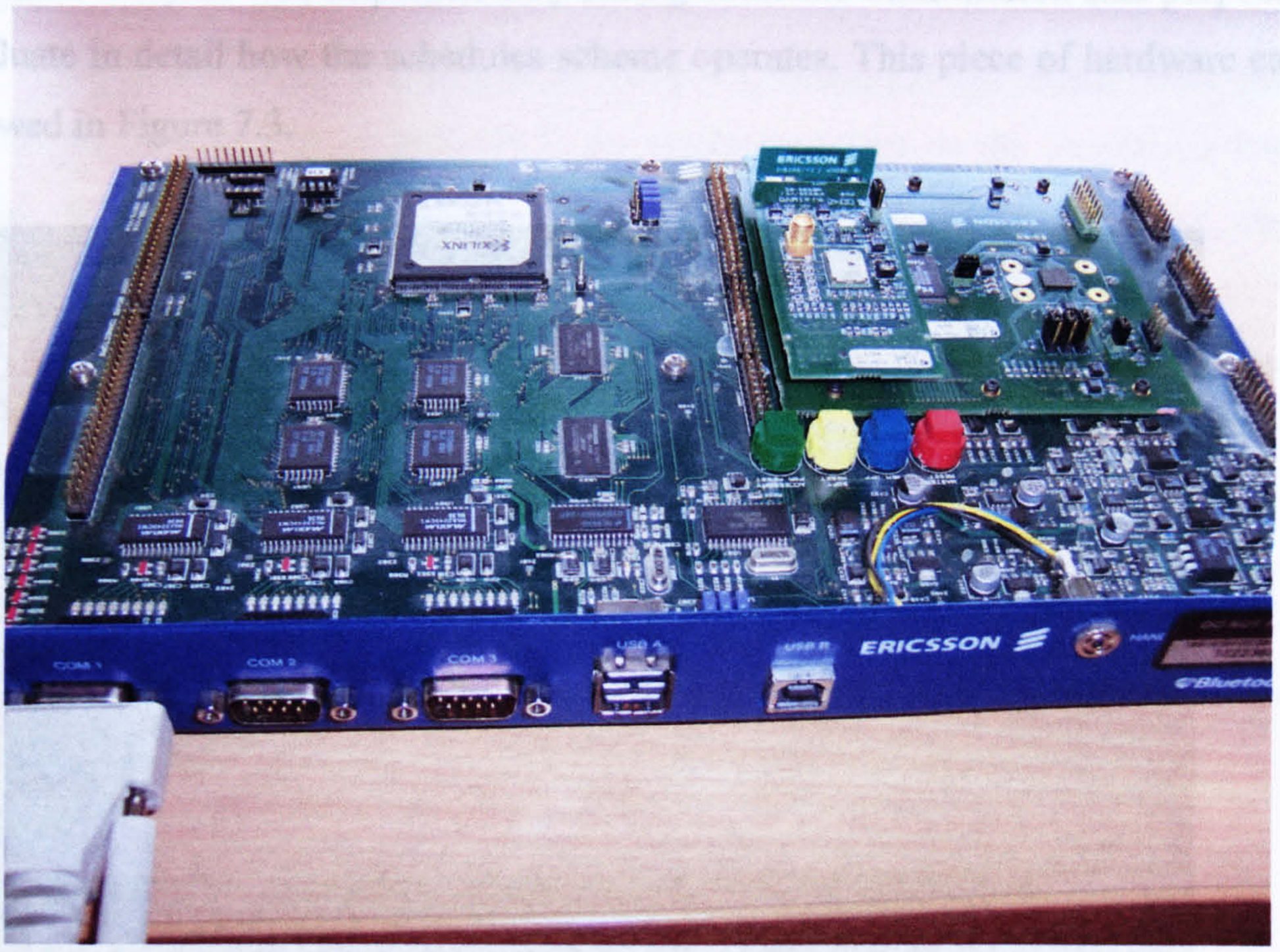


Figure 7.2: Ericsson Bluetooth Development Board.

Ericsson development Board was also employed for simulation purposes. These boards are commercially available and are very different from the $Gorgon_2$ boards. The Bluetooth network stack layers are implemented as firmware on the board, all

the way up to the HCI layer of the stack. The boards could be re-flashed with an updated version of the stack but regrettably this could not be carried out due to time reasons. The board could be simulated from the HCI layer by use of the BlueVelvet test suite on a PC environment. One of the four boards that were available is shown in Figure 7.2.

7.2.4 Merlin Bluetooth Protocol Analyser (Packet Snooper)

The Merlin Bluetooth protocol Analyzer works within the Bluetooth radio frequency band and has the ability to pick Bluetooth packets in a wireless environment and log them on a PC. The Analyzer logs all packet information along with packets timing. The Merlin analyzer was employed only during authentic transmission data purposes to evaluate in detail how the schedules scheme operates. This piece of hardware can be viewed in Figure 7.3.

Figure 7.3: Merlin Bluetooth Protocol Analyser.

7.3 SOFTWARE RESOURCES

7.3.1 BlueSoft: Bluetooth Network Stack Source Code

- IP description:

A Bluesky project has been composed using the IP of Bluetooth 1.01 release. It contains in C, micro-code sources and the Tality specification documents, and is organized as follows: /project/bluesky/clafon/

- ✓ doc: contains the documents and the specifications of the IP
- ✓ logs : test applications
- ✓ sw : contains the sources organised in different folders corresponding to the layers.

The C sources represent the Bluetooth program, designed to match with the specifications. It is written in portable C for implementation on the bluechips. The interface, exported by the Baseband Physical component, is defined by the BLUECHIP software interface specification.

These sources are compiled in object files that are linked with assembler files. The VHDL code files are implemented in a Xilinx FPGA, which is piloted by the Bluechip.

- Compiling and implementing the program files:

Microsoft visual C++ was used to develop the C program. The whole work was compiled in a workspace that contains different folders corresponding to each software layer. When building this workspace, a btpc.exe file was generated. Then the BTPC test mode used this executable file to test it. In this case, only the high level software files were involved. After development, the code was loaded on the Bluetooth boards $Gorgon_2$. This involved all the software layers including the baseband driver. The C code was related to micro code. Then, it was transferred to the microcontroller of the Bluechip. Finally, the program was tested in real conditions. Figure 7.4, shows a simplified test mode:

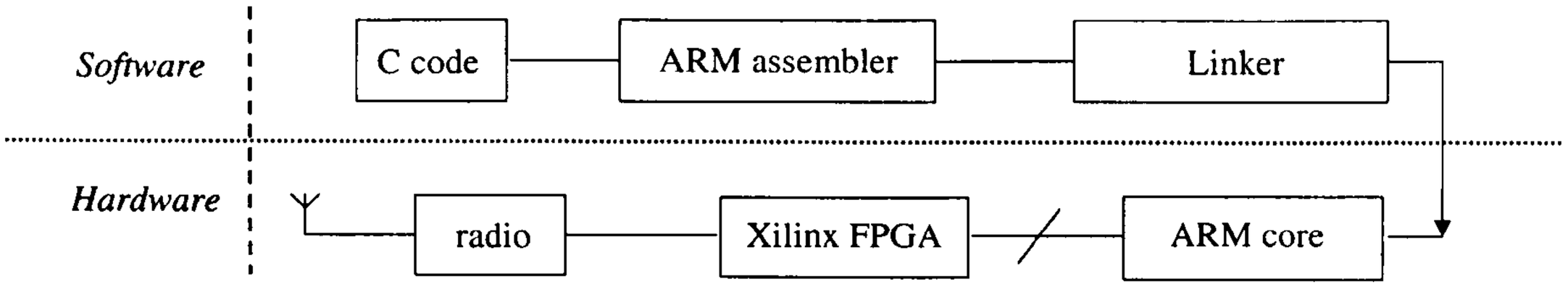


Figure 7.4: Simplified Testing method from software to hardware environment.

- Organization of the Bluetooth software and its testing modes on GORGON2. The Bluetooth software (bts) is an embedded system. The bts consists of hundred of thousands of code lines and contains multiple states that cannot be implemented using switch functions. Such software is a natural candidate for mechanization as a state machine. A state machine is defined as an algorithm that can be in one of a small number of states. A state is a condition that causes a prescribed relationship of inputs to outputs, and of inputs to next states. This software is developed as a Mealy machine. A Mealy machine is a state machine where the outputs are a function of both present state and input, as opposed to a Moore machine, in which the outputs are a function only of state. In both cases, the next state is a function of both present state and input. The BlueTooth General Interface Layer (BTGIL) was used for testing on these boards.

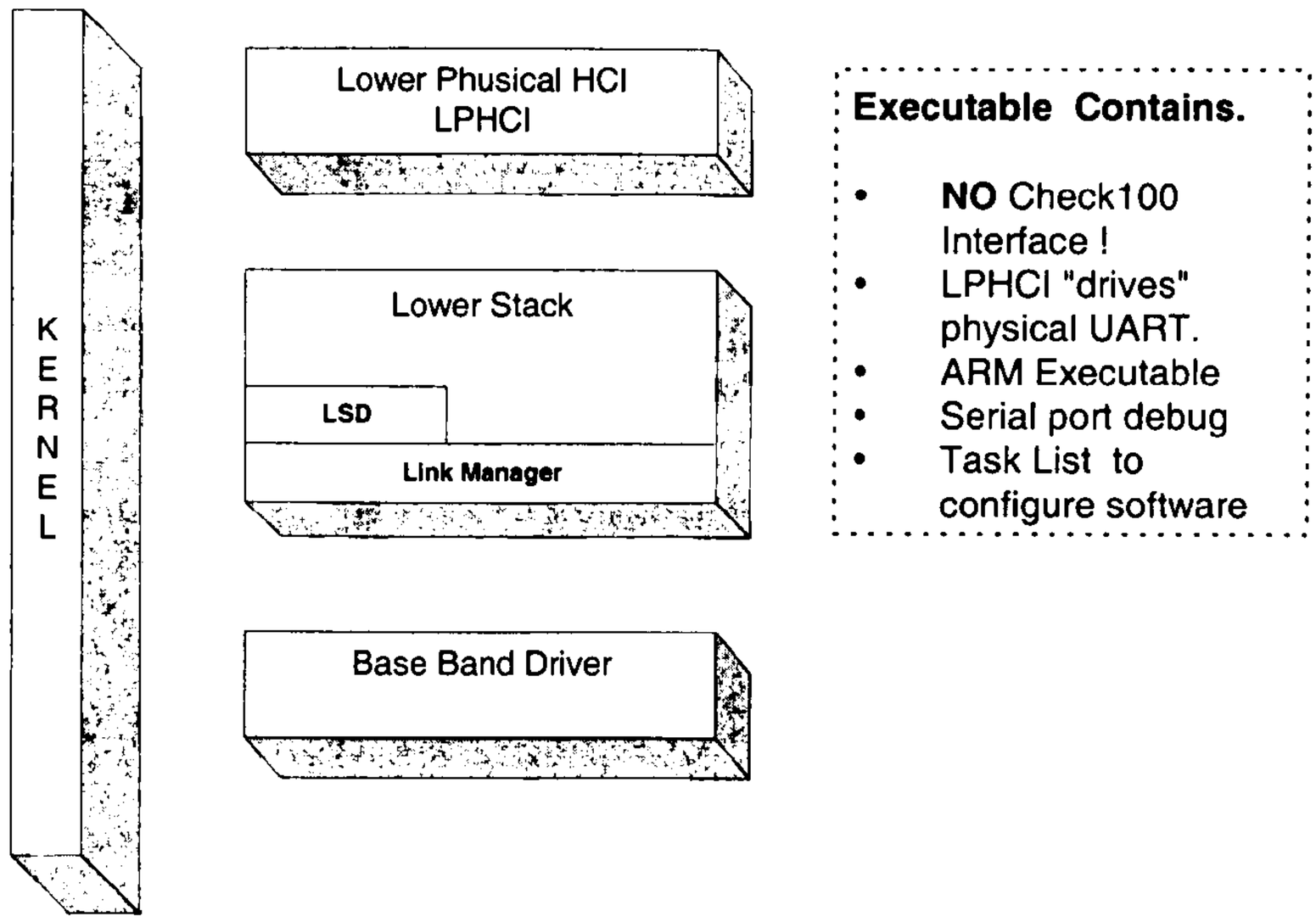


Figure 7.5: BTGIL Bluetooth Stack.

The BTGIL build consists of an ARM executable assembler for the $Gorgon_2$ development board. This is built with ARM Software Development Toolkit and mapped onto the board using the ARM Multi-ICE Protocol Converter. The executable consists of the lower layers of the Bluetooth stack as shown in Figure 7.5. It basically simulates from the HCI layer down. Testing on this build was carried out using the BlueVelvet test environment for entering HCI commands into the HCI interface through the PC serial connection.

The testing environment was the BlueVelvet operating on the Gorgon2 development boards. This is a HCI command line interface with the board that basically removes anything above the HCI layer of the stack and tests anything below it.

7.4 RESULTS

7.4.1 AMSQP Scheme Implementation

To adapt the intra-piconet traffic, the new scheduling scheme relies on the 2-bit code mentioned in chapter 4. These two bits need to be sent within every packet type. They need to be inserted into either the packet header or within the payload header. According to the Bluetooth specification version 1.1 there are four free bits available in the payload header as illustrate by Figure 7.6.

Segment	TYPE code $b_3b_2b_1b_0$	Slot occupancy	SCO logical transport	eSCO logical transport	ACL logical transport
1	0000	1	NULL	NULL	NULL
	0001	1	POLL	POLL	POLL
	0010	1	FHS	reserved	FHS
	0011	1	DM1	reserved	DM1
2	0100	1	undefined	undefined	DH1
	0101	1	HV1	undefined	undefined
	0110	1	HV2	undefined	undefined
	0111	1	HV3	EV3	undefined
	1000	1	DV	undefined	undefined
	1001	1	undefined	undefined	AUX1
3	1010	3	undefined	undefined	DM3
	1011	3	undefined	undefined	DH3
	1100	3	undefined	EV4	undefined
	1101	3	undefined	EV5	undefined
4	1110	5	undefined	undefined	DM5
	1111	5	undefined	undefined	DH5

Figure 7.6: Two bits combination placed within the Packet Type code.

Since only ACL links are employed, the two bits are positioned in the Packet Header, within the Packet TYPE code segment. It has been used in collaboration with the DH packets Type. Such that if a device receives the two first TYPE code bits (b_3b_2)

coded as "01" or "10" (within segment 2) it corresponds to AMSQP scheme, and the two other TYPE code bits (b_1b_0) represents the next polling sequence, i.e. the two bits symbolize Table 4.2 for the Slave and Table 4.3 for the master.

Hence from both tables, the combination such as "00" are covered by a packet type code "1000", while a combination of "11" is denoted by "0111". Therefore a Slave device without AMSQP implementation would still be compatible with the piconet, it will not recognize the other packet type apart from the initial Type code "0100" referred as DH1 packet.

The Sniff mode is used for setting up the proposed Round Robin scheme in AMSQP scheme. The sniff intervals involved depend purely on the number of Slaves present in the piconet. For a piconet of 3 slave devices the sniff interval is 6 slots. It was originally thought that the sniff interval could be 5 slots for this set up and this would be logically correct but the code implementation required that the sniff intervals had to be set to an even number and this even number should be the total number of slots available in the RR scheme, which in this case is 6 slots. During this sniff interval the Slave should be free to act in another piconet formation. By using the sniff mode for implementation, Slave devices should remain synchronized to the piconet for the duration of the sleep interval.

According to the equation 3.1, for a scatternet formation, the maximum sniff intervals would need to be limited to 400 slots for each link in a piconet to keep the piconet perfectly synchronized. This would make 396 slots, a suitable long sniff interval for a 3 Slaves piconet formation.

It was noted from experimental procedure that the clock drift does not affect the sniff time till a considerable increase of 10000 slots time within sleeping mode, on which the two devices could stay synchronised, which works out to be 6.25 seconds between slot-pairs.

Once the Slave device has received the two bits combinations, to be able to change the sniff interval, a new LMP message is sent. This is a significant problem since there is latency before the LMP message is confirmed. This latency is present both when sending LMP messages from the master to the slave, and from the slave back to the master.

To avoid changing sniff parameters over LMP messages, a new sniff mode configuration needs to be established in the hardware system. The fact that the sniff interval is safe in the memory on the bluechip and not within the C code, implies a change on the micro-code level, which depends on the bluechip characteristics. At this level of IP, information about the bluechip and about the micro-code has not been enough to carry out the modification. Hence, only through LMP messages the new sniff interval could be changed. This introduces latency before a new sniff interval occurs.

Thus, when a Slave connects or disconnects to the piconet, the master transmits a sniff_req LMP message to every Slave device in the piconet. This is done one Slave at a time and the sniff interval involved for each Slave device depends on how many devices are connected to the piconet. Sniff interval is equal to 2 * number of devices in the piconet. The master device knows when all the devices have changed into the AMSQP mode, so at this point the master device starts looking at the status of its data queues for each Slave device that is connected.

If no data is to send from the master device, the master will require the link to power down for a long period to save on power consumption. This sniff interval would need to be a multiple of the Number of Slaves * 2. For example, in a piconet of 3 Slave devices, the long sniff interval should be a multiple of 6 slots, i.e. 396 slots.

A slave device examines its queue status and if it has data to send to the master device then it decides to reject the sniff_req message by sending a not_accepted LMP message back to the master device. This means that the link stays with the same sniff interval that was previously being used. If the Slave device does not have any data to send to the master device then it sends an accepted LMP message to the master device. The long sniff interval is implemented on that link. Once master device wakes up from the long sniff interval, it looks again at its data queue and goes through the same process.

To make the slave aware of the next free slot (the slave next on the RR list is in sleeping mode), the master transmits a new sniff timeout. Basically when the link has come back from sniffing then a number of slot pairs can be specified where the Slave still listens to the master called sniff timeout. This value is set to 2 slot-pairs, and the Slave device listens to the next slot-pair.

7.4.2 Real Test for the AMSQP Scheme

This section deals with the test results that were carried out involving the Ericsson Development boards and the new implementations on the real hardware of the $Gorgon_2$ Development Boards. The test environment to simulate the new scheme was BlueVelvet, which is a test program from Tality. The test process was to send HCI commands down the stack, which would invoke the link manager to send LMP messages to a remote node.

Only two Gorgon2 boards were available for the test. As the Ericsson Development Board were not flashed with an updated version of the stack, they have been utilized as reference boards to show that the scheduling scheme being implemented could be used in conjunction with normal Bluetooth devices. The piconet is illustrated in Figure 7.7 with one of the Gorgon2 board specified as the master with three slaves. Another problem was that the developments boards were not using the round robin mode and thus it was not possible to evaluate the difference between the normal scheme and the AMSQP scheme, but only with the Gorgon2 board.

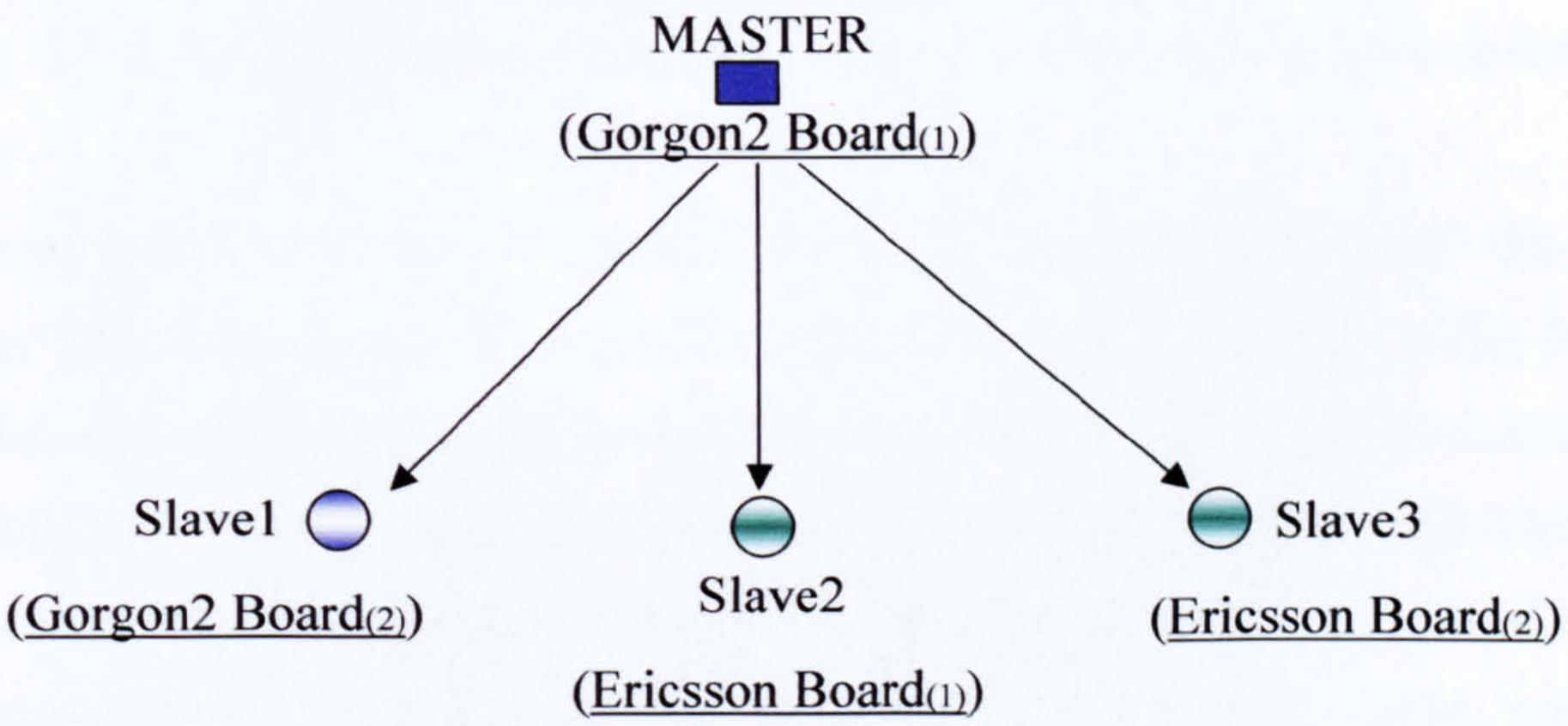


Figure 7.7: Real Simulation of a piconet, with two devices (gorgon2 boards) implemented with the AMSQP scheme.

7.4.2.1 Sniff Mode parameters test for both different Boards

To evaluate the two different boards among the sniff mode parameters, several experiments have been carried out. The first test was to determine if the boards were able to accept a sniff mode with 4 or 6 slots intervals. The second test, was to consider if the boards were capable of changing sniff interval while already in sniff mode.

To provide an answer, the connection was established between two Gorgon2 boards and the Ericsson boards as shown by Figure 7.7, and the results were studied using the Merlin Analyser on Figure 7.8.

From the four first LMP ($LMP_{(0)}$ to $LMP_{(4)}$) messages sent, as shown in Figure 7.8, the Gorgon2 Slave device (Addr=0x1) demonstrates that it accepts sniff intervals of 4 and 6 slots. In addition the $Gorgon_2$ device tolerates a different sniff_req message when already in sniff mode $LMP_{(12)}$ to $LMP_{(13)}$. While in both cases the Ericsson board cannot achieve a low sniff interval ($LMP_{(3)}$ and $LMP_{(5)}$). In fact the Ericsson Board could include a minimum of 14 slots interval within the sniff mode, and to change the parameters of the sniff interval, it should end the sniff mode; come back to active mode and then re-active the sniff mode with the proper interval.

Figure 7.8: LMP Sniff packets transmit between Ericsson and Gorgons boards.

A delay was noted when the LMP sniff_req message was sent and when the Slave accepts it. This delay is not the same for both boards and is around 15ms for the Gorgon boards, while only 5ms for the Ericsson board. This difference could be explained in part because of the extra complexity of the Development gorgon board. Moreover when the NULL and POLL packets were reinserted into the visualisation results, latency could be noted in the scheme between the time the accept message was sent until the actual sniff interval of the ACL link was changed to the new sniff interval. This can be seen in Figure 7.9 at packet 199948, the Time Stamp interval between the precedent packets is not 625µs but 1874µs (3 slots). Another lack of the performance is that the Ericsson with the Gorgon2 does not sequentially poll slaves in a simple round robin scheme but in a random way.

Packet	T	Freq	BTClock	CAC	HDR	Addr	POLL	Idle	Time Stamp
199883	M	2476	2654172			0x1	0x1	483.700 µs	00101.583 8019
Packet	T	Freq	BTClock	CAC	HDR	Addr	NULL	Idle	Time Stamp
199885	S	2475	2654174			0x1	0x0	482.600 µs	00101.584 4280
Packet	T	Freq	BTClock	CAC	HDR	Addr	POLL	Idle	Time Stamp
199887	M	2419	2654176			0x1	0x1	483.700 µs	00101.585 0520
LMP	T	Addr	LMP Len	TID	Opcode	accepted opcode		Time	
1	S	0x1	2	M	accept	sniff_req		101.585s	
Packet	T	Freq	BTClock	CAC	HDR	Addr	POLL	Idle	Time Stamp
199891	M	2411	2654180			0x1	0x1	483.700 µs	00101.586 3020
Packet	T	Freq	BTClock	CAC	HDR	Addr	NULL	Idle	Time Stamp
199893	S	2447	2654182			0x1	0x0	482.700 µs	00101.586 9280
Packet	T	Freq	BTClock	CAC	HDR	Addr	POLL	Idle	Time Stamp
199895	M	2403	2654184			0x1	0x1	483.600 µs	00101.587 5520
Packet	T	Freq	BTClock	CAC	HDR	Addr	NULL	Idle	Time Stamp
199897	S	2451	2654186			0x1	0x0	482.600 µs	00101.588 1780
Packet	T	Freq	BTClock	CAC	HDR	Addr	POLL	Idle	Time Stamp
199899	M	2474	2654188			0x1	0x1	483.800 µs	00101.588 8019
Packet	T	Freq	BTClock	CAC	HDR	Addr	NULL	Idle	Time Stamp
199901	S	2443	2654190			0x1	0x0	482.600 µs	00101.589 4281
Packet	T	Freq	BTClock	CAC	HDR	Addr	POLL	Idle	Time Stamp
199903	M	2472	2654192			0x1	0x1	483.700 µs	00101.590 0520
Packet	T	Freq	BTClock	CAC	HDR	Addr	NULL	Idle	Time Stamp
199905	S	2473	2654194			0x1	0x0	482.700 µs	00101.590 6780
Packet	T	Freq	BTClock	CAC	HDR	Addr	POLL	Idle	Time Stamp
199907	M	2464	2654196			0x1	0x1	483.600 µs	00101.591 3019
Packet	T	Freq	BTClock	CAC	HDR	Addr	NULL	Idle	Time Stamp
199909	S	2465	2654198			0x1	0x0	482.600 µs	00101.591 9279
Packet	T	Freq	BTClock	CAC	HDR	Addr	POLL	Idle	Time Stamp
199911	M	2456	2654200			0x1	0x1	483.700 µs	00101.592 5519
Packet	T	Freq	BTClock	CAC	HDR	Addr	NULL	Idle	Time Stamp
199913	S	2469	2654202			0x1	0x0	482.600 µs	00101.593 1780
Packet	T	Freq	BTClock	CAC	HDR	Addr	POLL	Idle	Time Stamp
199915	M	2448	2654204			0x1	0x1	483.700 µs	00101.593 8019
Packet	T	Freq	BTClock	CAC	HDR	Addr	NULL	Idle	Time Stamp
199917	S	2461	2654206			0x1	0x0	482.500 µs	00101.594 4281
Packet	T	Freq	BTClock	CAC	HDR	Addr	POLL	Idle	Time Stamp
199919	M	2447	2654208			0x1	0x1	483.700 µs	00101.595 0519
Packet	T	Freq	BTClock	CAC	HDR	Addr	NULL	Idle	Time Stamp
199921	S	2418	2654210			0x1	0x0	482.400 µs	00101.595 6782
Packet	T	Freq	BTClock	CAC	HDR	Addr	POLL	Idle	Time Stamp
199923	M	2415	2654212			0x1	0x1	483.700 µs	00101.596 3019
Packet	T	Freq	BTClock	CAC	HDR	Addr	NULL	Idle	Time Stamp
199925	S	2465	2654214			0x1	0x0	482.600 µs	00101.596 9280
Packet	T	Freq	BTClock	CAC	HDR	Addr	POLL	Idle	Time Stamp
199927	M	2431	2654216			0x1	0x1	483.700 µs	00101.597 5520
Packet	T	Freq	BTClock	CAC	HDR	Addr	POLL	Idle	Time Stamp
199936	M	2441	2654232			0x1	0x1	483.600 µs	00101.602 5521
Packet	T	Freq	BTClock	CAC	HDR	Addr	NULL	Idle	Time Stamp
199938	S	2438	2654234			0x1	0x0	482.500 µs	00101.603 1782
Packet	T	Freq	BTClock	CAC	HDR	Addr	POLL	Idle	Time Stamp
199942	M	2455	2654240			0x1	0x1	483.700 µs	00101.605 0521
Packet	T	Freq	BTClock	CAC	HDR	Addr	NULL	Idle	Time Stamp
199944	S	2434	2654242			0x1	0x0	482.700 µs	00101.605 6781
Packet	T	Freq	BTClock	CAC	HDR	Addr	POLL	Idle	Time Stamp
199948	M	2439	2654248			0x1	0x1	483.600 µs	00101.607 5522
Packet	T	Freq	BTClock	CAC	HDR	Addr	NULL	Idle	Time Stamp
199950	S	2430	2654250			0x1	0x0	482.900 µs	00101.608 1779

Figure 7.9: Delay between sniff acceptance and real affectation.

7.4.2.2 AMSPQ Scheme Test Within the Piconet

LMP	T	Addr	LMP Len	TID	Opcode	timing ctrl flags	Dsniff	Tsniff	sniff attempt	sniff timeout	Time
0	M	0x1	10	M	sniff_req	00000000	0 slots	6 slots	1 slots	1 slots	102.370s

LMP	T	Addr	LMP Len	TID	Opcode	accepted opcode	Time
1	S	0x1	2	M	accept	sniff_req	102.385s

Packet	T	Freq	BTClock	CAC	HDR	Addr	DM5	Ack'd	Idle	Time Stamp
236733	S	2434	2703570			0x4	0xE	Unknown	2.983 ms	00113.020 8563

LMP	T	Addr	LMP Len	TID	Opcode	timing ctrl flags	Dsniff	Tsniff	sniff attempt	sniff timeout	Time
2	M	0x2	10	M	sniff_req	00000000	0 slots	396 slots	1 slots	1 slots	114.501s

LMP	T	Addr	LMP Len	TID	Opcode	accepted opcode	Time
3	S	0x2	2	M	accept	sniff_req	114.505s

LMP	T	Addr	LMP Len	TID	Opcode	timing ctrl flags	Dsniff	Tsniff	sniff attempt	sniff timeout	Time
4	M	0x1	10	M	sniff_req	00000000	0 slots	6 slots	1 slots	3 slots	114.506s

LMP	T	Addr	LMP Len	TID	Opcode	accepted opcode	Time
5	S	0x1	2	M	accept	sniff_req	114.524s

Packet	T	Freq	BTClock	CAC	HDR	Addr	DM3	Ack'd	Idle	Time Stamp
239545	M	2429	2819264			0x3	0xA	Unknown	1.734 ms	00115.175 6471

LMP	T	Addr	LMP Len	TID	Opcode	timing ctrl flags	Dsniff	Tsniff	sniff attempt	sniff timeout	Time
6	M	0x3	10	M	sniff_req	00000000	0 slots	396 slots	1 slots	1 slots	116.585s

LMP	T	Addr	LMP Len	TID	Opcode	accepted opcode	Time
7	S	0x3	2	M	accept	sniff_req	116.601s

LMP	T	Addr	LMP Len	TID	Opcode	timing ctrl flags	Dsniff	Tsniff	sniff attempt	sniff timeout	Time
8	M	0x1	10	M	sniff_req	00000000	0 slots	6 slots	1 slots	5 slots	116.602s

LMP	T	Addr	LMP Len	TID	Opcode	accepted opcode	Time
9	S	0x1	2	M	accept	sniff_req	116.617s

Figure 7.10: LMP sniff packets recorded for full piconet transmission using the AMSQP scheme.

To verify the AMSQP capability, a test was prepared, with first connecting the three devices, then one slave, $slave_{(2)}$, enters in a long sniff period, while the two others stay in active mode, and then after some time, $slave_{(3)}$ enters in long sniff interval too.

Figure 7.9 captures the different connection states. The results of the LMP sniff message transmission are shown by Figure 7.9, and according to the first LMP packet sent, it could be noted that as soon as the three devices are connected, $Slave_{(1)}$ and master enter in sniff mode with 6 slot interval. Which means that the slave enter in power saving state by just avoiding to listen to every packet sent by the master, except when the master sends a packet to the slave.

Then, as soon as $Slave_{(2)}$ goes into a longer sniff period, the master indicates to $Slave_{(1)}$ that the next two slots on every cycle are free which leads to an increase in the sniff timeout for two more slots, as illustrated by $LMP_{(4)}$. The same process

occurs when the third $Slave_{(3)}$ goes into a longer sniff period; the sniff timeout increases by two more bits to become equal to 5 slots ($LMP_{(8)}$) which increases the bandwidth for $Slave_{(1)}$.

7.5 CONCLUSION

In this chapter a real implementation of a new intra-piconet scheme based on the AMSQP scheme has been established. The Bluetooth were operated with simple round robin scheme (recommended by the current Bluetooth specification), and shown to be able to accept sniff mode with minimum of 4 slots interval. Therefore by using straightforward simplification on Bluetooth implementation, such as placing all slaves in sniff mode with the adequate sniff interval corresponding to a RR cycle the power could be reduced by at least 15% for a piconet with three slaves, without decreasing the throughput, without increasing delay. This is due to the fact that the slave listens to the master only once during a cycle, which is when the master transmits data to it.

Moreover to have a better fairness within the piconet transmission, with less delay and higher throughput, the sniff timeout parameters should be employed in relationship with the master and Slave queue, as considered during the test.

The contribution of these tests shows that the Bluetooth standardised system for power consumption performs inadequately; two Bluetooth boards from two different companies differ on some basics such as the sniff mode, or Round Robin scheme. This shows to small degree the inattention of Bluetooth SIG over improving the power consumption performance, which could have a significant impact since the wireless market has been influenced by new product similar to Bluetooth such as *"ZigBee" from Motorola, which consumes* less power.

In a comparable way, a lot of research has been carried out to evaluate a new scatternet. The fact that the sniff mode does not function correctly between two supposed identical commercial products show a real need for scatternet standardization. Therefore as shown by the chapter 6, a scatternet could be developed in particular "package", with a specified Bluetooth environment. A primitive

Bluetooth chip might not be able to function in the scatternet if originally it was not designed for this purpose.

CHAPTER 8
CONCLUSION AND FURTHER WORKS

8.1 CONCLUSION

Wireless network have proliferated worldwide, and now represents a fundamental element of our daily lives. Nowadays habitat and business environments are increasingly characterized by the capacity and demand for wireless networks. To meet the divers and insatiable needs of this market, different types of wireless networking are being established.

Bluetooth differs from traditional wired and wireless LAN inter-networking, since it is a point-to-point link formation mechanism, which operates by Time Division Duplex and Frequency Hopping schemes for communication data. This dissimilarity makes Bluetooth unique in terms of connection establishment and network formation. The responsibility for orchestrating frequency synchronization does not depend on a central point of control but on the devices themselves. Hence devices, with absolutely no knowledge of each other's timing or frequency dwelling, are supposed to find each other and to develop a connection network as efficient as possible with no outside helps.

The construct for such large network is referred as Bluetooth scatternet formation. Obviously realizing this challenge suggests the development of a new structure for Bluetooth wireless technology.

This thesis has attempted to solve many of theses issues including topology formation, intra and inter-piconet scheduling and packet routing. The aim of this work, which has never been covered by any research publications previously, was to produce a modern and unique scatternet environment without piconet interferences and requiring no guard time while switching piconet. In such environment, the communication operates with improved fairness, higher throughput and maintains latency under control.

To begin, the thesis has presented an overview of Bluetooth history and a comprehensive view of Bluetooth technology, in Chapter 2.

Research investigations on Bluetooth scatternet topology have been given in *Section 3.2*. Here, the Tree Hierarchy topology is described to ensure single and multi-hop communications with incremental node arrival. To this effect, an innovative Tree Hierarchy structure has been presented in detail in Chapter 3. Currently available scatternet structures suffer intensively from piconets interference degradation, particularly when the piconet numbers expand within the scatternet.

To eradicate these interferences, *Section 3.3* shows that a single root node, the Leader, synchronizes in time and in frequency, all communication between devices affiliate to its tree. A total of 57 piconets (up to 400 devices) can then collaborate without creating interference with each other. Furthermore to improve piconets coordination, *Section 3.4* shows that guard time could be avoided while devices switch from one piconet to another by synchronizing all piconets perfectly. Both time and frequency synchronisation engender a significant impact on the reliability of the scatternet network topology.

Chapter 4 provided review on intra-piconet research, and presented details on the measurement of the power consumption within the piconet. *Section 4.4* described the new flexible adaptive scheme namely AMSQP proposed for improving QoS in Bluetooth intra-piconet. AMSQP is defined as a predictable polled sequence, depending on the queue states of each device. It is described how the combination of two bits adapts the traffic fluctuation to improve the pure Round Robin scheme.

To illustrate working performance of the AMSQP scheme, a simulation model with Poisson arrival traffic, was implemented. The simulation results showed a major improvement in terms of power consumption and fairness.

A review of recent research on inter-piconet scheduling was given in *Section 5.2*, in which the majority of the issues apply a rendezvous tome for bridge scheduling. A predictive inter-piconet scheduling that examines on time-to-time the traffic depending on the queue status of every devices was explained in *Section 5.3*.

The novel inter-piconet scheduling uses an extension of the AMSQP scheme, in which the bridge scheduling is applied for the scheduling of the three different hierarchies. Using scenarios of simple scatternet for each hierarchy, with a Poisson

arrival traffic model, simulations were conducted which showed that the bridge scheduling maintains the performance of the intra-piconet in term of power consumption and fairness within the overall scatternet. The fact that all piconets are perfectly synchronised facilitates the harmonisation of the schedule. The influences of the position of the device in the tree hierarchy do not noticeable affect its performances. The simplicity and performances achievement of the inter-piconet confirms that this scheme is among the scheduling policies of choice for Bluetooth inter-piconet.

In Chapter 6, another significant drawback for constructing Bluetooth scatternet is highlighted in that as yet the forwarding packet concept is not standardized within the piconet. In this Chapter, a forwarding packet method is proposed to enable communication through the master. A simple routing protocol applicable to the entire scatternet is proposed in detail in *Section 6.2*. It is explained how each device could be assigned with a 9 bits address, and how each packet could determine the route path itself. This addressing keeps the overhead network low, and the straightforward routing guarantees that any packets that are forwarded, will reach their destination within a transmission period of six-hops. From a simple model of Bluetooth scatternet simulation, the results shows that the forwarding issues create a bottleneck at the master side when the traffic becomes significant between two separated devices. To prevent bottlenecks, a multi-slave communication is developed. *Section 6.3* explains how any device could get quickly into contact with another device, which is within range. Simulations show a significant enhancement in terms of delay performance. To analyse the effectiveness of the new scatternet, diverse scatternet topologies have been selected which reflect every-day circumstances. These networks are composed of Bluetooth nodes that establish a spontaneous network at system initialization.

In *Section 6.4.1* a scatternet within a lecture room is described. The topology includes a considerable number of devices that must be connected to the lecturer. The fact that the creation of multiple piconets in such a small area, does not affect the scatternet interferences itself is a main advantage of the new scatternet, which required less retransmission, as explained in detail by [Sun *et al [2002]*] where the

inter-piconet interference can significantly reduce the system throughput (by more than 30%).

The scatternet formation along with its routing and forwarding scheme allow the Leader to contact any device. Moreover the fact that the AMSQP can save up to 90% on normal consumption power remains a crucial benefit for the new scatternet, since power in nodes such as "*skinny PDAs*" is extremely limited.

The second topology represents an office with Internet access, in which the main attention converges to the Internet flow rate within the scatternet. *Section 6.2* explains that the scatternet could make use of the DH3 packets type in order to increase the link bandwidth. Additionally every slave could communicate to any device within the scatternet. Thus offers supplementary traffic and thus increases the overall throughput.

A model of the Bluetooth office scatternet with arrival bursty traffic such as CBR traffic model has been encoded. Employing low cycle numbers while slave is sleeping, preserve the QoS for small Bluetooth devices such as mouse, keyboards... while reducing delay and power consumption. The simulation confirms the fairness of the forwarding system and the reliability of the scatternet in the presence of alternative paths.

A real implementation of this innovative scatternet structure has been undertaken and described in Chapter 7. The exploitation of Bluetooth Development Boards along with the source code has provided a better perception of the difficulty behind standardizing a scatternet concept, and highlighted the lack of attention paid by Bluetooth SIG companies towards power consumption. The fact that the Development Boards, based on the Bluetooth concept, have difficulty in interacting properly in some trivial aspects, such as round robin mode and sniff mode, has made the work especially complex.

However, a partial implementation of the AMSQP scheme has been presented in Chapter 7. In *Section 7.4.1* it is shown that the power consumption could be decreased by at least 15%, while the structure preserves the identical throughput, fairness and delay. Further, while applying longer sleeping cycle time, as simulated in Chapter 4, a piconet, with dissimilar traffic rates, should produce a throughput increase along with better fairness. Since the Erickson development does not permit

sniff timeout changes, these could not be evaluated, however this work could be conducted in the future.

In summary this thesis has presented an entirely new protocol for the establishment of multi-hop wireless network based on Bluetooth technology. Starting from a single node called Leader, the protocol ensures proper local topology discovery, allows devices to self organise, following the criterion of the tree hierarchy as given in Chapter 3. The protocol significantly improves the QoS within a piconet using the AMSQP scheme as presented in Chapter 4. The system enables the interconnection of several piconets by exploiting the inter-piconet schedule described in chapter 5, in which several devices could be connected to each other through the routing system given in Chapter 6. Finally, a primary implementation of this concept has been explored in Chapter7. Thus, the entire protocol features contribute to a unique and eminent improvement in Bluetooth scatternet concept.

8.2 FURTHER RESEARCH DIRECTIONS

From this thesis description, it is obvious that the initial implementation on the Bluetooth development board has to be carried on in order to observe the effect of this scatternet structure in real circumstances. The concept of inter-piconet schedules using the AMSQP scheme, while synchronising piconets in time and in frequency can be implemented in a future work along with the new routing strategy.

Furthermore, the fact that Bluetooth is a recent technology, and its scatternet concept is still under development, many areas for further studies and insightful developments are opened.

For example the influence of real time transmission in wireless network is becoming a major concern. And since Bluetooth transmits real time services such as voice through the Synchronous Connection oriented SCO links (with time allocation period of 6 TDD slots account for 64Kbps bandwidth), the quality of the scatternet links is directly related to how well the scatternet implementation is able to deal with critical timing issues such as clock drift overlap. For instance it is not potentially feasible for a device to have SCO links in two piconets at once because over time, the piconet clocks will drift until eventually the two SCO links overlap.

But, in the new scatternet concept presented in this thesis, all masters clocks are synchronized. This brings an opportunity for further researches to study detailed SCO link transmissions between two piconets, which could further reinforce the preference for this new scatternet concept.

Another study should be defined for the discovery process, which requires considerable time consuming, resulting in agreement between nodes that have already been discovered. In the discovery protocol of the proposed scatternet, nodes are operating only using inquiry mode, which avoid "handshakes" between devices part of the scatternet. But instead of having for example three nodes discovering devices at the same time and within the same range, a sequence could be proposed depending on the devices hierarchy position. And a cycle will be predefined in order to discover other devices.

The reduction of interferences caused by Bluetooth on the 2.4GHz frequency range with other unlicensed industrial wireless communications could be the object of further studies.

REFERENCES

Aggarwal *et al [2000]*: A. Aggarwal, M. Kapoor, L. Ramachandran, and A. Sarkar, "*Clustering algorithms for wireless ad hoc networks*", in Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, Boston, Massachusetts, USA, August 2000, pp. 54-63.

Bhagwat *et al [1999]*: P. Bhagwat, A. Segall: "*A Routing Vector Method (RVM) for routing in Bluetooth Scatternet*", in Proceeding in IEEE International Workshop on Mobile Multimedia Communications, (MoMuC), 1999, pp 375 –379.

Bray *et al [2001]*: J.Bray and C. F. Sturman, "*Bluetooth: Connect Without Cables*", Prentice Hall, Englewood Cliffs, NJ, 2001.

Capone *et al [2001]*: A. Capone, R. Kapoor, M. Gerla, "*Efficient Polling Schemes for Bluetooth Picocells*", proceeding of the IEEE International Conference on Communication (ICC 2001), Vol. 7, Finland, June 2001, pp. 1990-1994.

Carey *et al [2001]*: K. Carey, R.A. Guinee and F. O' Reilly "*Bluetooth Enabled Wireless Mouse and Keyboard Interconnectivity*", Institute of Technology, Cork, IRELAND, http://telecoms.eeng.dcu.ie/symposium/papers/A5.pdf .

Chakrabarti *et al [2001]*: S.Chakrabarti, A. Mishra "*QoS Issues in Ad-Hoc Wireless Network*", IEEE Comunications Magazine, February 2001, pp 142-148.

Chakraborty *et al [2000]*: I. Chakraborty, A. Kashyap and A. Rastogi "*Policies for increasing throughput and decreasing power consumption in Bluetooth MAC*", IEEE International Conference on Personal Wireless Communications, 2000, pp 90 –94.

Chakraborty *et al [2001]*: I. Chakraborty, A. Kashyap, A. Kumar, A. Rastogi, H. Saran and R. Shorey, "*MAC scheduling policies with reduced power consumption*

and bounded packet delays for centrally controlled TDD wireless networks", IEEE International Conference on Communications, ICC 2001, Aarhus, Denmark, No. 1, June 2001, pp. 1980-1984.

Chen *et al [2002]*: W. P. Chen and C. J. Hou, "*Provisioning of Temporal QoS in Bluetooth Networks*", 4th IEEE Conference on MWCN, Stockholm, Sweden, September, 2002.

Choi *et al [2002]*, C. S. Choi and H. W. Choi, "*DSR Based Bluetooth Scatternet*", ITC-CSCC 2002, Phuket, Thailand, July 2002.

Chun-Choong *et al [2002]*: F. Chun-Choong and C. Kee-Chaing, "*Bluerings - bluetooth scatternets with ring structures*", in IASTED International Conference on Wirelessand Optical Communication (WOC 2002), Banff, Canada, July 2002.

Cordeiro *et al [March 2004]*: C. Cordeiro, S. Abhyankar, and D. P. Agrawal, "*Design and implementation of QoS- driven dyamic slot assignment and piconet partitioning algorithms over Bluetooth WPANs*", IEEE Conference on Computer Communications, INFOCOM 2004, Hong-Kong, China, Vol. 23, No. 1, March 2004, pp. 1253-1264.

Cordeiro *et al [June 2004]*: C. Cordeiro, S. Abhyankar, and D. P. Agrawal, "*Reducing power consumption and enhancing performance by direct slave-to-slave and group communication in Bluetooth WPANs*" in Elsevier Computer Networks (COMNET) Journal, Vol. 45, No. 2, June 2004.

Cuomo *et al [2003]*: F. Cuomo, G. Di Bacco, T. Melodia, "*Shaper: a self-healing algorithm producing multi-hop bluetooth scatternets*", In IEEE Globecom 2003, San Francisco USA, December 2003.

Das *et al [2001]*: Das, A. Ghose, A. Razdan, H. Saran, R. Shorey, "*Enhancing Performance of Asynchronous Data Traffic over the Bluetooth Wireless Ad-Hoc*

Network", Proceedings of IEEE INFOCOM 2001, Anchorage, Alaska, USA, April 22-26, 2001.

Fabian *et al [1993]*: O.Fabian, H. Levy, "*Polling system Optimization Through Dynamic Routing Policies*", IEEE INFOCOM '93, vol. 1, pp. 194-200.

FAQ *[2004]*: Bluetooth Overview - Bluetooth FAQ, http://www.thewirelessdirectory.com/Bluetooth-Overview/Bluetooth-Overview.htm.

Gerla *et al [2002]*: M. Gerla R. Kapoor, A. Zanella, "*A fair and traffic dependent polling scheme for bluetooth*", ICN 2002, Atlanta, USA, 26-29 Aug. 2002.

Haartsen *et al [Oct 2000]*: J. Haartsen and S. Mattisson, "*Bluetooth –A New Low Power Radio Interface Providing Short-Range Connectivity*" Proceedings of the IEEE, vol. 88, no. 10, pp. 1651-1661, October 2000.

Haartsen *[2000]*: J.C Haartsen Ericsson Radio System, "*The Bluetoooth Radio System*", IEEE Personal Communications 7, Vol.1, Feb 2000 pp 28-36.

Haas *et al [2002]*: Z. Wang, R. J. Thomas, and Z. Haas, "*Bluenet—a new scatternet formation scheme*", in Proceedings of the 35th Hawaii International Conference on System Science (HICSS-35), Big Island, Hawaii, January 7–10 2002.

Infrared *[2001]*: Serial Infrared Physical Layer Specification (IrLAP), version 1.4, Infrared Data Association, Walnut Creek, California, May 30th, 2001.

Johansson *et al [2000]*: N. Johansson, M. Kihl, and U. Körner, "*Performance simulation of TCP/IP over a Bluetooth ad-hoc network*", ITC Specialist Seminar on Mobile Systems and Mobility, Lillehammer, Norway, 2000.

Johansson *et al [2001]*: N. Johansson, L. Tassiulas. "*A Distributed Scheduling Algorithm for a Bluetooth Scatternet*", The Seventeenth International Teletraffic Comgress, ITC'17, Sept.24-28, 2001, Salvador da Bahia, Brazil.

Kalia *et al [99]*: M. Kalia, S. Garg, R. Shorey, "*MAC Scheduling Policies and SAR policies for Bluetooth: A Master Driven TDD Pico-Cellular Wireless System*", MoMuc 99, pp. 384-386.

Kalia *et al [May 2000]*: M. Kalia, S. Garg, R. Shorey: "*Efficient Policies for Increasing Capacity in Bluetooth: An Indoor Pico-Cellular Wireless System*", IBM India Research Laboratory; Indian Institute of Technology, IEEE Vehicular Technology Conf. (VTC 2000-Spring), Tokyo, pp. 907-911, vol. 2, May, 2000.

Kalia *et al [2000]*: M. Kalia, D. Bansal, R. Shorey, "*Data Scheduling and SAR for Bluetooth MAC*", IEEE VTC 2000-Spring Tokyo, pp. 717-720.

Kleinschmidt *et al [2004]*: J. H. Klieinschmidt, M. E. Pellenz and L. A. P. Lima Jr, "*An alternative metric for channel estimation with application in bluetooth scheduling*", 6th IFIP IEEE International Conference on Mobile and Wireless Communication Networks, MWCN'04, Paris, France, October 25-27.

Law *et al [2001]*: C. Law, A. Mehta, K-Y Siu, "*Performance of a new Bluetooth scatternet formation protocol*", Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing 2001, Long Beach, CA, USA, October 2001.

Lilakiatsakun *et al [2001]*: W. Lilakiatsakun, A. Seneviratne, "*Wireless Home networks based on a hierarchical Bluetooth scatternet architecture*", Proceeding of the Ninth IEEE International Conference on Networks 2001, Bangkok, Thailand, October 10-12, pp. 481-485.

Lin *et al [2003$_{(1)}$]*: T. Y. Lin, Y. C. Tseng, and Y. T. Lu, "*An efficient link polling policy by pattern matching for Bluetooth piconets*", Proceedings of the 36th Hawaii

International Conference on System Sciences, HICSS'03, Big Island, Hawaii, January 3-6, 2003.

Lin *et al [2003(2)]*: T. Y. Lin, Y. C. Tseng, K. M. Chang and C. L. Tu, "*Formation, Routing, and Maintenance Protocols for the BlueRing Scatternet of Bluetooths*", Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03), Big Island, Hawaii, January 3-6, 2003.

Liu *et al [2003]*: Yong Liu, Myung J. Lee, Tarek N. Saadawi, "*A bluetooth scatternet route structure for multihop ad hoc networks*", IEEE Journal on Selected areas in communications, Vol.21, No 2, February 2003.

Maric *[2000]*: I. Maric, "*Connection Establishment in the Bluetooth Piconet*", Master of Science Thesis, Electrical and Computing Engineering, New Jersey, US October 2000, www.winlab.rutgers.edu/~ryates/thesis/ivana-ms-final.ps.

Miklos *et al [2000]*: G. Miklós, A. Rácz, Z. Turányi, A. Valkó, P. Johansson: "*Performance Aspects of Bluetooth Scatternet Formation*", Proceedings of the First Annual Worshop on Mobile and Ad Hoc Net working and Computing, MobiHOC 2000, Boston, Massachusetts, USA, August 11, pp. 147-148.

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

Miorandi *et al [2004]*: D. Miorandi, A. Zanella, G. Pierobon, "*Performance evaluation of Bluetooth polling scheme: an analytical approach*", ACM MONET, vol. 9, No. 1, Feb. 2004, pp 63-72.

Misic *[2003]*: V.B Misic: "*Modeling Bluetooth Piconet Performance*". IEEE Comm. Letters, Vol. 7, Jan 2003, pp18-20.

Misic *et al [2004]*: V. B. Misic, E. W. S. Ko, and J. Misic. *" Load And QoS-Adaptive Scheduling In Bluetooth Piconets"*, HICSS-37 Minitrack on Quality of Service Issues in Mobile and Wireless Networks, Hawaii, January 2004.

Nokia *[2003]*: Bluetooth Technology overview, forum Nokia, Version 1.0 April, 2003, http://ncsp.forum.nokia.com/download/?asset_id=219;ref=devx.

Palowireless Bluetooth *[2004]*: Palowireless Ressource Center http://www.palowireless.com/Bluetooth.

Perillo *et al [2003]*: M. Perillo and W. Heinzelman, "*ASP: An Adaptive Energy-Efficient Polling Algorithm for Bluetooth Piconets*", Proceedings of the 36th International Conference on System Sciences, HICSS '03, Big Island, Hawaii, January 6-9, 2003.

Persson *[2004]*: K. E. Persson, D. Manivannan and M. Singhal, "*Bluetooth scatternets: criteria, models and classification*", Accepted for publication, Ad Hoc Networks Journal, Elsevier Science, March 2004.

Petrioli *et al [6-9 May 2002]*: C. Petrioli and S. Basagni "*A scatternet formation protocol for ad hoc networks of Bluetooth devices*" in Proceedings of the IEEE Semiannual Vehicular Technology Conference, VTC Spring 2002, Birmingham, AL, May 6–9 2002.

Petrioli *et al [2002]*: C. Petrioli, S. Basagni, and I. Chlamtac, "*Bluemesh: Degree-constrained multihop scatternet formation for Bluetooth networks*" ACM/Kluwer Journal on Special Topics in Mobile Networking and Applications (MONET), Special Issue on Advances in Research of Wireless Personal Area Networking and Bluetooth Enabled Networks (G. Zaruba and P. Johansson, eds.), 2002.

Petrioli *et al [Nov 2002]*: C. Petrioli and S. Basagni, "*Degree-constrained multihop scatternet formation for bluetooth networks*", in Proceedings of the IEEE Globecom 2002, Taipei, Taiwan, November 2002.

Petrioli *et al [May 2002]*: C. Petrioli and S. Basagni, "*Multiphop scatternet formation for bluetooth networks*", in Vehicular Technology Conference, ser. IEEE 55th, Birmingham, Alabama, vol. 1, May 2002, pp. 424-428.

Prabhu *et al [2002]*: B.J. Prabhu, A. Chockalingam, "*A Routing protocol and energy efficient techniques in Bluetooth Scatternets*", Proceedings of the IEEE International Conference on Communications, ICC 2002, College Park, Maryland, vol. 5, 2002, pp. 3336-3340.

PRS *[2004]*: InterWrite PRS, CalComp, http://www.gtcocalcomp.com/interwriteprs.htm

Rao *et al [2001]*: R. Rao, O. Baux and G. Kesidis, "*Demand-Based Bluetooth scheduling*", presented at the 3rd IEEE Wireless LAN Conference, WLAN 03, Boston, MA, Sept. 2001.

Salonidis *et al [2001]:* T. Salonidis, P. Bhagwat, L. Tassiulas, R. La Maire, "*Distributed topology construction of Bluetooth personal area networks*", Proc. Of the *IEEE Infocom 2001*, Anchorage, Alaska, April 2001, pp. 1577-1586.

Salonis *et al [2000]*: T. Salonidis, P. Bhagwat, L. Tassiulas, "*Proximity awareness and fast connection establishment in Bluetooth*", in First Annual Workshop on Mobile and Ad Hoc Networking and Computing, MobiHOC 2000, Boston, Massachusetts, pp.141-142.

Silicon Wave *[2001]*: "*Bluetooth and Power Consumption: Issues and Answers*", Nov 2001, http://rfdesign.com/images/archive/1101Linsky74.pdf .

Shreedhar *et al [1996]*: M. Shreedhar, G. Varghese, "*Efficient Fair Queuing using Deficit Round Robin*", IEEE/ACM Transactions on Networking, Vol. 4, No. 3, June 1996, pp375-385.

SIG *[2001]*: Specification of the Bluetooth System, Ver.1.1 February 22, 2001, http://www.bluetooth.com.

SLI *[2004]*: System Level of Integration and Source of Concept Explain, http://www.sli-institute.ac.uk/about/soc.htm .

Son *et al [2001]*: L. T. Son, H. Schioler, O. B. Madsen, "*Predictive scheduling approach in inter-piconet Communications*", Proceeding of the 4th international symposium on Wireless personal multimedia communications (WPMC'01), Aalborg, Denmark, Sep 2001.

Sun *et al [2002]*: Min-Te Sun, Chung-Kuo Chang and Ten-Hwang Lai, "*A self-routing topology for bluetooth scatternets"*, in Proceedings of I-SPAN 2002, Manila, Philippines, May 2002.

Stojmenovic *[2002]*: I. Stojmenovic, "*Dominating set based Bluetooth scatternet formation with localized maintenance*", in Proceedings of the Workshop on Advances in Parallel and Distributed Computational Models, Fort Laud erdale, FL, April 2002.

Tan *et al [2001]*: G. Tan, A. Miu, J. Guttag, and H. Balakrishnan, "*Forming scatternets from bluetooth personal area networks"*, Massachusetts Institute of Techonology, http://lcs.mit.edu/, Technical Report. MIT-LCS-TR-826, October 2001.

Tan *et al [2002]*: G. Tan, A. Miu, J. Guttag, H. Balakrishnan, "*An Efficient Scatternet Formation Algorithm for Dynamic Environments*", in *IASTED*

Communications and Computer Networks (CCN), Cambridge, Massachusetts, November 2002.

Technology *[2002]*: Bluetooth Wireless Technology - A presentation, Dr. Farmer (SE 4C03 Winter 2002), http://www.cas.mcmaster.ca/~wmfarmer/SE-4C03-02/projects/student_work/mitrovm.html.

Whitaker *et al [2004]*: R.M. Whitaker, L. Hodge, I. Chlamtac, "*Bluetooth Scatternet Formation: a Survey*", Accepted for publication, Ad Hoc Networks Journal, Elsevier Science, Feb 2004.

Yaiz *et al [2001]*: R. A. Yaiz, G. Hejenk. "*Polling in Bluetooth a Simplified Best Effort Case*" CTIT Workshop on Mobile Communications, ISBN 90-3651-546-7, February 2001.

Yaiz *et al [2003]*: R. A. Yaiz, G. Hejenk. "*Providing Delay Guarantees in Bluetooth*", 23rd International Conference on Distibuted Computing Systems Workshops (ICDCSW'03), Rhode Island, USA, May 19 - 22, 2003.

Yang *et al [2004]*: C-C. Yang, C-F. Liu, "*A bandwidth-based polling scheme for QoS support in Bluetooth*" Journal of Computer Communication, Vol. 27, No.1, Jan 2004.

Zabura *et al [2001]*: Gerely V. Zaruba, Stefano Basagni, and Imrich Chlamtac, "*Bluetrees – scatternet formation to enable bluetooth-based ad hoc networks*", in IEEE International Conference on Communications, ICC2001, Hong-Kong, China, December 2001, pp. 273-277.

Zhu *et al [2004]*: H. Zhu, G. Cao, G. Kesidis, and C. Das "*An Adaptive Power-Conserving Service Discipline for Bluetooth (APCB) Wireless Networks*", Journal of Computer Communications, Vol. 27, pp 828-839, 2004.

Zussman *et al [2003]*: G. Zussman, A. Segall, "*Exact Probabilistic Analysis of the limited scheduling algorithm for symmetrical Bluetooth Piconets*". Proceedings of the 2003 Conference on Personal Wireless Communications (PWC 2003), Venice, Italy, September 2003, LNCS 2775, M. Conti et al. (Eds.) pp. 276-290.

Zussman *et al [2004]*: G. Zussman, L. Har-Shai, R. Kofman and A. Segall, "*Load-Adaptive Inter-Piconet Scheduling in Small-Scale Bluetooth Scatternets*", IEEE Communicatons Magazine, Vol. 42, No 7, Jul 2004, pp. 136-142.