Congestion Control by Sleep Management in Wireless Sensor Networks using Bluetooth Technology

by

Gonapati Rajashekar Reddy

A dissertation submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Faculty of Graduate Studies
University of Manitoba

COMMITTEE SIGNATURE PAGE

This dissertation was presented

by

Gonapati Rajashekar Reddy

It was defended on

Yet to be decided

and approved by

(Signature) ______________________________

Committee Chairperson
Dr. Jelena Misic

(Signature) ______________________________

Committee Member
Dr. Ekram Hossain

(Signature) ______________________________

Committee Member
Dr. Neil Arnason

Abstract

Sensor networks are event driven networks and reliable event detection at the sink (sink is used to inject a query into the network and receive the sensed data from the source) is based on the collective information provided by multiple nodes, known as the source. The event reliability is defined as the number of data packets with the sensing information collected by the sink per second. Due to large information redundancy from the source, protocols offering total end-to-end reliability may transmit more data packets than necessary for reliable event detection. This situation is not desirable since each node is equipped with a battery and has a fixed lifetime. Hence, by efficiently managing the power usage of these sensor nodes, the lifetime of the whole sensor network can be increased. Since sensor nodes are equipped with limited buffer space, the network may be overloaded when a large amount of data is transferred form source to sink, giving rise to congestion in the network and thereby reducing the event reliability observed at the sink. In order to analyze the effect of congestion in a sensor network, I have simulated a wireless sensor network with finite buffers operating on Bluetooth technology. The simulated network was tested without any congestion avoidance or power management algorithms. The obtained results were clearly showing the effects of congestion such as data packet drop rates, end-to-end delays and reduced throughput in the simulated network.

In order to alleviate the above mentioned problems due to congestion in the network, energy efficient congestion control algorithms have been designed. These algorithms operate by forcing the sensor nodes at the source to operate in a power saving mode based on the observed event reliability at the sink. The first algorithm keeps the whole network within the acceptable range of packet losses using the minimal slave activity. In this case source piconets use the information measured at the sink in order to regulate the activity of the slaves. The second algorithm maintains the required (fixed) event reliability at the sink using minimal slave activity. It uses pre-calculated activity values obtained from the analytical and simulation models of the network. The event reliability obtained at the sink, the bridge buffer loss rates, the end-to-end delays and the throughput are analyzed for power-controlled piconets within the sensor scatternet. After analyzing the obtained results I conclude that the designed algorithms significantly reduce congestion and source-to-sink delays, while minimizing packet losses due to finite buffers in the bridge nodes.

Dedication

The people for the very reason I am here in Canada,
people who cant live without me,
people who are offering prayers every day for my good health,
people who have taught me the very essence of life,
people who have put great trust in me,
people who have stood by me when the going was tough,
people who have bought joy into my life,
people who have shared every thing they had,
people who have dedicated their life to their son's and expecting nothing form them,
people who haven't seen their son's for past two years and are counting the days to see them and so do I,
people who are waiting for my success from past 26 years,
people who have done million things to me.

It gives me great pleasure in dedicating this thesis to my father **Mr. Venkatarama Reddy, G.**, to my mother **Srimathi. Rama Devi, G.** and to my brother **Mr. Kishore Reddy, G.**

Acknowledgements

The two years that I have spent researching on Bluetooth and Sensor Network technology's has been challenging at times, but ultimately very rewarding. The success behind this work is equally shared by my advisor **Dr. Jelena Mišić** who was there when it mattered most, helping me to understand the subject, guiding me in the right direction and at times insisting me to think like a researcher. I don't know many other advisors who would take that much time out of their day to attend to their graduate students. For past one year I have seen her almost every day and on weekends when required. The workmanship which I have learnt under her will always help me in future. The full extent of this research could not have been achieved without the help of **Dr. Vojislav Mišić**. I would like to thank him for all his help regarding the Artifix simulator and providing with the Artifix and Bluetooth manuals.

I would also like to acknowledge **Dr. Neil Arnason** with much appreciation for the crucial role he played in shaping my thesis proposal and accepted to supervise my thesis defense. Dr. Arnason inspite of his busy schedule took time to read my thesis proposal and provided with valuable comments regarding the grammar and the proposal writing style. This helped me to submit the thesis proposal in time.

Many thanks are due to **Dr. Ekram Hossain** for accepting to supervise my thesis defense.

Special thanks to **Dr. Peter Graham** and **Graduate Studies Committee** for reviewing my thesis proposal more than once and providing with some valuable comments.

I would like to thank the **Administrative** and the **Systems** staff for all their help for the past two years. Many thanks go to **Mr. Gilbert Detillieux** for all his help regarding the Artifix server.

I want to thank all my colleagues working under Dr. Jelena Mišić and all my friends here in Winnipeg for all their help in any form.

I would like to give my special thanks to **Mr. Raghavendra, E. S.** pursuing his masters in Computational Engineering at University of Erlangen, Germany, **Mr. Jayateertha Joshi, S.** pursuing his masters in Computer Science at National Institute of Technology Karnataka, Surathkal, India, and **Mr. Manjunath, A.** obtained his masters in Mechanical Engineering at Mysore, India. All these guys helped me a lot in various stages of my yesteryears.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A wireless sensor network consists of a number of wireless sensor nodes spread across a geographical location. These networks are mainly used for event detection and reporting; for example, monitoring the levels of humidity in a building [3]. Recent advances in integrated chip production technology have made embedding onboard computation, wireless communication, and sensing in a single tiny chip possible. These tiny chips are known as Wireless Integrated Network Sensors (WINS) [3]. Using WINS, it is possible to develop low-cost, low-power, and multi-functional sensor nodes. A sensor node can communicate, coordinate, and process data and hence, has modernized information gathering for a given sensing task. Wireless sensor networks have a wide variety of applications and can work in hostile environments. The ability of computing, communicating and decision making, helps in analyzing a real-time application. The concept of micro-sensing and wireless connection of these nodes promises many new application areas [3]. Broadly speaking, sensor network applications are classified in four categories they are: Environmental and Agricultural observations, Health applications, Military applications, Home applications, Office and Automobile applications. Sensor networks can be used to prevent the effects of fire hazard in industrial and home locations, tracking and monitoring doctor and patients inside a hospital [3]. Wireless sensor networks have been used in various operations such as surveillance, exploration, communication, intelligence etc. Some of the military applications of sensor networks are battlefield surveillance, monitoring friendly forces, estimating the battle damage, biological and chemical attack detection [3]. Most of the above-mentioned applications of sensor networks involve transfer of sensed data (images) periodically observed at the phenomenon.

Sensor networks are made up of many tiny sensor nodes, which are densely deployed near the site of event occurrence [3]. Nodes in a wireless sensor network possess limited power, computational capability, and memory [41]. When sensor networks are put into operation in hostile environments, node failures may occur. Failure of the node may be due to loss of battery, external disturbances, or physical damages. When a node is out of order due to physical damage or lack of power, neighboring nodes surrounding the failed node should take over its sensing task. Hence, waking up additional nodes and re-routing of the data around the failed node is necessary. Since sensor networks transport different types of traffic, from simple periodic reports to unpredictable bursts of messages triggered by events that are being sensed, The limited buffer size of a node, affects the ability of the network to handle this transient bursts of traffic. Hence, a constant monitoring of loss rates at the buffers and congestion notification is necessary.

Whenever we require some data from the network, a query is injected into the network from a sensor node, known as the sink. The sink propagates the query throughout the network. When a match for the query is made by a group of nodes (called the source), the required actions in response to the query are performed by the source. The data obtained by the source is processed and sent back to the sink. The process of finding a match for the query is called event detection and the process of collecting data and sending it back to the sink is known as data aggregation [3].

Since sensor networks are mainly used for event detection tasks, the rate at which data is propagated from source node to sink must be high enough to obtain the desired event reliability R (R is the number of data packets required per second for reliable event detection at the sink) [2]. For reliable event detection using minimal resources, the packet loss along the path from source to the sink has to be minimized. Some of the main factors to be considered in a sensor network are packet loss due to congestion and the power consumption in order to retransmit the lost packets.

Some of the main features for a sensor network are:

- Ensuring high reliability by combining information from various sources,

- Ensuring reliable transmission of data with minimal power consumption by exploiting redundancy of sensed data,

- Dynamic change of topology when a node failure occurs, and

- Improving the performance of the sensing task by including multiple sensor types.

Reliable event detection using minimal energy resources requires simultaneous achievement of several sub-goals. First, packet loss along the path from source to the sink has to be minimized; at the PHY layer, packets can be lost due to noise and interference, while at the MAC layer, losses may be incurred by collisions. (Since sensors are continuously monitoring the environment and sending data, retransmissions of lost packets are not necessary.) Second, packet waiting time has to be minimized, including queuing delays experienced in various devices along the data path, but also delays due to congestion in the network. (Queueing delays are the responsibility of the MAC layer, while congestion detection and control are performed at the transport layer.) Finally, packet propagation should take place along the shortest paths, while avoiding congested nodes and paths; this is the responsibility of the network layer.

The lower sub-layer of the OSI (Open Systems Interconnect) data link layer is Medium Access Control (MAC), which acts as an interface between a node's Logical Link Control (LLC) and the network's physical layer [40]. A MAC for wireless sensor networks has been developed to assist each node in deciding when and how to access the network. One fundamental task of the MAC protocol is to avoid collisions so that two nodes do not transmit data at the same time over a single channel. Many types of MAC protocols have been developed for wireless voice and data communication networks. Typical examples include the Time Division Multiple Access (TDMA), Code Division Multiple Access (CDMA), and contention-based protocols like IEEE 802.11 [40]. The IEEE 802.15 series of standards can also be used in the MAC layer of a sensor network. However, IEEE 802.15.3 offers high data rate leading to higher congestion in the network. IEEE 802.15.4 (ZigBee) is used for Ultra Low Rate PAN (Personal Area Network). Since IEEE 802.15.4 offers ultra low rate of data transfer, congestion can be avoided, but the event reliability factor R is lowered. IEEE 802.15.1 (Bluetooth) offers medium data rate that suits wireless sensor network applications [2]. Hence I intended to simulate a sensor network operating on Bluetooth technology.

Some of the other reasons to chose Bluetooth technology to build a sensor network are as follows: For reliable event detection using minimal energy resources, the packet loss along the path from source to the sink has to be minimized. This packet loss is affected by all the layers of protocol stack running in the sensor network. At the physical layer, packets can be lost due to the interference. Bluetooth operates in the Industrial, Scientific and Medical band (2.4GHz), together with other networks such as IEEE 802.11b and IEEE 802.15.4 [35]; however, its use of the Frequency Hopping Spread Spectrum, as opposed to the Direct Sequence Spread Spectrum used by those other networks, makes it highly resilient to noise and interference [33]. The raw data rate of 1Mbps (3Mbps in some cases) and the default transmission range of 10 to 100 meters [37] make Bluetooth networks suitable for low cost coverage of sensing areas with diameter of several tens to several hundred meters [2]. In the MAC layer, Bluetooth uses a TDMA/TDD polling protocol where all communication is performed under control of the master. Compared to the collision-based MAC in IEEE 802.11 and IEEE 802.15.4, this MAC is collision free and thus more energy efficient. However, the queuing delays at the bridges and the bridge buffer losses (bridges are used to transfer data between two or more piconets. The issues related to the bridges are discussed in Section 2.1) are affected by the piconet polling algorithm as well as by the bridge management algorithm [29].

Transport protocols provide transparent transfer of data between the end-systems using the services of the network layer to move PDUs (Protocol Data Units) between the source and the sink. Though numerous transport protocols have been proposed for sensor networks, very few of them such as Directed Diffusion [18], Pump Slowly Fetch Quickly (PSFQ) [39], Event to Sink Reliable Transport Protocols (ESRT) [2], and Congestion Detection and Avoidance (CODA) [38] suit the unique functionality of sensing a given phenomenon by multiple sensor nodes. All the above mentioned protocols deal with end-to-end (point-to-point) data transfer and have some drawbacks such as high power consumption, resource utilization, and congestion. Hence, we need to consider them.

At the network layer, routing and transport algorithms are coupled in sensor networks since routing algorithms should choose data paths which avoid congestion, while congestion control is normally the responsibility of the transport layer. Energy management is also placed in one of these two layers. In this work, we assume that routing algorithm is choosing shortest paths, while congestion and energy control are performed together by the same control mechanism at the transport layer. Since sensors are continuously sending data, packet retransmissions are not necessary; however, minimizing the packet losses (and improving the overall energy efficiency) requires that only a minimal number of slaves is kept awake in each piconet. At the same time, the reliability at the sink has to meet application requirements. Therefore the main contribution of this work is the cross-layer integration of congestion control with reliability and energy management in the Bluetooth based sensor network.

In this thesis, as the first phase of my research, I have simulated a wireless sensor network using Bluetooth at the MAC layer. The simulated network is equipped with finite buffers and has no congestion control or power management algorithms. The motivation behind this work was to analyze the effect of congestion on end-to-end delays and data packet loss in the network. Later, two sleep management algorithms are designed based on the outcomes obtained by varying the number of active slaves at the source piconet, in a wireless sensors network operating on Bluetooth technology.

The design, implementation and evaluation of these algorithms on a Bluetooth-based sensor network constitute the second phase of my research. In the simulated wireless sensor network, reliability can be categorized as follows: desired event reliability, relative event reliability and absolute event reliability. The desired reliability is defined as the number of data packets required (in %) for reliable event detection at the sink, where desired reliability is a user specified value which is application dependent. Relative event reliability is defined as the number of data packets received (in %), during decision interval t_i, at the sink, where decision interval is the period of time after which the relative event reliability is reassessed. The absolute event reliability corresponds to the number of packets successfully received per second at the sink. The first algorithm makes use of relative event reliability and desired event reliability and the second make use of the absolute event reliability and desired event reliability. In both of these algorithms, if the relative event reliability (or absolute) is lower than the desired reliability, then the data packets travelling from source to sink are lost at the intermediate bridges due to buffer overflows (congestion) in the network. These algorithms try to put some of the source nodes to power saving mode, thereby reducing the number of packets pumped into the network along the path from source to sink and avoiding congestion and reducing power consumption in the entire network.

The issues I intend to address in a wireless sensor network are reliability control, congestion control, and minimization of power consumption.

The rest of the thesis is organized as follows. In **Chapter** 2, I present some basic features of Bluetooth standards and its Application. In **Chapter** 3, a Bluetooth scatternet without congestion control is introduced and analyzed. In **Chapter** 4, a Bluetooth scatternet equipped with wireless sensor network functionalities is presented and the problems related to congestion and power consumption are discussed. **Chapter** 5 gives an overview of various congestion control protocols employed in sensor networks. In **Chapter** 6, a sleep management algorithm is designed and in **Chapter** 7, the designed algorithm is applied to a sensor network and evaluated. In **Chapter** 8, a new Algorithm to maintain fixed reliability at the sink is introduced and analysed. **Chapter** 9 concludes the thesis. **Appendix** A discusses the implementation of the simulator.

Chapter 2

Overview of Bluetooth Standards & its Application

In this chapter I present an overview of Bluetooth standard from [37] and its application to wireless sensor networks.

2.1 Overview of Bluetooth technology

Bluetooth is an emerging standard for short-range low-cost wireless connectivity for consumer devices and for Wireless Personal Area Networks (WPANs), which enables portable devices to connect and communicate within a short distance. Originally designed to serve as a cable replacement for connections between computers, personal digital assistants (PDA) and other devices, it has grown to become a personal area network (PAN) standard whose applications have been growing over a period of time. It is a frequency hop spread spectrum system intended for worldwide operation in the unlicensed 2.45 GHz Industrial Scientific Medical (ISM) band [10]. It is capable of transmitting both voice and data, and its software stack is designed to allow it to interface with a wide range of applications. Communications proceeds by hopping from channel to channel in a pseudo-random sequence at a rate of 1,600 hops per second [37]. Transmitting and receiving devices must synchronize on the same hop sequence in order to communicate. This high hop rate and the use of short data packets make this technology more robust than other wireless technologies that operate in the ISM band and also more immune to interference from other RF sources, thereby enhance its security [11]. With Bluetooth's reduced interference between devices, multiple hopping sequences can be in use in the same physical area, which allows more devices to share the available bandwidth.

Bluetooth devices are organized into piconets. A piconet is a network that contains up to 255 nodes (slaves), of which only eight may be active and the rest may be in one of the power saving modes such as: sniff mode, hold mode, or park mode [34]. Within the eight active nodes, one node acts as the master while the rest act as slaves. The device that initiates a connection is considered to be the master of a piconet. All slaves are linked together using a specific frequency-hop sequence defined by the master device. In addition, a sensor node operating on Bluetooth technology can act as a slave for two different piconets, or it can act as a master in one piconet and slave in the other [37].

2.1.1 Bluetooth Link Establishment Techniques

The Bluetooth link establishment protocol describes the set of rules by which all Bluetooth devices must abide in order to establish a link to communicate with one another. In this thesis, I assume the the scatternet is in its full functional state at the beginning and no scatternet formation techniques are employed.

PAGE/INQURY Commands: If a device wishes to make a link with another device, it sends out a PAGE message, if the address is known, or an inquiry followed by a page message is sent, if it is unknown. The inquiry method requires an extra response from the slave unit, since the MAC address is unknown to the master unit [11].

2.1.2 Modes of operation

A Bluetooth device can be in any of the five following modes: Standby, Active, Sniff, Hold and Park mode. The Sniff, Hold and Park modes are considered as the sleep or power saving modes.

- **STANDBY Mode:**

  Devices not connected in a piconet are in STANDBY mode. In this mode, they listen for messages every 1.28 seconds over 32 hop frequencies.

- **ACTIVE Mode:**

  In the ACTIVE mode, the Bluetooth unit actively participates on the channel performing data transmission. Active slaves listen in the master-to-slave slots for packets. If an active slave is not addressed, it may sleep until the next new master transmission.

- **SNIFF Mode:**

  Devices synchronized to a piconet can enter power-saving modes in which device activity is lowered. In the SNIFF mode, a slave device listens to the piconet at reduced rate, thus reducing its duty cycle. The SNIFF interval is programmable and depends on the application. It has the highest duty cycle (least power efficient) of all 3 power saving modes.

- **HOLD Mode:**

  Devices synchronized to a piconet can enter power-saving modes in which device activity is lowered. The master unit can put slave units into HOLD mode or the Slave units can also demand to be put into HOLD mode, a hold mode can be established, during which no data is transmitted. The hold mode is typically used when connecting several piconets or managing low-power devices. Data transfer restarts instantly when units transition out of HOLD mode. It has an intermediate duty cycle (medium power efficient) of the 3 power saving modes.

- **PARK Mode:**

  In the PARK mode, a device is still synchronized to the piconet but does not participate in the traffic. Parked devices have given up their MAC address and occasional listen to the traffic of the master to re-synchronize and check on broadcast messages. It has the lowest duty cycle (power efficiency) of all 3 power saving modes.

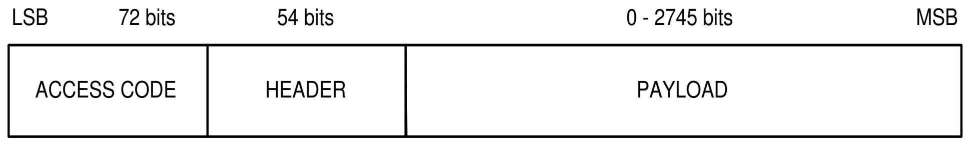


Figure 2.1: General packet format [37].

2.1.3 Packet Format

Each packet consists of 3 entities, the access code, the header, and the payload. The general basic rate data packet format is shown Figure 2.1.

- **Access Code:** Access code are used for timing synchronization, offset compensation, paging and inquiry. There are three different types of Access code: Channel Access Code (CAC), Device Access Code (DAC) and Inquiry Access Code (IAC). The channel access code identifies a unique piconet while the DAC is used for paging and its responses. IAC is used for inquiry purpose.

- **Header:** The header contains information for packet acknowledgement, packet numbering for out-of-order packet reordering, flow control, slave address and error check for header.

- **Payload:** The packet payload can contain either voice field, data field or both. If it has a data field, the payload will also contain a payload header.

2.1.4 Bluetooth Packet Types for SCO and ACL Links

The physical links set up by a Bluetooth piconet determine the packet types used to transmit data. Currently, the Bluetooth standard offers two types of physical links the SCO (Synchronous Connection-Oriented link) and the ACL (Asynchronous Connection-Less link). The SCO link reserves slots and can therefore be considered as a circuit-switched connection between the master and the slave. The SCO packet types are used for time-bounded information transmissions like voice [37]. On the other hand, the ACL links provide packet-switched connections between a master to any slave. To indicate

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

Figure 2.2: SCO and ACL link packet types [37].

the different packets on a link, a 4-bit TYPE code is used as shown in the Figure 2.2. For each of these links, 12 different packet types can be defined. Four control packets will be common for both SCO and ACL, including their TYPE code. Figure 2.2 gives an overview of the packets used for both the links along with their TYPE codes and slot occupancies.

- **NULL Packet:**

  The NULL packet doesn't have an payload, it is a 126-bit packet consisting of the CAC (Channel Access Code) and packet header only. The NULL packet is transmitted by the slave unit and is used to return link information to the master regarding the success of the previous transmission. An acknowledgement to the

NULL packet is not required.

- **POLL Packet:**

  POLL packet is similar to the NULL packet; it does not have any payload. However, an acknowledgement to the POLL packet is required. The POLL packet is initiated by the master. When a slave receives a POLL packet, it must respond with a packet, as an acknowledgement to the POLL packet. The POLL packet can be used by the master in a piconet to poll the slaves, and they must respond on receiving the POLL packet from the master, even if they have no information to transmit.

- **FHS Packet:**

  FHS (Frequency Hopping Synchronization) is a control packet which provides the Bluetooth device address and clock of the sender. Its payload consists of 144 information bits and a 16-bit CRC (Cyclic Redundancy Check) code added to the packet to determine whether the payload is correct or not. The FHS packet is used for frequency hop synchronization before the piconet channel has been established, or when an existing piconet changes to a new piconet.

- **DM1, DM3, and DM5 Packets:**

  DM stands for Data Medium rate. The DM1 can be used to support control messages in any link type (SCO or ACL), it can also carry regular user data. The payload of the DM1 consists of up to 18 information bytes plus a 16-bit CRC code. They are encoded using 2/3 FEC (Forward Error Correction) and the packet can cover up to a single time slot. DM3 and DM5 are used only with ACL link data packet types. The DM3 packets are the same except they have extended payload and can cover up to 3 time slots, and can carry up to 123 information bytes. DM5 packets are the same as that of DM1 and DM3 except that they have extended payload and can cover up to 5 time slots and can hold up to 226 information bytes.

- **DH1, DH3, and DH5 Packets:**

  DH stands for Data High rate. The DH data packets are transferred on ACL links. DH1 packets are similar to DM1 packets, except the information in the payload is not FEC encoded. As a result of this, DH1 packet can carry up to 28 information bytes and covers a single time slot. The DH3 is the same except it can cover up to 3 time slots and contain up to 185 information bytes. The DH5 packet is the same again except it can cover up to 5 time slots and contains up to 341 information bytes.

- **AUX1 Packet:**

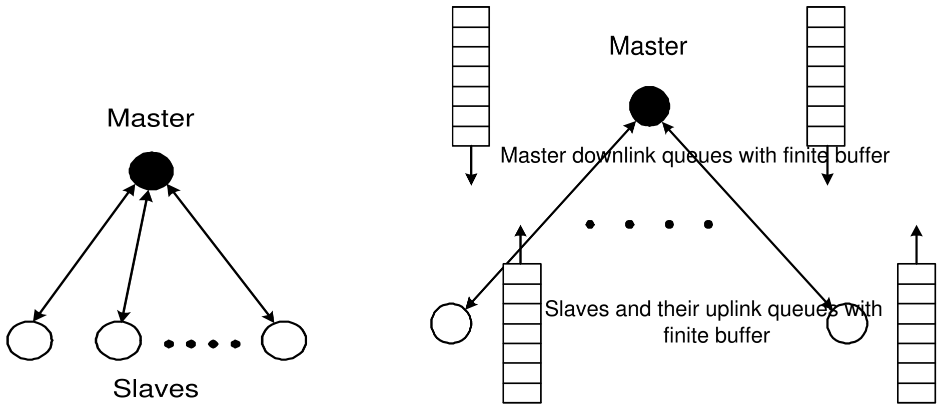
  The AUX1 packet is used to transfer data on the ACL link. An AUX1 packet resembles a DH1 packet except it has no CRC code. As a result it can carry up to 30 information bytes. This packet is not retransmitted if received erroneously.

- **HV1, HV2, and HV3 Packets:**

  HV stands for High Quality Voice and is transmitted on SOC link. The HV packets are used for transmission of voice and other transparent synchronous data which are time-bounded. HV1 packets carry 10 information bytes, which are protected by 1/3 FEC. HV2 packets carry 20 information bytes, and are protected by 2/3 FEC and HV3 packets can carry 30 information bytes, and not protected by FEC. HV packets do not have a CRC or payload header. HV1 packet can carries 1.25ms of speech at a 64 kb/s rate, HV2 packet carries 2.5ms of speech at a 64 kb/s rate and HV3 packet carries 3.75ms of speech with a 64 kb/s rate.

- **DV Packet:**

  DV stands for Data Voice. The DV data packet is divided into voice and data fields, where the voice field is of 80 bits and the data field is of 150 bits. The voice field is not protected by FEC. The data field contains of up to 10 information bytes, which includes a 16-bit CRC. The data field is encoded with a rate 2/3 FEC. The voice and data fields are treated separately. The voice field is handled like normal SCO data and is never retransmitted, on the other hand if data field is erroneous, a retransmission is essential.



(a) Piconet. (b) Queueing model of a single piconet.

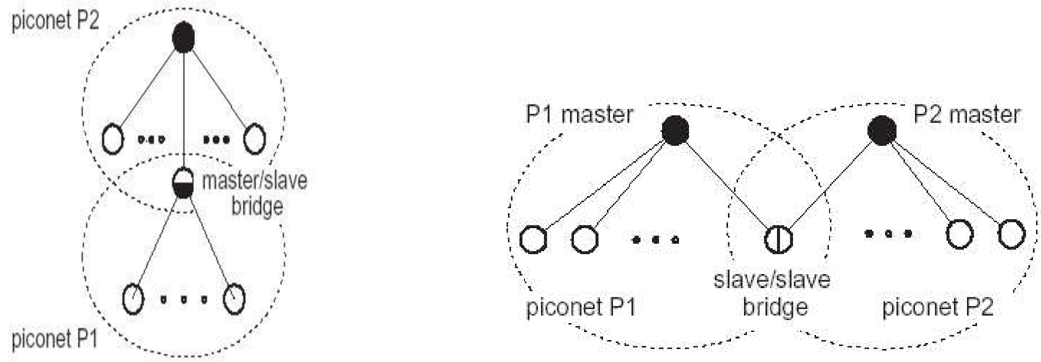
Figure 2.3: Piconet with uplink and downlink queues [27].

2.1.5 Operational Functionalities of a Piconet

Figure 2.3 shows the uplink and downlink queues of a piconet. All communication in Bluetooth devices follow a master-slave scheme, i.e. there is no slave-slave or master-master direct communication. The master determines the hopping sequence and timing, whereas a slave waits for the downlink transmissions from the master as discussed by [27]. The slave responds with an uplink transmission if and only if explicitly addressed by the master. As all communications in a piconet takes place in the form of data packets, both the master and the slaves can transmit packets using 1, 3, 5 time slots as specified in the Bluetooth standard (time slots define number of packets to be transmitted for a given time). Communication that takes place between the slaves of a single piconet is called intra-piconet communication, whereas communication that takes place between the devices of different piconets, via shared devices known as bridges [31], is called inter-piconet communication [27].

2.1.6 Scatternet Formation

Multiple piconets may cover the same area. Since each piconet has a different master, the piconets hop independently, each with their own channel hopping sequence and phase as



(a) Topology with Master/Slave Bridge.

(b) Topology with Slave/Slave Bridge.

Figure 2.4: Scatternet Formation Topologies [27].

determined by the respective master. A Bluetooth device can act as a slave in several piconets, but only as a master in a single piconet. Interconnecting multiple piconets via shared devices (known as bridges) forms a scatternet [14]. The bridge can act as master in one piconet and slave in another or it may act as slave in both the piconets. A group of independent piconets that share at least one common Bluetooth device form a scatternet, where each piconet has a unique frequency-hop sequence [23]. Figure 2.4(a) shows interconnection of two piconets using Master/Slave bridge, and Figure 2.4(b) shows interconnection of two piconets using a Slave/Slave bridge. By joining two or more shared devices (bridges) complex networks can be formed. A piconet can have more than one bridge device. The bridging device keeps on switching between all the piconets it belongs to in a Round Robin fashion. Data packets with destinations in other piconets are queued in the master till the bridge enters its piconet [27]. The actual duration of bridge residence in a piconet and the fashion in which they are scheduled is entirely dependent on the bridge scheduling algorithms. The nominal range limit for two piconet devices to communicate is 10 meters, but in the context of a scatternet this range may be extended to more than 100 meters by using inter-piconet communications [19].

2.1.7 Polling Schemes

The performance of Bluetooth networks largely depends on the polling schemes used to poll the Bluetooth devices in a piconet. The performance of Bluetooth scatternets is largely based on the end-to-end packet delay. It should also satisfy two other requirements. First, the polling scheme chosen must maintain fairness among all the slaves. Each slave must be allocated some fair bandwidth during a piconet cycle, taking the slave's previous traffic into account. The current Bluetooth standards do not recommend any specific polling scheme [5]. Many polling schemes have been proposed for Bluetooth scatternets. However, Bluetooth makes use of specific communication mechanisms:

- All the communications in Bluetooth are bi-directional.
- Initially the master polls each slave with a packet which has no data payloads.
- The data between the slaves must be routed through the master.
- There is no mechanism to update the master regarding the status of the queues at the slave [27].

Due to the above characteristics in Bluetooth communication, the performance of various polling schemes has to be re-assessed. Many of the polling schemes proposed are just modifications of traditional 1-limited and exhaustive service scheduling schemes [1]. Several adaptive schemes have also been proposed. The polling scheme determines the order in which the slaves are to be served. It also decides the number of packets to be exchanged in a single visit. The number of packets to be exchanged can be set to a predefined value or it can be dynamically changed based on the previous history of the slaves. The amount of bandwidth provided for each slave depends on the polling scheme used. Some of the polling schemes proposed and the specific methodology used by them and their drawbacks are discussed below.

Pure Round Robin (PRR) (1-limited service polling)

In this scheme, a fixed cyclic order is defined. A single frame is given to the master-slave queue, when ever poling takes place. This scheme has a high end-to-end delay. This scheme is also known as 1-limited service polling [12].

Exhaustive Round Robin (ERR)

In case of ERR, a fixed order for the scheduling is predefined. But, this scheme is exhaustive and the master does not switch to the next pair until both the master and the slave queues are empty. This procedure does not provide fairness among the slaves, as a single slave with high traffic can monopolize the piconet. This leads to starvation of other active slaves and end-to-end packet delays [17].

E-limited Service Polling

Misic et al. proposed this scheme in [24], in this scheme the master stays with the slave until there are no more packets to exchange or for a fixed number of M (where M determines the maximum number of packets that can be transmitted whenever a master polls a slave). 1-limited and Exhaustive Round Robin are special cases of E-limited scheduling scheme. When M is equal to 1, E-limited scheme acts as 1-limited and when M is equal to infinity, it acts as Exhaustive Round Robin [24].

Exhaustive Pseudo-Cyclic Master Queue Length (EPM)

A dynamic cyclic order is defined at the beginning of each cycle, where each master-slave pair is visited exactly once per cycle according to a decreasing master-to-slave queue length order. One more drawback is, if the bridge is not polled immediately by the master as it enters the piconet, the bridge may end up wasting an average of half the piconet cycle time. The performance of the walk-in bridge scheme depends on the intra-piconet scheme [1].

Walk-in Bridge Scheme

Misic et al. proposed this scheme in [27]. In this bridge scheduling, the master polls its slaves using a chosen intra-piconet polling scheme. In this scheme, the master polls all the slaves in a round robin fashion. If the bridge is present in the piconet while polling, data packets are exchanged between the master and the bridge. Otherwise the master continues to poll the next slave in the piconet. This scheme has a few disadvantages. If the bridge is not present during the polling, some time slots are wasted. However, such delays are negligible when compared to the other delays in the scatternets. Suppose, if the bridge is not available for several piconet cycles, then the performance of the Bluetooth scatternet is drastically reduced. Misic et al. have made use of the E-limited scheme in walk-in bridge scheduling, which overcoming the above mentioned problems.

Though Bluetooth technology appears as a feasible solution for WPANs, a number of issues that affect the performance of Bluetooth scatternets have to be resolved. One important issue to be addressed is the choice of inter-piconet scheduling scheme. Also, all communications that take place in Bluetooth is bi-directional [14]. The master polls a slave using a regular packet (empty packet) through a downlink queue and the slave responds by sending a data packet or a null packet through an uplink queue. The performance of Bluetooth networks depends on the manner in which the master polls the slaves and the bridges in a piconet [24]. Since sensor nodes are error prone, failure of a node may lead to dynamic topological changes.

2.2 Bridge Scheduling using Sniff mode

A scheduling algorithm for bridges in Bluetooth network has been proposed by Wang et al. in [34]. The proposed algorithm tries to solve various problems encountered while building a large scatternet for sensor network applications. All the traffic between various piconets is exchanged via bridges. Hence, the timings of a bridges presence and participation in different piconets need to be carefully scheduled for a smooth operation of a sensor network. This bridge scheduling problem is also known as Rendezvous Point (RP) scheduling in Bluetooth literature [20]. In sensor networks, a node sleeps for most of the time and needs to wake up only when a packet is ready to be sent, or pick up the radio signal when it is the intended receiver for a transmitting packet [34]. The problem of bridge scheduling can be effectively addressed by using a good RP scheduling algorithm. In a scatternet, each pair of sender and receiver needs to be synchronized with each other to make a network work efficiently. Hence, the aim of this paper is to design an RP scheduling scheme which minimizes the power consumptions by effective bridge scheduling.

All the nodes in sensor networks have low duty cycle. Hence, the authors propose to use SNIFF mode as the low duty cycle mode. The SNIFF mode defines a regular recurrent wakeup/sleep cycle, which is suitable for low duty cycle devices. The SNIFF mode in Bluetooth enabled network consists of the following parameters: T*sniff*, D*sniff*, N*sniff attempt*, and N*sniff timeout* [34].

- T*sniff* : Represents the length of the sniff cycle also known as the superframe.
- D*sniff* : Represents the time at which a device needs to wakeup within the superframe.
- N*sniff attempt* : Provides with the time the node should be active.
- N*sniff timeout* : Provides with the time the node should sleep.

Bluetooth specification defines that at slot D*sniff* within a superframe, the device is woken up (activated) to listen to incoming packets for N*sniff attempt* slots. After receiving a packet, it waits for N*sniff timeout* slots for the continuing packets. The algorithm calculates the RP schedule for a bridge, based on local information. It is processed as follows: when a piconet is formed, the master decides a pair of RPs for its piconets, which are a half superframe apart. Once this is done, a slave initializes the SNIFF negotiation process by picking up an arbitrary D*sniff* for the SNIFF request message from a master. The master picks an RP close to the RP supplied by the slave, and replies with the modified SNIFF parameter. Once a link is established between the bridge and the second master, the bridge sends a SNIFF request, the procedure in acquiring RP for first master is followed, provided the second master is assigned with a different RP. If in case the bridge detects that the two RPs it owns for different piconets are likely to drift towards each other and may interfere with each other, the bridge renegotiates for a new RP with one of the masters. Following the above procedure the authors try to achieve a conflict free RP scheduling scheme for sensor networks operating on Bluetooth technology.

Some of the drawbacks in the above model are as follows: whenever a node goes for a lengthy sleep time, the bridge needs to turn on its receiver 8.2 slots in advance in order to synchronize with the master, resulting in wastage of power [37]. In Bluetooth enabled networks, the bridges act as the backbone of the network and most of the time they are busy carrying the traffic. Hence, the bridges need to shift from the sniff mode to active mode often, resulting in a faster drift in RP, needing renegotiation often. Since the sensor nodes are equipped with minimal energy and computation capabilities, lot of energy is wasted in calculating the RPs, comparing the RPs, and for a constant renegotiation processes. For any given sensor network there are more ordinary slaves than bridges, and the authors have failed to specify the power saving techniques for the slaves. Power saving of ordinary (non-bridge) slaves is an important issue that needs to be addressed.

2.3 Bluetooth and Sensor Networks: A Reality Check

An examination of the suitability of Bluetooth as a MAC interface for sensor networks has been carried out by Leopold et al. in [22]. Performance of sensor nodes rely not only on the type of transmitting signals they are using, but also on the protocol stack that they operate on. The two types of transmitting signals existing for sensor nodes are fixed frequency carriers and spread spectrum transmissions.

In fixed frequency carriers, all nodes within the communication range compete for the same channel to transmit data. Whereas in spread spectrum transmissions, nodes within the communication range communicate or transmit data using different channels. Bluetooth radios (transmitting signals) make use of spread spectrums for communications. Bluetooth favors connectionless data transfers. Hence, Bluetooth is a feasible option for sensor networks. Leopold et al. have designed and implemented a tiny Bluetooth stack for TinyOS. The experiments were conducted on actual Bluetooth-based devices known as BTnodes developed at ETH Zurich [8]. These BTnodes were equipped with two radios to enable multihop networking. The network was tested for throughput and energy consumption. These results suggest that Bluetooth based sensor networks could be appropriate for event-driven applications that exchange bursts of data for a limited time period.

Chapter 3

A Bluetooth Scatternet without Congestion control

In the first phase of my research, a Bluetooth scatternet of triangular topology has been simulated, where all the piconets have 3 slave nodes. The intent behind this work was to analyze the impact of finite buffers on the performance of Bluetooth scatternet operating under E-limited intra-piconet polling and walk-in bridge scheduling and in the absence of congestion control or power management algorithms. The simulation results obtained from this initial exploratory work demonstrate the impact of finite buffers on end-to-end delays between the piconets, bridge buffer loss rates, slave buffer loss rates and throughput for each piconet in the simulated Bluetooth network.

3.1 Introduction

It is well known that network performance may suffer when the buffers through which the packets must pass have finite sizes; the consequences may range from negligible to disastrous. When a buffer is full, some of the packets arriving from the application or from the network will be dropped and, effectively, removed from the network due to buffer overflows. These dropped packets will have to be retransmitted, which increases the effective packet arrival rate. At low loads, such an increase may go unnoticed; at high loads, a feedback effect may occur in which packet drops lead to more packets being sent and even more of them being dropped, until the traffic reaches saturation, while packets that manage to get through will experience increased end-to-end delays.

This important aspect of network operation has so far been ignored in the context of Bluetooth networks, in particular in Bluetooth scatternets, where authors have generally focused either on scatternet formation [7] or scatternet scheduling [6, 26, 30, 36], with most of the latter work done assuming that buffers with infinite capacity are available. This assumption, while useful as the first approximation, is simply unrealistic. Not only will those buffers be finite, but most of them will be rather small in absolute terms, in particular those on mobile devices where energy consumption is directly proportional to the chip real estate.

In this exploratory work, I have analyzed the impact of finite baseband buffers on the performance of a Bluetooth scatternet. I first present a brief overview of the queuing model used and other assumptions. This is followed by a discussion on the packet blocking rate at the slave uplink queues and bridges, respectively. Throughput and end-to-end packet delays are also calculated and presented. Some practical recommendations regarding buffer size allocation conclude the analysis on my initial exploration of the Bluetooth scatternet with finite buffers.

3.2 Scatternet operation

Regarding scatternet operation, I assume that both piconet masters poll their slaves using E-limited polling scheme. At most M_s data packets are exchanged between the master and an ordinary slave during a single master's visit to a slave, and all piconet masters use the same value for M_s. This polling scheme has been shown to offer lower end-to-end delays for bursty traffic than either exhaustive or 1-limited service [25]. Furthermore, delays can be minimized if the polling parameter is set to $M_s \approx \overline{B}$ (Where $\overline{B}$ is the mean packet burst generated by each slave), in which case packet bursts are mostly preserved from uplink to downlink queues, which simplifies routing and flow control [25].

I consider a scatternet in which bridge scheduling is performed according to the walk-in scheduling policy without predefined rendezvous points, shown schematically in Figure

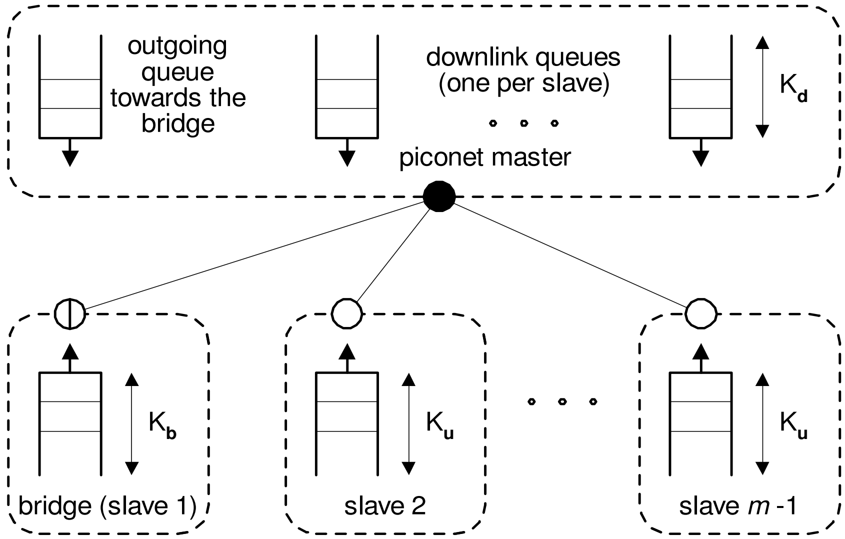


Figure 3.1: Queues are implemented with finite size buffers.

3.2 [28]. While simple to implement, this scheduling policy offers good performance and excellent scalability [27]. A Timing diagram (clock pulse) for walk-in bridge scheduling is shown in Figure 3.2 with clock pulse going low and high for the bridge and the piconets P_1 and P_2 it is associated with. If the clock pulse for bridge is high then it is associated with P_2 and if the clock pulse for bridge is low then it is associated with P_1. If the clock pulse is high for the bridge and low for P_2, then the bridge is being served by the master of P_2 or else if the clock pulse is low for the bridge and high for P_1, then the bridge is being served by the master of P_1. If the clock pulse go high for P_1 and if the bridge pulse is high indicates that P_1 is trying to check for the bridges presence in the piconet.

Under walk-in scheduling, the bridge switches between the piconets in a round robin fashion without a predefined schedule. When present, the bridge is polled just like any other slave, except that the first packet to be sent to the bridge is always an empty POLL. The absence of a response to the initial POLL packet means that the bridge is not present, in which case the master simply moves on to the next slave. If the bridge is present, it responds with a NULL; the master then starts the exchange with actual data packets.

The exchange lasts for M_b packets, or less if both queues are emptied, which is detected

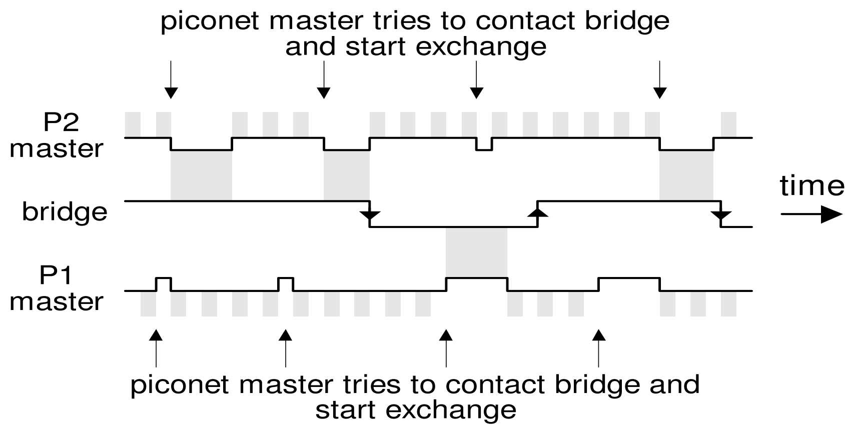


Figure 3.2: Scatternet operation under walk-in scheduling.

through a NULL response from the bridge to a POLL packet from the master. All piconet masters use the same value for M_b, but separate values for each piconet, or even each bridge, may be readily accommodated by my model. As soon as the exchange is finished, the bridge leaves the piconet and joins another one (this policy is referred to as *limited exchange* in [28]).

I assume that different queues, shown in Figure 3.1, are implemented with finite size buffers. The buffer lengths for uplink, downlink and bridge queues are denoted with K_u, K_d, and K_b, respectively. For simplicity, queue lengths will be expressed in baseband packets, but conversion to Kbytes would be straightforward. Some of the other factors which will be used to test the model are as follows: M_b and M_b the slave and bridge E-limited factors, P_l the locality factor which defines the local and non-local traffic, λ defines the packet burst arrival rate for each slave and BRT defines the number of piconet cycles the bridge resides in a piconet.

3.3 Scatternet topology

A Bluetooth scatternet simulator operating at the MAC level has been built using the object-oriented Petri Net-based simulation engine Artifex by Artis Software Inc. [13] running on a Linux platform. I have considered a symmetric topology with six piconets

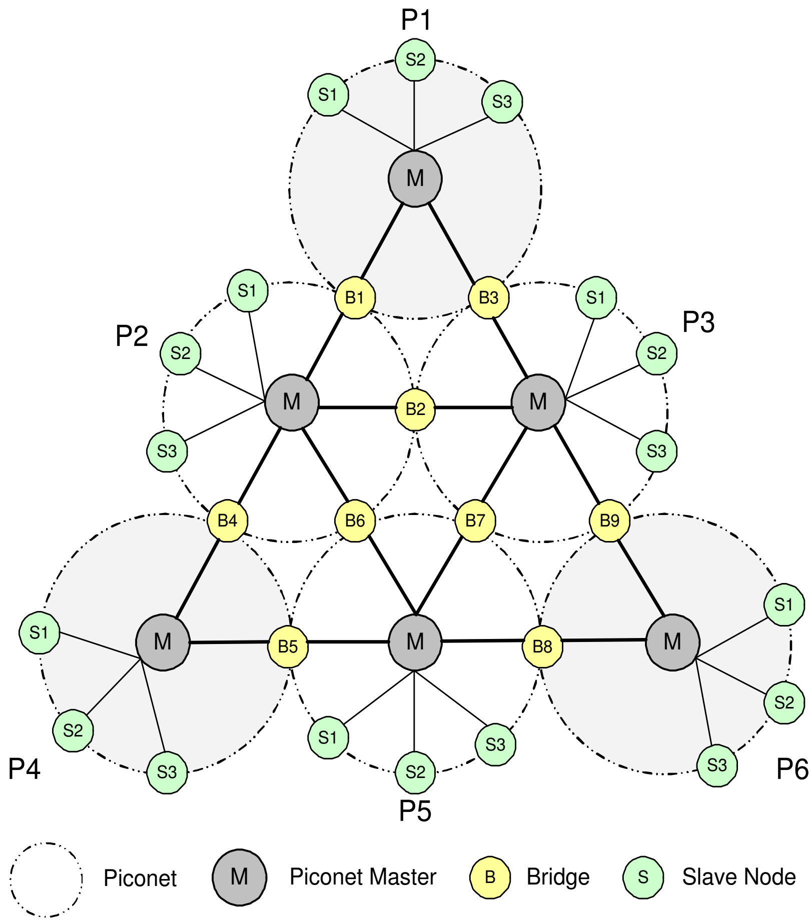


Figure 3.3: Topology of the scatternet under consideration.

shown in Figure 3.3. The piconets P_1, P_4, and P_6 (which will be hereafter referred to as 'exterior' piconets) have two bridges each, while the piconet P_2, P_3, and P_5 (the 'interior' piconets) have four bridges each. The bridges use the walk-in scheduling with the limited exchange policy.

In addition to bridges, each piconet has three ordinary slaves. Each slave generates traffic which targets other slaves in the same piconet with the probability of P_l, and slaves in any other piconet with the probability $(1 - P_l)/5$. In case of nonadjacent piconets, the inter-piconet traffic is routed through the shortest path. When two such paths exist (e.g., traffic from piconet P_1 to piconet P_5 may go through P_2 or P_3), traffic is split evenly between those paths.

Each slave generates packets in bursts with mean burst size of $\overline{B} = 3$, and five-slot DH5 packets are used throughout the simulation [37]. All devices are assumed to use the same segmentation/reassembly protocol, therefore the mean burst size has the same value for all devices. Traffic locality was set to $P_l = 0.6$, and the values of the polling parameters were $M_s = 3$ and $M_b = 15$, unless specified otherwise. The buffer sizes, when fixed, were $K_d = 100$, $K_b = 40$, and $K_u = 30$, for master downlink queues, bridge outgoing queues, and slave uplink queues, respectively.

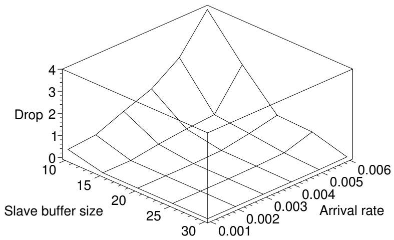
Although such a symmetric topology is unlikely in practice, it can nevertheless serve as a 'stress test' setup in which the performance of Bluetooth scatternets under walk-in scheduling and the impact of the finite buffer sizes on said performance, can be readily assessed.

3.4 The impact of finite uplink buffers

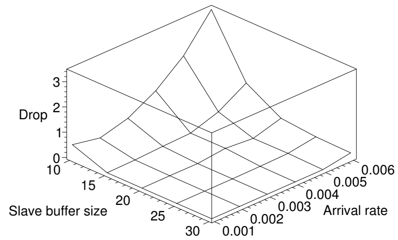
The first set of simulation results shows the impact of finite size of buffers that implement the slave uplink queues. The size of the slave buffers K_u was varied between 10 and 30 baseband packets, and the packet burst arrival rate λ was varied between 0.001 and 0.006 (burst arrivals per Bluetooth slot). The polling parameter for the bridges was fixed at $M_b = 15$.

The diagrams in Figure 3.4 show the packet drop rates at the slave, expressed in percents, when the slave polling parameter M_s is varied. Higher values of M_s mean that more packets will be transmitted to the master during a single visit, and the uplink queues will be serviced faster. Consequently, the packet drop rate will decrease. Increasing the slave buffer size has the same effect.

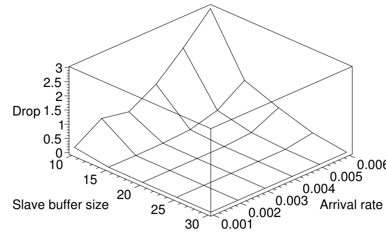
Note that the drop rates are higher in the interior piconets on account of their higher traffic load. Each of the interior piconets have three ordinary slaves that generate traffic and bridges that carry the inter-piconet traffic. However the exterior piconets have only the traffic of their own (incoming and outgoing), while the interior ones also route the traffic between other piconets. As a consequence, their traffic load will be much higher, nearly twice as much in the setup I have used, and drop rate will be higher as well.



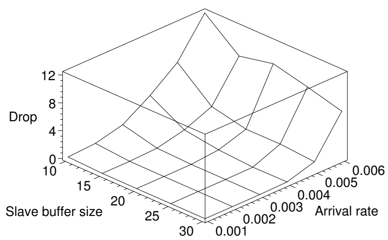
(a) Exterior piconet, M_s = 2.



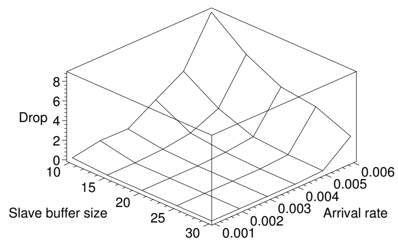
(b) Exterior piconet, M_s = 5.



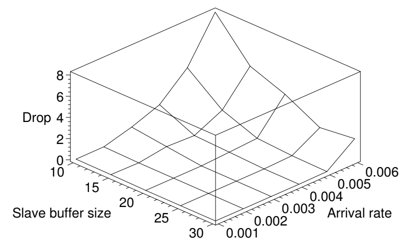
(c) Exterior piconet, M_s = 8.



(d) Interior piconet, M_s = 2.



(e) Interior piconet, M_s = 5.



(f) Interior piconet, M_s = 8.

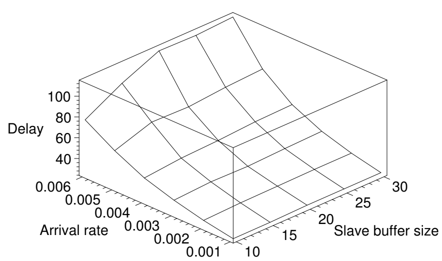
Figure 3.4: Slave buffer drop rate as a function of the polling parameter $M_s = 2,\ 5,\ 8$ [Arrival rate = 0.001 (0.001) 0.006, $K_u = 10$ (5) 30, $M_b = 15$, $P_l = 0.4$, $BRT = 1$, $K_d = 100$ and $K_b = 40$].

The diagrams in Figure 3.5 illustrate the access delay at the slave uplink buffer, expressed in Bluetooth time slots $T = 0.625ms$, averaged over the piconets in the exterior and interior group, respectively. As can be seen, increasing the slave buffer size leads to a increase in access delay, and the increase of the slave polling parameter M_s has leads to a decrease in the delays. As before, the delays are much higher in the interior piconets due to their higher traffic load, which leads to longer cycle times [25] and these lead to higher access times and end-to-end delays.

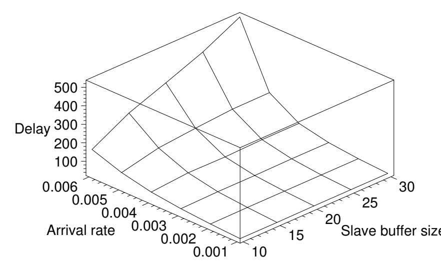
3.5 The impact of finite bridge buffers

The second set of experiments investigates the packet drop rate at the bridge as a function of the bridge polling parameter M_b. In this case, I have also varied the bridge buffer size from $K_b = 10$ to 20 baseband packets, as well as the locality probability from $P_l = 0.2$ to 0.8. I have considered the two bridge buffers in the bridge B_1 from Figure 3.3 as representative for the bridge behavior in exterior and interior piconets, respectively.

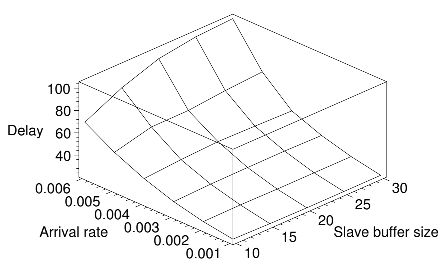
The diagrams in Figure 3.6 show the packet drop rate at the bridge buffer as a function of the bridge polling parameter M_b. As could be expected, the blocking probability decreases with the polling parameter M_b. Namely, more packets per exchange will empty the bridge queue faster, and the probability that both the bridge queue and the corresponding outgoing queue at the master will empty increases. Furthermore, emptying the



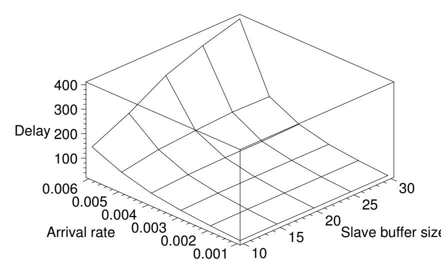
(a) Exterior piconet, $M_s = 5$.



(b) Interior piconet, $M_s = 5$.

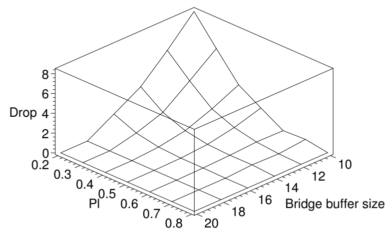


(c) Exterior piconet, $M_s = 8$.

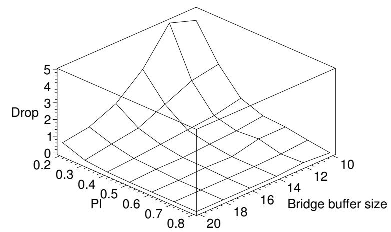


(d) Interior piconet, $M_s = 8$.

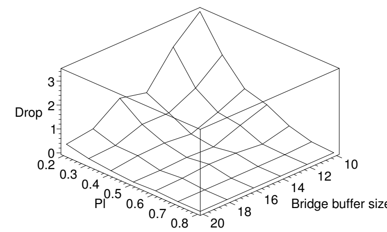
Figure 3.5: Access times at the slave $M_s = 2$, 5 [Arrival rate = 0.001 (0.001) 0.006, K_u = 10 (5) 30, $M_b = 15$, $P_l = 0.4$, $BRT = 1$, $K_d = 100$ and $K_b = 40$].



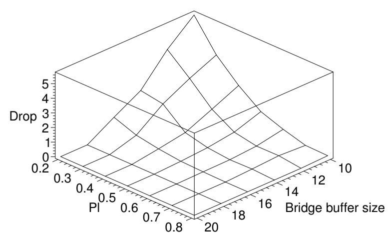
(a) Bridge from P_1 to P_2, $M_b = 9$.



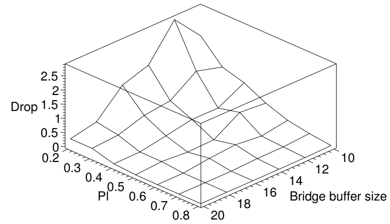
(b) Bridge from P_1 to P_2, $M_b = 12$.



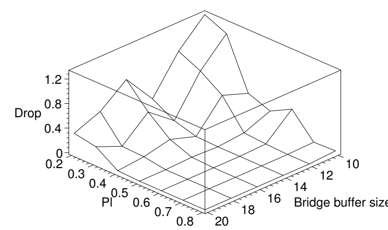
(c) Bridge from P_1 to P_2, $M_b = 15$.



(d) Bridge from P_2 to P_1, $M_b = 9$.



(e) Bridge from P_2 to P_1, $M_b = 12$.



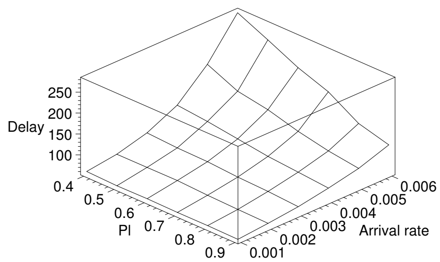
(f) Bridge from P_2 to P_1, $M_b = 15$.

Figure 3.6: Bridge buffer drop rate in % from P_1 to P_2 and from P_2 to P_1 as a function of the polling parameter $M_b = 9,\ 12,\ 15$ [$K_b = 10\ (2)\ 20$, $P_l = 0.2\ (0.1)\ 0.8$, $M_s = 3$, Arrival rate $= 0.002$, $BRT = 1$, $K_d = 100$ and $K_u = 30$].

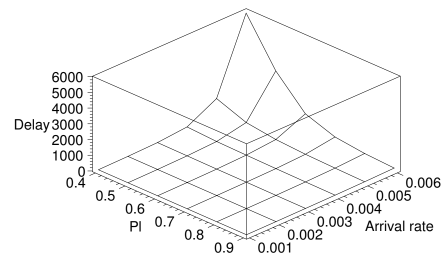
bridge will cause the bridge to terminate its residence and switch to another piconet. If the frequency of such events increases, the piconet cycle time will decrease, leading to a decrease of slave buffer drop rates, access delays, and end-to-end delays for both local and non-local traffic.

The decrease of traffic locality means that more traffic will be sent through the bridges, and packet drop rates will increase. For values of P_l above 0.7 (which correspond to 70:30 ratio of local vs. non-local traffic), the drop rates are below 1%.

It may be noted that the packet drop rate is lower at the bridge that 'resides' in the interior piconet P_2, despite its higher total load and longer piconet cycle time. This is caused by the difference in bridge traffic. Namely, there are four bridges in an interior



(a) Within exterior piconets.



(b) Within interior piconets.

Figure 3.7: End-to-end packet delays for local traffic [Arrival rate = 0.001 (0.001) 0.006, $P_l = 0.4$ (0.1) 0.9, $M_s = 5$, $M_b = 15$, $BRT = 1$, $K_d = 100$, $K_b = 40$, and $K_u = 30$].

piconet, as opposed to only two in an exterior one, and the traffic per bridge is lower, which leads to lower drop rates.

3.6 End-to-end packet delays

The next set of simulation results illustrate end-to-end delays for intra-piconet and inter-piconet (i.e., local and non-local) traffic. The slave buffer size was fixed to $K_u = 30$ and the traffic locality P_l was varied in the range $P_l = 0.4 \,..\, 0.9$. The packet burst arrival rate λ was varied in the range $0.001 \,..\, 0.006$ bursts per Bluetooth time slot. The polling parameters were fixed at $M_s = 5$ for ordinary slaves, and at $M_b = 15$ for the bridges.

End-to-end delays for local (intra-piconet) traffic, averaged over all exterior and interior piconets, respectively, are shown in Figure 3.7; as before, units of Bluetooth time slots $T = 0.625ms$ are used. The interior piconets can be seen to exhibit much higher intra-piconet delays, but this is to be expected since the traffic load of those piconets is higher due to longer piconet cycle times.

End-to-end delays for non-local (inter-piconet) traffic are shown in Figure 3.8. The corresponding delays were averaged over all the alternative paths that lead from an exterior piconet to another exterior one, and from an interior piconet to another interior one, respectively.

(a) Between exterior piconets.

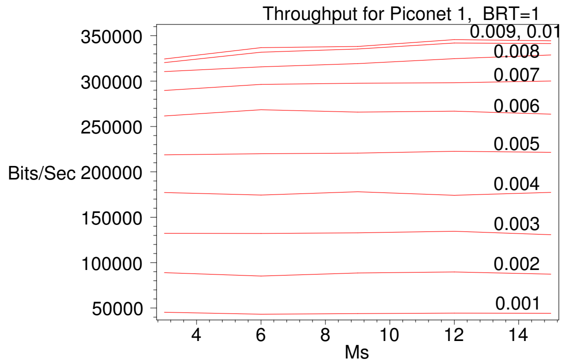
(b) Between interior piconets.

Figure 3.8: End-to-end packet delays for non-local traffic [Arrival rate = 0.001 (0.001) 0.006, P_l = 0.4 (0.1) 0.9, $M_s = 5$, $M_b = 15$, $BRT = 1$, $K_d = 100$, $K_b = 40$, and $K_u = 30$].

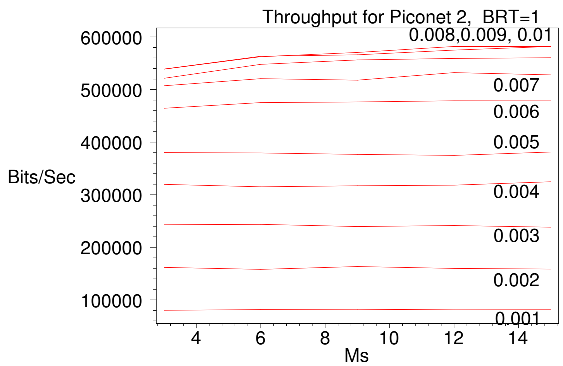
As in the case of local delays, exterior piconets exhibit lower delays than the interior ones, due to their smaller aggregate traffic load. However, the diagrams show that the bulk of the delay experienced by the traffic from exterior to another exterior piconet is, in fact, incurred whilst passing through one or the other of the interior piconets. Note that the traffic from an exterior piconet to another exterior piconet must pass through one of the interior piconets, and thus has to experience a total of six hops: from source slave to the exterior master, then to the bridge, from the bridge to the interior master, then to another bridge, from the bridge to another exterior master, and finally from that master to the destination slave. The traffic from an interior piconet to another interior piconet will, however, experience only four hops, since it will have to pass through a single bridge only.

3.7 Throughput

My final experiment shows the throughput through the piconets. Throughput for a piconet is defined as the rate at which the sensed data packets can be sent and received through that piconet, measured in bits per second. The slave polling parameter value was varied in the range of $M_s = 3..15$, while the corresponding bridge polling parameter



(a) External piconet (P_1).



(b) Internal piconet (P_2).

Figure 3.9: Throughput in the example scatternet for P_1 and P_2 [$M_s = 4$ (2) 14, Arrival rate = 0.001 (0.001) 0.01, $M_b = 15$, $P_l = 0.6$, $BRT = 1$, $K_d = 100$, $K_b = 40$, and $K_u = 30$].

was kept constant at $M_b = 15$. The packet burst arrival rate per slave was varied in the range of $\lambda = 0.001..0.01$ packet burst arrivals per Bluetooth time slot, and traffic locality was fixed at $P_l = 0.6$. Buffer sizes were kept at values $K_u = 30$, $K_d = 100$, $K_b = 40$.

As can be seen from Figure 3.9, the throughput for interior piconets is higher when compared to the exterior ones, which is again due to their higher traffic load. I note that the maximum throughput in interior piconets is around 600Kbps, which is about two-thirds of the theoretical maximum with five slot packets [37]. This difference seems

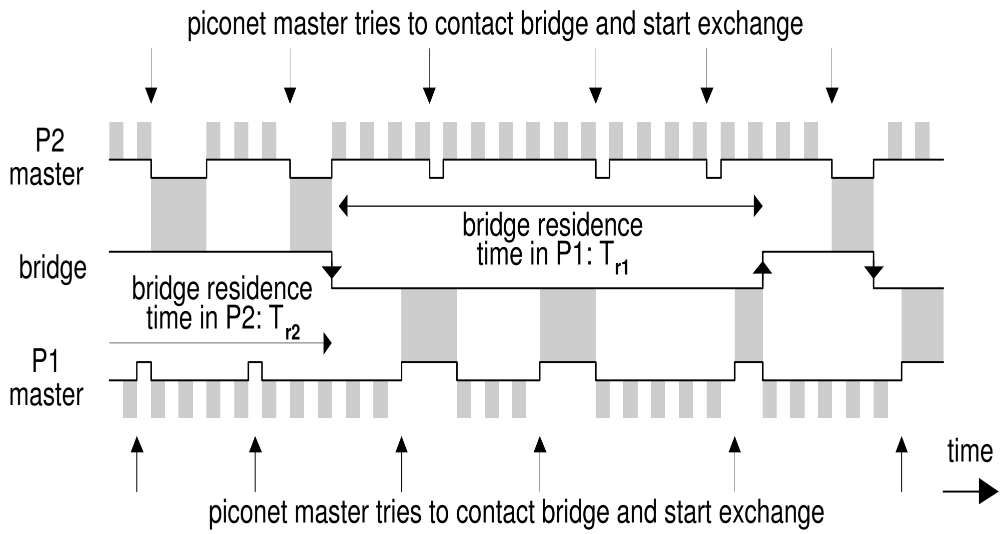


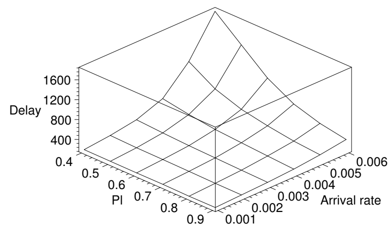
Figure 3.10: Bridge Residence Time (BRT) in a Piconet.

reasonable in view of the losses due to packet waste when the bridge is absent from the piconet and when there are no packets to exchange with a slave or a bridge, and packet blocking due to buffers filled up to capacity.

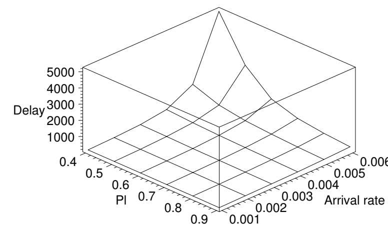
3.8 Effect of BRT on Network Performance

Figure 3.10 shows how the bridge moves back and forth under walk-in bridge scheduling scheme between the two piconets it is participating. The Bridge Residence Time (BRT) defines the amount of time the bridge resides in a piconet. BRT is represented in terms of piconet cycles, if BRT is set to 1 then the bridge resides for 1 piconet cycle and is served by the master only once before moving to the other piconet. If BRT is set to 2 then the bridge resides for 2 piconet cycles and is served twice in a round robin fashion before moving to the other piconet. In this section I analyze the effect of the BRT on slave buffer drop rates, bridge buffer drop rates, end-to-end delays and throughput on the designed Bluetooth network.

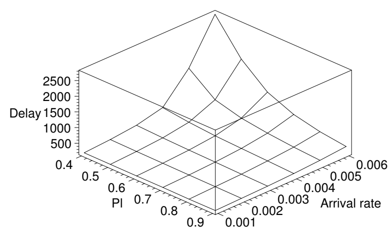
Figure 3.11 represent the end-to-end packet delays for non-local traffic between the interior and the exterior piconets. The packet delays are calculated by taking a mean



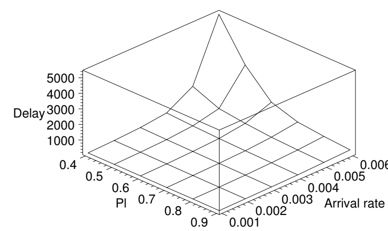
(a) Between exterior piconets, $BRT = 1$.



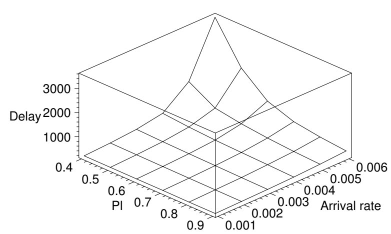
(b) Between interior piconets, $BRT = 1$.



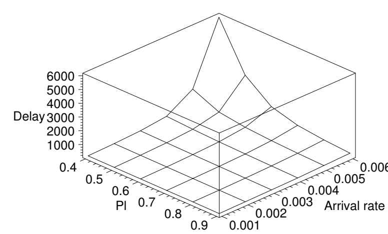
(c) Between exterior piconets, $BRT = 2$.



(d) Between interior piconets, $BRT = 2$.



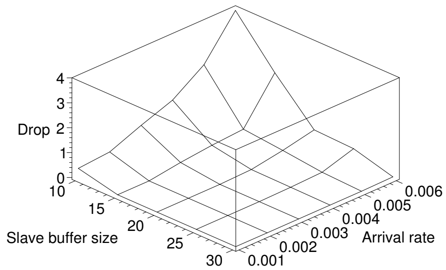
(e) Between exterior piconets, $BRT = 3$.



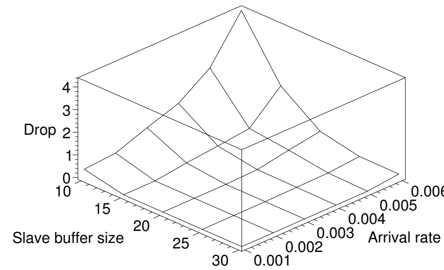
(f) Between interior piconets, $BRT = 3$.

Figure 3.11: End-to-end packet delays Between exterior and interior piconets $BRT = 1$, 2, 3 [Arrival rate = 0.001 (0.001) 0.006, $P_l = 0.4$ (0.1) 0.9, $M_s = 5$, $M_b = 15$, $K_d = 100$, $K_b = 40$, and $K_u = 30$].

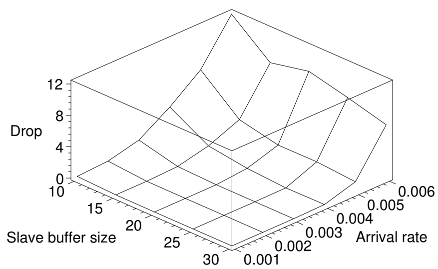
for the interior and for the exterior piconets. Simulations were carried out for bridge residence time of 1, 2 and 3 piconet cycles, in order to explore the effect of BRT on end-to-end delays. From the above three Figures I infer that a increase in BRT increase



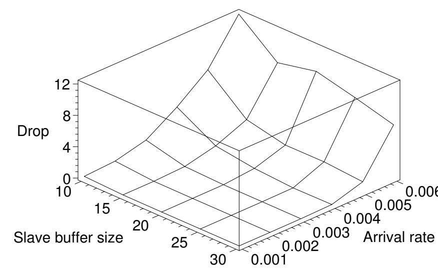
(a) Exterior piconet, $BRT = 1$.



(b) Exterior piconet, $BRT = 2$.



(c) Interior piconet, $BRT = 1$.



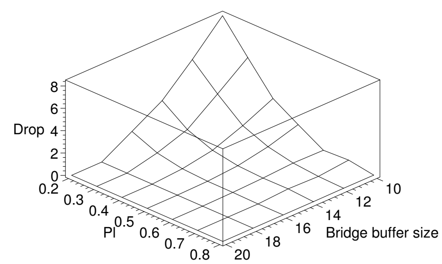
(d) Interior piconet, $BRT = 2$.

Figure 3.12: Slave buffer drop rate for $BRT = 1$, 2 [Arrival rate = 0.001 (0.001) 0.006, $K_u = 10$ (5) 30, $M_s = 2$, $M_b = 15$, $P_l = 0.4$, $K_d = 100$ and $K_b = 40$].

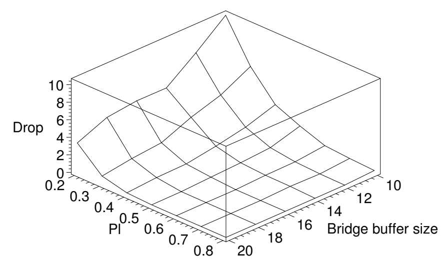
the amount of time the bridge spends in a piconet and thereby increasing the end-to-end packet delays between interior and exterior piconets.

In order to analyze the effect of BRT on the slave buffer drop rates, simulations were carried out for bridge residence time (BRT) of 1 and 2 piconet cycles. The slave buffer drop rates are obtained by plotting the graph between slave buffer size of 10, 15, 20, 25 and 30 and packet burst arrival rates of 0.001 .. 0.006. By analyzing Figure 3.12 I conclude that the increase in BRT has no effect on slave buffer drop rates.

Figure 3.13 gives the bridge buffer blocking rates for BRT equals to 1 and 2 piconet cycles. As the bridges are equipped with limited buffer size, increase in BRT results in increases number of data packets transferred to the bridge before its switching to the



(a) $BRT = 1$.

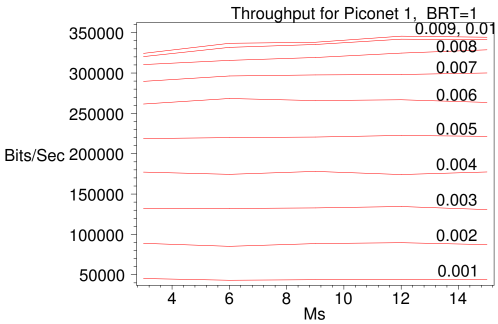


(b) $BRT = 2$.

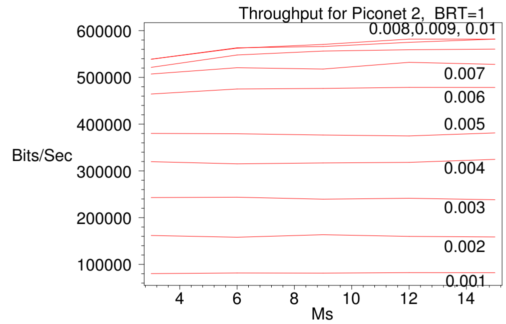
Figure 3.13: Bridge buffer blocking rate for P_1 to P_2 for $BRT = 1, 2$ [$K_b = 10$ (2) 20, P_l $= 0.2$ (0.1) 0.8, $M_s = 3$, $M_b = 9$, Arrival rate $= 0.002$, $K_d = 100$ and $K_u = 30$].

other piconet, thereby resulting in increased loss rates at the bridges.

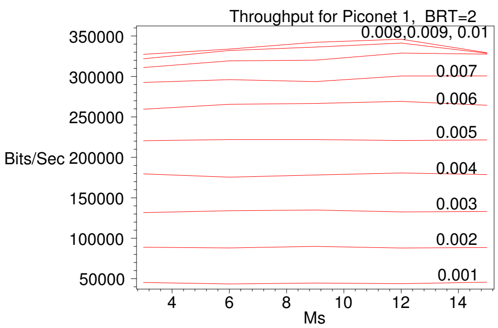
The final set of simulations carried out to analyze the effect of BRT for the throughput. Once again the BRT was varied for 1, 2 and 3 piconet cycles and the throughput was calculated for P_1 and P_2. As can be seen from Figure 3.14, increase in BRT results in longer cycle time and hence increases the packet drop rate and decreases the throughput for both P_1 and P_2.



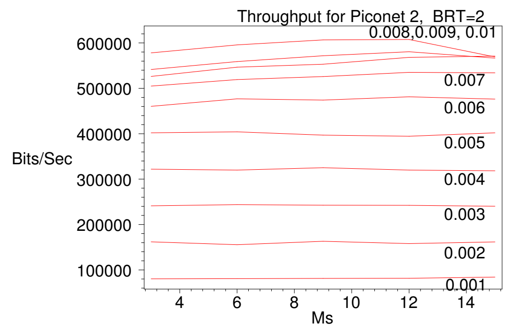
(a) External piconet (P_1), $BRT = 1$.



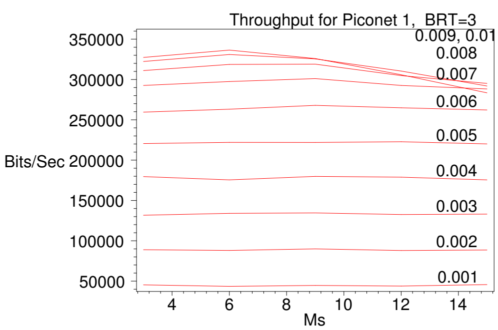
(b) Internal piconet (P_2), $BRT = 1$.



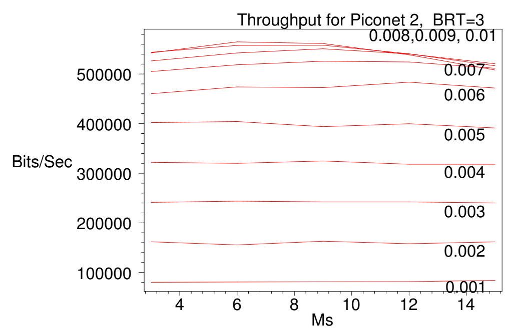
(c) External piconet (P_1), $BRT = 2$.



(d) Internal piconet (P_2), $BRT = 2$.



(e) External piconet (P_1), $BRT = 3$.



(f) Internal piconet (P_2), $BRT = 3$.

Figure 3.14: Throughput in the example scatternet for P_1 and P_2 with $BRT = 1, 2, 3$ [$M_s = 4\ (2)\ 14$, Arrival rate $= 0.001\ (0.001)\ 0.01$, $M_b = 15$, $P_l = 0.6$, $K_d = 100$, $K_b = 40$, and $K_u = 30$].

3.9 Performance of a Scatternet without Congestion Control

In this work, I have analyzed the performance of a Bluetooth scatternet in which all device queues—uplink, downlink, and bridge queues alike—are implemented through buffers with finite size. I have considered scatternet operating under walk-in bridge scheduling, in which the masters poll both ordinary slaves and bridges using the E-limited polling scheme. I have shown that the dependence of packet drop rates and packet delays on the size of different buffers; larger buffers lead to reduced drop rates as well as reduced delays. From the results, I was able to conclude that as the arrival rate of slave data increases, the slave buffer drop rate and access delay for piconets also increases, hence resulting in loss of data packets. The simulations were also carried out for Bridge Residence Time (BRT) of 1, 2, 3 cycles (BRT defines the amount of time the bridge resides in a piconet). From the obtained results, I observed that as the bridge residence time (BRT) increases in steps of 1, 2, and 3, the bridge buffer loss rate also increases. This is because the cycle time for each piconet increases with the increase of bridge residence time. Hence, the bridge buffer loss rate increases with the BRT. The throughput for each piconet was calculated, and it was observed that the internal piconets acted as the backbone of the network by performing a lot of data transfers between the exterior piconets. I have also shown that both delays and packet drop rates are critically dependent on the aggregate load of individual piconets; a piconet with too heavy load will exhibit high packet drop rates and high end-to-end delays. The above results and conclusions were made in the absence of congestion control and power management algorithms.

Chapter 4

Bluetooth Scatternet as a Sensor Network

In this chapter, I present a detailed description of the simulated scatternet acting as a wireless sensor network and the reasons for choosing a triangular topology. Subsequently, I will explain the traffic model used and follow with the implementation details. Later in this chapter, I discuss the cause and effect of congestion and power consumption in wireless sensor networks.

4.1 Description of the Bluetooth scatternet as a Sensor Network

I have considered each piconet as a cluster of sensor nodes that is controlled and coordinated by the piconet master. Each slave maintains an uplink queue towards the master, and for each slave, the master maintains a downlink queue. The master also maintains downlink queues for each bridge. Each bridge has two queues, namely the outgoing queue and the incoming queue. The networking environment will be the same as described in section 3.

To evaluate the sleep management algorithm presented in Section 6.2, I need to simulate a Wireless Sensor Network (scatternet) operating on Bluetooth Technology which will be the environment for testing the designed sleep management algorithm. The requirement for such a scatternet is that one piconet must act as a sink and the other piconets must do the sensing and forwarding operations of sensed data. The target scatternet could have any topology, say polygon or 3D structure, yet, for the sake of simplicity I have decided to use a non-trivial triangular topology of six piconets, where one vertex piconet (P_1) acts as a sink as shown in Figure 4.1. Using this topology I can evaluate the effects of multi-hop data transmission and can distinguish between the piconets which are mainly involved in sensing and those that are involved both in packet-forwarding and sensing. This is important since forwarding piconets make some sort of a network backbone. This sensor network backbone is expected to be the first to be affected by signs of congestion (packet loss). Therefore, I hypotheses that my proposed solution will also satisfy other more complex scatternet topologies. In the proposed model, I assume a shortest path routing algorithm, while congestion and energy control are performed together by the same control mechanism at the transport layer.

In the simulated sensor network the interior piconets are equipped with 3 slaves and the exterior piconets are equipped with 5 slaves. As mentioned earlier, the master visits the slaves in a round robin fashion. In the walk-in bridge scheme, the master polls the bridges in the same fashion as it polls the slaves. If a bridge is present in the piconet, data exchange between master and bridge takes place. If the bridge is not present in the piconet the master simply polls the next slave. M_s is the value which determines the maximum number of packets that a slave can exchange with the master from a single poll. M_b is the value which determines the maximum number of packets that a bridge can exchange with the master from a single poll. The bridge residence time BRT determines the number of cycles the bridge will reside in the same piconet. BRT is variable in my experiment and can take values of 1, 2 and 3 piconet cycles. However, I have fixed the BRT to 1 piconet cycle based on the conclusions from Chapter 3.

In the simulated scatternet, the control and coordination operations are shared by the sink and the source piconet masters, thereby eliminating the need for a computationally powerful sink. The life of a sensor node depends on the battery it is carrying. Hence, in order to increase the life of a sensor node, the node should be operated so as to maximize

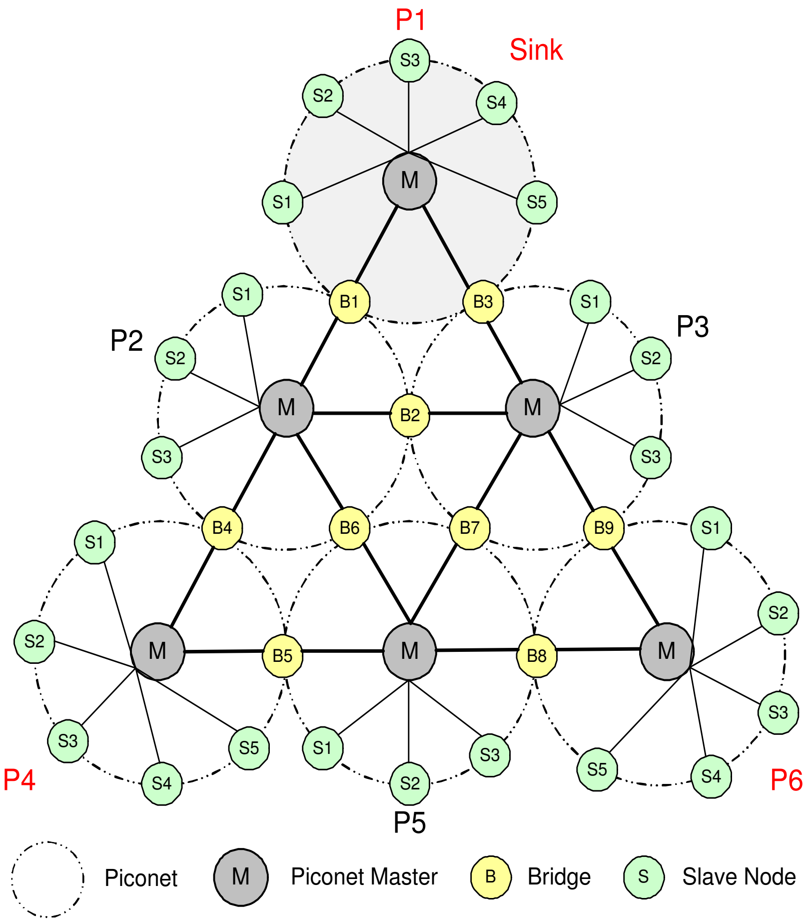


Figure 4.1: Wireless Sensor Network with Triangular Topology

its life. The aim of this work is to design an algorithm which will balance the number of active/sleeping slaves to maintain satisfactory throughput of data at the sink and to reduce congestion in the network.

4.1.1 Traffic model

The traffic model is derived from a relatively high-bandwidth, low cost surveillance based sensing application where compressed still images are taken as result of event detection and sent to the sink. (This setup may be used in applications such as road traffic control or asset protection.) I assume that the sensed data can not fit in a single Bluetooth packet—even with the Enhanced Data Rate facility defined in Bluetooth v2.0, the largest packet size is still only 1023 bytes [37]—and, therefore, the traffic must consist of packet bursts. I assume that the packet burst size is geometrically distributed with mean burst size B (in this model I use $B = 3$), and each packet has a length of five Bluetooth time slots $T = 625\,\mu\,s$. The packet burst arrival rate λ to each sensing node is presented as number of packet burst arrivals per one basic Bluetooth slot of 625μs. For example, the values which I use in my model are 0.002-0.005 packet bursts (images) per Bluetooth slot, which translates into approx. 3.2 to 8 packet bursts (images) per second. Packet burst arrival rate and the probability of traffic locality for all the slaves are uniform.

For the above model, each slave creates traffic with a packet length of 5 Bluetooth time slots. The arrival rate λ and the probability of traffic locality for all the slaves are uniform. Traffic is bursty, with packet burst size geometrically distributed, with average packet burst size B equal to 3 packets. Locality probability that traffic generated by the slave will have destinations in the same piconet is P_l, while probability that destination is in a different piconet is (1-P_l). I assume that the nodes within a piconet can exchange some local data. According to the selected topology shown in Figure 4.1 the probability that destination is in given non-local piconet is (1-P_l)/5. When the traffic is generated by a slave for another piconet, the packets are routed through the master, through the bridges, and through various intermediate piconets, to the destination piconet, by taking the shortest path. In the above model, the master and the bridge do not generate any traffic, they only route the traffic.

4.2 Congestion Control & Energy Management in Sensor Networks

The main objective of my thesis is to achieve reliable event detection in a sensor network, by spending minimal energy and enhancing congestion control. Since sensor nodes at the source are continuously sending the sensed data towards the sink, retransmission of the lost data packets due to buffer overflows at the bridges is not necessary; however, minimizing the packet losses (and improving the overall energy efficiency) requires that only a minimal number of slaves be kept awake in each piconet. At the same time, the reliability at the sink has to meet application requirements. Therefore the main contribution of this work is the integration of congestion control with reliability and energy management in the Bluetooth-based sensor network. This is achieved by introducing a sleep management algorithm for the source piconets. This sleep management algorithm tries to put the source piconets slaves into hold mode based on the observed reliability for that piconet at the sink, thereby reducing the traffic generated by that source piconet toward the sink and reducing power consumption and buffer overflow at the intermediate bridges. The outcome of the sleep management algorithm depends on, (a) the choice of polling scheme employed to poll the slaves, by the master, in the source piconets, (b) the bridge buffer sizes, (c) the slave buffer sizes, (d) the bridge residence time, and (e) the routing algorithms. I will now discuss the problems created by buffer overflows and the need for an energy efficient congestion control scheme.

4.2.1 Congestion Control

Sensor networks transport different types of traffic, from simple periodic reports to unpredictable bursts of messages triggered by events that are being sensed, all of which might lead to congestion. Congestion in a sensor network causes an increase in data loss rate, power consumption, access delays, and (in some cases) high resource utilization. All the nodes in a sensor network generate data packet bursts at some application dependent rate λ_i. Packets are further forwarded to the sink by the co-operation of multiple nodes. If the packet burst arrival rate is too high, the packet buffers of the nodes along the path will overflow. This sudden increase in loss rate is the main symptom of congestion. Since the sensor nodes are equipped with small storage, the buffer size cannot be increased dynamically when congestion occurs. Hence, by putting some of the source nodes to sleep, the traffic load and congestion in the network can be reduced. To overcome the problems created by congestion, efficient congestion control protocols need to be employed. I plan to implement congestion control by dynamically putting sensor nodes to sleep based on the reliability factor that is periodically calculated at the sink.

4.2.2 Energy Efficiency

Later in this thesis, a wireless sensor network operating on Bluetooth technology has been simulated. In order to analyze the congestion levels in the simulated network, initial exploratory simulations were carried out with P_1 acting as the sink and by varying the number of active slaves at P_4 (source). From the obtained results I infer that the desired reliability can be achieved in two regions (No congestion, Desired reliability) and (High congestion, Desired reliability) as shown in Figure 6.3 of Section 6.2. However, operating the network in (High congestion, Desired reliability) is not feasible, as it uses a greater number of active slaves and leads to high congestion in the network. Hence, there is a need to operate the number of active slaves in (No congestion, Desired reliability) region to minimize the congestion and power consumption. More of this is discussed in Chapter 6.

Chapter 5

Congestion Control protocols for Sensor Networks

Though numerous schemes have been proposed for event detection and data transmission in wireless sensor networks, the issue of reliable data transfer, with energy efficient congestion control for wireless sensor networks operating on Bluetooth technology, has yet to be studied and addressed. There is a need for a comprehensive set of congestion control mechanisms specifically designed to fit the requirements of sensor networks. These mechanisms should provide a general set of components that can be plugged into applications or into the MAC to support reliable data transfer with energy efficient congestion control for wireless sensor networks. In this chapter I present various transport protocols that employ congestion control techniques for wireless sensor networks.

5.1 Directed Diffusion (DD)

Directed diffusion has been proposed in [18] for event detection and reliable data transfer in wireless sensor networks. Directed diffusion is data-centric, i.e. communication between the nodes in the network is restricted to named-data (structured data). All sensor nodes in the directed diffusion scheme are equipped with small amount of memory and processing ability. In directed diffusion, a node requests data by sending an interest (query) for named-data. Once a match for the required data is found, the results are transferred to the querying node. In this process of data transfer, intermediate nodes can aggregate the obtained data, store them in their cache and redirect them to their neighboring nodes.

Whenever there is a need for data pertaining to a specific event, a query is injected into the network in the form of an interest. Interests in directed diffusion give the description of the task to be performed. An interest consists of type, interval and duration. The type defines the type of query, the interval parameter specifies the event data rate and the duration represents initiation of diffusion after inserting the interest at the sink (sink is a node where the interest is injected and the collected data from the source is received).

For a given task, the sink periodically broadcasts the same interest with a new time stamp. Initially these interests are called exploratory interests, and are used to set up initial gradients and to check if there are any nodes in the network that detects the required data. Gradients in directed diffusion are used to transfer the sensed data from source to sink via an optimal path. Gradients specify the direction in which the sensed data is to be propagated. When the source nodes receive a query, the request is processed and the obtained results are sent back towards the sink through multiple paths indicated by the gradients. At some point of time a single high data-rate path is established between the source and sink by local interaction and reinforcement in order to transfer the events indicated by the gradients [2].

The drawback of directed diffusion is that all nodes should be equipped with some amount of computational capabilities and memory in order to store the interest with various time stamps. This might cause significant overhead for sensor networks with power and processing limitations [15]. Though directed diffusion offers guaranteed end-to-end data delivery, this is not required in case of event detection due to the fact that correlated data flows from several source nodes are loss tolerant (to the extent that event features are reliably detected) [2], [4].

5.2 Pump Slowly Fetch Quickly (PSFQ)

Campbell et al. [39] discussed the Pump Slowly Fetch Quickly (PSFQ) mechanism for reliable control and management of wireless sensor networks. PSFQ supports a simple, robust, and scalable transport scheme that can be customized to meet the requirements of reliable data transfer applications in wireless sensor networks [39]. In their study, Campbell et al. have observed that in sensor networks data that flows from source to sink is generally tolerant of loss of data packets. This observation is based on the theory that the source transfers correlated data. However, the data that flows from sink to source for the purpose of control or management of the source nodes is vulnerable to message loss (control messages are initiated periodically and are not loss tolerant).

PSFQ is based on propagation of data from source node by injecting data at relatively low speed and allowing nodes that experience data loss to fetch any missing data packets from immediate neighbors by requesting for retransmission.

The problem with PSFQ is that the authors assume that packet loss, in this scheme, is due to the poor quality of wireless links and the resulting transmission errors, while the traffic-congestion and the resulting packet blocking due to buffer overflows at various stages in the network are not considered [2]. This is not a realistic assumption for sensor networks, especially in view of the fact that some packet loss may be acceptable due to correlation of sensed data.

5.3 Event to Sink Reliable Transport Protocol (ESRT)

ESRT, proposed in [2], has been designed for reliable event detection in wireless sensor networks. ESRT necessitates an event-to-sink (multipoint-to-point) reliability notion in contrast to existing end-to-end (point-to-point) reliable transport protocols such as: Transfer Control Protocol (TCP) and Pump Slowly Fetch Quickly (PSFQ). Event-to-sink reliability is based on the fact that correlated data that flows from several source nodes to a single sink are loss tolerant to the extent that the event features are reliably detected at the sink [2]. Correlated data gives a numeric measure of the similarities between the sensed data from various source nodes for a given phenomenon. Due to this data correlation, it is not necessary to receive all the data packets at the sink, but some smaller number of data packets according to required accuracy of detection. Smaller number of data packets received at the sink can be achieved either by preventing the nodes to send extra packets (which is desired) or by dropping the packets due to congestion (which is not desired) [2].

The objective of ESRT is to achieve reliable event detection in wireless sensor networks with minimum energy expenditure and maximum congestion control. In ESRT, the sink estimates the event reliability for every t time units (t represents duration of a decision interval). At the end of each time interval t, based on the reports obtained from the source nodes, the sink makes a decision whether or not to send a control signal to increase the reporting rate of the source nodes. Also, at the end of each interval t_i, the sink calculates the event reliability indicator r_i. The reliable transfer of an event from source to sink is measured in terms of number of data packets received at the sink for a given time interval. Hence, r serves as a function for event reliability measure at the sink. Reliability of event detection is based on two definitions given as follows:

- Observed event reliability: For observed event reliability, r_i defines the number of received data packets in decision interval t_i at the sink.
- Desired event reliability: For desired event reliability, R defines number of data packets required for reliable event detection.

From the above two definitions we can infer that, if $r_i >$ R, the event is determined to be reliably detected at the sink or else appropriate actions must be taken to achieve the desired event reliability R [2]. To achieve the desired event reliability in case of congestion, the reporting rate f of source nodes have to be varied dynamically. The reporting rate f of a sensor node is defined to be the number of packets sent per unit time by that node [2]. The idea behind ESRT is to configure the appropriate reporting rate f for a source node, to achieve the required event detection reliability R at the sink, with minimum resource utilization.

In ESRT, though the sink tries to adjust the report rate of the source nodes, it does not try to optimize the number of nodes required to sense the data for a given task.

Also, ESRT transfers raw data to the sink, rather than employing any kind of in-network processing or reducing the redundant sensed transferred from source to sink [16], [38].

5.4 Congestion Detection and Avoidance (CODA)

Sensor networks used for event detection operations are either idle or lightly loaded until an event occurs. The occurrence of an event results in a sudden burst of data transports in the network. This sudden transport of data packets is likely to lead to various levels of congestion at various stages of the network. The event impulses (observed data) generated by the source is of utmost importance as the likelihood of it (event impulses) experiencing congestion while transferring is high. Retransmission of data that was lost due to congestion incurs a substantial energy cost on each sensor, which should be avoided. To address these challenges, an energy efficient transport protocol for wireless sensor networks has been proposed by Chieh-Yih et al. in [38].

CODA, the proposed congestion handling technique, uses energy efficient heuristic mechanisms for monitoring network operations to avoid congestion. These mechanisms are as follows: receiver based congestion detection, open-loop hop-by-hop back-pressure and closed-loop multi-source regulation [38].

In receiver based congestion detection, CODA uses combinations of present and past channel loading conditions, and current buffer levels, to predict congestion occurrence in the network. CODA makes use of a simple technique for monitoring the message queue. If there is an overflow at the queue, congestion at the node is implied. A periodic sampling of the network's performance is then employed rather than continuous monitoring of the channel. By a periodic check of message queue (buffer) levels, the network state can be accurately monitored at the expense of higher power consumption.

The second mechanism used in CODA is open-loop hop-by-hop back-pressure regulation. When a sink detects congestion or potential signs of congestion in the network, it broadcasts a back-pressure message (control signal) to all its neighbors. Each node then handles this back-pressure message depending on its own state. If the node experiences congestion, it propagates the message to its neighbors and this continues until the back-pressure message reaches the source nodes. A node that receives the back-pressure message may reduce its sending rate or drop data packets based on network policies.

The third mechanism employed in CODA is closed-loop multi-source regulation. This mechanisms operates over a long time scale, and is capable of regulating congestion on multiple sources in the network, from a single sink. When congestion in the network crosses threshold level, the sink initiates a control signal to all source nodes to receive an acknowledgment from the sink at certain regularity or else the source nodes need to considerably reduce their sending rates. This results in sink oriented selective regulation of data transmissions.

Though CODA tries to achieve congestion control, the extra messaging required in controlling congestion for closed-loop multi-source regulation leads to higher energy consumption than open-loop hop-by-hop regulation.

The aforementioned approaches can be roughly classified into those which achieve individual packet reliability using packet re-transmissions, and those which try to obtain sufficient number of packets at the sink by using some kind of feedback to inform the sensing nodes to decrease the reporting rate. Neither of them considers the effects of finite buffer limitations of sensing and bridging devices.

Furthermore, all those approaches either do not consider the impact of the MAC protocol at all, or assume the use of collision-based (and thus generally inefficient) protocols such as CSMA-CA. This was the motivation that lead me to investigate the possibility of implementing wireless sensor networks using Bluetooth and its collision-free MAC protocol.

5.5 MAC with adaptive sleeping for Wireless Sensor Networks

A novel sensor media access control protocol (S-MAC) with adaptive sleeping for sensor networks has been proposed by Ye et al. in [40]. The aim of the proposed protocols is to conserve energy by dynamically putting the nodes into sleep. S-MAC also achieves good scalability and collision avoidance by utilizing a combined scheduling and contention scheme. In order to achieve the above stated factors. It is essential to identify main sources that cause inefficient use of energy as well as the tradeoffs required to reduce energy consumption. These were the major source of energy consumptions identified by the authors.

- **Collision:** When a transmitted packet is lost, it has to be retransmitted leading to increase in energy consumption.

- **Overhearing:** There are chances of nodes picking up packets that are destined to other nodes.

- **Control packet overhead:** Sending and receiving control statements consume energy.

- **Ideal listening:** listening to the channel for a possible traffic.

Most of the sensor networks are designed to operate for long time. Thus, ideal listening state is one of the major sources of power consumption in wireless sensor networks. Measurements have shown that ideal listening consumes 50-100% of the energy required for receiving data [40]. S-MAC tries to reduce energy consumed for the above mentioned tradeoffs, in exchange to some performance reduction such as latency. The S-MAC employs a low-duty cycle operation on nodes in a multihope network. For sensor networks operating on traditional MAC, the nodes are active for most of the time unless some sleep-management techniques are employed. In case of S-MAC, the low-duty-cycle mode is the default mode of operation, nodes are in sleep state for most of the time and become active only when there is traffic in the network. However, to reduce control overhead and latency, S-MAC adopts a coordinated sleeping technique among the neighboring nodes in the network. When there is no sensing taking place in the network, there is very little data flow in the network. Therefore, S-MAC lets nodes periodically sleep or else they are idle. Although, this design reduces energy consumption, it leads to increase in latency (sender must wait for the receiver to wakeup before it can transfer data).

In the basic scheme employed by S-MAC, each node sleeps for some time and then wakes up and listens to the network to see if any other neighboring nodes are willing to transmit data. During the sleep period, the node turns off its radio and sets a timer to awake itself later. If the network is active and if multiple neighbors are willing to communicate simultaneously whenever a node starts listening, this leads to collision and increase in energy consumption for retransmission. S-MAC follows similar procedures used by 802.11 for contention avoidance [40]. A duration field is inserted into the transmission packet to indicate how long the node must keep silent. In this way, S-MAC effectively address energy wasted due to ideal listening of the channel, and congestion control with low-duty-cycle operation and contention mechanism.

Chapter 6

Design of a Sleep Management Algorithm

In this chapter a sleep management algorithm is designed based on the observations at the sink (P_1) with P_4 acting as the source.

6.1 Analysis of congestion with varying slaves at the source

In the Figure 6.1, piconet 1 (P_1) acts as a sink and piconet 4 (P_4) acts as the source and other piconets (P_2, P_3, P_5, P_6) generate background traffic, where the slaves are uniformly loaded and are active all the time. In this experiment, the number of active slaves at the source (P_4) are increased in steps of one and the observations at the sink are noted. Based on these observations a sleep management algorithm is designed, which regulated the amount of traffic generated in the network. Whenever there is a need to acquire some information from the network, a query is injected at the sink (P_1 in this case). Once the query reaches the target piconet (P_4), required actions in response to the query are performed and the collected data is sent back to the sink. Traffic model, polling and scheduling and the implementation details are the same as discussed in Chapter 4.

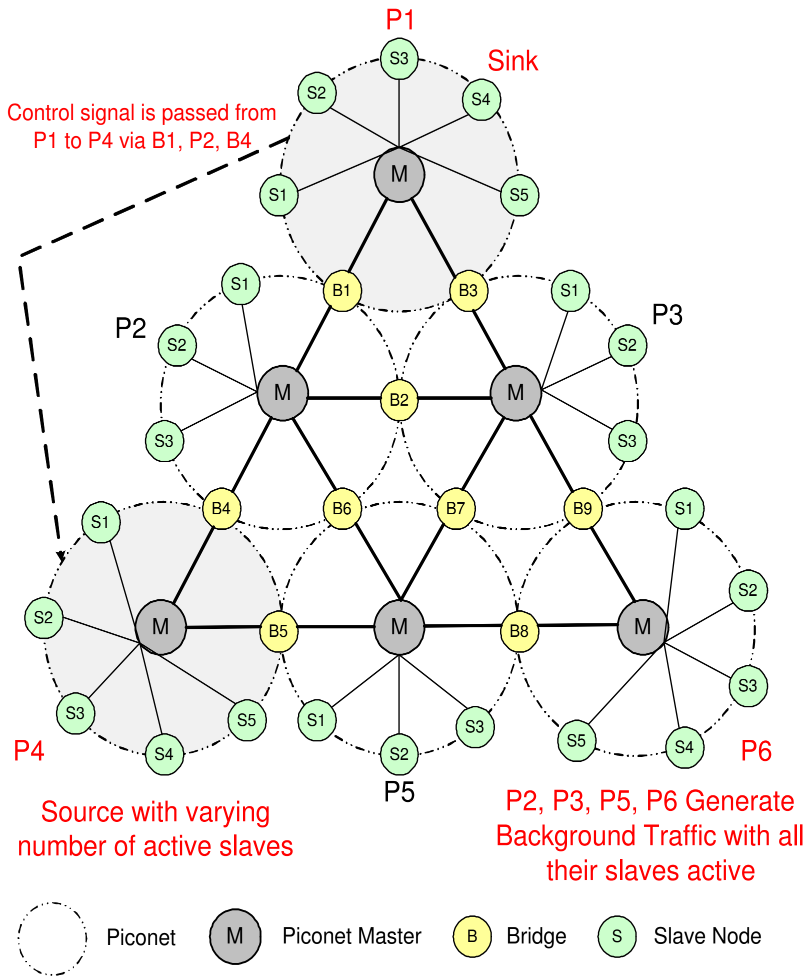
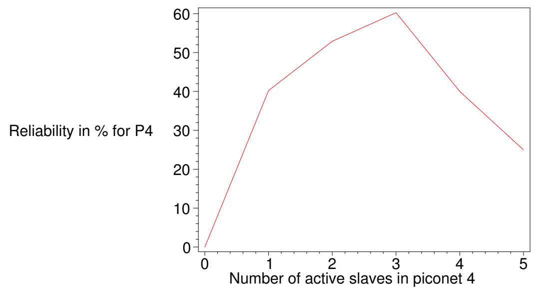


Figure 6.1: Wireless Sensor Network with P_1 as Sink and P_4 as Source

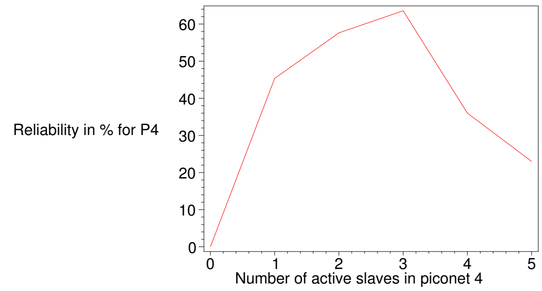
6.1.1 On reliability at the sink

Considering reliability (defined above as the number of data packets required per second for reliable event detection at the sink), we can distinguish between three related concepts.

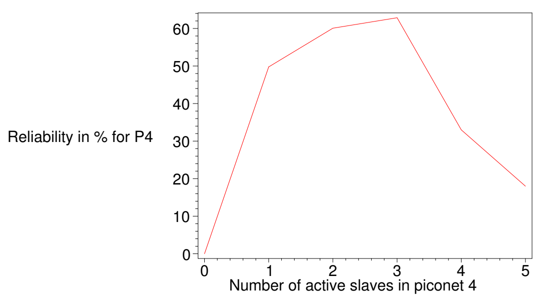
- Relative event reliability is defined as the portion of the maximum reliability at the sink, estimated at every t_i (known as the decision interval).
- Absolute event reliability corresponds to the number of packets received per second at the sink, as requested by the sensing application.
- Finally, the desired event reliability (which is specified by the user application) is



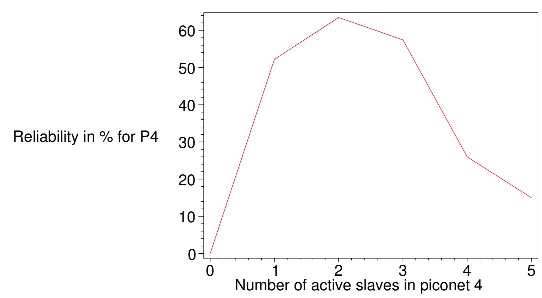
(a) Relative reliability in % for arrival rate of 0.002



(b) Packet burst arrival rate of 0.003.



(c) Packet burst arrival rate of 0.004.



(d) Packet burst arrival rate of 0.005.

Figure 6.2: Relative reliability at the sink (in percents) vs. the number of active slave in piconet P_4.

the number of data packets required (in %) for reliable event detection at the sink.

Initially, simulations were carried out to explore the behavior of the relative reliability at the sink. The piconet P_4 contained a varying number of active slaves, while the other piconets (P_2, P_3, P_5, P_6) were simply generating background traffic with all slaves carrying uniform load. Figure. 6.2 shows the relative reliability for packets sent from the slaves in P_4 measured at the sink as a function of number of active slaves at P_4, with arrival rates of 0.002, 0.003, 0.004, and 0.005 packet bursts per Bluetooth slot, in all the piconets. Other parameters were: traffic locality $P_l = 0.3$, polling parameters $M_s = 3$

and $M_b = 12$, mean burst size of 3, slave buffer size of 30 baseband packets, master buffer size of 100 packets, and bridge buffer size of 40 packets.

From the diagrams, I note that the relative reliability has a peak of 60 to 65% which is slowly moving toward lower number of active slaves: from about 3 in Figure. 6.2(b) to around 2 in Figure. 6.2(d). Such low values at the peak, as well as the drop in reliability beyond the peak, are due to lack of congestion control. Namely, other piconets generate a lot of traffic towards the sink (which is in P_1); this traffic overloads the bridge buffers along the way and reduces the relative reliability for traffic from P_4.

Therefore, it may be expected that, by controlling the number of active slaves in all the source piconets, I can minimize packet losses at the bridge buffers and maximize the relative reliability at the sink. Controlling the number of active slaves must be performed by the master nodes in source piconets at the request of the sink. The sink should inform the masters in source piconets whether the number of active slaves should be increased or decreased; the masters will then deactivate or activate some of their slaves accordingly. Activation and deactivation may be accomplished by unparking some parked slaves and parking previously active slaves; an alternative (and much faster) procedure is to put active slaves in the so-called HOLD mode for a specified time interval [37].

The HOLD mode may be entered upon request from either master or the slave, and the duration of the HOLD interval is negotiated between the two. While in the HOLD mode, the slave retains its network address, and may enter a low power mode or do something else – the master will not try to poll it. (In my experiments, I assume that the slave will enter a low power mode and thus conserve energy.) Upon expiry of the HOLD mode, the slave again begins to listen to master's transmissions, while the master is free to poll the slave at will. Finally, each HOLD interval is negotiated anew, hence it may be adjusted to any required time interval. Because of the simplicity and flexibility of the procedure, I have chosen to implement the activation and deactivation of slaves using the HOLD mode.

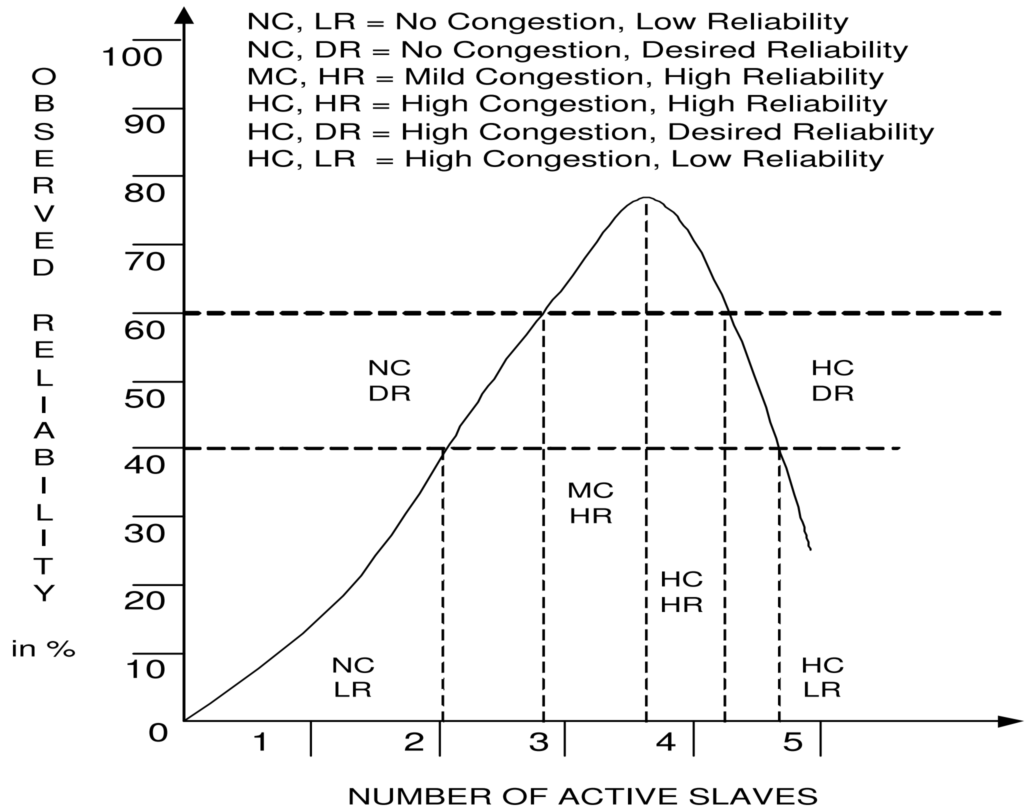


Figure 6.3: Characteristic regions for reliability and congestion at the sink as a function of number of active slaves in the P_4.

6.2 Managing reliability at the sink

Based on the previous conclusions drawn from Figure. 6.2, I can distinguish the regions of the (reliability, congestion) pair behavior at the sink for varying number of active slaves in P_4, as shown in Figure. 6.3. According to the levels of congestion and reliability at the sink, the network can be in any of the following six states: (No congestion, Low reliability), (No congestion, Desired reliability), (Mild congestion, High reliability), (High congestion, High reliability), (High congestion, Desired reliability), and (High congestion, Low reliability). Taking these six (congestion, reliability) states and the corresponding operating regions into consideration, a scheme may be devised to simultaneously execute sleep management and congestion control, and thus accomplish reliable event detection.

Informally, the algorithm operates as follows. Initially, the exterior piconets operate with five active slaves, while the interior ones operate with only three, because of their higher carried load. The desired reliability is chosen by the user, usually in the range of 40 to 60%, and the relative reliability is periodically calculated at the sink, once every, say, $60s$. If the value obtained is above 60%, congestion has occurred (regions (HC,HR) (HC,DR) and (HC,LR)) and the number of active slaves is reduced. Depending on the current number of active slave, the reduction may affect one or two slaves; in the latter case, different HOLD intervals are used for those slaves, one below the decision interval and another one above it. The number of active slaves is also reduced if reliability is below 40% but the number of active slaves is above two (region HC,LR). If the value obtained is somewhat below the peak value of 60%, but not too low, the number of active slaves is not changed.

The control algorithm is distributed between the sink and the masters of the source piconets. The master's and sink's algorithms are shown in Figures. 6.4 and 6.5 respectively. Whenever data transfer from source to sink takes place, the sink periodically calculates the relative event reliability and passes it to the masters in source piconets. Once the source receives the relative event reliability from the sink a comparison is made with the desired event reliability. If relative event reliability is higher that the desired event reliability, the source master place two of its slaves in HOLD mode. Once a slave is put into HOLD mode, it stops sending the sensed data to the source master, thereby conserving its battery power and reducing congestion in the network. The slave returns from the HOLD mode on expiration of the sleep time.

When the network experiences congestion, the data packets transferred from source to sink are lost due to buffer overflow at the intermediate bridges. This overflow results in a sudden drop of the relative reliability at the sink, which is taken as the sign of congestion and low reliability in the network. At this point of time, the relative event reliability calculated at the sink and passed on to the source master will drop below the desired event reliability. Hence, the source master has to put two of its active slaves into HOLD mode for given time, thereby reducing the load on the network and avoiding further congestion. Once the node is put into the HOLD mode, it will remain in that mode until its sleep time expires, as shown in Figure. 6.6. Once the sleep time expires,

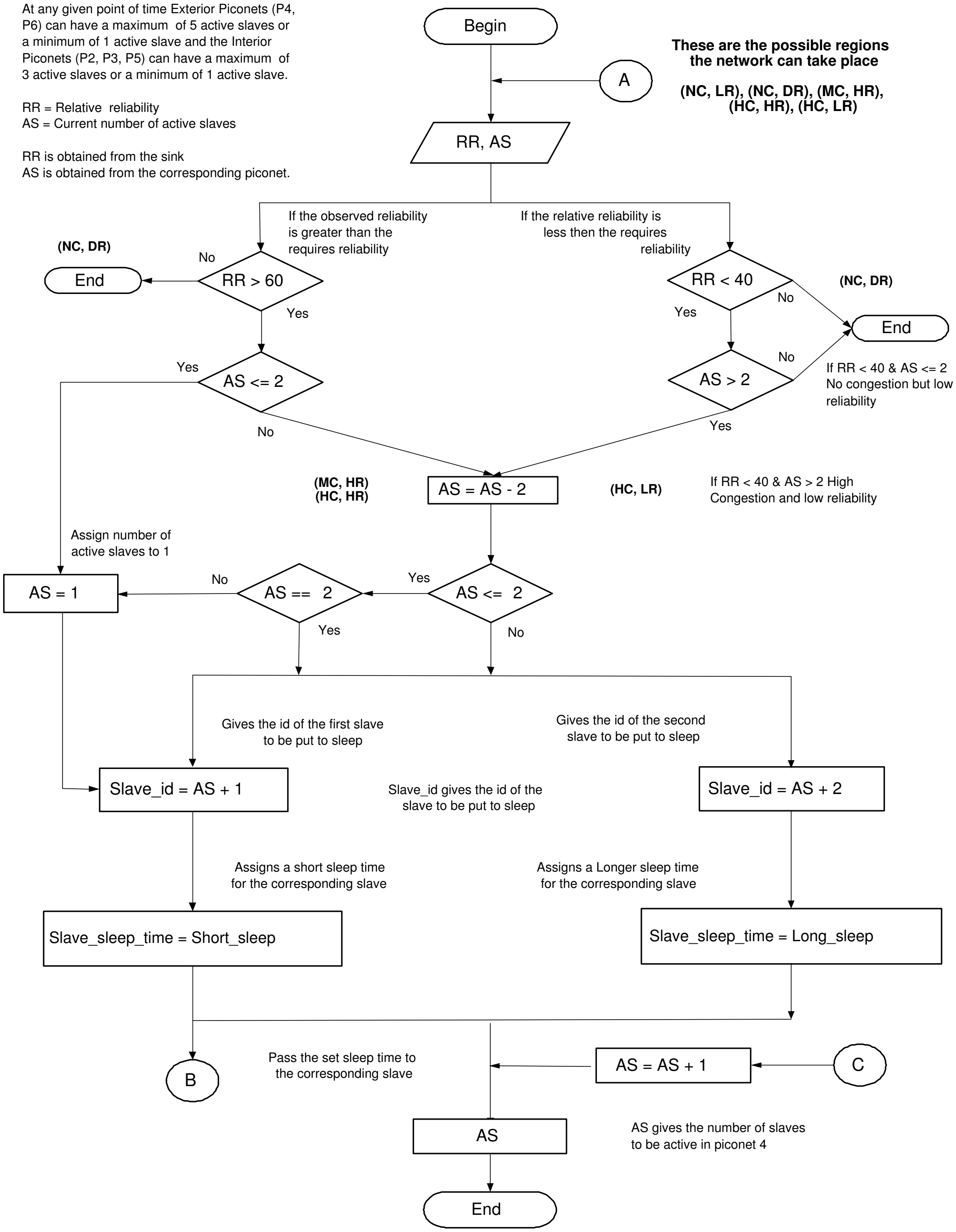


Figure 6.4: Algorithm to calculate the number of active slaves in source piconet – master portion.

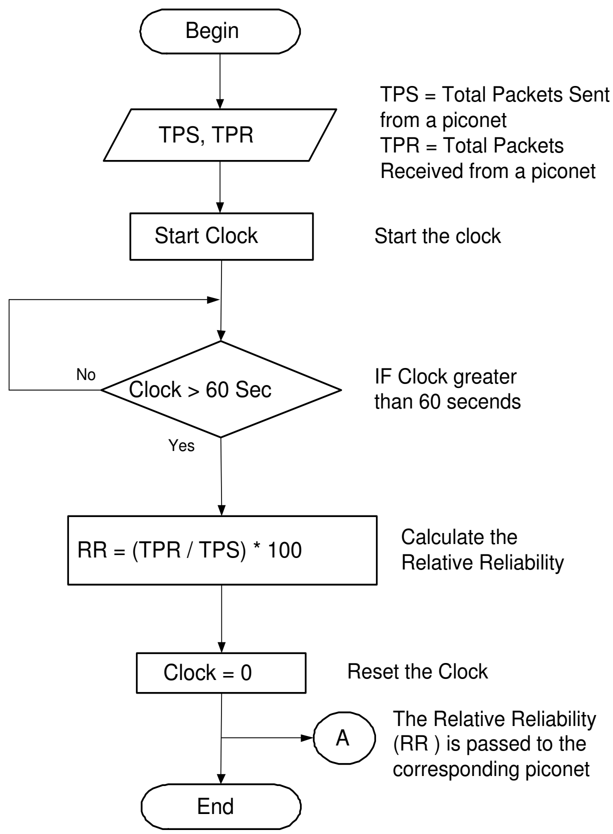


Figure 6.5: Algorithm to calculate the number of active slaves in source piconet – sink portion.

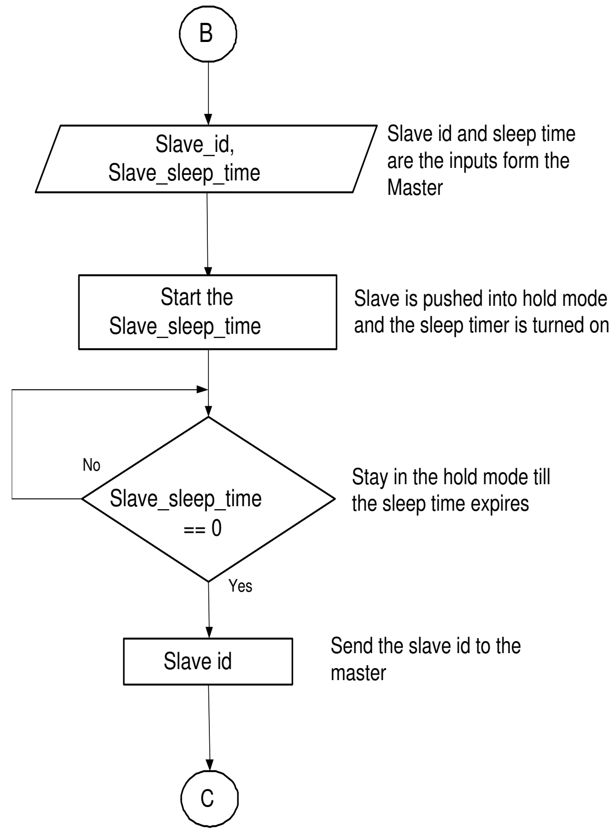


Figure 6.6: Activation of slaves from HOLD mode.

the slave starts collecting data and is ready to be polled.

Table 6.1 gives one measured trace from the simulator in order to illustrate the sleep regulation technique. As before, I apply the sleep management to only one piconet, P_4 in this case.

Initially P_4 has five active slaves, and the desired reliability is any value between 40-60%, say, 60%. At first check, relative reliability of 55% is measured, and the regulation algorithm makes no change to the current number of active slaves at P_4. At second check, relative reliability has increased to 64%, which is above than the desired reliability. As the number of active slaves is four, this is interpreted as the sign of congestion coming from P_4, and the number of active slaves is reduced by two. At third check, relative reliability has increased to 75% and the number of active slaves is 3. Hence, two more slaves are put to sleep. In the fifth measurement I observe that one slave is put into HOLD mode,

Table 6.1: Simulator trace for P_4.

Interval	Relative reliability	Active slaves in P_4	Slaves put on HOLD	Slaves back from HOLD
1	55	5	0	0
2	64	5	2	0
3	75	3	2	0
4	88	1	0	0
5	86	2	1	1
6	85	3	2	2
7	90	3	2	2
8	91	2	1	1
9	88	3	2	2
10	89	3	2	2
11	91	2	1	1
12	88	3	2	2
13	92	3	2	0
14	88	1	0	1

while another one has returned from the HOLD mode. In this way the control algorithm regulates the relative reliability observed at the sink for a source piconet.

Though the designed sleep management algorithm is not scalable, which is based on the heuristic values obtained by varying the number of active slave at P_4 and observing the event reliability for P_4 at the sink. However, the concept of putting the slaves into HOLD mode reduces congestion and power consumption in the designed network.

Chapter 7

Evaluation of Sleep Management Algorithm

In this chapter the designed sleep management algorithm is applied to the whole sensor network shown below. Later, an analysis of the sleep management algorithm in controlling congestion and power consumption in the network is made.

7.1 Sensor network with power controlled piconets

In the Figure 7.1, piconet 1 (P_1) acts as a sink and piconets (P_2, P_3, P_4, P_5, P_6) acts as the source. All the source piconets are equipped with the sleep management algorithm discussed in Chapter 6. Initially piconets P_4 and P_6 have five active slaves and piconets P_2, P_3, P_5 have 3 active slaves. In the below shown model all the source piconets transfer the sensed data packets to the sink via various intermediate piconets. The sink measures the relative reliability for each piconet for every $60s$ and a comparison is made with the desired reliability. Based on this comparison a control signal is generated and propagated back to the corresponding source piconet to put one or two of its slave nodes to Hold mode or not. Traffic model, polling and scheduling and the implementation details are the same as discussed in Chapter 4.

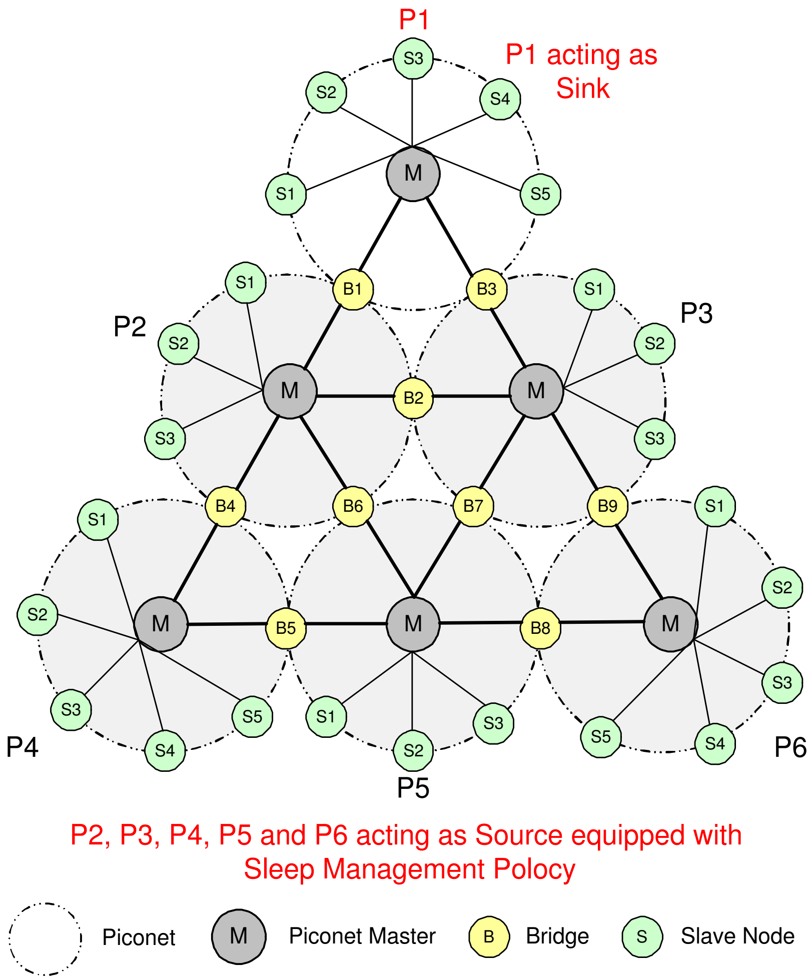


Figure 7.1: Wireless Sensor Network with power controlled piconets

7.2 Implementation details:

In order to study the performance of the proposed network architecture working on Bluetooth technology, a sensor network has been simulated making use of Artifex simulation software. Artifex is a Petri Net-based simulation engine developed by Artis Software Inc. used for discrete-event systems modelling [13]. I choose to use Artifex for its effectiveness in event-driven systems modelling and analysis. Furthermore, Artifex allows users to graphically model a system, build simulators, and prototype a network. It comes with graphical modelling language tools that use object-based concepts. C or C++ code will be used to create the models describing the operations. Artifex allows significant reductions in systems development time and provides a fast and reliable validation of system behavior and operations. Readers interested in in-depth details of simulating a sensor network using Artifex can go through the Appendix.

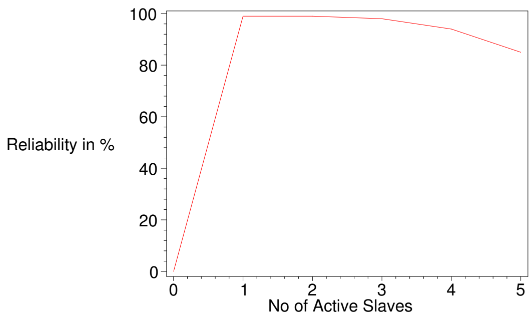
7.3 Performance of sleep management algorithm

7.3.1 Observations related to Relative and Absolute Reliability at the sink

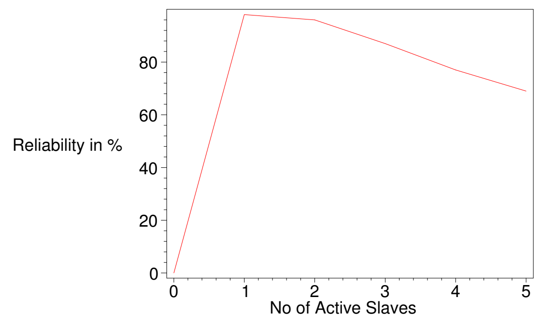
Simulations were carried out to explore the behavior of the relative reliability observed at the sink for each source piconet when the sleep management scheme is applied in the entire scatternet. Figure. 7.2 presents the relative reliability observed at the sink for packets sent from slaves in P_4 (which is an exterior piconet) for packet arrival rates of 0.002, 0.003, 0.004 and 0.005 packet burst arrival rates per Bluetooth time slot. All other parameters were set to the same values as before. Since all piconets operate under the sleep management scheme, the relative reliability at the sink is much higher than in the case when only one piconet uses the scheme, shown in Figure. 6.2; the peak value exceeds 95% under a wide range of packet arrival rates. Note that the reliability peak is reached with only one active slave per piconet, and that the reduction in reliability is fairly mild when the number of active slaves increases.

Figure. 7.3 presents the analogous dependency, only this time the relative reliability corresponds to packets sent to the sink from slaves in P_2, which is an interior piconet. Since the same congestion control mechanism is used in all the piconets, the shape of the dependencies is almost identical to those from the previous set of diagrams.

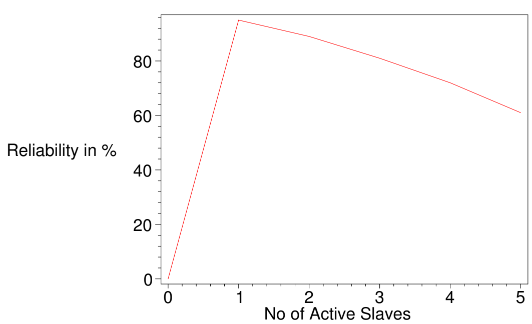
In order to calculate the dispersion of the relative reliability, its mean, variance and standard deviation are calculated for the data in Figures. 7.2 and 7.3. I note that the increase in arrival rate leads to a decrease in mean value and an increase in variance, which may be used to indicate serious congestion. Figure. 7.6 shows the development of the number of active slaves in P_4 over time including the warm-up period of the simulator (more on the warm-up period is given in the Appendix).



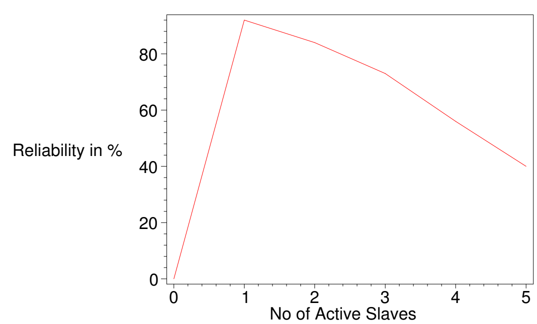
(a) Packet burst arrival rate of 0.002.



(b) Packet burst arrival rate of 0.003.

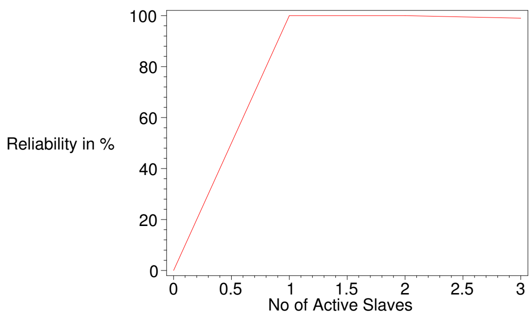


(c) Packet burst arrival rate of 0.004.

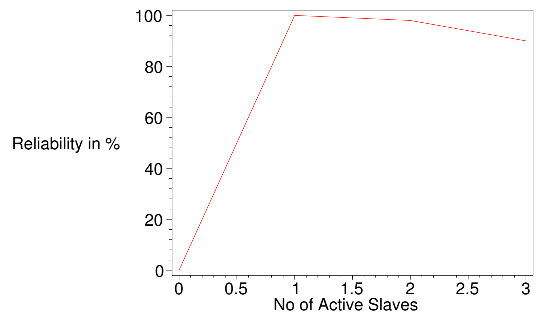


(d) Packet burst arrival rate of 0.005.

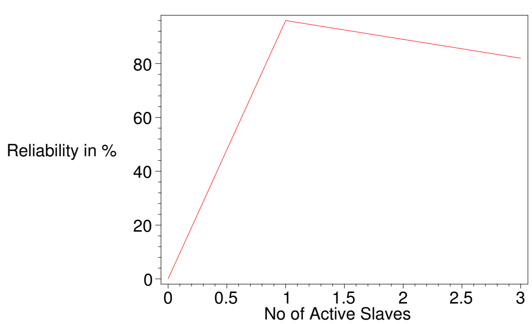
Figure 7.2: Relative reliability (in percent) of packets from the slaves in piconet P_4 at the sink vs. the number of active slaves in P_4.



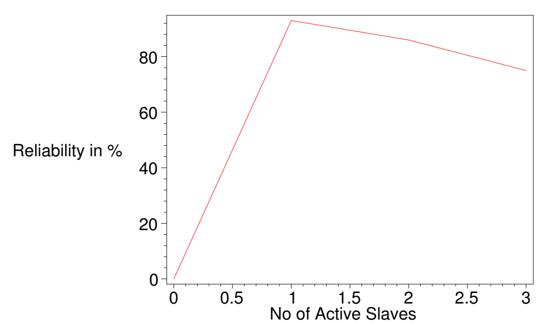
(a) Packet burst arrival rate of 0.002.



(b) Packet burst arrival rate of 0.003.

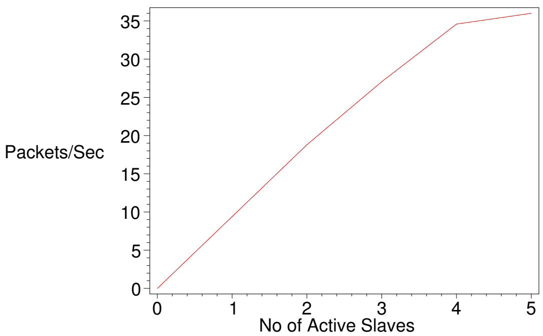


(c) Packet burst arrival rate of 0.004.

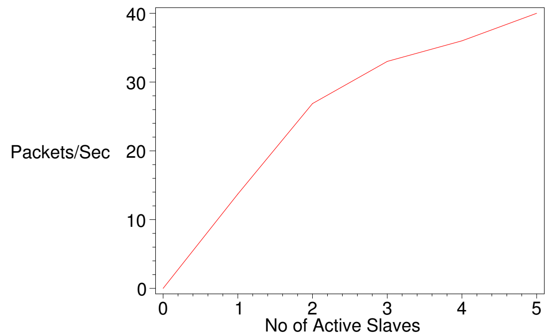


(d) Packet burst arrival rate of 0.005.

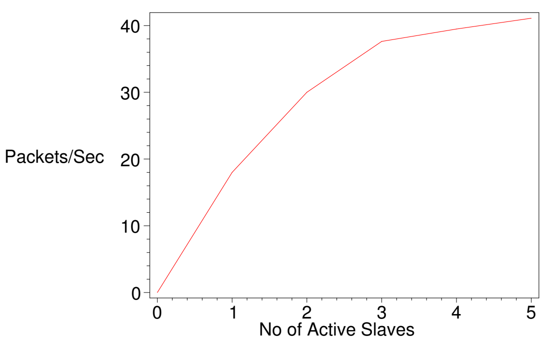
Figure 7.3: Relative reliability (in percent) of packets from the slaves in piconet P_2 at the sink vs. the number of active slaves in P_2.



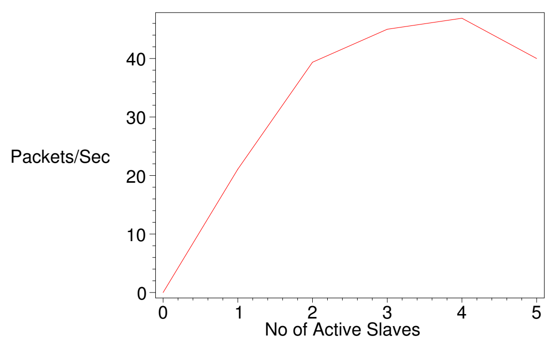
(a) Packet burst arrival rate of 0.002.



(b) Packet burst arrival rate of 0.003.

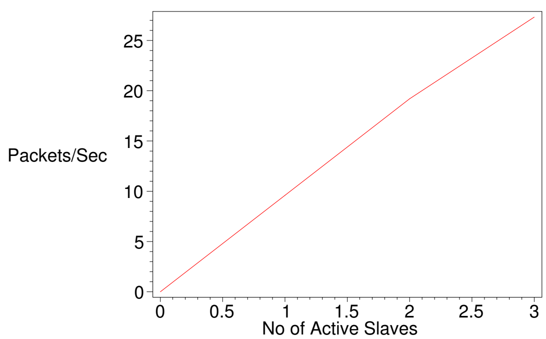


(c) Packet burst arrival rate of 0.004.

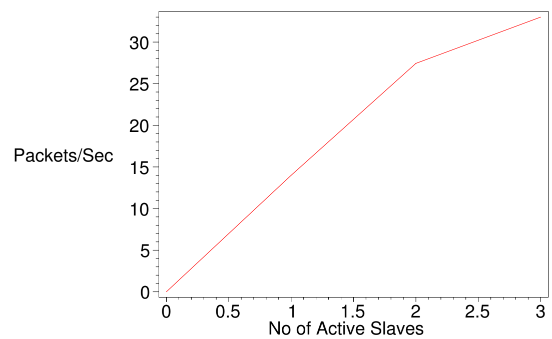


(d) Packet burst arrival rate of 0.005.

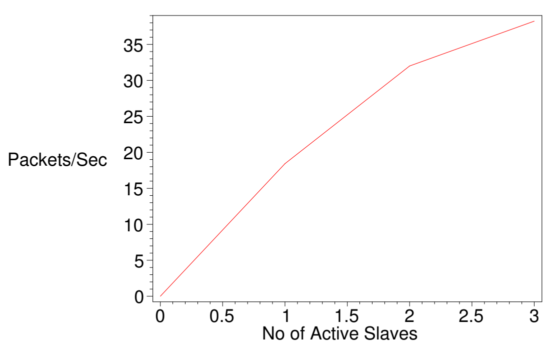
Figure 7.4: Absolute reliability for packets from slaves in P_4 at the sink vs. the number of active slaves in P_4.



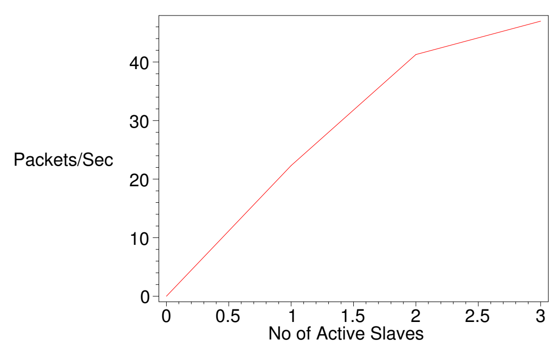
(a) Packet burst arrival rate of 0.002.



(b) Packet burst arrival rate of 0.003.



(c) Packet burst arrival rate of 0.004.



(d) Packet burst arrival rate of 0.005.

Figure 7.5: Absolute reliability for packets from slaves in P_2 at the sink vs. the number of active slaves in P_2.

Table 7.1: Mean, Variance and Standard Deviation for relative reliability at the sink for P_4 and P_2 over a given period of time.

Interval	Piconet	Packet burst arrival rate	Mean	Variance	Standard Deviation
1	P_4	0.002	95.25	27.32	5.22
2	P_4	0.003	85.46	40.39	6.35
3	P_4	0.004	79.61	130.04	11.40
4	P_4	0.005	69.18	250.04	15.81
1	P_2	0.002	99.66	6.05	2.46
2	P_2	0.003	96.54	15.05	3.88
3	P_2	0.004	89.15	49.05	7.04
4	P_2	0.005	84.66	125.05	11.18

Figures. 7.4 and 7.5 show absolute reliability observed at the sink, for packets originating from slaves in piconets P_4 and P_2, respectively. At lower packet burst arrival rates, the absolute reliability is monotonically increasing function of the number of slaves. While this result differs from the corresponding dependencies of relative reliability from Figures. 7.2 and 7.3, one should keep in mind that the sleep management scheme has been designed with the goal of maintaining the *relative* reliability, not its absolute counterpart, within certain limits. Of course, congestion control could be designed the other way around, i.e., by specifying the desired absolute reliability and trying to achieve it with the highest possible relative reliability, which translates into the lowest number of active slaves in source piconets.

Figure 7.6 represents the number of active slaves in P_4 over a period of time. By analyzing the figure I notice that the sleep management algorithm tries to put the slaves into HOLD mode, thereby reducing the traffic pumped into the network, reducing congestion and power consumption in the network, increasing the relative and absolute reliability at the sink.

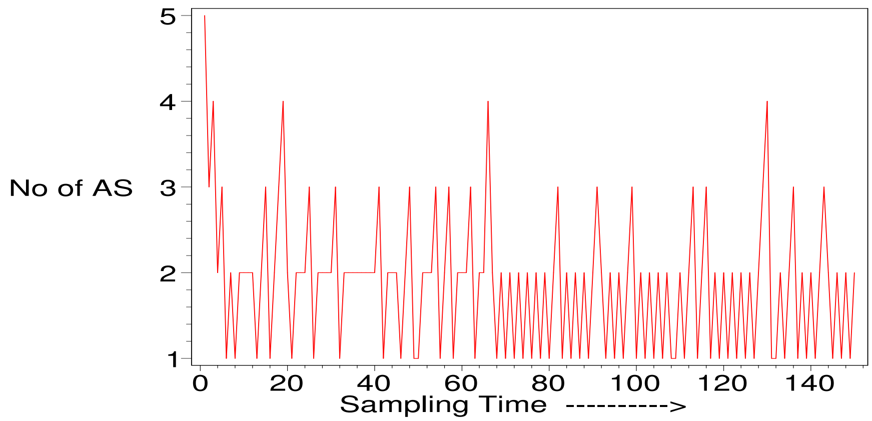
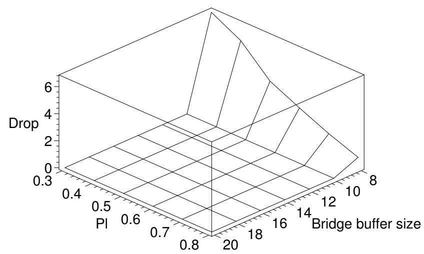


Figure 7.6: Fluctuations of the number of active slaves for P_4.

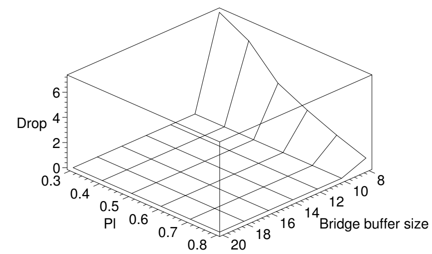
7.3.2 Packet loss at the bridge buffers

Another sign of congestion (and, by extension, decrease in reliability) is the increase in packet loss rates at the bridge buffers. I have measured packet loss rates in the designed scatternet using the same setup as above: the sink was in piconet P_1, the desired reliability has been set to 60%, packet burst arrival rate was set to 0.005 (bursts per Bluetooth time slot), bridge residence time set to one piconet cycle, and polling parameters for slaves and bridges were $M_s = 3$ and $M_b = 12$, respectively. However, in order to get better insight, I have varied the traffic locality probability in the range $P_l = 0.3 \ldots 0.8$ and bridge buffer size in the range $8 \ldots 20$.

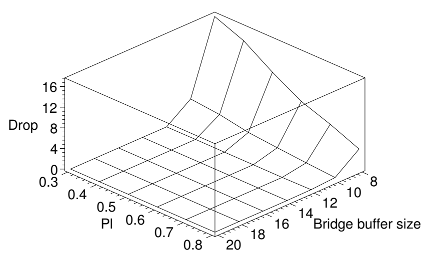
Packet losses at the buffers of bridges B_4, B_9, B_1 and B_3 are shown in Figure. 7.7. Because of the symmetry of the network, B_4 and B_9 exhibit similar packet loss rates; I note that packet losses become significant for high inter-piconet traffic (i.e., with $P_l = 0.3$ and lower) which is characteristic of sensor networks. Similar conclusions hold for packet loss rates at the buffers of 'interior' bridges B_1 and B_3 (which carry the data from P_4, P_6, P_2 and P_3 to P_1), shown in Figures. 7.7(c) and 7.7(d). Since B_1 and B_3 carry data packets from two piconets each, the loss of data packets at these bridges are higher when compared to B_4 and B_9. I observe that under a realistic locality probability of $P_l = 0.3$, buffers sizes of 12 packets or more suffice to keep the packet loss very low; this offers substantial advantage over the value of 40 or more which is necessary in the network



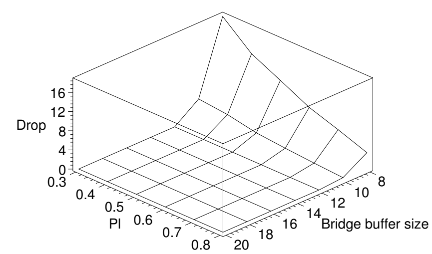
(a) Bridge Buffer Drop Rate in % for B_4



(b) Bridge Buffer Drop Rate in % for B_9



(c) Bridge Buffer Drop Rate in % for B_1



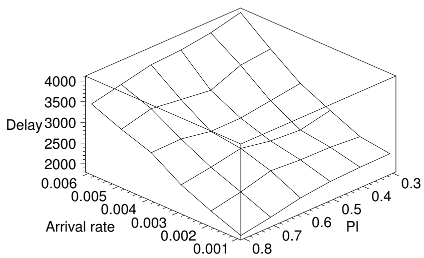
(d) Bridge Buffer Drop Rate in % for B_3

Figure 7.7: Bridge Buffer Drop rate in % for B_1, B_3, B_4, B_9 with $M_s = 3$, $M_b = 12$, Burst size =3, Packet length = 5 slots, Slave Buffer Size = 30, Master Buffer Size = 100, Packet burst arrival rate = 0.005.

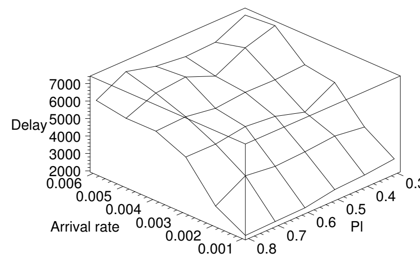
without sleep management [29].

7.3.3 End-to-End Delay

Finally, end-to-end packet delays for traffic from P_6 to P_1 and from P_3 to P_1, are shown in Figures. 7.8(a) and 7.8(b), respectively. Both queuing and transmission delays are taken into consideration to calculate end-to-end delays. In this case, P_l was varied in the range 0.3 . . 0.8 and packet burst arrival rates were in the range 0.001 . . 0.006. Since the interior piconets have more bridges than exterior ones, their carried load is higher, and



(a) End-to-End delays from P_6 to P_1



(b) End-to-End delays from P_3 to P_1

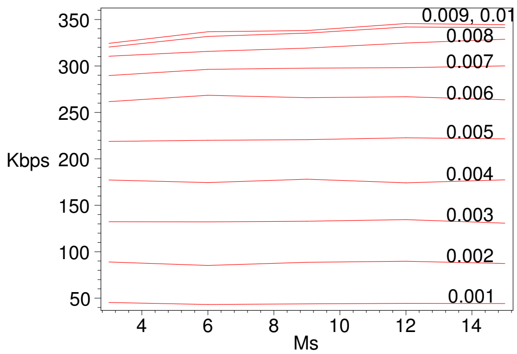
Figure 7.8: End-to-End delays from P_6 to P_1 and P_3 to P_1 with $M_b = 15$, Burst size =3, Packet length = 5, Slave Buffer Size = 30, Master Buffer Size = 100, Bridge Buffer size = 40.

so are the delays.

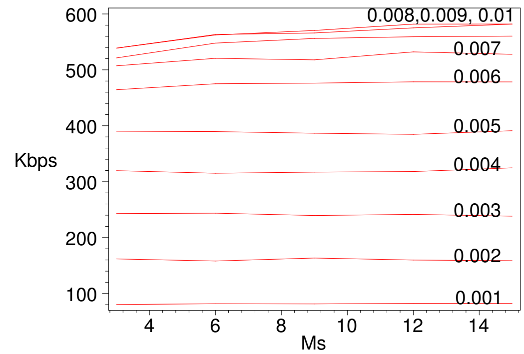
7.3.4 Throughput:

Throughput for a piconet is defined as the rate at which the sensed data packets can be sent and received through that piconet, measured in bits per second. Throughput is obtained by varying the slave parameter $M_s = 4, 6 \,..\, 14$ and arrival rates $\lambda = 0.001, 0.002 \,..\, 0.01$ packet burst arrival rate per Bluetooth time slot. Figures 7.9 represents the throughput for P_4 and P_2 before applying the sleep management algorithm and Figures 7.10 represents the throughput for P_4 and P_2 after applying the sleep management algorithm.

From the figures I observe that, for Figure 7.9 as the arrival rate increases the throughput increases, and reaches saturation for packet burst arrival rates of $0.006, 0.007 \,..\, 0.01$. However, after applying the sleep management algorithm P_4 and P_2 exhibits lower throughput for higher packet burst arrival rates as shown in Figure 7.10. Hence, I conclude that by applying sleep management algorithm for the source piconets we can decrease the amount of traffic generated and thereby reduce congestion, power consumption and end-to-end delays in the network.

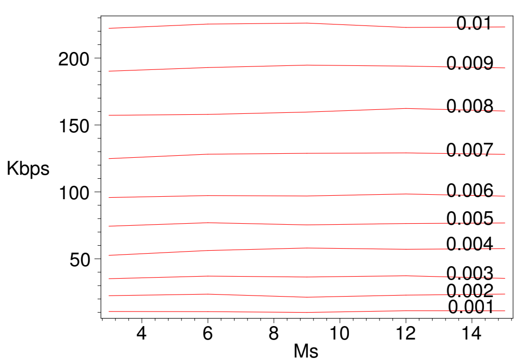


(a) Throughput for P_4

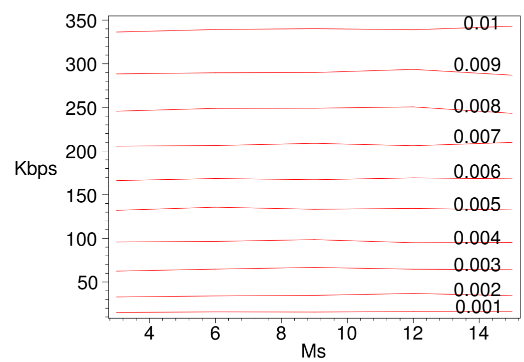


(b) Throughput for P_2

Figure 7.9: Throughput for P_4 and P_2 before applying the sleep management algorithm with $M_b = 15$, Burst size = 3, Packet length = 5, Slave Buffer Size = 30, Master Buffer Size = 100, Bridge Buffer size = 40.



(a) Throughput for P_4



(b) Throughput for P_2

Figure 7.10: Throughput for P_4 and P_2 after applying the sleep management algorithm with $M_b = 15$, Burst size = 3, Packet length = 5, Slave Buffer Size = 30, Master Buffer Size = 100, Bridge Buffer size = 40.

Chapter 8

Algorithm to Maintain Fixed Reliability at the Sink

In this chapter a second sleep management algorithm is introduced. This algorithm maintains the required (fixed) absolute event reliability at the sink using minimal slave activity at the source. It uses pre-calculated activity values obtained from the analytical and simulation models of the network based on which the sleep time of the slaves is calculated. The algorithm is analyzed for the absolute event reliability observed at the sink.

8.1 Maintaining fixed reliability at the sink

Let us assume that N_p piconets are reporting the sensing information to the master in sink piconet. Piconets are indexed by index $i = 1 \,..\, N_p$, and each piconet P_i has m_i ordinary slaves (i.e., slaves without the bridging function). The upper limit of reliability of sensed information is determined by the piconet capacity. If five-slot packets are used and there is no downlink data traffic (which, in fact, is needed to carry control information), the maximum absolute reliability is $R_{max} = 1/(T + 5\,T) = 266$ packets per second, where $T = 625\,\mu\,s$ is the duration of Bluetooth time slot. In practice, the maximum achievable number will be lower, due to the losses at the bridges and presence of downlink traffic needed to send queries and control information. In many cases it will

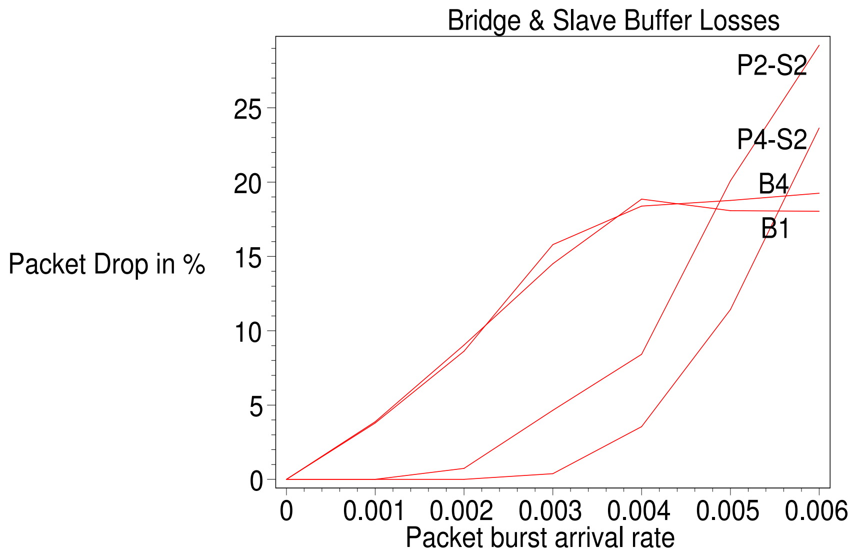


Figure 8.1: Blocking probability vs. offered load, at the slaves in P_2 and P_4 and the bridges B_1 and B_4

suffice to maintain the reliability at some application-defined level R; assuming uniform conditions, the absolute reliability to be contributed by each piconet is $R_i = R/N_p$.

We also need to estimate packet losses at the slave and bridge buffers. The bridge loss rate is a function of total piconet load, bridge load, bridge polling parameter M_b, slave polling parameter M_s and bridge buffer size. In case the topology is fixed and the polling parameters are known, we may assume that the bridge loss rate depends on the bridge packet arrival rate and total piconet load. In the topology shown in Figure. 7.1, the bridge loss rate may be approximated with $P_{b,i} = K_i \overline{B} L \lambda_{b,i}$, where $\overline{B}$ is the average burst size, L is the packet size in slots, $\lambda_{b,i}$ is the burst arrival rate towards the bridge and K_i is the proportionality constant. Measured values of blocking probabilities are shown in Figure. 8.1.

Therefore, the sink can calculate losses from source piconets and communicate them to source piconets in order to adjust the slaves' activities. Of course, these losses should not be too high—say, up to a few percent—otherwise the network is operating in the congestion regime, in which case it is better to partition it into sections with separate (and different) sinks, and thus avoid congestion. When losses along the path are known, the source piconet can compensate for the losses by scaling its absolute reliability to

$$R'_i = \frac{R_i}{\prod_{overpath}(1 - P_{b,i})} \tag{8.1}$$

Finally, absolute reliability has to be transformed into the average number of active slaves per piconet. Mean number of packets contributed by the slave per second is $R_s = \lambda \overline{B}/T$, while the mean number of active slaves per piconet is

$$A_{s,i} = \frac{R'_i T}{\lambda \overline{B}} \tag{8.2}$$

Procedure `LONG` : managing the long activity period.

Data: a, b, m_i

Result: initial value of the short activity management counter C_s

begin
- **if** $a \leq b$ **then**
  - put $m_i - 1$ most recently used slaves to HOLD mode for bT_u seconds;
  - remaining slave should be active for aT_u seconds;
- **else**
  - put $m_i - \lceil \frac{a}{b} \rceil$ most recently active slaves to sleep for bT_u ;
  - among $\lceil \frac{a}{b} \rceil$ remaining slaves, activate $\lfloor \frac{a}{b} \rfloor$ least recently used slaves for next bT_u seconds ;
  - the remaining slave S^* should be active for $(a \bmod b)T_u$ seconds ;
- **end**
- $C_s = a \bmod b$;

end

The mean value of $A_{s,i}$ slaves at any given time can be obtained in the following manner: assume that $A_{s,i}$ is a rational number: $A_{s,i} = \frac{a}{b}$ where a, b are integers. Further, assume that the activity control process consists of basic time units T_u when the slave can be put in HOLD state. (Note that T_u should be much larger than Bluetooth time slot; in this work, we assume that T_u is one second.) Then, a units of activity have to be executed by the slaves over every b time units. Let us denote the long activity management period with bT_u, and the short activity management period with T_u.

Procedure SHORT: managing the short activity period.

Data: C_s

begin
    if $C_s > 0$ **then**
        $C_s = C_s - 1$;
    else if $C_s = 0$ **then**
        $C_s = C_s - 1$;
        put slave S^* to HOLD for $(b - a \bmod b)\,T_u$ seconds ;
    end
end

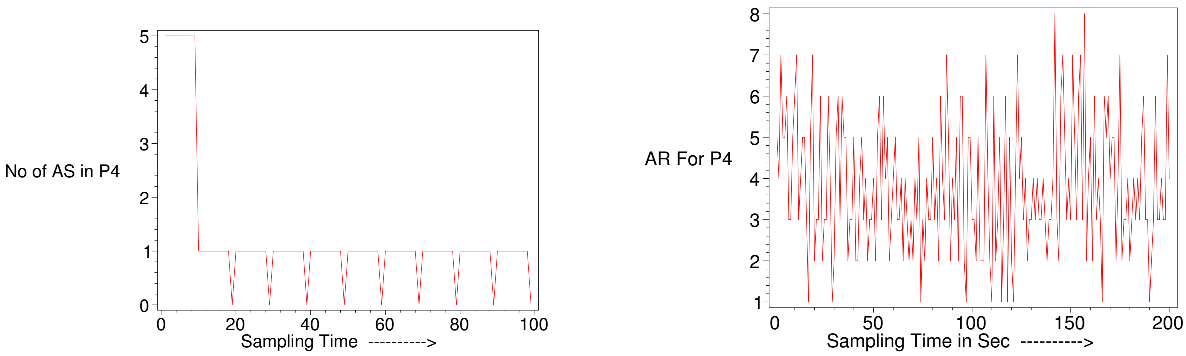
Within the long activity management period, we try to minimize the number of slaves needed to accomplish this activity requirement. In effect, this is an attempt to minimize the protocol overhead since the slaves will sleep in the HOLD mode and this has to be negotiated; the less negotiation we undertake, the more efficient the protocol becomes.

During the short management cycles, we will try to balance the utilization of various slaves in order to extend the battery life of each slave. Additionally, feedback from the sink can be communicated to the source piconets in order to slightly decrease or increase the average number of active slaves, which will result in decrementing or incrementing the value of a.

In this manner, we are able to maintain the reliability at the sink at the desired level. The entire procedure is shown in Algorithm 3.

Algorithm 3: Maintaining the fixed reliability at the sink.

```
Data: total event reliability at the sink R, scatternet topology, N_p,
      m_i, i = 1 .. N_p, packet burst arrival rate λ per slave, mean burst size B̄
begin
    for each piconet P_i do
        estimate event reliability R_i ;
        estimate load through outgoing bridges ;
        estimate packet loss through each bridge ;
        estimate total packet loss towards the sink ;
        recalculate R'_i;
        find A_{s,i}, a, b ;
        C_0 = 0 ;
        after every T_u seconds do
            C_0 = C_0 + 1 ;
            management of long activity period ;
            if (C_0 mod b == 0) then
                call LONG ;
            end
            management of short activity period ;
            call SHORT ;
        end
    end
end
```



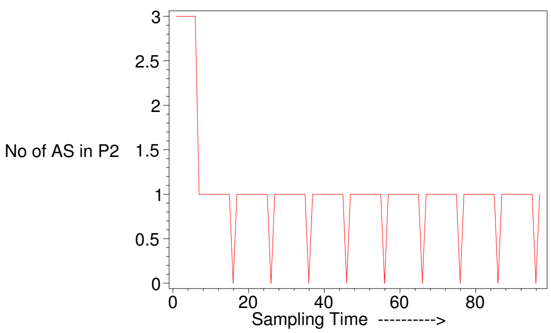
(a) Number of active slaves over time.

(b) Absolute Reliability from P_4 at the sink.

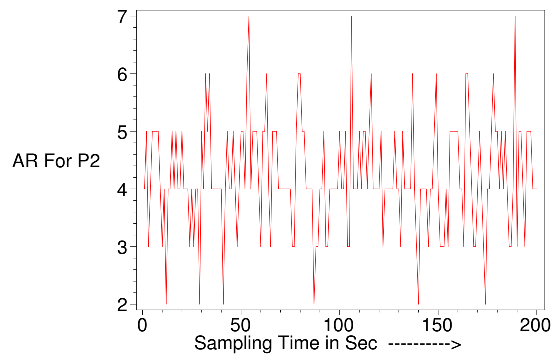
Figure 8.2: Mean number of active slaves and absolute reliability at the sink, for packets from slaves in P_4.

In order to validate this algorithm, we have performed simulation experiments with the required event reliability of 20 packets per second at the sink from all the source piconets(20 packets/5 piconets = 4 packets/Sec from each source piconet). Packet burst arrival rate for each slave, when active, was set to $\lambda = 0.001$. Once the slaves are put to HOLD mode, they remain in that mode till their sleep time expires. In this simulation the long activity period was set to 10s and the short activity period may have any value between 1 and 10s. The reliability requirement was mapped into bridge packet burst arrival rates and losses through the bridges were estimated as 3% for B_4 and 5% for B_1, respectively. Then the source piconet transmission rates were set to 4.3 packets per second for $P_4, P_5,$ and P_6 and 4.21 packet per second from P_2 and P_3. The resulting activity of the slaves in piconet P_4 and P_2 and their absolute reliability observed at the sink are shown in Figures 8.2 and 8.3; as can be seen, the algorithm manages to maintain the mean value of absolute reliability around the desired value, while the number of active slaves is minimized.

Figure 8.4 shows the total absolute event reliability observed at the sink for the source piconets (P_2, P_3, P_4, P_5, P_6) for the above specified arrival rate and the required event reliability. From the Figure we can analyze that the absolute event reliability observed



(a) Number of active slaves over time.



(b) Absolute Reliability from P_2 at the sink.

Figure 8.3: Mean number of active slaves and absolute reliability at the sink, for packets from slaves in P_2.

at the sink fluctuates around the required event reliability. The fluctuation in absolute event reliability observed at the sink are the results of the transmission delays and the queuing delays at various bridges from source to sink. Figures 8.2(a) and 8.3(a) gives the number of active slaves fluctuating over a period of time for P_4 and P_2. For the above given inputs (required event reliability of 20 and Packet burst arrival rate $\lambda = 0.001$) and after taking the packet loss rates for each piconet into consideration. the sink has calculated 1 slave to be active in P_4 and P_2 and the short activity period was calculated as 9s. Hence, by both the Figures 8.2(a) and 8.3(a) we can observe that 4 slaves in P_4 and 2 slaves in P_2 are placed on Hold for the whole 10s (long activity period) and one slave is active for only 9s (Short activity period).

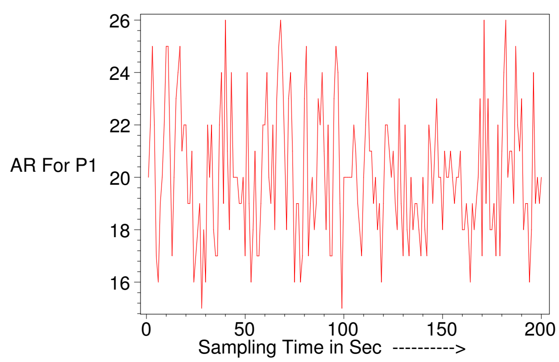


Figure 8.4: Total absolute event reliability at the sink, from the source (P_2, P_3, P_4, P_5, P_6).

Chapter 9

Conclusion

The focus of this thesis was to reduce congestion and power consumption in a Bluetooth based sensor network. In order to analyze the behavior of a sensor network and the effect of congestion on it, a Bluetooth-based wireless sensor network has been simulated and tested in the absence of congestion control algorithms. The simulated network is of triangular topology consisting of six sensing piconets with all the nodes equipped with finite buffers. Each piconet has a master and three slaves and the piconets are interconnected with bridges. The interior piconets have 4 bridges when compared to that of 2 in the exterior piconets. I have considered scatternet operating under walk-in bridge scheduling, in which the masters poll both ordinary slaves and bridges using the E-limited polling scheme. The simulated network was tested by altering various factors such as packet burst arrival rates, bridge buffer and slave buffer sizes, probability locality, M_s and M_b and BRT. The obtained results show the dependence of data packet drop rates and data packet delays on the size of different buffers; larger buffers lead to reduced drop rates and increased delays. I have also shown that both delays and packet drop rates are critically dependent on the aggregate load of individual piconets; a piconet with too heavy load will exhibit inordinately high drop rates and high delays. The simulations were also carried out to analyze the effect of BRT over the network and from the obtained results it was clear the lower values of BRT results in lower data packet losses and lower packet delays.

After analyzing the simulated network, a need for congestion control and power management was identified to minimize the power consumption and the packet losses at the bridges. In order to analyze the effect of number of active slaves at the source piconet for the observed event reliability at the sink, simulations were carried out by varying the number of active slaves at the source piconet. The exterior piconets were equipped with five sensing slaves. P_1 acted as sink while P_4 acted as source and the rest of them were need to generate background traffic towards the sink with all their slaves active. From the obtained results a reliability/congestion graph was drawn and by analyzing the graph it was clear that the congestion and power consumption in the network can be reduced by dynamically altering the number of active slaves at the source. Two sleep management algorithms were developed and applied to the above described Bluetooth based sensor network. In this case P_1 was assigned as the sink and P_2, P_3, P_4, P_5 and P_6 were assigned as the source. All the source piconets were equipped with the sleep management algorithm. Both algorithms are based on sleep scheduling of Bluetooth slaves and the HOLD mode was used as the power saving mode. The first algorithm keeps the whole network within the acceptable range of packet losses using the minimal slave activity. In this case source piconets use the information measured at the sink in order to regulate the activity of the slaves. The second algorithm maintains the required (fixed) event reliability at the sink using minimal slave activity. It uses pre-calculated activity values obtained from the analytical and simulation models of the network. The power controlled piconets within the sensor scatternet were tested for the event reliability obtained at the sink, the bridge buffer loss rates, the end-to-end delays and the throughput. After analyzing the obtained results I conclude that the designed sleep management algorithms significantly reduce congestion, power consumption, source-to-sink delays, while maintaining a control on the observed event reliability and minimizing packet losses due to finite buffers in the bridge nodes.

Bibliography

[1] R. Kapoor A. Capone and M. Gerla. Efficient Polling Scheme for Bluetooth Picocells. In *In Proceedings of IEEE International Conference on Communications ICC 2001*, volume 7, pages 1900–1994, Helisink, Finland, 2002.

[2] O. B. Akan and I. F. Akyildiz. ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks. In *IEEE/ACM Transaction on Networking (to appear)*, 2005.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless Sensor Networks: A Survey. *Computer Networks, (Elsevier) Journal*, 38:393–422, March 2002.

[4] I. F. Akyildiz, M. C. Vuran, and O. B. Akan. On Exploiting Spatial and Temporal Correlation in Wireless Sensor Networks. In *Proc. WiOpt'04: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, pages 71–80, March 2004.

[5] Raffaele B., Marco C., and Enrico G. Wireless Access to Internet via Bluetooth: Performance Evaluation of the edc Scheduling Algorithm. In *WMI '01: Proceedings of the first workshop on Wireless mobile internet*, pages 43–49, New York, NY, USA, 2001. ACM Press.

[6] S. Baatz, M. Frank, C. Kühl, P. Martini, and C. Scholz. Bluetooth Scatternets: An Enhanced Adaptive Scheduling Scheme. pages 782–790, New York, June 2002.

[7] S. Basagni, R. Bruno, and C. Petrioli. Bluetooth Scatternet Formation in Bluetooth Networks. In Stefano Basagni, Marco Conti, Silvia Giordano, and Ivan Stojmenovic, editors, *Ad Hoc Networking*, New York, NY, 2003. IEEE Press.

[8] J. Beutel, O. Kasten, and M. Ringwald. BTnodes - A Distributed Platform for Sensor Nodes. In *Proc. 1st Intl. Conf. on Embedded Networked Sensor Systems (SenSys)*, pages 292–293, November 2003.

[9] Bluetooth SIG. Bluetooth Network Encapsulation Protocol (BNEP) Specification. Technical report, Revision 0.95a, June 2001.

[10] Bluetooth SIG. *Specification of the Bluetooth System – Architecture & Terminology Overview*, volume 1. Version 1.2, November 2003.

[11] L. Ching, A. K. Mehta, and S. Kai-Yeung. A new Bluetooth Scatternet Formation Protocol. *Mob. Netw. Appl.*, 8(5):485–498, 2003.

[12] M. Daniele, Z. Andrea, and P. Gianfranco. Performance Evaluation of Bluetooth Polling Schemes: An Analytical Approach. *Mob. Netw. Appl.*, 9(1):63–72, 2004.

[13] Rsoft Design Group. *Artifex v.4.4.2.* San Jose, CA, 2003.

[14] L. Har-Shai, R. Kofman, G. Zussman, and A. Segall. Inter-Piconet Scheduling in Bluetooth Scatternets. Technical report, Department of Electrical Engineering, Israel Institute of Technology, Haifa 32000, Israel, August 2002.

[15] B. Hong and V. K. Prasanna. Optimizing System Life time for Data Gathering in Networked Sensor Systems. In *Algorithms for Wireless and Ad-hoc Networks (A-SWAN) (Held in conjunction with MobiQuitous)*, August 2004.

[16] B. Hong and V. K. Prasanna. Optimizing a Class of In-network Processing Applications in Networked Sensor Systems. In *The 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2004)*, October 2004.

[17] Z. Honghai and Jennifer C. H. A Scheduling Algorithm for Transporting Variable Rate Coded Voice in Bluetooth Networks. In *WOWMOM '02: Proceedings of the 5th ACM international workshop on Wireless mobile multimedia*, pages 25–32, New York, NY, USA, 2002. ACM Press.

[18] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed Diffusion for Wireless Sensor Networking. *ACM/IEEE Transactions on Networking*, 11(1):2–16, February 2003.

[19] N. Johansson, U. Körner, and L. Tassiulas. A Distributed Scheduling Algorithm for a Bluetooth Scatternet. In *Proceedings of the International Teletraffic Congress – ITC-17*, pages 61–72, Salvador de Bahia, Brazil, September 2001.

[20] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *Mobile Computing*, 353, 1996.

[21] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach Featuring The Internet*. Addison-Wesley Longman, Boston, MA, 3rd edition, 2005.

[22] M. Leopold, M. Dydensborg, and P. Bonnet. Bluetooth and Sensor Networks: A Reality Check. In *1st ACM Conference on Sensor Networks*, November 2003.

[23] G. Miklos, A. Racz, Z. Turanyi, A. Valko, and P. Johansson. Performance Aspects of Bluetooth Scatternet Formation. In *In Mobile Ad Hoc Networking and Computing (MobiHOC)*, pages 147–148, 2000.

[24] J. Mišić, K. L. Chan, and V. B. Mišić. Performance of Bluetooth Piconets under E-limited Scheduling. Technical Report TR 03/03, Department of Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada, May 2003.

[25] J. Mišić, K. L. Chan, and V. B. Mišić. Admission Control in Bluetooth Piconets. 53(3):890–911, May 2004.

[26] J. Mišić and V. B. Mišić. Bridges of Bluetooth county: Topologies, Scheduling, and Performance. 21(2):240–258, February 2003.

[27] J. Mišić, V. B. Mišić, and K. L. Chan. Stability and Scalability of Bluetooth Scatternets under Walk-In Bridge Scheduling. In *Dynamics of Continuous, Discrete and Impulsive Systems, special issue on Routing/Signaling Protocols in Wireless/Wired Communication Networks*, 2003.

[28] J. Mišić, V. B. Mišić, and K. L. Chan. Analysis of Loosely Coupled Scatternets. In *Proc. SPECTS 2004 (CD ROM)*, San Jose, CA, July 2004.

[29] J. Mišić, V. B. Mišić, and G. R. Reddy. On the performance of bluetooth scatternets with finite buffers. In *2nd International Workshop on Wireless Ad Hoc Networking (WWAN 2005)*, 2005.

[30] V. B. Mišić and J. Mišić. Adaptive Inter-Piconet Scheduling in Small Scatternets. *ACM MC^{2}R – Mobile Computing and Communications Review*, 7(2), April 2003.

[31] V. B. Mišić and J. Mišić. Polling and Bridge Scheduling Algorithms in Bluetooth. Technical Report TR 03/04, Department of Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada, R3T 2N2, September 2003.

[32] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, April, 1989.

[33] R. Nada Golmie, E. Van Dyck, and A. Soltanian. Interference of Bluetooth and IEEE 802.11: Simulation Modeling and Performance Evaluation. In *Proceedings 4th ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 11–18, Rome, Italy, July 2001.

[34] Q. Wang and D. P. Agrawal. A Dichotomized Rendezvous Algorithm for Mesh Bluetooth Scatternets. *Ad Hoc and Sensor Wireless Networks Journal*, 1(1-2):65–88, 2005.

[35] Standard for part 15.4: Wireless MAC and PHY specifications for low rate WPAN. IEEE Std 802.15.4, IEEE, New York, NY, October, 2003.

[36] G. Tan and J. Guttag. A Locally Coordinated Scatternet Scheduling Algorithm. In *Proceedings of the 26th Annual Conference on Local Computer Networks LCN 2002*, pages 43–49, Rome, Italy, November 2002.

[37] The Bluetooth Special Interest Group Ver. 2.0. Draft specification of the Bluetooth System, November 2004.

[38] C. Wan, S. B. Eisenman, and A. T. Campbell. CODA: Congestion Detection and Avoidance in Sensor Networks. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 266–279. ACM Press, November 2003.

[39] C.Y. Wan, A. T. Campbell, and L. Krishnamurthy. PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks. In *First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA 2002)*, pages 1–11, September 2002.

[40] W. Ye, J. Heidemann, and D. Estrin. Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks. *ACM/IEEE Transactions on Networking*, 12(3):493–506, June 2004.

[41] M. Younis, M. Youssef, and K. Arisha. Energy-aware Management for Cluster-based Sensor Networks. *Computer Networks*, 43(5):649–668, November 2003.

Appendix A

Simulating a Scatternet acting as a Sensor Network

Artifex simulation models are based on colored, timed Petri Nets in which tokens are arbitrary user-defined data structures, and transitions may optionally contain priority information and enabling predicates. I will omit the basic operation of the Petri Nets; the interested reader can consult any among the many books and tutorials available, such as [32]. Further details about the operation of the Artifex simulation engine can be found in the corresponding user manuals.

Before describing the simulator in more detail, let me once again describe the sensing scatternet acting as a sensor network simulated in this thesis. The sensing scatternet consists of six piconets interconnected through bridges operating in a walk-in fashion, as shown in Figure 4.1. Although such a symmetric topology is unlikely in practice, it can nevertheless serve as a 'stress test' setup in which the performance of sensing scatternets under E-limited polling and walk-in bridge scheduling, as well as the impact of the congestion and power consumption, can be readily assessed.

Due to the symmetry of the topology, each of the 'exterior' piconets 1, 4, and 6 will behave in an identical manner, as is the case with the 'interior' piconets 2, 3, and 5. The interior piconet have three ordinary slaves and the exterior piconets have five ordinary slaves. These slaves generate traffic toward other slaves in the same piconet

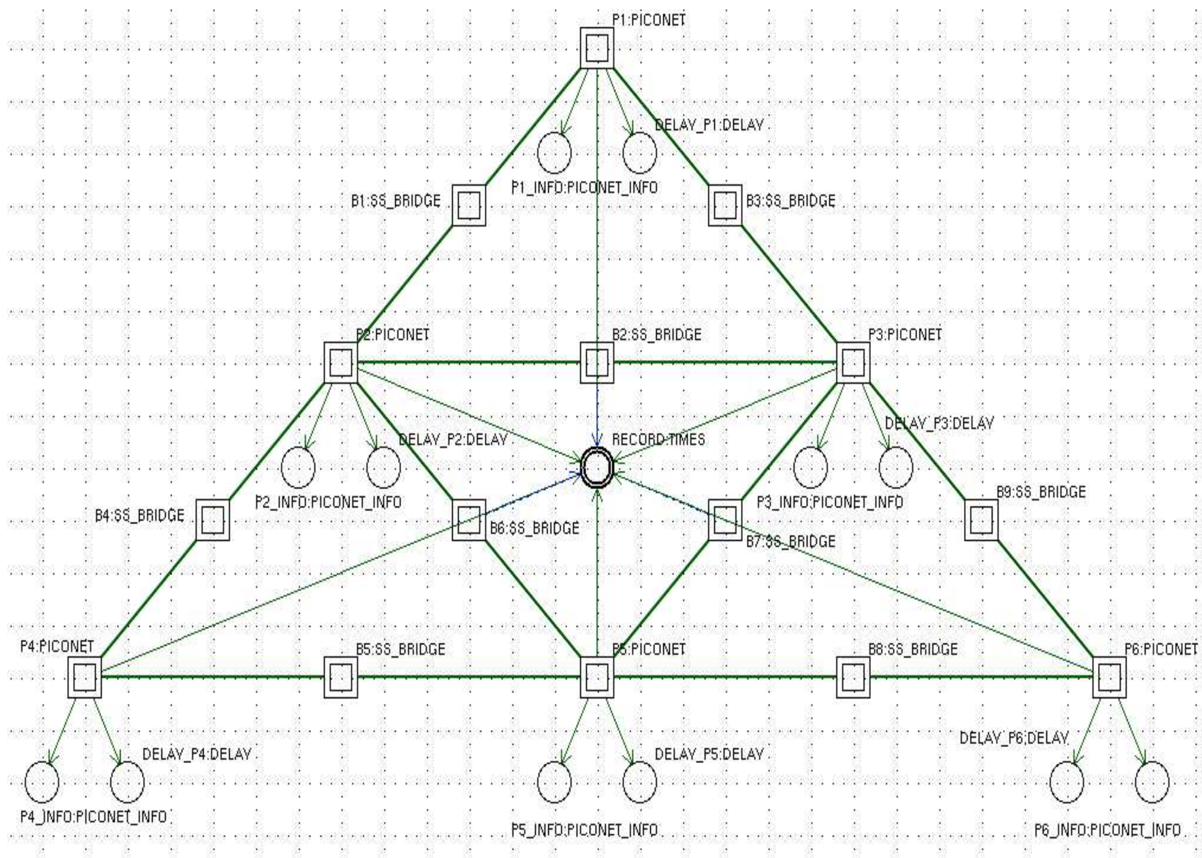


Figure A.1: Simulator top level: SCATTERNET with piconet and bridge classes and auxiliary structures for measurements.

with the probability of P_l, and toward slaves in any other piconet with the probability $(1 - P_l)/5$. In the sensing scatternet P_1 acts as the sink and the rest of them act as the source piconets, transferring the sensed data to the sink. On the other hand the sink tries to regulate the number of active slaves in the source piconets to reduce congestion and power consumption the network.

In case of nonadjacent piconets, the inter-piconet traffic is routed through the shortest path. When two such paths exist (e.g., traffic from piconet P_5 to piconet P_1 may go through P_2 or P_3), traffic is split evenly between those paths. Each master polls its slaves in a fixed cyclical sequence, using E-limited polling.

This logical topology is modelled through the Artifex simulation model shown in Figure A.1. The entire model contains a number of such classes, each of which contains places, transitions, and arcs which connect them, as well as the instances of other classes. The hierarchical structure of the model allows for easy development of sophisticated models, through simple repeated decomposition into logical submodels. For example, the top level (SCATTERNET) class contains a number of named instances of the PICONET class and a number of named instances of the BRIDGE class. (Note that the classes are shown as rectangles with double lines.) As there are no global variables in Artifex models, all communication must be performed by token passing, and all information is contained within the appropriate token data structures. This facilitates both development and validation of such models.

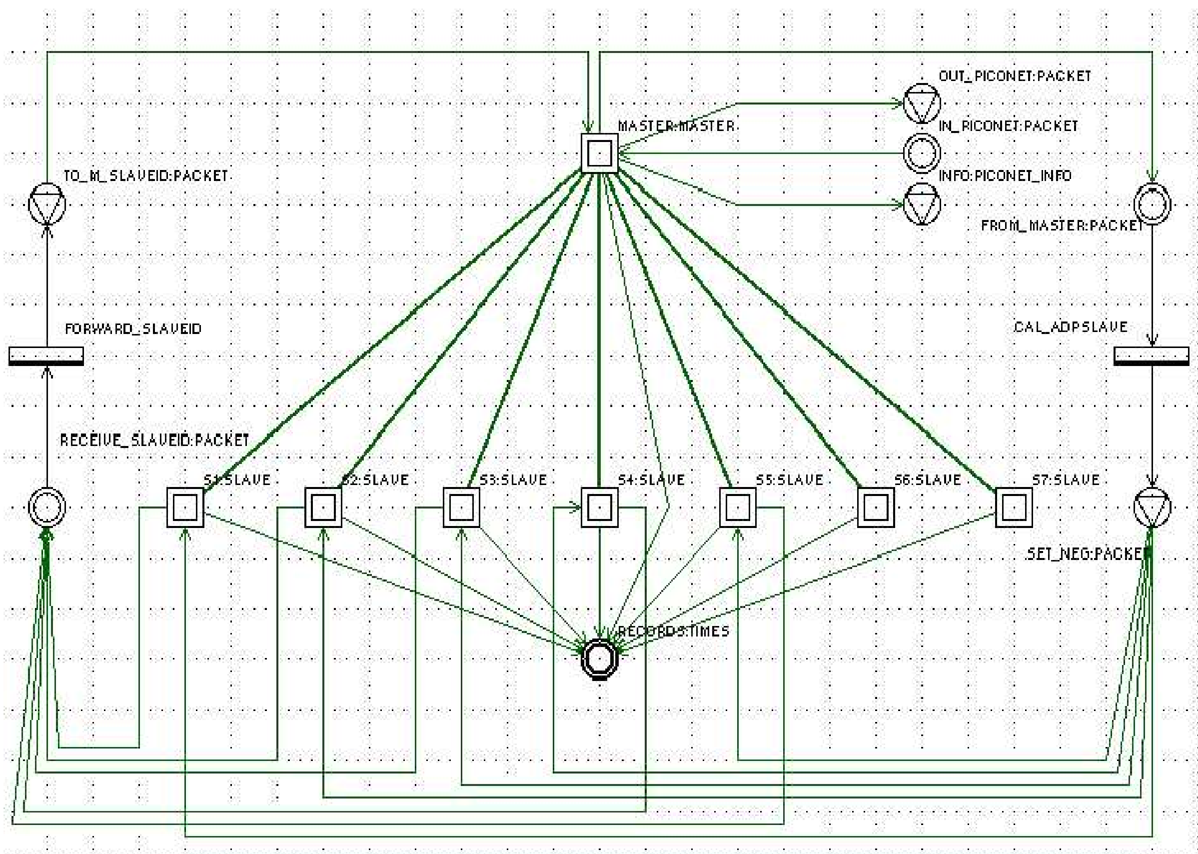


Figure A.2: Simulator: PICONET class with master and slaves.

Additional places and arcs perform activities related to simulation management and measurement. Places shown in double lines are the so-called input places; by convention, input places can accept tokens from outside the class instance. Places with an inscribed triangle are output places, which are a notational convenience as they don't contain as token queue – instead, their incoming arcs are directly connected to places outside of the class they belong to.

Figures A.2 and A.3 show the structure of the PICONET and BRIDGE classes, respectively. As can be seen, the PICONET class contains instances of the MASTER and SLAVE classes, the structure of which is shown in Figures A.4 and A.5 respectively. Although the structure of the MASTER class is nontrivial, the inclusion of more sophisticated aspects can be accomplished quite simply in most cases. For example, adding other polling schemes (instead of the E-limited scheme) requires a simple modification to a single transition in Figure A.6.

Since the simulator operates at the MAC level, the time unit in all simulations is the basic time slot of the Bluetooth clock, $T = 0.625ms$. This is the basic time interval unit to which all activities at the MAC level are synchronized.

I note that the simulator contains no provisions to model noise and interference at the PHY level. All simulations were set to run for a predefined number of time slots in two steps, as follows. The first step, the time interval of T_0 time slots, is used to 'warm up' the simulator so as to avoid measuring any transient effects. When this interval ends, all the measurement variables are reset, and the simulator is run for a predefined measurement interval of T_m. Since both time intervals are user defined, fine control of simulator operation can be achieved with ease. In our measurements, warm up and measurement times were set to $288,000T$ and $960,000T$, which correspond to three and ten minutes of real time, respectively.

Each slave generates packets in bursts with geometrically distributed burst sizes with a mean value of $\overline{B} = 3$ packets. This corresponds to the case where the traffic to be

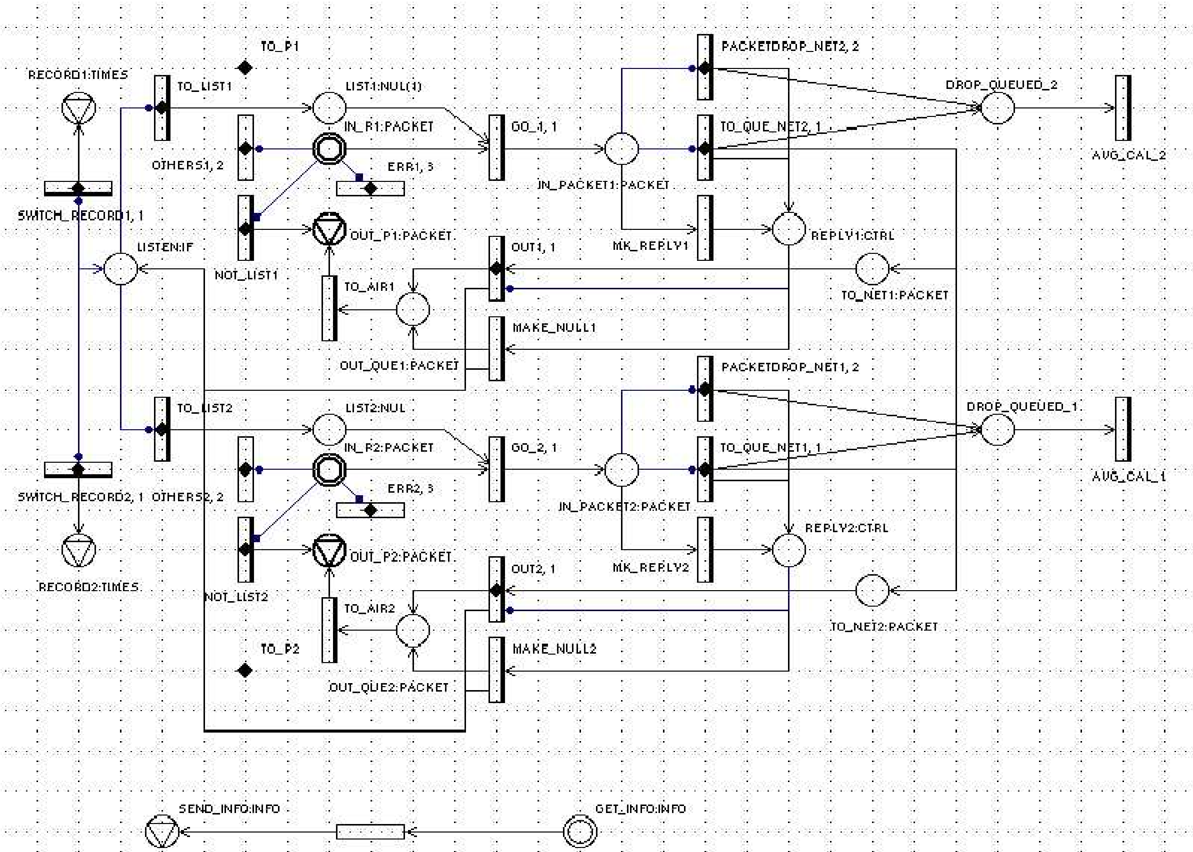


Figure A.3: Simulator: portion of the structure of the BRIDGE class.

transported over Bluetooth links is generated and consumed by the applications running on Bluetooth (and other) devices. This traffic will be formatted as a stream of packets (or protocol data units, PDUs) conforming to the rules of the protocol used by the application. In most practical cases, that protocol will belong to the ubiquitous TCP/IP family [21].

Longer application packets have, then, to be repackaged or *segmented* into Bluetooth packets, with appropriate administrative information added to the Bluetooth packet headers. At the receiving device, the complementary *reassembly* operation will take place. An example of such a segmentation approach is the Bluetooth Network Encapsulation Protocol, or BNEP [9] All devices are assumed to use the same segmentation/reassembly

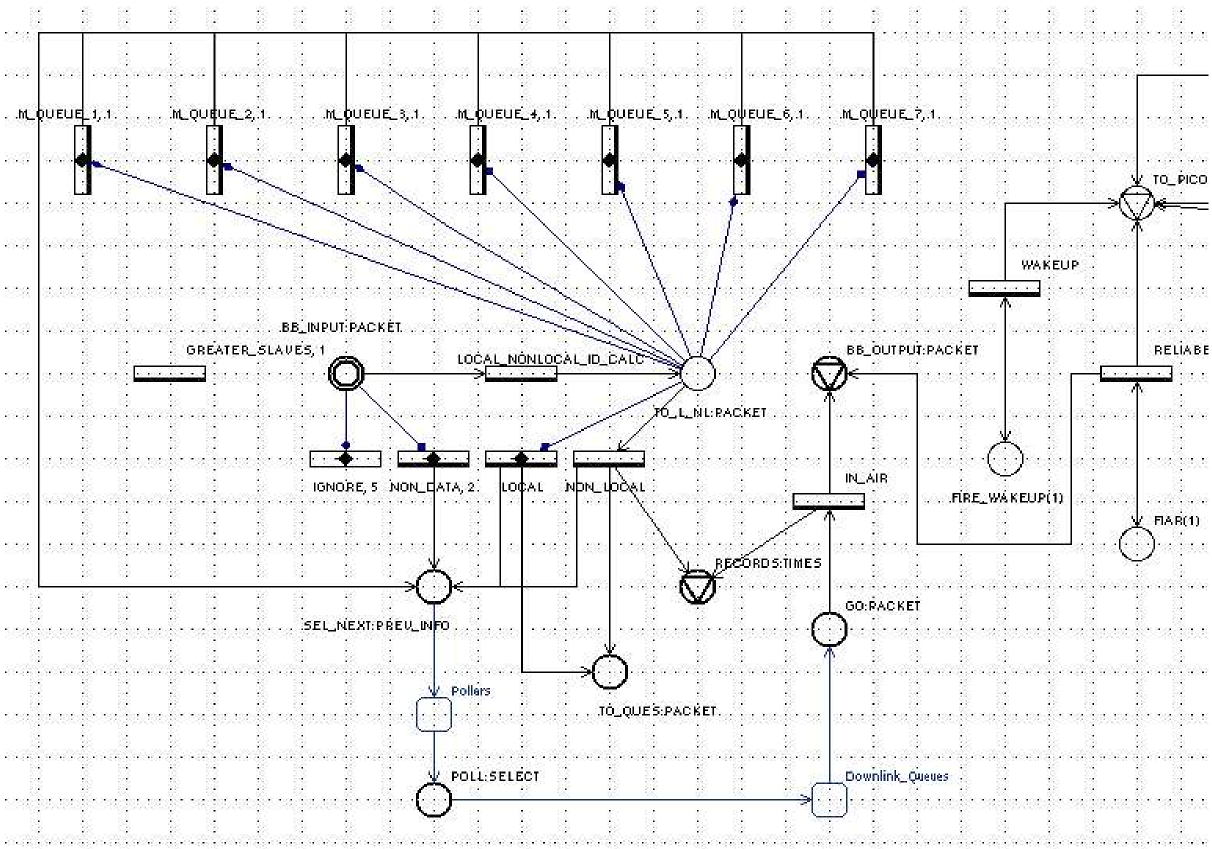


Figure A.4: Simulator: structure of the MASTER class.

protocol, therefore the mean burst size has the same value for all devices.

It should be noted that the Bluetooth specification does not discuss any routing scheme; in fact, routing must be performed at the higher layers of the Bluetooth protocol stack. On account of that, I have implemented addressing and routing through a simple hard-wired scheme.

Five-slot packets of the DH5 type with a 341 byte payload are used throughout the simulation [37]. Note that the current version of the standard allows for packets with larger information carrying capacity, such as the so-called 2DH5, which will put even more strain on device buffers and lead to increased packet blocking. However, I have chosen not to include those packets since they do not differ conceptually from the more

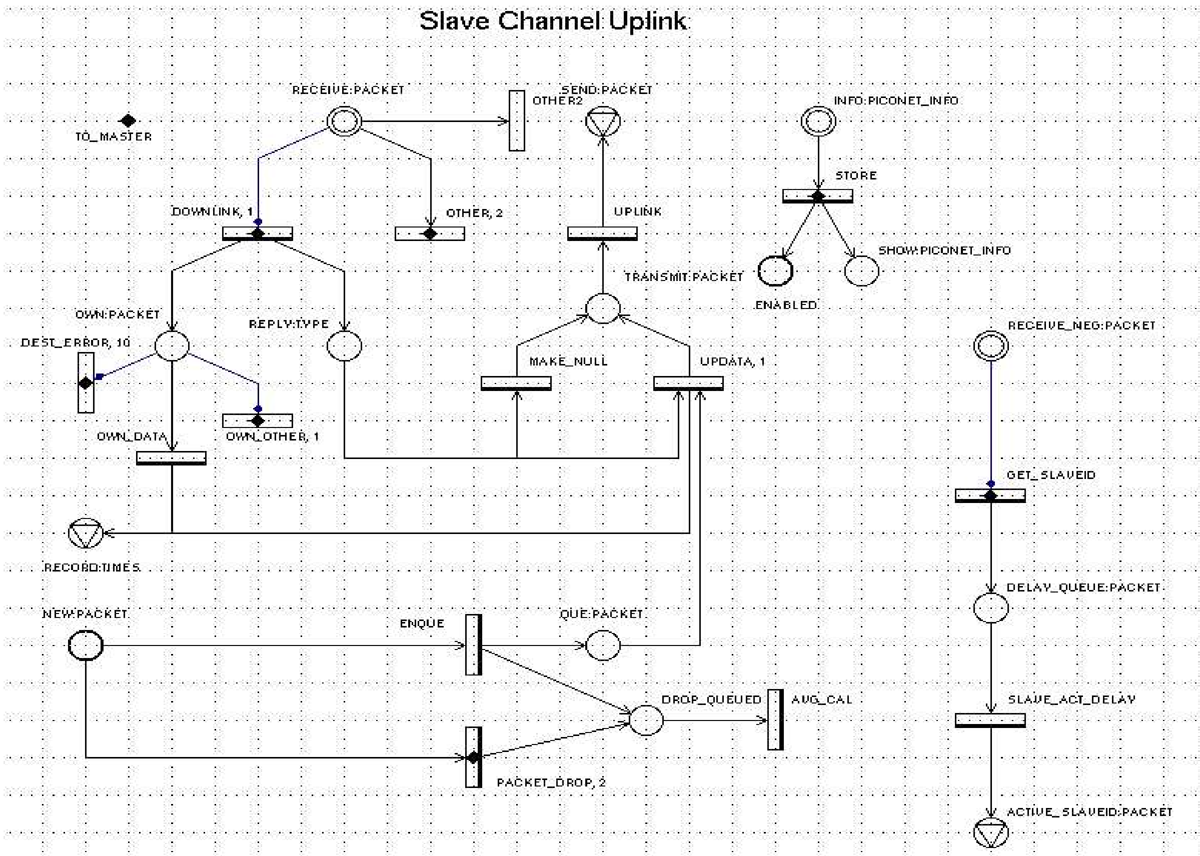


Figure A.5: Simulator: structure of the SLAVE class.

common DH5 ones, so that the obtained results hold – after scaling to accommodate the different capacity of such packets, of course. Furthermore, there are few, if any, Bluetooth devices capable of supporting the new packet types as yet.

The piconet and the bridge was designed and implemented by K. L. Chan. Chan was a former MSc student of Dr. J. Mišić at Hong Kong University of Science and Technology, China. Making use of this piconet and bridge a Bluetooth scatternet of triangular topology with finite buffers was designed, simulated and analyzed by me. Later this Bluetooth scatternet was incorporated with sensors network functionalities and analyzed for congestion in the network. Based on these results, congestion control algorithms were designed, simulated and analyzed.

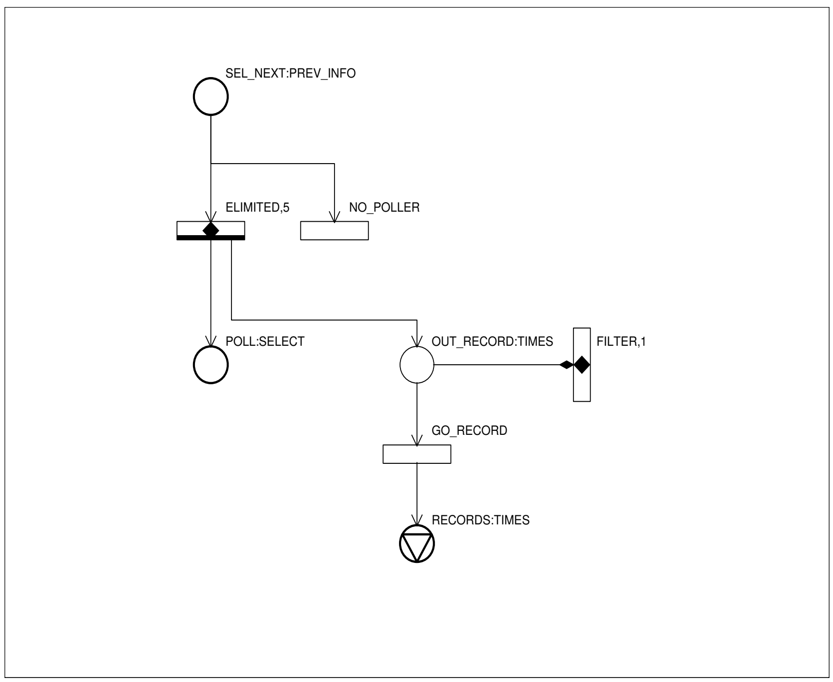


Figure A.6: Polling discipline.