Name: Chong Foh Wooi

Degree: Master of Engineering

Dept: Mechanical Engineering

Thesis Title: Bluetooth Wireless Communication for *MEMSwear*

ABSTRACT

MEMSwear is a wearable system to monitor vital signs for elderly at home based on Smart-shirt. Health vital signs such as ECG, blood pressure and fall sensor will be transmitted wirelessly to a gateway for data storage and analysis. Bluetooth is found to be the most promising technology for this application. However, due to the limited coverage range of the low power Bluetooth device and high mobility of the wearer, roaming (not supported by Bluetooth) has to be devised. The roaming scheme suggested in this thesis is then tested based on the *blue-ns* simulation software which is developed in this project as well.

Keywords: Wearable system, Bluetooth, MEMSwear, Software simulation, Roaming

Bluetooth Wireless Communication For MEMSwear

CHONG FOH WOOI

NATIONAL UNIVERSITY OF SINGAPORE

2003

Bluetooth Wireless Communication

For MEMSwear

CHONG FOH WOOI

(B.Eng. (Hons), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2003

Acknowledgements

I would like to thank the following people for their support and help throughout the project.

- Supervisor, A/Prof Francis Tay Eng Hock, for his continuous guidance and support.
- Programme manager, Mr. Tan Kwong Luck, for his work in coordinating project meeting.
- Mr. Tan Yee Yuan, for his help in electrical design and fabrication of prototypes.
- Project members, Mr. Nyan Myo Naing, Mr. Wong Yeong Shiong and Mr. Phang Jyh Siong for their help and support in making this project a success.
- Lab technician, Mr. Mohamed Ali, for his support in providing the necessary tools and equipments.

Last but not least, I would like to thank all my friends for their helps in many ways. Thank you from the bottom of my heart.

Table of Contents

Summary

MEMSwear is a Smart-shirt based vital signs monitoring wearable system for the elderly at home. By incorporating Bluetooth technology into the *MEMSwear* system, wireless and distant communication can be achieved in order to improve the mobility of the wearer. Some features offered by the Bluetooth are very attractive to the *MEMSwear* (such as its small size, low power and low cost). However, there are specific requirements of *MEMSwear* that the current Bluetooth technology cannot fully fulfil.

One of these is the limited coverage range of the Bluetooth device. To overcome the limitations, roaming for Bluetooth with a few access points connected to fixed network infrastructure is proposed in this report to solve the problem. Various strategies will also be explored to improve the performance of the network. To evaluate the performance of the roaming mechanism, Bluetooth simulator called *blue-ns* has been developed.

The *blue-ns* simulator is developed not only for roaming performance evaluation, it is also used to provide better understanding of the Bluetooth communication needed for *MEMSwear*. It is used to simulate flow of data between the Bluetooth devices and it is particularly useful in situation where multiple Bluetooth devices are within the vicinity.

List of Figures

List of Tables

CHAPTER 1 Introduction

According to research, more than 19% of the population of Singapore will be above 65 by 2030 [1]. With the incidence of chronic diseases, falls and dementia, expenditure in geriatric healthcare will make up a larger part of the total national expenditure. In view of the emerging emphasis on better and more affordable healthcare for the elderly, a wearable monitoring system for the elderly is required.

1.1 MEMSwear

MEMSwear is a Smart-shirt based vital signs monitoring wearable system for the elderly at home. Various human vital sign sensors such as ECG and blood pressure sensor as well as MEMS-based fall detection system will be incorporated into the Smart-Shirt. To enhance the mobility and convenience of the wearer, the continuous monitoring system needs to be transmitted wirelessly from the various sensors for data storage. More importantly, the continuous monitoring enables the trigger of emergency signal in the event of critical conditions of the wearer. By incorporating Bluetooth technology into the *MEMSwear* system, wireless and distant communication can be achieved. Bluetooth is a promising short-range radio network technology. The reason for using Bluetooth networking is to allow small and high-speed networks to be established around a user at a very low cost.

1.2 Bluetooth

While there are some features offered by the Bluetooth that are very attractive to the *MEMSwear* (such as its small size, low power and low cost), there are specific requirements of *MEMSwear* that the current Bluetooth technology cannot fully provide. One of these is the limited coverage range of the Bluetooth device. To overcome the limitations, cellular-like network roaming with a few access points connected to fixed network infrastructure is suggested in this paper. Various strategies will also be explored to improve the performance of the network.

1.2.1 Roaming for Bluetooth

Bluetooth was originally designed to replace propriety cables that connect devices. However, the usage of Bluetooth is found to be more than just a cable replacement technology by incorporating different profiles. One of the useful features is its use as an interface for mobile devices to access network system. Two profiles in the Bluetooth specification are defined to support network access - LAN Access Profile (LAP) and Personal Area Network (PAN).

Accessing the network requires access points which behave like base stations in the cellular network. Bluetooth devices have limited range 15 m to 100 m. To be able to cover a large area, many access points need to be deployed in a similar fashion as cellular network. Most Bluetooth devices are personal device and therefore move around with their owner. Thus, there is a definite need to support the mobility of the user to move freely from one access point to another without the need for user intervention and in a timely fashion. It must also be able to migrate network connections transparently. In this project, the roaming mechanism of Bluetooth is examined and optimised.

1.2.2 Bluetooth Simulation

When evaluating the feasibility of using Bluetooth for *MEMSwear,* there is a need to test the reliability and performance. This can be done using real hardware, but often tests are performed using a simulator. This provides a more cost efficient solution that does not require any special hardware. However, there is lack of support provided by the current simulation package for Bluetooth (i.e. *BlueHoc*). Therefore, there is a need to develop a Bluetooth simulator that provides a flexible and dynamic platform for Bluetooth simulation.

Based on current implementation of *BlueHoc*, a new simulator called *blue-ns* is developed in this project. Before actual hardware implementation of the suggested roaming mechanism, the mechanism is simulated under the *blue-ns* simulator environment.

1.3 Objectives

The objectives of this project are: -

- Develop a Bluetooth simulation platform.
- Design a mechanism of roaming for Bluetooth.
- Evaluate the performance and reliability of the roaming mechanism.

1.4 Outline of this documents

This thesis is outlined in the following manners: -

Chapter 2 Literature Review – gives an overview of research and works done in the related field.

Chapter 3 Bluetooth Technology – gives a brief explanation of the Bluetooth technology.

Chapter 4 Bluetooth Simulation – gives a detailed implementation of the new Bluetooth simulation platform, blue-ns, developed in this project.

Chapter 5 Roaming for Bluetooth – discusses the implementation of roaming for Bluetooth and its performance.

Chapter 6 Conclusions – gives conclusions of the project and the results achieved.

CHAPTER 2 Literature Review

2.1 MEMSwear

There is a need for an effective and portable monitoring system to address the needs of an increasingly greying population and raising healthcare costs worldwide. Taking advantage of advancements in micro-machining technology and coupling with an innovative breakthrough in textile engineering, a complete vital signs monitoring system in the form of a wearable garment can be realised.

MEMS technology, which uses techniques originally intended for fabrication of IC (Integrated circuits), allows mechanical devices like accelerometers, pressure sensors to be miniaturised and integrated into the Georgia Tech Wearable Motherboard™. The Wearable Motherboard™ acts as a Personalised Mobile Information Processing (PMIP) platform [2][3]. This platform allows various vital sensors of *MEMSwear* to be attached for sensing the vital signs (see Figure 2.1).

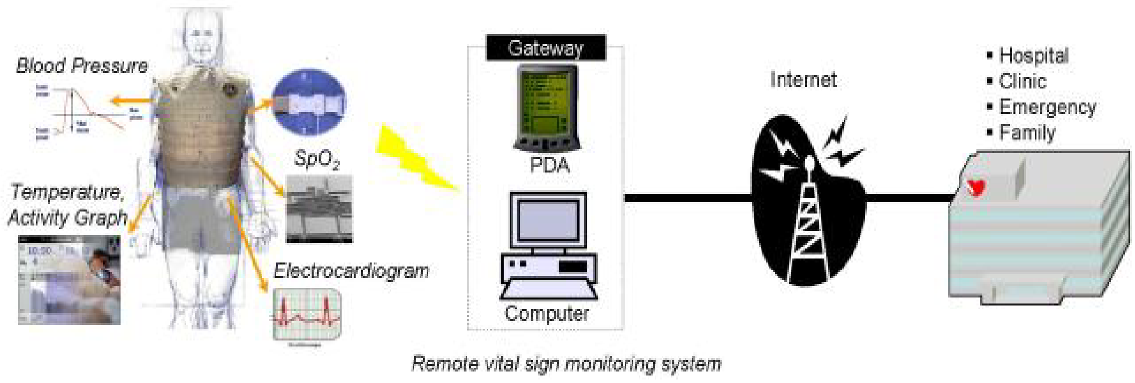


Figure 2.1 – MEMSwear technology platform overview

The Wearable Motherboard™, also called Smart Shirt, from Georgia Institute of Technology is a washable Meraklon (polypropylene fibre) shirt that has electrical and optical fibres weaved across the entire shirt for collection of biomedical information. As a result, it provides a very systematic way of monitoring the vital signs of humans in an unobtrusive manner. Furthermore, the flexible data bus integrated allows the structure to transmit information from the sensors effectively. Figure 2.2 shows an example of a Smart Shirt.

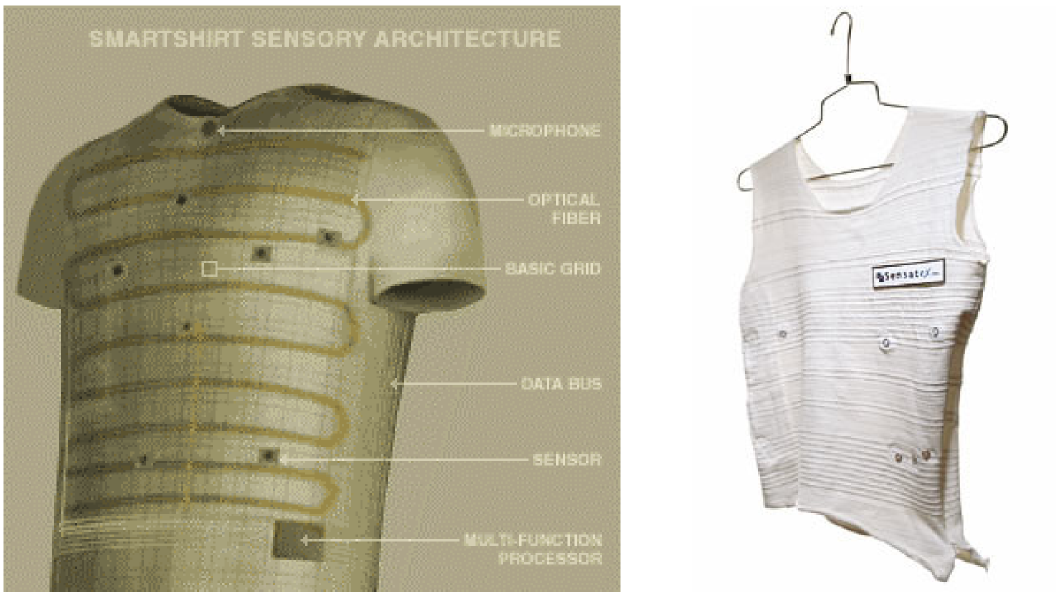


Figure 2.2 – Example of a Smart Shirt

Incorporating specially designed MEMS devices on various positions of the shirt, vital signs like heart rate, blood pressure and respiration rate can be measured. For example, micro pump and pressure sensors can replace the bulky counterparts in conventional blood pressure monitoring system to optimise space. In addition, the person wearing the Smart Shirt can be made portable by using Bluetooth for wireless communication to a computer for data recording and monitoring. The vital signs data can then be sent to the doctor or specialist anywhere in the world via the Internet.

With the advantage of portability and completeness, this shirt can be used to speed up medical check and help in monitoring firemen and soldiers during duty. In the future, the shirt may even make virtual medical exams possible. A patient could wear one of these shirts, and a physician would be able to monitor his or her heart rate, lung sounds, and other life signs from a remote location via the Internet.

2.2 Bluetooth

Bluetooth is used as the wireless communication means for *MEMSwear* mainly because of its attractive features: -

- **Small size** – the latest Bluetooth chip from Ericsson Microelectronics is shown in Figure 2.3 that has a size as small as 15 x 10 mm.
- **Low power** – the minimum power consumption of a Bluetooth chip (Class 3) is only around 1mW.
- **Network capability** – Bluetooth support network connection via the LAN Access Profile and Personal Area Network (PAN) Profile.
- **Moderate data rate** – the maximum data rate for a connection is 723kbps which is sufficient for *MEMSwear* requirement.

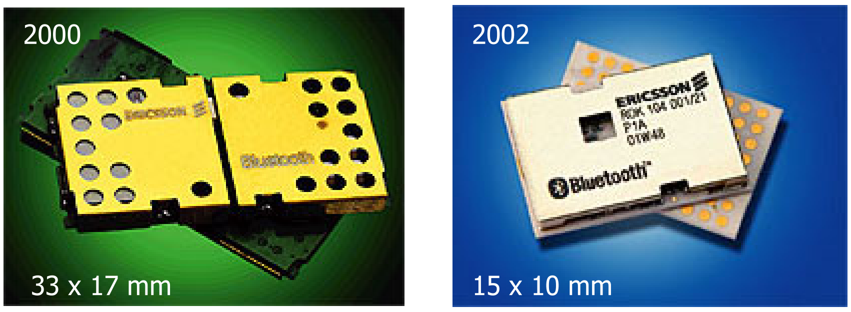


Figure 2.3 – Ericsson's Bluetooth chips

The Bluetooth Special Interest Group (SIG) is a trade association comprised of leaders in the telecommunications, computing and network industries that is driving the development of a low-cost, short-range wireless specification (Bluetooth) for connecting mobile products. The Bluetooth SIG promoters include 3Com, Agere, Ericsson, IBM, Intel, Microsoft, Motorola, Nokia and Toshiba, and hundred of Associate and Adopter member companies. Over the years, the Bluetooth SIG has written extensive requirements to implement Bluetooth technology in the Bluetooth Specification [4]. The latest version of the Bluetooth Specification is version 1.1. The Bluetooth Specification includes wireless specification of both link layer and application layer definitions for product developers which support data, voice and content-centric applications. It also includes radio specification that operates in the unlicensed 2.4 GHz radio spectrum. The Bluetooth Specification contains the information necessary to ensure that diverse devices supporting the Bluetooth wireless technology to communicate with each other worldwide.

Haartsen from Ericsson Radio Systems has described the radio system behind the Bluetooth concept in his paper. It said, "Bluetooth technology eliminates the need for wires, cables and the corresponding connectors" and "paves the way for new and completely different devices and applications" [5]. Haartsen, in this paper, also explained the basic design and technology trade-offs of the Bluetooth radio system and the fundamental issues regarding ad-hoc radio systems.

In addition to protocols which guarantee that two units speak the same language, Bluetooth Profiles [6] are also defined. Profiles are associated with applications. The profiles specify which protocol elements are mandatory in certain applications. This prevents devices with little memory and processing power to implement the entire Bluetooth stack when they only require a small fraction of it. Simple devices like a headset or mouse can thus be implemented with a strongly reduced protocol stack.

Profiles are dynamic in the sense that for new applications, new profiles can be added to the Bluetooth Specification.

There are two profiles that support network connections: LAN Access Profile and Personal Area Network Profile. In both cases, the assumptions is that a specific Bluetooth device called Access Point is available to enable Bluetooth client devices to be connected to a backend network infrastructure, such as Local Area Network (LAN) or the Internet.

As specified in the Bluetooth Specification, up to a maximum of seven simultaneous connections can be established and maintained by a Bluetooth device. This forms a piconet. Bluetooth network is ad-hoc, meaning that the device connects and disconnects as it enters and leaves the network. Due to the ad-hoc nature of Bluetooth, the participants of the piconet are highly dynamic and unpredictable. Much research has been done on the performance of piconet in the area of its data throughput [8][13][15], interference model [9][11] and link scheduling [7][10][12][14].

Frequency hopping is used by Bluetooth to permit multiple concurrent Bluetooth communication within the radio range of each other, without adverse effects due to interference. This facilitates high densities of communicating devices, making it possible for multiple piconets to co-exist and independently communicate in close proximity without significant performance degradation. This raises the possibility of inter-networking multiple piconets called scatternet. Bluetooth scatternet has been an active area of research, particularly in the area of link scheduling [18][21][22], scatternet formation [18][19][20] and routing [16][17].

One limitation of Bluetooth is its limited range. Although larger transmission range (around 100 meter) is possible for Power Class 1 devices, the power consumption is unacceptably high at 100mW (refer to Section 3.8.1). In dealing with this limitation, various methods have been devised. One way is to use the routing method (see Figure 2.4). In routing method, the data is routed through the devices in the network to reach its destination. There are many proposals for routing protocol for ad-hoc wireless network [23]-[26]. Bhagwat *et al* proposed a Routing Vector Method (RVM) that encodes source route paths in Bluetooth scatternet [16].

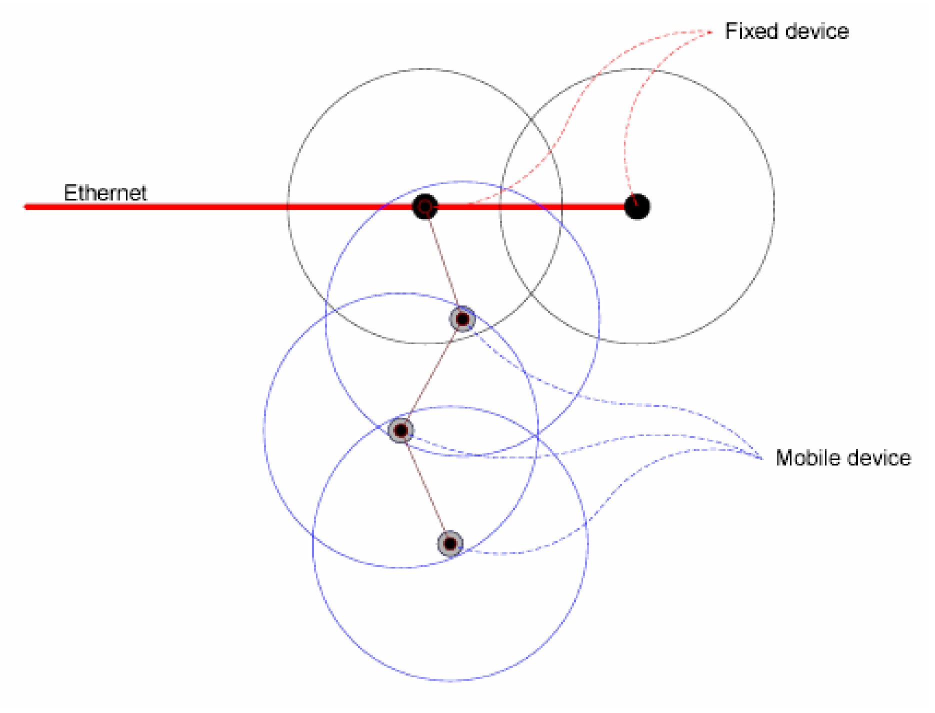


Figure 2.4 – Routing for Bluetooth

Another method is to make use of the roaming method. Roaming involves fixed Access Points attached to a backend network infrastructure. To gain network access, mobile Bluetooth devices connect to the Access Points. One problem of using Access Point occurs when the mobile device moves from one Access Point to another. The connection would be disconnected and the user has to start the inquiry and connection process again. Thus many researchers suggested using handoff/handover algorithm to deal with this problem. The main concern of the handover algorithm is the long handover delay and its effect on data throughput.

Lee *et al* proposed to integrate Bluetooth with the already existed wireless network in the building such as office [27][28]. To include coverage of the whole building, the authors suggested that the traffic to be ricocheted through several Bluetooth nodes to/from the gateway. In a way, this method is similar to routing the data traffic through nearby Bluetooth devices to reach its final destination. This method assumes that there will always be some Bluetooth nodes around current node. However, this is not always the case.

Another handover protocol called Bluetooth Public Access (*BluePAC*) is proposed by M. Frank *et al* [29][30]. Public Access Points are installed in public places like airport, train stations, hotel rooms, departmental stores and etc. *BluePAC* aims to provide IP services over Bluetooth, and tries to address issues such as IP addressing, routing issues and handoff support. It has the network architecture as shown in Figure 2.5.

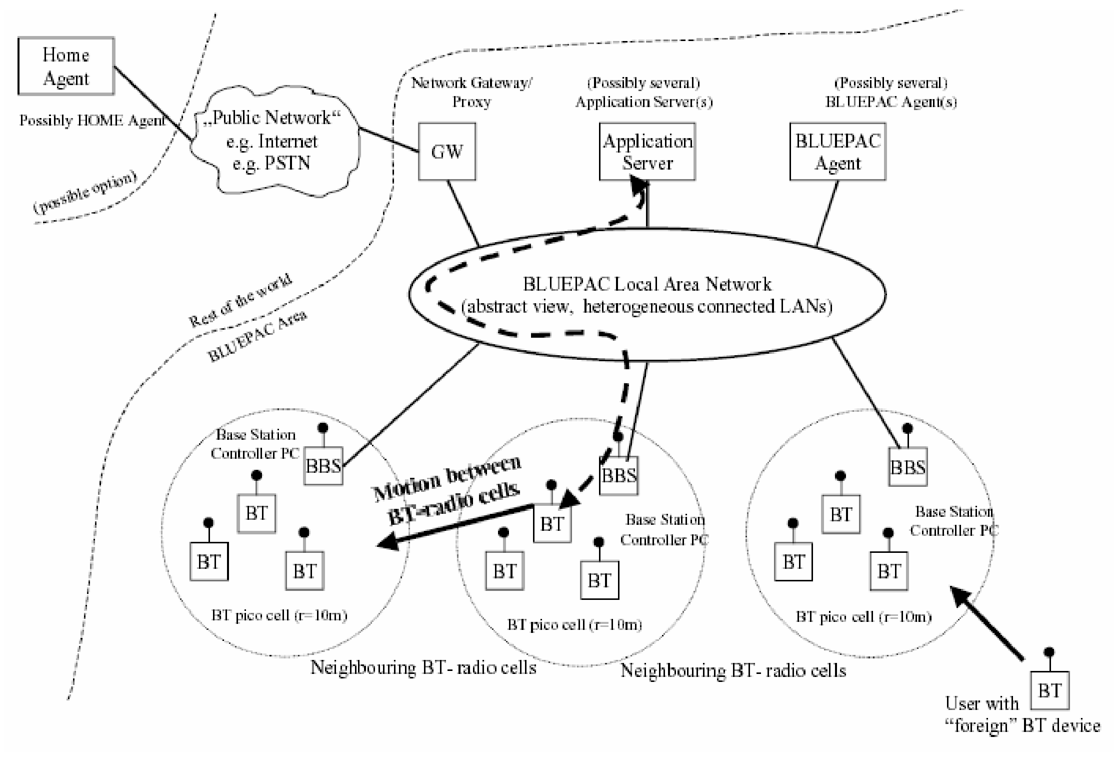

Home Agent
Possibly HOME Agent
(possible option)
„Public Network"
e.g. Internet
e.g. PSTN
Rest of the world
BLUEPAC Area
Network Gateway/ Proxy
GW
(Possibly several) Application Server(s)
Application Server
(Possibly several) BLUEPAC Agent(s)
BLUEPAC Agent
BLUEPAC Local Area Network
(abstract view, heterogeneous connected LANs)
Base Station Controller PC
BBS
BT
Motion between BT-radio cells
BT pico cell (r=10m)
Neighbouring BT- radio cells
User with "foreign" BT device

Figure 2.5 – BluePAC reference network architecture

The handoff support of *BluePAC* is made possible with the use of router as shown in Figure 2.6. A routing mechanism with routing regardless of the location within the inter-network is required to accommodate devices in motion. When leaving the range of the old piconet, the Bluetooth device connects to the next *BluePAC* Base Station available and continues sending this information through the new Base Station. These packets going through the new Base Station towards the servers update the routing caches to the new route leading to the device.

The main disadvantage of *BluePAC* is the need of a router to enable handoff mechanism. As the context of *MEMSwear* is for home use, the solution needs to be simple and it is not suitable to install dedicated hardware such as router at home.

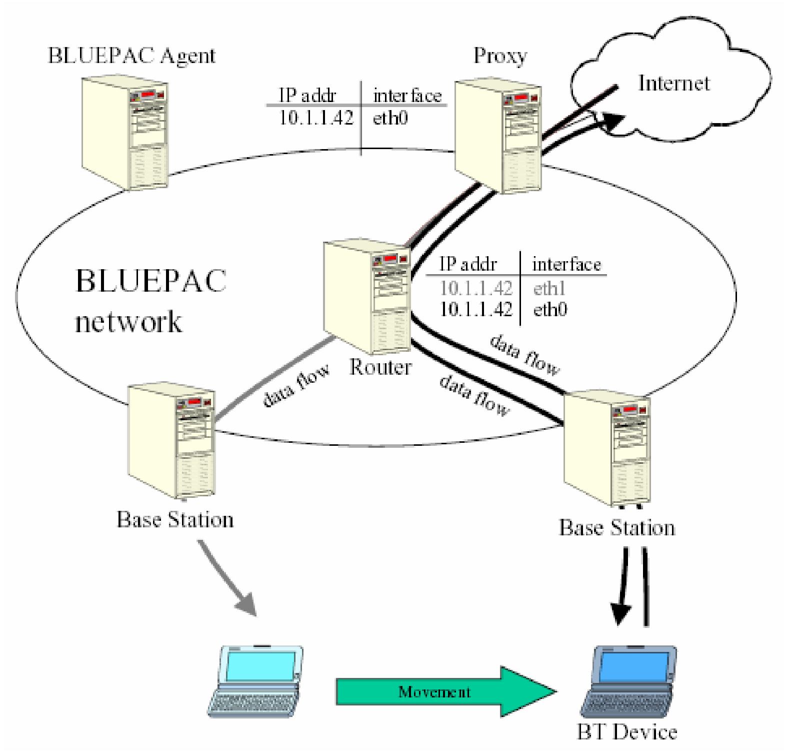


Figure 2.6 – Handoff support for mobile device in BluePAC area

Kansal *et al* proposed a new handoff protocol called Neighbourhood Handoff Protocol (NHP) to achieve fast handoff. This method is said to be able to prevent packet loss during the handoff and attempts to achieve efficient utilization of bandwidth at the Bluetooth layer [31]. The advantage of this method is that the protocol does not require any changes to the mobile nodes, as it has to be implemented at the Access Points alone. NHP eliminates the time-consuming inquiry procedure from handoff and provides for very rapid detection of connection loss.

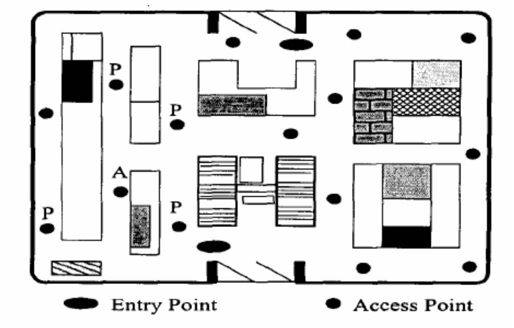


Figure 2.7 – Sample arrangement of Access Points and Entry Points for NHP

NHP requires the use of Entry Points and Access Points in the network as shown in Figure 2.7. Mobile nodes are only allowed to enter the network at the Entry Points and not at any arbitrary location inside the network. The Entry Points constantly carries out inquiry to obtain information (its address and its clock information of) any new devices entering the network. The Access Points know all the information required for connection establishment and thus eliminate the inquiry process. The strict requirement that the mobile node has to enter the network from the Entry Points only makes it unsuitable for use in *MEMSwear* context.

Literature review shows that there is currently no suitable roaming mechanism for use in the context of *MEMSwear* and suggests that a simple yet reliable roaming mechanism is required specifically for the application of *MEMSwear*.

CHAPTER 3 Bluetooth technology in a nutshell

The Bluetooth wireless technology allows for the replacement of the numerous proprietary cables that connect one device to another with a universal short-range radio link. The Bluetooth technology "enables the design of low power, small-sized, low-cost radios that can be embedded in existing (portable) devices" [5]. Beyond un-tethering devices by replacing cables, Bluetooth provides a universal bridge to existing data networks, a peripheral interface and a mechanism to form small private *ad-hoc* groupings of connected devices away from fixed network infrastructures.

The Bluetooth Special Interest Group (SIG) has written extensive requirements to implement the Bluetooth technology called The Bluetooth Specification [4] (latest version is 1.1). The specification highlights the various aspects and implementations of the technology, some of which will be discussed in this section. For more detailed information, the Bluetooth Specification shall be referred to.

3.1 Protocol Stack

The Bluetooth Specification defines the Bluetooth protocol stack to allow devices from different manufacturers to work with one another. This ensures compatibility and integrity of various components. The Bluetooth SIG does not only define the radio system (hardware), but also define the software stack to enable applications to find other Bluetooth devices in the area, discover device services, just to name a few. The Bluetooth stack is defined as a series of layers and some features apply across several layers.

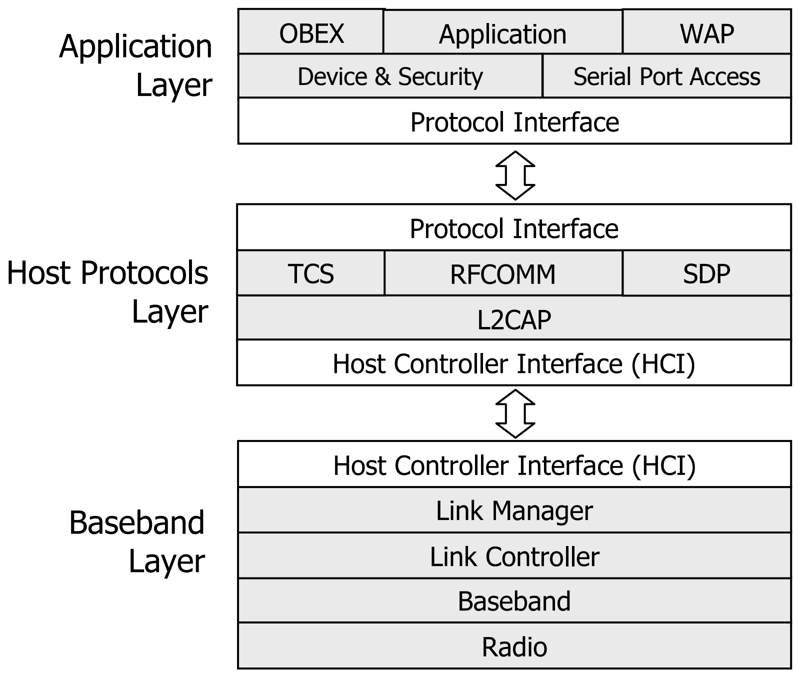


Figure 3.1 – The Bluetooth protocol stack

It is inefficient to include every aspects and features of Bluetooth into a single device. Hence, it is suggested the protocol stacks to be divided into several layers where one layer can talk to another layer through appropriate interface.

Figure 3.1 shows one of the many possible ways in which the Bluetooth protocol stack is partitioned into three layers: Baseband layer, Host Protocol layer and Application layer. This kind of system partitioning allows application developers more flexibility as products from different manufacturers may be integrated together seamlessly.

The lowest Baseband Layer is usually implemented in as a single chip, while the middle Host Protocol Layer is employed as another single chip that can access the Baseband Layer through the Host Controller Interface. There are really no strict requirements as to where a specific component should be implemented. It is up to the developers to choose product that has the support for the Bluetooth stacks of interest, even though it may be from different manufacturers.

3.2 Bluetooth Radio Spectrum

Data can be transmitted over the air at certain frequency. Both the transmitter and receiver must be at the same frequency for a successful transmission. The Bluetooth radio is the lowest layer of the Bluetooth protocol that defines the frequency used to transmit the data. It defines the requirements of the Bluetooth transceiver device operating in the range of 2400 ~ 2483.5 MHz Industrial, Scientific and Medical (ISM) band, which is an unlicensed radio band that is globally available. Bluetooth is based on Frequency Hopping – Code Division Multiple Access (FH-CDMA). FH-CDMA offers properties most appropriate for Bluetooth ad-hoc radio system. The signal is spread over a large frequency range, but instantaneously only a small bandwidth is occupied, thus avoiding most of the potential interference in the already busy ISM band [5].

In the 2.45 GHz ISM band, a set of 79 hop frequency carriers have been defined at 1 MHz spacing. Frequency hopping is a feature where the radio transceiver hops to different frequency after each transmission and reception of data over the air. In this ISM band, the range of 2400 ~ 2483.5 MHz is divided into 79 RF channels at 1 MHz spacing apart. The RF channel used can be defined as: -

$$Frequency, f_k = 2402 + k \text{ MHz} \qquad \text{(where k = 0, 1, 2, … 78, 79)}$$

In most implementation, the radio layer is a hardware analogue circuitry that provides modulation and demodulation of the data for transmission and reception over the air.

Under the Bluetooth specification, there are 79 frequencies used for data transmission. Only one of the 79 frequencies is used at any time slot (which is 625 μsec). It will hop to different frequency on next time slot and the hop rate is 1600 hops/second.

3.3 The Baseband

The Baseband is a lowest software layer that sits on top of the analogue radio circuitry layer. To understand the Baseband layer, we need to distinguish the concepts of physical channel, physical link, device addressing, packet types and format, controller states and inquiry & connection establishment operations. It is undeniable that the Baseband layer is the most important layer among the rest of the layers.

3.3.1 Physical Channel

Data are transferred in packet at every time slot. Each time slot is 625 μsec in length. The time slots are numbered according to the Native Clock (CLKN) of the device, ranging from 0 to 2^{27}-1 at which it will cycle approximately after one day under continuous use. At each slot, transmission/reception occurs at certain frequency both the transmitter and receiver agreed upon.

A Time Division Duplex (TDD) scheme is used where master and slave transmit alternatively data packets as illustrated in Figure 3.2. At even numbered time slot, the master will transmit data packet to the slave while on odd numbered time slot, the slave will transmit data packet to the master. This is to prevent collision of data during transmission.

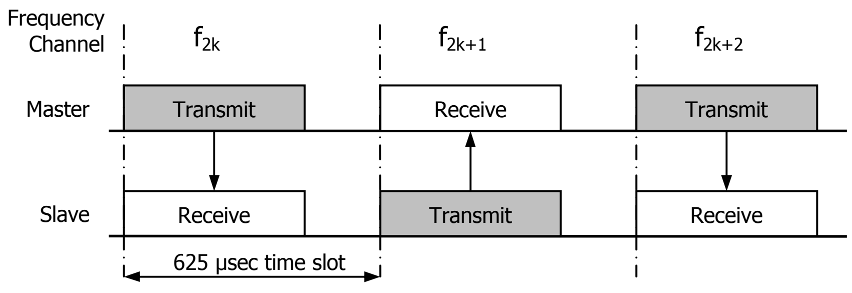


Figure 3.2 – Time Division Duplex (TDD) scheme for Bluetooth

Different frequencies are used for each time slots. In other words, the frequency changes from one time slot to another and this is called frequency hopping. A Bluetooth channel can be defined as a pseudo-random hopping sequence through the 79 channels as shown in Figure 3.3. Two or more Bluetooth devices using the same channel form a piconet. There is a master and one or more slave(s) in each piconet. The hopping sequence is unique for the piconet and is determined by the Bluetooth Device Address (BD_ADDR) of the master of the piconet while the phase is determined by the internal Native Clock (CLKN) of the master.

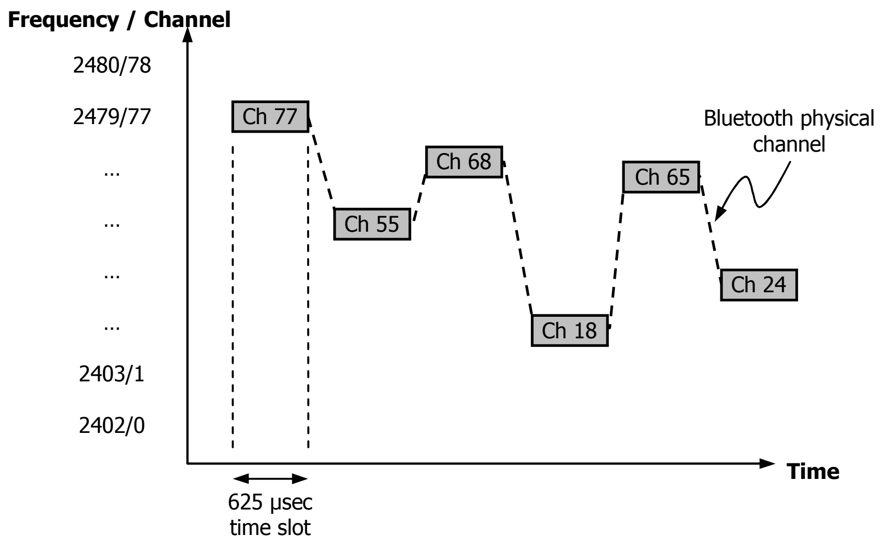


Figure 3.3 – Frequency hopping for Bluetooth

As the BD_ADDR of a device is unique, there will not be two devices using the same frequency channel at the same time. This reduces the interference between devices in situations where there are many Bluetooth devices at an enclosed environment and minimize the effect of fading. The mechanisms to determine the frequency channel are clearly defined in the Bluetooth Specification.

3.3.2 Physical Link

There are two types of links that can be established between master and slave(s): -

- Synchronous Connection-Oriented (SCO) link
- Asynchronous Connection-Less (ACL) link

The SCO link is a point-to-point link between a master and a single slave in the piconet and is usually used for 64 KB/s speech transmission. The ACL link is a point-to-multipoint link between the master and all slaves participating on the piconet. Only ACL link will be discussed throughout this report.

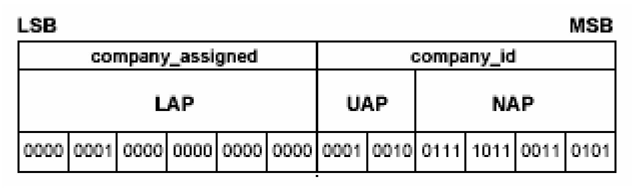
For ACL link, lost packet is retransmitted to ensure data integrity. A slave can return a packet only if it has been addressed in the previous master-to-slave slot and only if it successfully decodes the slave address in the packet header.

3.3.3 Device Addressing

There are four possible types of addresses that can be assigned to a Bluetooth device:-

- Bluetooth Device Address (BD_ADDR)
- Active Member Address (AM_ADDR)
- Parked Member Address (PM_ADDR)
- Access Request Address (AR_ADDR)

Each Bluetooth device is allocated a unique 48-bit BD_ADDR and it can be divided into three fields as shown in Figure 3.4.



LAP (24 bits): lower address
UAP (8 bits): upper address
NAP (16 bits): non-significant address

Figure 3.4 – Structure of BD_ADDR

The AM_ADDR is a 3-bit number assigned by the master for each slave upon joining the piconet and as long as the slave is an active member of the piconet. The AM_ADDR is attached to all messages between master and slave in the packet header. In the special case, the AM_ADDR of all-zeros means a broadcast message to all slaves in the piconet.

3.3.4 Packet Format

Information is exchanged through the use of packets. Each packet is broadcast on a different frequency hop. All packets have the general structure as shown in Figure 3.5. It consists of an access code, a header, and a payload.

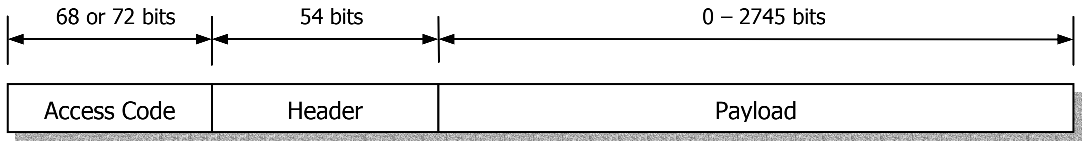


Figure 3.5 – Structure of a Bluetooth packet

3.3.4.1 Access Code

Different access codes are used in different operating modes: -

- **Channel Access Code (CAC)** – used to identify a piconet and is included in all packets exchanged on the piconet channel. It contains the address of the master.
- **Device Access Code (DAC)** – used for special signalling procedures, e.g. during page, page scan and page response sub-states. It contains the address of the device being paged.

- **Inquiry Access Code (IAC)** – used for inquiry operation. Two types of IAC are defined: - General Inquiry Access Code (GIAC) and Dedicated Inquiry Access Code (DIAC).

Access Code is the first parameter that a device will examine to determine if this packet is intended for itself. It simply discards packets that are not of its interest.

3.3.4.2 Packet Header

The header contains link control information and consists of 6 fields as shown in Figure 3.6. The packet header contains information pertinent to the connection within the piconet.

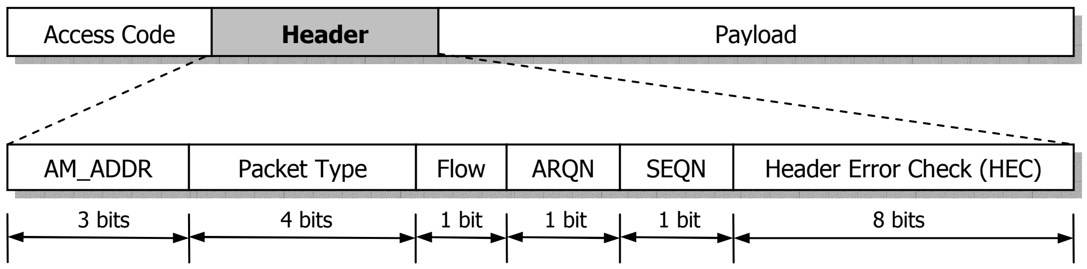


Figure 3.6 – Structure of a Bluetooth packet header

- **AM_ADDR** – used in all communications to the slave and for the master to differentiate between responses from different slaves.
- **Packet Type –** used to identify which type of traffic is carried by this packet.
- **Flow** – used to inform the sender that the device is unable to receive any more data due to its receive buffer being full.
- **ARQN** – used to inform sender that previous packet is received successfully.
- **SEQN** – used to identify whether this packet is a duplication of previous packet by the receiver.
- **Header Error Check** – used to perform error check using the Cycle Redundancy Checksum (CRC) function.

One of the important fields in the packet header is the AM_ADDR field. It provides the second level filter of packets after the access code filtering (refer to Section

3.3.4.1). Packet from the master to one of the slaves contains the AM_ADDR of the slave. If the recipient's AM_ADDR does not match, this packet will be discarded. In packet from the slave to master, it contains the AM_ADDR of the slave's assigned by the master. When the master receives this packet, it will pass this packet up to the appropriate application layer corresponding to this slave.

3.3.4.3 Packet Payload

The packet payload can be further divided into three parts: - payload header, payload data, and CRC as shown in Figure 3.7.

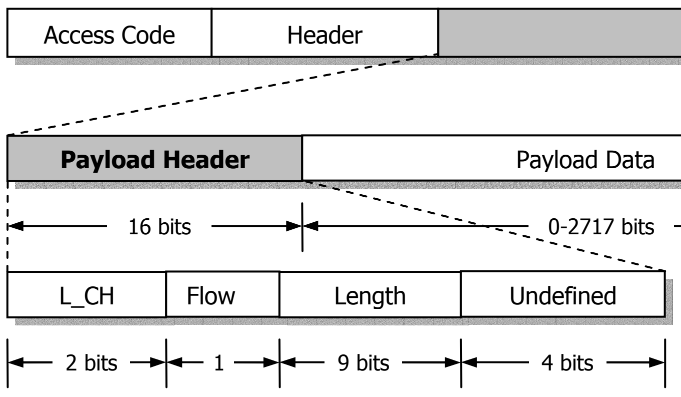


Figure 3.7 – Structure of a Bluetooth packet payload

- **L_CH** (Logical Channel) – used to identify the type of data carried in the payload data (see Table 3.1).

L_CH code	Descriptions
00	Undefined
01	Continuation fragment of an L2CAP message
10	Start of an L2CAP message or no fragmentation
11	LMP message

Table 3.1 – Meanings of the Logical Channel (L_CH) field

- **Flow** – used to controls data transfer at the L2CAP level.
- **Length** – used to identify the total length of the payload data.

The payload carries two types upper layer message: the Link Manager Protocol (LMP) and the Logical Link and Control Adaptation Protocol (L2CAP). The actual data itself is embedded inside the L2CAP data message (refer to Section 3.7).

3.3.4.4 CRC

The CRC field carries information required to verify the data integrity of the payload data received. It ensures that the data is not corrupted in the process of transmission.

3.3.5 Packet Types

There are four special packet types (ID, NULL, POLL, FHS) and five common data packets (DM1, DH1, DM3, DH3, DM5, DH5).

3.3.5.1 Identity (ID) Packet

The ID packet consists only of the access code and is used only in pre-connection operation. In the inquiry process, the Inquiry Access Code[1] (IAC) is carried in the access code field. In the paging process the Device Access Code[1] (DAC) is carried in the access code field.

3.3.5.2 NULL Packet

Another special packet is the NULL packet where it carries only Channel Access Code[1] (CAC) and packet header[2] only. It is normally used to return link information to the source regarding the success of the previous transmission (ARQN bit of packet header[2]) or the status of the receive buffer (FLOW bit of packet header[2]).

[1] Refer to Section 3.3.4.1 for information on different types of Access Code.
[2] Refer to Section 3.3.4.2 for information on packet header.

3.3.5.3 POLL Packet

The POLL packet has only the Channel Access Code (CAC) and the packet header as well. Upon reception of a POLL packet, the slave must respond with a packet to acknowledgment the reception. This packet is used by the master in a piconet to poll the slaves, which must then respond even if they do not have information to send.

3.3.5.4 FHS Packet

The Frequency Hop Synchronisation (FHS) packet provides all information required by the recipient to address the sender in terms of timing and thus the frequency hop channel synchronisation and correct device access code. It is used in inquiry scanning device to answer the inquirer during the inquiry procedure. It is also used in paging process where the master sends this data to synchronise the channel hopping selection.

3.3.5.5 DM and DH Packets

The DM and DH types of packets are data packets. The difference between the DM and DH packets is the inclusion of Forward Error Correction information inside the DH packet. FEC is used to protect data from corruption. Table 3.2 lists all the different packet types and the sizes of the payload data.

Packet Type	Payload Header	User Payload (bytes)	FEC
ID	No	No	No
NULL	No	No	No
POLL	No	No	No
FHS	No	No	2/3
DM1	Yes	0~17	2/3
DH1	Yes	0~27	No
DM3	Yes	0~121	2/3
DH3	Yes	0~183	No
DM5	Yes	0~224	2/3
DH5	Yes	0~339	No

Table 3.2 – Summary of Bluetooth packet types

Data of different sizes are carried by different data packets. The tradeoffs between data packet with and without FEC are the robustness and higher data throughput. FEC protected packets are more robust with lower throughput while those without FEC enjoy higher data throughput with less robustness.

3.3.6 Controller States

At any one time, the Bluetooth device is in one of the many states. Figure 3.8 shows the various sub-states that a device needs to go through in transition from the **STANDBY** mode to **CONNECTION** state.

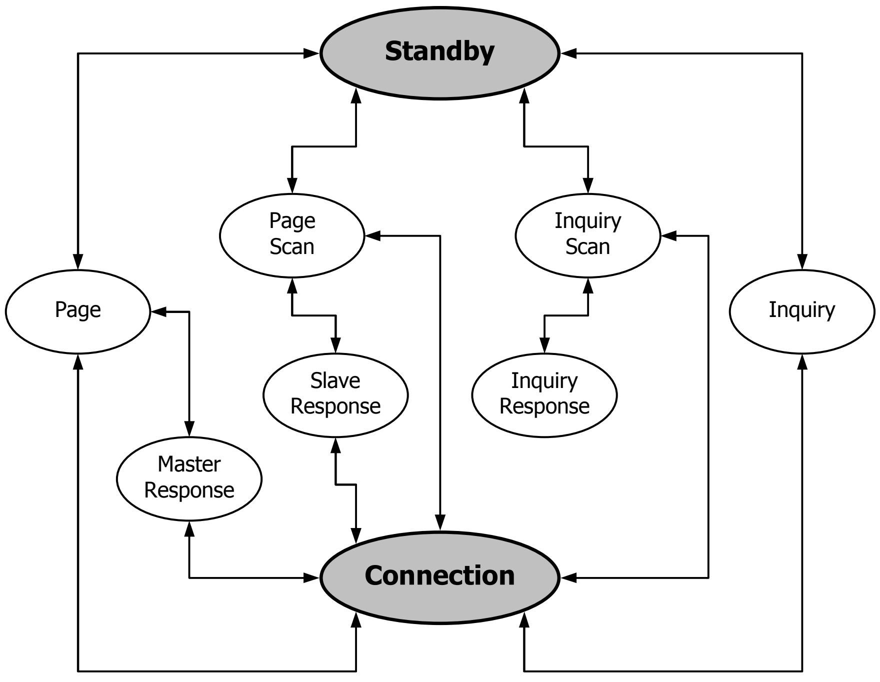


Figure 3.8 – Controller states of a Bluetooth device

This gives a clear and simple picture of how a single link is established from scratch. However, in practice, the state transitions can be much more complex.

3.3.6.1 Standby Mode

In **STANDBY** mode, the device is inactive, no data is being transferred and the radio is not switched on. Thus, the device is not able to receive any packet in this state. The **STANDBY** is also used to enable low power operation.

3.3.6.2 Inquiry Mode

A device enters into **INQUIRY** mode when it wants to discover other Bluetooth devices within the range. During the inquiring process, the inquirer sends out ID packets. Upon receiving this ID packet, the inquired device sends out FHS packets to the inquiring device (refer Section 3.3.7.2 to for inquiry process).

3.3.6.3 Inquiry Scan Mode

To allow other device to discover itself, the device needs to enter into **INQUIRY SCAN** mode. To conserve power, it makes itself available to inquiring device for a short period of time over a specified interval (see Figure 3.9). When it receives a valid inquiring message, it enters the **INQUIRY RESPONSE** mode where it responds with the FHS information (refer to Section 3.3.7.2).

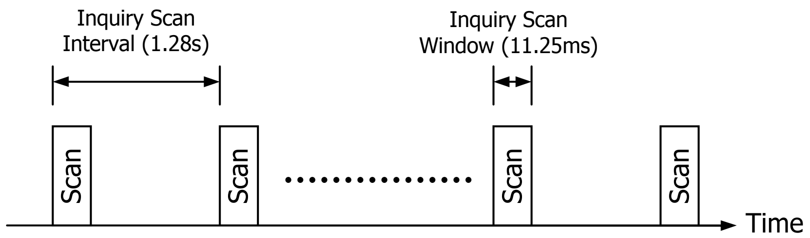


Figure 3.9 – Timing of inquiry scan

3.3.6.4 Page Mode

To establish a connection, the device that is to become master is to carry out the paging procedure and enter into **PAGE** mode. In this mode, the master constantly sends out paging ID packets directed to a specific slave device. When the slave responded to this paging message, the master enters the **MASTER RESPONSE** state and sends the slave a FHS packet (refer to 3.3.7.3 for paging procedure).

3.3.6.5 Page Scan Mode

Only if the device wishes to be connected to as a slave by a master enters the **PAGE SCAN** mode. In this mode, the slave device is listening for the paging message sent out by the master. It works the same ways as with the **INQUIRY SCAN** mode with default page scan window of 1.28s and page scan window of 11.25ms. Upon the receipt of the paging message, it goes into the **SLAVE RESPONSE** state to await further packets from the master (refer to 3.3.7.3 for paging procedure).

3.3.6.6 Connection Mode

After successful paging, both the master and slave are in the **CONNECTION** mode. The slave is synchronised to the master clock. During this state, various data exchanges are possible. The master must keep transmitting periodically to keep track of all of its slaves. Even if there is no data to send, the master could send the POLL packet and the slave responded with the NULL packet.

3.3.7 Connection Establishment

Before data can be transmitted across the air, the devices need to perform certain steps. This includes: -

- Parameters initialisation
- Inquiry/device discovery
- Paging/link establishment

3.3.7.1 Initialisation

To ensure proper operation of the device, the initial settings of the device must be changed to the desired values. Some of the settings are shown in Table 3.3. Some of these parameters can be modified using various commands.

Settings	**Possible value**	**Default value**
Inquiry Scan Enable	True / false	False
Page Scan Enable	True / false	False
Page Timeout	0.625ms ~ 40.9s	8192 time slots / 5.12s
Page Response Timeout	Fixed	8 time slots / 5ms
Inquiry Response Timeout	Fixed	128 time slots / 80ms
New Connection Timeout	Fixed	32 time slots / 20ms
Supervision Timeout	0.625ms ~ 40.9s	32000 time slots / 20s

Table 3.3 – Initial setting for Bluetooth device

3.3.7.2 Inquiring and device discovery

The discovery procedure is the mechanism to manage devices wishing to discover and to be discovered respectively. To identify other devices within the range, the Bluetooth has to start the inquiring/discovery process. The inquiry ID packets[3] are broadcasted to all devices. Only devices that have set inquiry scan enabled will be able to receive this packet. The recipients of this ID packet may respond by sending back a respond message. The response message consists of necessary information needed to make a connection between the two devices. Note that there may be more than two devices within the area that responded to the inquiry. In this case, the inquirer device may receive more than two inquiry results. Therefore, the responder must wait for a random back-off time before sending back the response. The inquiring process is illustrated in Figure 3.10.

[3] An inquiry ID packet is an ID packet with specific IAC in the access code field of the packet header.

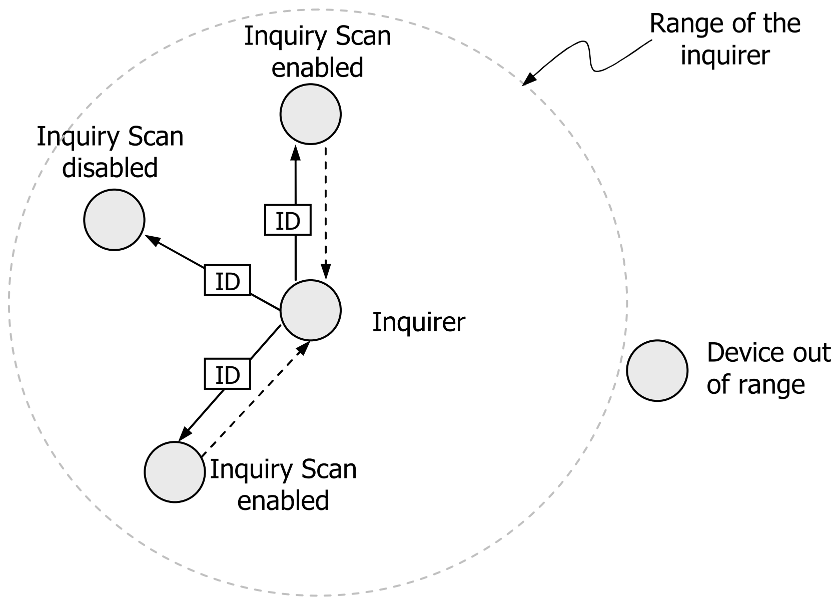


Figure 3.10 – The inquiring process

3.3.7.3 Paging and link establishment

With the information obtained from the inquiry process, the device may now page for device that it intended to make a connection to. Paging process starts when the master sends out paging ID packets[4]. When the paged device received this packet, it replies by sending a respond packet back to the paging device. From this point onwards, a series of messages are exchanged between these two devices to ensure that connection can be made. After the messages exchange and connection has been completed, the upper layer may start sending data through the Baseband to the other device.

3.4 Link Controller (LC)

The Link Controller is responsible for the implementation of ARQ mechanism in order to provide reliable data transmission of a single link. It makes sure that a packet

[4] A paging ID packet is an ID packet that contains the page scanner device's address in the access code field of the header.

sent is acknowledged by the recipient. ARQN is a flag available the every packet header. The ARQN flag is asserted by a device to indicate that the previous reception was successful. If the sender received a positive ARQN in the packet, it can assume a successful reception and will send the next packet. If, however, the ARQN was lost due to failure of the returned header, then the original sender of the data will assume a Negative-Acknowledge (NAK) condition and re-transmit the first packet.

It also ensures that no two packets received are the same. Each time a new packet is sent, another flag in the packet header, SEQN flag is toggled. However, in this case, if the packet is resent the SEQN flag will stay the same and the recipient knew that the packets are identical packets because the SEQN flag has not changed. Thus, it will ignore the second or all subsequent packets with the same SEQN until the SEQN flag changes. Even if the link is very bad, an ACK will eventually get through and the sequence will move on.

3.5 Link Manager (LM)

The Baseband level connection merely enables data to be transmitted across the air between two devices. To further customise and configure the link, various negotiations need to be carried out. This is usually done at the Link Manager level. The Link Manager carries out link setup and shutdown, authentication, link configuration and other protocols. The Link Manager uses the services provided by the underlying Link Controller and the Baseband. In summary, the Link Manager will translate control command from the upper layer into operations at the Baseband level, managing the following operations: -

- Attaching slaves into a piconet and allocating AM_ADDR to the slaves.
- Breaking connections to detach slave from a master.
- Configuring the link including controlling Master/Slave switches.
- Establishing ACL (data) and SCO (voice) link.
- Putting connections into low power modes: Hold, Sniff and Park.
- Controlling test modes.

It communicates with other devices via the Link Manager Protocol (LMP). The LMP defines a set of messages, which are passed between Link Managers of the devices. The messages are exchanged in the form of Protocol Data Units (PDU) as shown in Figure 3.11.

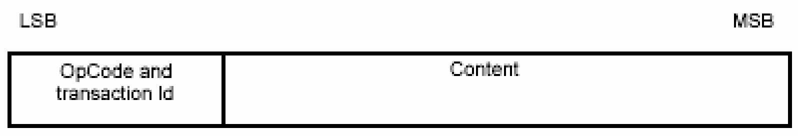


Figure 3.11 – Protocol Data Units (PDU) for Link Manager

The transaction ID is a single bit field, where 0 means a master initiated transaction while 1 means a slave initiated transaction. This is followed by a 7-bit Operation Code (OpCode) that identifies the type of LMP message being sent. The rest of the PDU carries the message parameters. The Baseband level connection between two devices must be established before the exchange of LMP messages.

The exchange of LMP messages between the Link Managers on both devices is carried out through the ACL link established at the Baseband level. It is carried in the payload data portion of a packet and is distinguished by the L_CH field in the payload header (refer to 3.3.4.3 for packet payload format).

3.6 Host Controller Interface (HCI)

The Host Controller has access to various layer on the protocol stack, including, the Baseband, the Link Controller, the Link Manager and the device hardware. The Host Controller Interface provides a command interface to the Host Controller by exchanging HCI commands and HCI events. The HCI commands are used to control the Bluetooth device and to monitor its status. The Host Controller translates these commands into appropriate actions and in return, the Bluetooth device replies with HCI events to report the status of the HCI commands issued. All HCI commands and events are issued in the form of packet. Beside HCI command and event packet, ACL data packet that carries upper layer protocol or data can also be transferred (see Figure 3.12). These packets are exchanged only between the Host and the Host Controller.

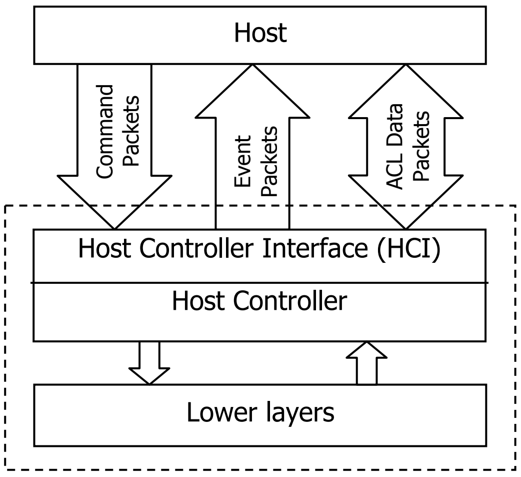


Figure 3.12 – HCI command, event and data packet

There are 5 categories of HCI commands defined in the Bluetooth Specification: -

- Link Controller Commands
- Link Policy Commands
- Host Controller & Baseband Commands
- Informational Parameters Commands
- Status Parameters Commands

3.6.1 Link Control Commands

These commands allow the Host Controller to control connections to other Bluetooth devices. These commands also instruct how the Link Manager to control, establish and maintain the Bluetooth piconet and scatternet. The Link Manager is instructed to create and modify link layer connections with other Bluetooth devices, including performing inquiring and paging as well as other LMP commands.

3.6.2 Link Policy Commands

The Link Policy Commands provides methods for the Host Controller to affect how the Link Manager manages the piconet. These commands are usually used to modify various parameters or policies on the link/connection. They modify the Link Manager behaviour that can result in changes to the connection with other Bluetooth devices.

3.6.3 Host Controller & Baseband Commands

The Host Controller & Baseband Commands provide access and control to various capabilities of the Bluetooth hardware. These parameters provide control of Bluetooth devices and of the capabilities of the Host Controller, the Link Manager, the Link Controller and the Baseband. The host device can use these commands to modify the behaviour of the local device.

3.6.4 Informational Parameters Commands

Some settings are fixed by the manufacturer of the Bluetooth hardware. These settings are called Informational Parameters. They can be retrieved using the Informational Parameter commands. These parameters are information about the Bluetooth device, capabilities of the Host Controller, the Link Manager and the

Baseband. These commands will only return these parameters but will not be able to modify these parameters.

3.6.5 Status Parameters Commands

These commands provide information about the current state of the Host Controller, the Link Manager and the Baseband. The host device cannot modify any of these parameters other than to reset certain specific parameters.

3.6.6 HCI Packet Format

To command the HCI, the host must send packet compliant to the HCI packet format in order for the Host Controller to understand. Note that none of the HCI packets are sent over the air. It is used as a common interface of message exchange between the host and the host controller.

3.6.6.1 HCI Command Packet

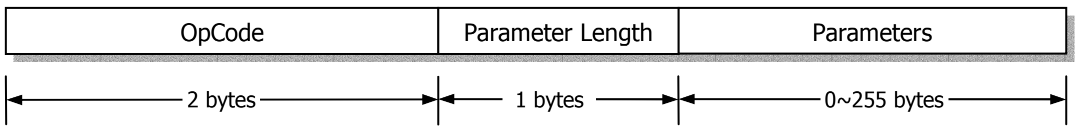


Figure 3.13 – Structure of HCI command packet

- **OpCode** – used to identify the type of the command of this packet.
- **Parameter Length** – total length of the parameters contained in the packet measured in bytes.
- **Parameters** – each command has a specific number of parameters associated with it. These parameters and its size are defined for each command.

3.6.6.2 HCI Event Packet

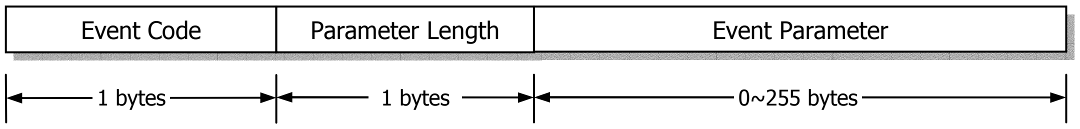


Figure 3.14 – Structure of HCI event packet

- **Event Code** – used to identify the different types of events.
- **Parameter Length** – total length of all of the parameters contained in this packet, measured in bytes.
- **Event Parameters** – each event has a specific number of parameters associated with it. These parameters and its size are defined for each event.

3.6.6.3 HCI Data Packet

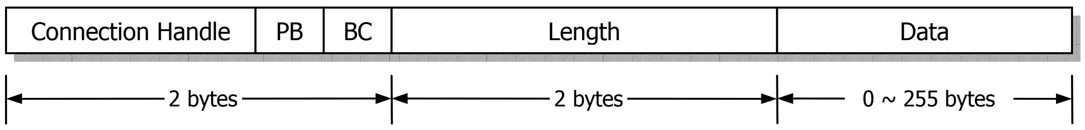


Figure 3.15 – Structure of HCI data packet

- **Connection Handle** – used to identify to which remote Bluetooth device this packet is intended for. Every device connected is assigned a connection handle.
- **Packet Boundary (PB)** – used to identify whether this packet is a first packet (0x01) or a continuation of fragment of packet of higher layer (0x10).
- **Broadcast (BC)** – used to identify whether this packet is a broadcast (0x01) or a point-to-point packet (0x00).
- **Length** – total length of the data measure in bytes.
- **Data** – the data of the packet.

Upon receiving the HCI data packet, the Host Controller extracts and interprets the content of the packets, particularly the connection handle. It then uses this information to compose the Bluetooth packet according to the format described in

Section 3.3.4. The data portion of the HCI data packet makes up the whole portion of the data payload of the Bluetooth packet.

3.7 Logical Link Control and Adaptation Protocol (L2CAP)

The L2CAP layer is located in the middle Host Protocol layer as seen in Figure 3.1. It takes request and data from higher layers of the Bluetooth stack or from application and sends it over the lower layers of the stack. It main functions are: -

- Multiplexing between different higher layer protocols, allowing them to share a single lower layer link.
- Segmentation and reassembly (SAR) to allow the transfer of larger packets than the lower layer can support.
- Quality of service management for higher layer protocols.

L2CAP sits on top of the Host Controller. Hence, it is required to communicate with lower layers using the Host Controller Interface. Because L2CAP entity on one Bluetooth device needs to communicate with the L2CAP entity on another Bluetooth device, the L2CAP packet is embedded in the HCI data packet (see Section 3.6.6.3). The Host Controller then translates the HCI data packet into appropriate Bluetooth packet and transfers it over the air using the already established ACL link at the Baseband level. The format of the L2CAP packet is shown in Figure 3.16.

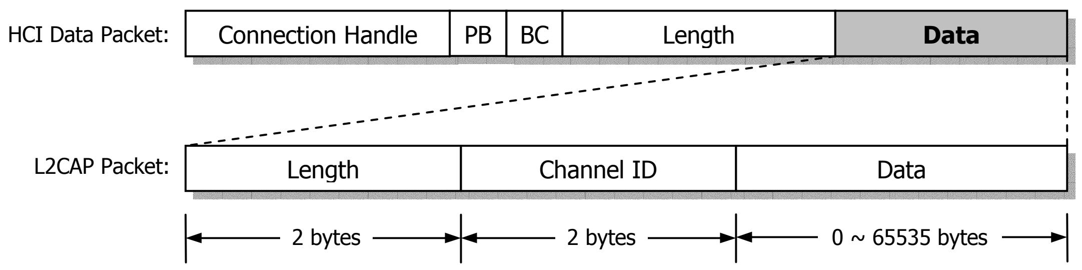


Figure 3.16 – Structure of L2CAP packet

- **Length** – total length of the L2CAP packet measured in bytes.
- **Channel ID (CID)** – local names representing a logical cannel end-point on the device.
- **Data** – the actual data of the packet.

Protocol Multiplexing can be achieved using channels in the L2CAP. Every service from the upper layer is assigned a Channel ID. This is to identify the intended recipient of the packets. Protocol multiplexing allows several services from upper layer to use a single entity of L2CAP. The Channel ID 0x0001 is reserved for signalling commands that are passed between two L2CAP entities on remote devices. The signalling commands are used for requests and responses between the L2CAP entities.

L2CAP supports segmentation and reassembly (SAR) of large packets as well. The maximum size of a L2CAP packet is 65,535 bytes. However, the maximum size that a Bluetooth packet can carry is one 343 bytes (which is packet type DH5, see Section 3.3.5.5). Because the all L2CAP packets are sent as Bluetooth packet, the larger L2CAP must be segmented into chunks of smaller packets suitable for transmission. On the receiving side, the chunks of packets will then be reassembled again as a whole L2CAP packet. Once the L2CAP packet is assembled completely, the packet is sent up to appropriate upper layer.

3.8 Other aspects

3.8.1 Radio Power Classes

The Bluetooth Specification allows for 3 different types of radio powers as shown in Table 3.4: -

Power Class	Maximum Output Power	Range
1	100 mW (20 dBm)	100 m
2	2.5 mW (4 dBm)	30 m
3	1 mW (0dBm)	10 m

Table 3.4 – Bluetooth power classes

These power classes allow Bluetooth devices to connect at different ranges. Most personal devices at the market now implement the power Class 3 for its low power consumption, though power Class 3 devices have a maximum range of only about 10m. Obviously higher power radios have longer ranges. The maximum range for a Class 1 device is 100m. For this feature, a Class 1 device consumes around 100mW of power, which is 100 times more the Class 3 devices of only 1mW. There is also a minimum range of 10cm for Bluetooth connection. If radios are put together too close, the receiver saturates. Figure 3.17 shows a mixture of high and low power devices in different piconets occupying an area.

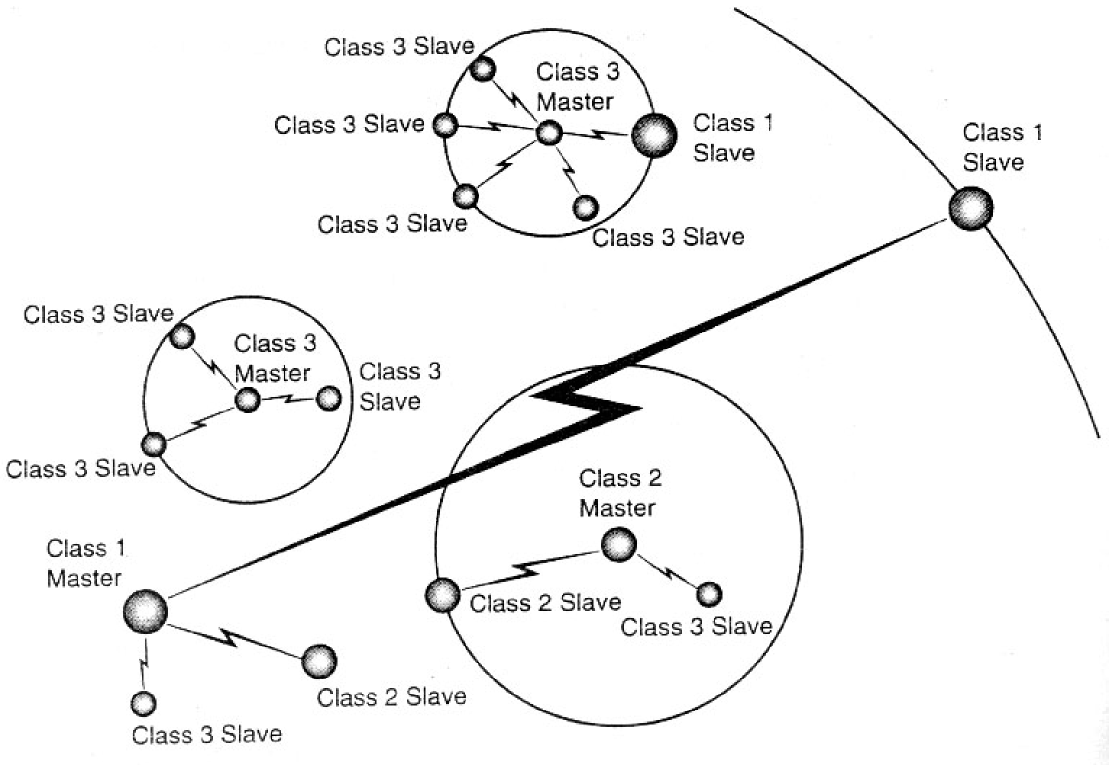


Figure 3.17 – Bluetooth power classes

3.8.2 Ad-hoc Network, Piconet and Scatternet

Bluetooth is an *ad-hoc* network. Bluetooth devices are dynamic and move around. When it moves out of range, the device is disconnected from a network. When the device moves within the range again, it may join the network again. Because of this highly dynamic behavior, the network participants change constantly and this makes the network administration far more difficult to manage and configure than traditional wired network.

A *piconet* is an arbitrary collection of Bluetooth-enabled devices which are connected. A piconet is formed by a master device and one to seven slave devices connected together. All slaves adopt the timing of the master and will respond to any messages that include the master's access code in the packet header.

A *scatternet* is formed when a group of piconets joined together by devices that are in more than one piconet. Scatternet exists because one of the devices is the master to a piconet and acts as a slave to another piconet at the same time. Figure 3.18 depicts the network formation of piconet and scatternet.

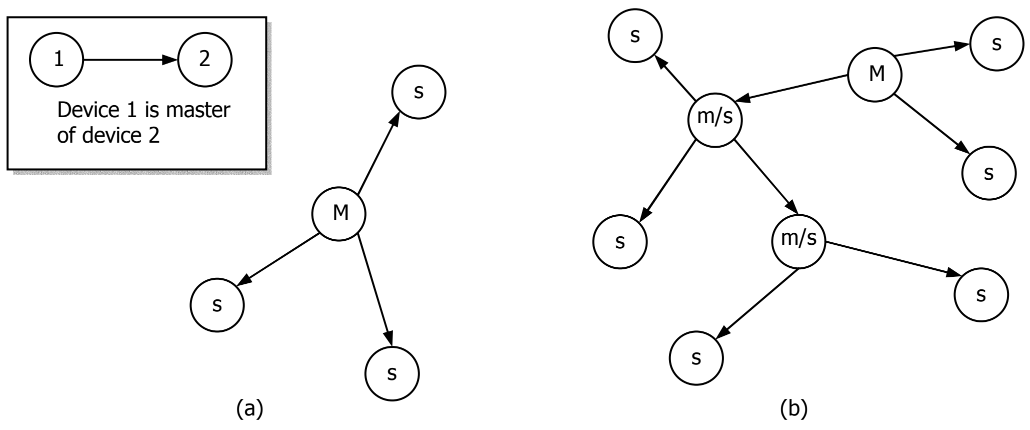


Figure 3.18 – Bluetooth network (a) piconet and (b) scatternet

When there are more than one slave are connected to a master, the master has to carefully select and schedule which slave device that the master will transmit to and receive from. Even if there is nothing to send, the master may simply omit that slave or transmit a NULL packet. Figure 3.19 shows how the master device is communicating with several slaves.

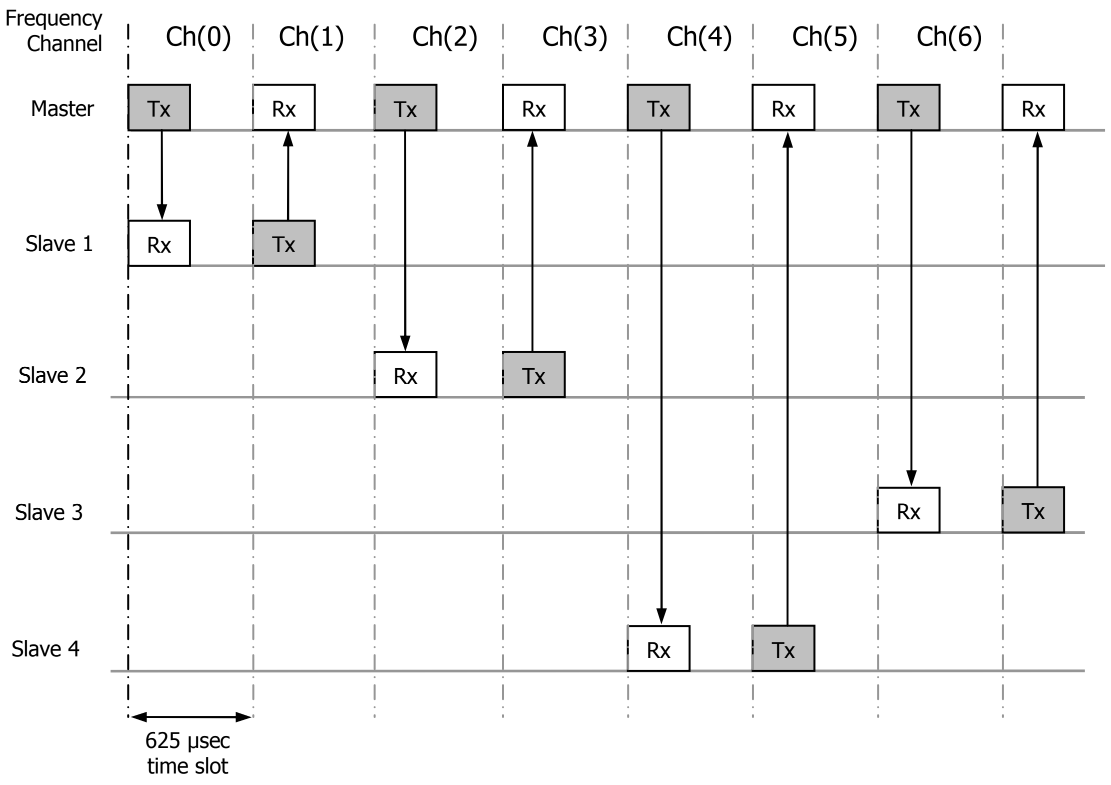


Figure 3.19 – Transmission of packets between master and several slaves

We can see that the transmission is totally controlled by the master. There are two directions where the packet is likely to flow: - master-to-slave and slave-to-master.

Master to Slave

The master simply sends out the packet to that particular slave in even-numbered slot. Upon received, the slave acknowledges the successful reception of packet by sending back an acknowledgement packet on the next odd-numbered slot.

Slave to Master

If a slave wishes to send a packet to the master, it has to wait until the master send out a packet (usually a POLL packet) on even-numbered slot. The slave is then able to send the packet to the master on next time slot.

3.9 Prototyping

A few prototypes have been developed to further understand and evaluate the performance of Bluetooth protocol. The Bluetooth Application Toolkit from Ericsson used for MEMSwear is shown in Figure 3.20.

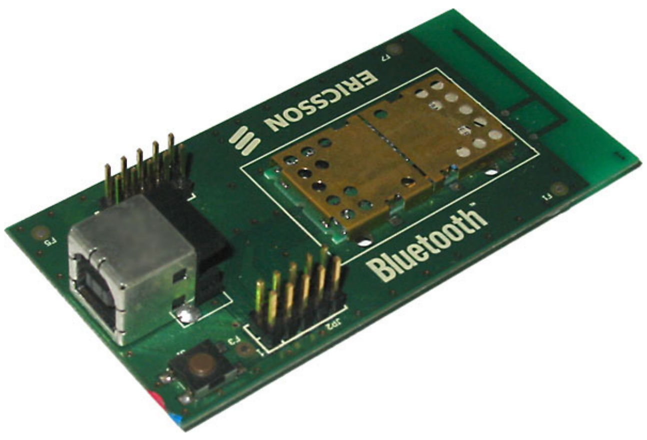


Figure 3.20 – Bluetooth Application Toolkit from Ericsson

One Bluetooth module is connected to a micro-controller that is used to collect signal from various sensors. These signals are then sent through the attached Bluetooth module that is already connected to another Bluetooth module on a computer. The MEMSwear software on the computer stores and displays the collated sensor signal and executes necessary analysis algorithm based on the signal, for instance the fall detection algorithm is executed based on the signal from the accelerometer. Figure 3.21 give an overview of the MEMSwear system using Bluetooth as the wireless communication mean.

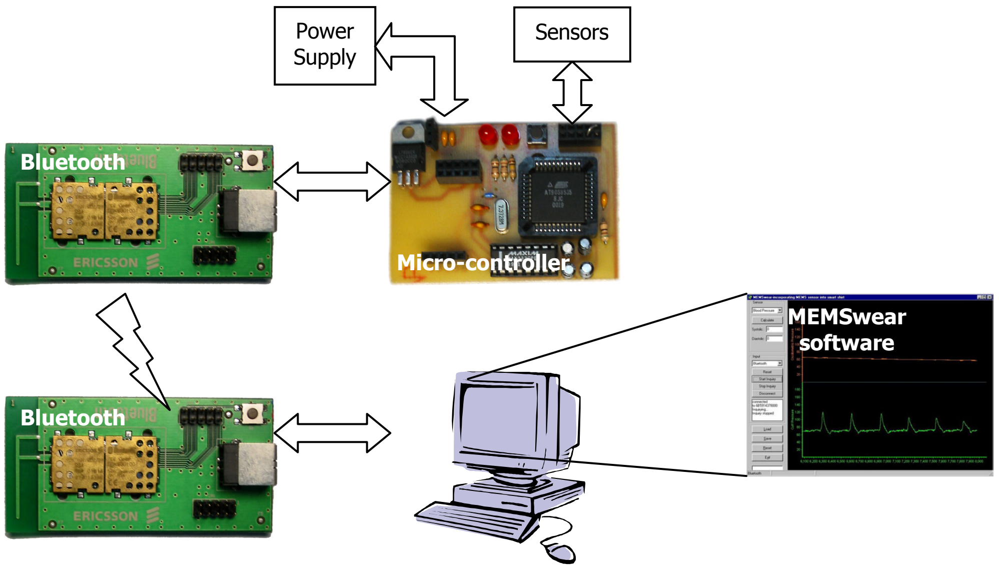


Figure 3.21 – MEMSwear using the Bluetooth as communication tool

The micro-controller shown in Figure 3.22 was fabricated. It uses Atmel AT90S8535 microprocessor as the main processing unit. Signal from various sensors such as accelerometer (both one axis and three axes), blood pressure sensor and ECG sensor have been successfully collected by the microprocessor. The sensors are connected to the sensor interface and ADC (analog-to-digital converter) of the microprocessor. The microprocessor will collect the signal from these sensors and send wirelessly to the computer through the Bluetooth module. The microprocessor communicates with the Bluetooth modules through the serial port.

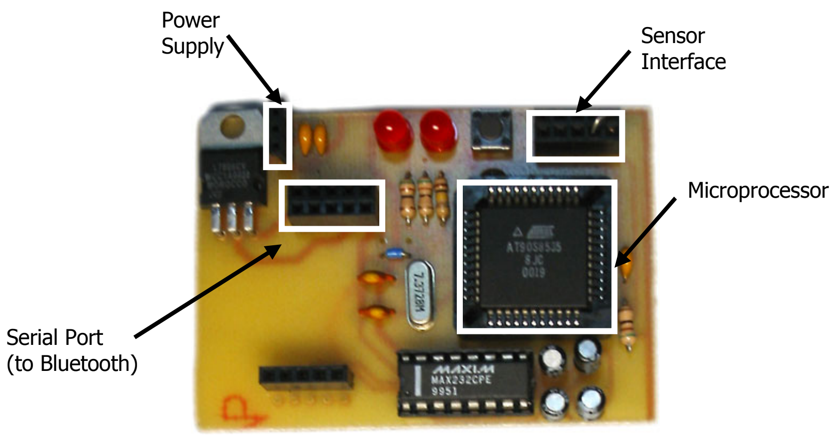
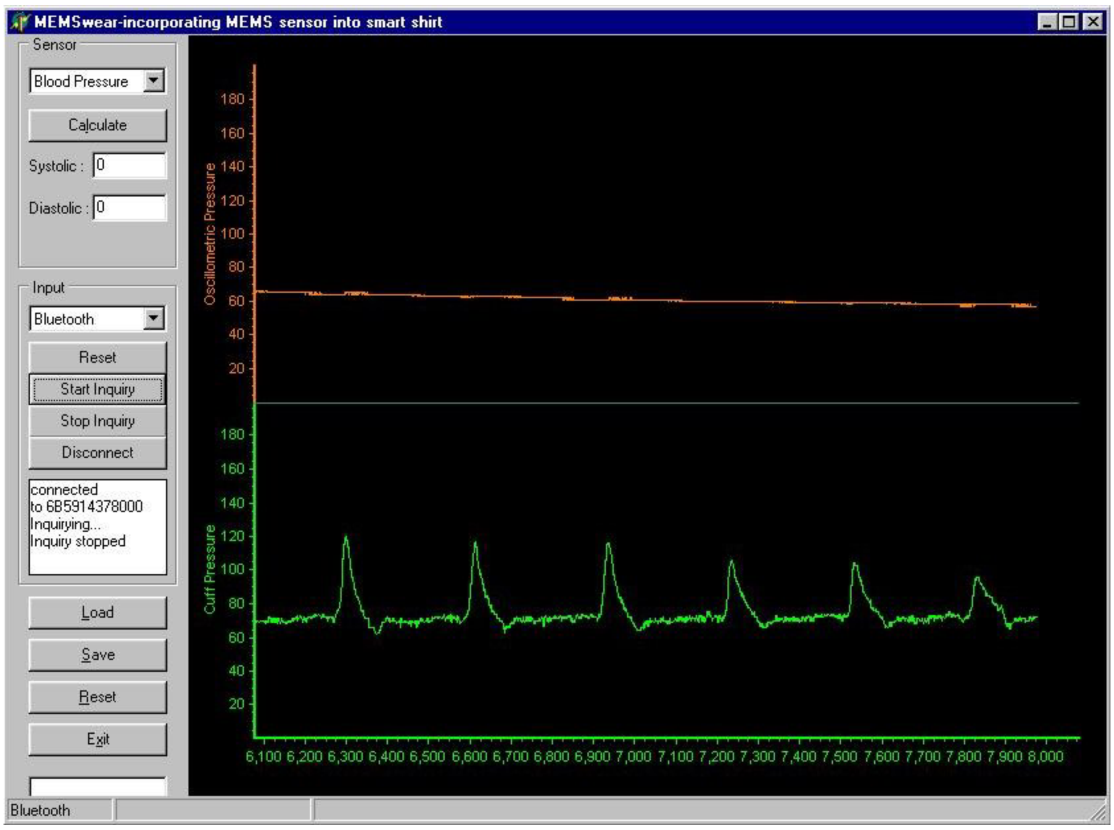


Figure 3.22 – Components of the micro-controller

In conjunction with this, the MEMSwear software is also developed to facilitate data collection, data display and data storage. Using this software (see Figure 3.23), data from the sensors can be further analyzed to observe the health situation of the wearer.



connected
to 6B5914378000
Inquirying...
Inquiry stopped

Figure 3.23 – MEMSwear software showing signal from the blood pressure sensor on computer

Through the prototyping, it is observed that Bluetooth technology is very appropriate to be used for *MEMSwear* technology. Prototyping is not suitable if several units are required, such as evaluating the network capability of Bluetooth as in the case of *MEMSwear*. This is due to the high prototyping cost of using the Bluetooth evaluation kit. Software simulation for Bluetooth proves to be a more cost effective way of evaluating networking with Bluetooth for *MEMSwear*.

CHAPTER 4 Bluetooth Simulation

Network simulation is a very useful and important tool for network performance evaluation and analysis. This is because networking is very complicated and highly dynamic. Besides, it also provides a cost effective way to network analysis as compared to prototyping analysis. Simulation gives us a way to test the reliability and correctness of software produced in the development of networking protocols.

Hence, there is a need for good Bluetooth simulation software. However, it has been found that currently there is not any Bluetooth simulation software that provides flexibility and yet accurate results for *MEMSwear* project. Therefore, the author has developed a new Bluetooth simulation software, *blue-ns*, the development will be discussed in details in this section. The main purpose of developing the simulation software is to provide better understanding the Bluetooth protocol stack so as to evaluate the suitability of Bluetooth as the wireless communication means for *MEMSwear*.

Since the wearers of the *MEMSwear* are highly mobile within the house compound, the Bluetooth simulator will be able to simulate the situations where the wearers move about and understand the flow of data between the devices. The Bluetooth simulator will even be more useful and handy when analyzing the situation where conflicts happen. Prototyping does not provide enough details when these situations happen and these can only be examined accurately using Bluetooth simulation software.

4.1 Network Simulator 2 (ns-2)

Network Simulator 2 (*ns-2*) is open source and is a non-commercial network simulation software [32]. It is an object-oriented, time-driven, discrete event driven network simulator developed at UC Berkeley written in C++ and OTcl. *ns-2* is primarily useful for simulating local and wide area networks. It provides complete support for most wired and wireless environment, including TCP, routing and multicast protocol. It is widely used by researchers for simulation analysis. The software can be easily extended by implementing extension to the *ns-2* software. This feature makes *ns-2* to continuously expand and grow. The open source nature of *ns-2* offers the flexibility to modify the source code to adapt to necessary changes.

ns-2 implements network protocols (such as TCP and UPD), traffic source behavior (such as FTP, Telnet, Web, CBT and VBR), router queue management mechanism (such as Drop Tail, RED and CBQ), routing algorithm (such as Dijkstra) and more. It also implements multicasting and some of the MAC layer protocols for LAN simulations.

ns-2 is actually an OTcl script interpreter that has a simulation event scheduler and network component object libraries and network setup module libraries (see Figure 4.1). In other words, to use *ns-2* for network setup and simulation, one needs to write simulation script in OTcl script language. The script consists of routines to initiate an event scheduler, setup the network topology using the network objects, setup the data paths among the network objects and tells the data traffic sources when to start and stop.

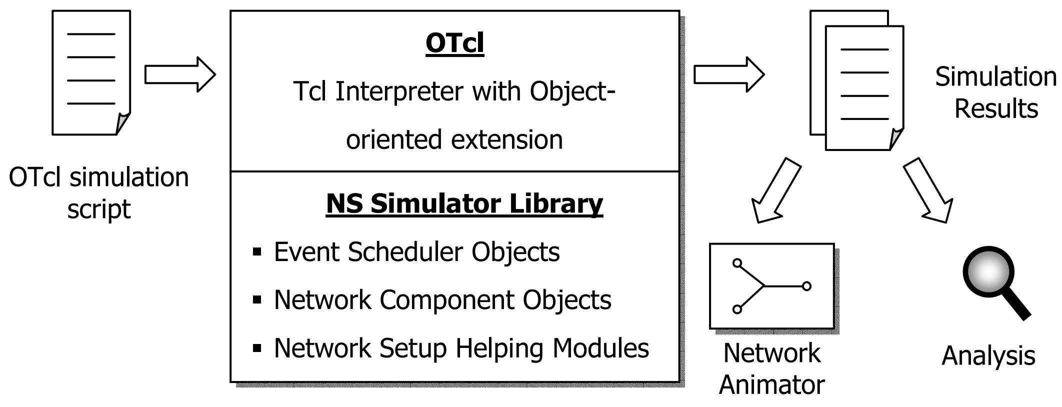

OTcl simulation script
OTcl
Tcl Interpreter with Object-oriented extension
NS Simulator Library
▪ Event Scheduler Objects
▪ Network Component Objects
▪ Network Setup Helping Modules
Simulation Results
Network Animator
Analysis

Figure 4.1 – Network Simulator 2

When the simulation finishes, the simulator produces one or more simulation results in text based files that contained detailed simulation data. The data can be used for simulation analysis or as an input to a graphical simulation display tool (such as Network Animator).

ns-2 is written not only in OTcl but in C++ as well. For efficiency reason, the network components are written and compiled using C++. These compiled objects are made available to the OTcl interpreter through an OTcl linkage that creates a matching OTcl object for each of the C++ objects. In this way, the controls of C++ objects are given to OTcl. Figure 4.2 shows an object hierarchy example in C++ and OTcl. For every object and method written in C++, there is a corresponding object and method in OTcl language.

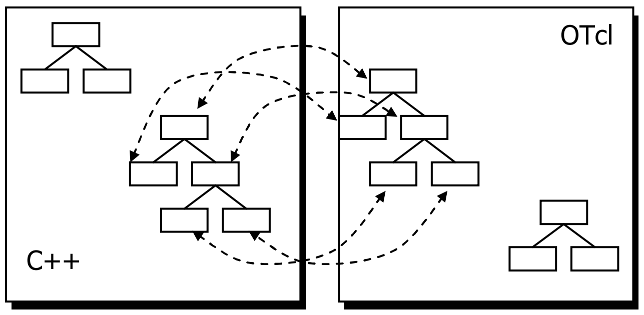


Figure 4.2 – OTcl and C++ duality

4.1.1 Network Animator

Network Animator (*NAM*) is a Tcl/Tk based animation tool for viewing network simulation traces and real world packet traces. It is a simple graphical user interface for animating packet trace data and this trace data is typically derived as output from *ns-2*. It supports topology layout, packet level animation and various data inspection tools. It also graphically presents information such as throughput and number of packet drops at each link. Figure 4.3 shows a screenshot of the Network Animator.

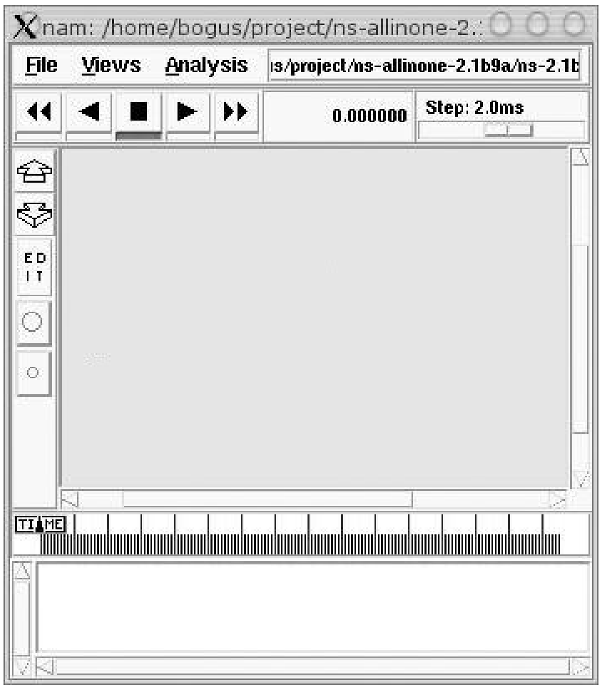

nam: /home/bogus/project/ns-allinone-2.
File
Views
Analysis
s/project/ns-allinone-2.1b9a/ns-2.1b
0.000000
Step: 2.0ms
EDIT
TIME

Figure 4.3 – Screenshot of a Network Animator

4.1.2 BlueHoc

BlueHoc (Bluetooth Ad-hoc Network Simulator) is developed by IBM developer Apurva Kumar [33] as a Bluetooth extension to *ns-2*. The latest version of *BlueHoc* supports simulation of various layers of Bluetooth protocol stack including the Radio, the Baseband, the LMP, the Host Controller and the L2CAP.

BlueHoc uses the TCP/IP simulations of network simulator to provide a complete simulation model for Bluetooth performance evaluation. It also provides a simulation model for testing performance of various routing and service discovery protocols over an ad-hoc network. As the simulation model of *BlueHoc* closely approximates Bluetooth protocols, it can be used as a platform to evaluate the performance of proposed improvements to the technology. The key issues addressed by the *BlueHoc* are: -

- Device discovery performance of Bluetooth
- Connection establishment and QoS negotiation
- Medium access control scheduling schemes
- Radio characteristics of Bluetooth system
- Statistical modeling of indoor wireless channel
- Performance of TCP/IP based application over Bluetooth

BlueHoc provides a very good foundation of Bluetooth simulation. However, further analysis shows that *BlueHoc* has its shortcomings, among these are: -

- **Different structure for master and slave**

  The design adopted by *BlueHoc* is such that the role of a Bluetooth is predefined. There is no way to switch the role between master and slave as suggested by the Bluetooth Specification. This is mainly because *BlueHoc* uses different structure for master and slave.

- **No mobility support**

  The Bluetooth device is static and pre-located. There is no support of node movement in *BlueHoc*. Thus, it would be difficult to simulate situation that requires flexible movement and highly dynamic nodes movement.

- **Lack of common control interface**

  BlueHoc supports only a small subset of all the Host Controller Interface (HCI) commands and events as specified in the Bluetooth Specification. The HCI provides a common control interface for which the Bluetooth can be controlled.

- **No graphical support**

  Though *BlueHoc* is able to produce simulation result for performance evaluation, it does not produce any trace file for animation such as Network Animator. The reason is because it does not support nodes movement. Animation support is very important for visualizing how the nodes are moving and how the movement affects the performance.

4.2 Blue-ns (Bluetooth simulation for ns-2)

To evaluate the performance of Bluetooth in this project, an extensible Bluetooth simulator called *blue-ns* has been developed, as an extension to *ns-2*. Based mainly on the *BlueHoc*, the *blue-ns* simulator implements many aspects of the Bluetooth protocol stack according to the Bluetooth Specification (version 1.1). The *blue-ns* overcomes the disadvantages of *BlueHoc* and implements more features.

Each layer in the Bluetooth protocol stack is implemented as module in the *blue-ns* extension. The general structure of *blue-ns* is shown in Figure 4.4. In essential, the Baseband layer, the Link Controller, the Link Manager, the Host Controller and the L2CAP layer are included.

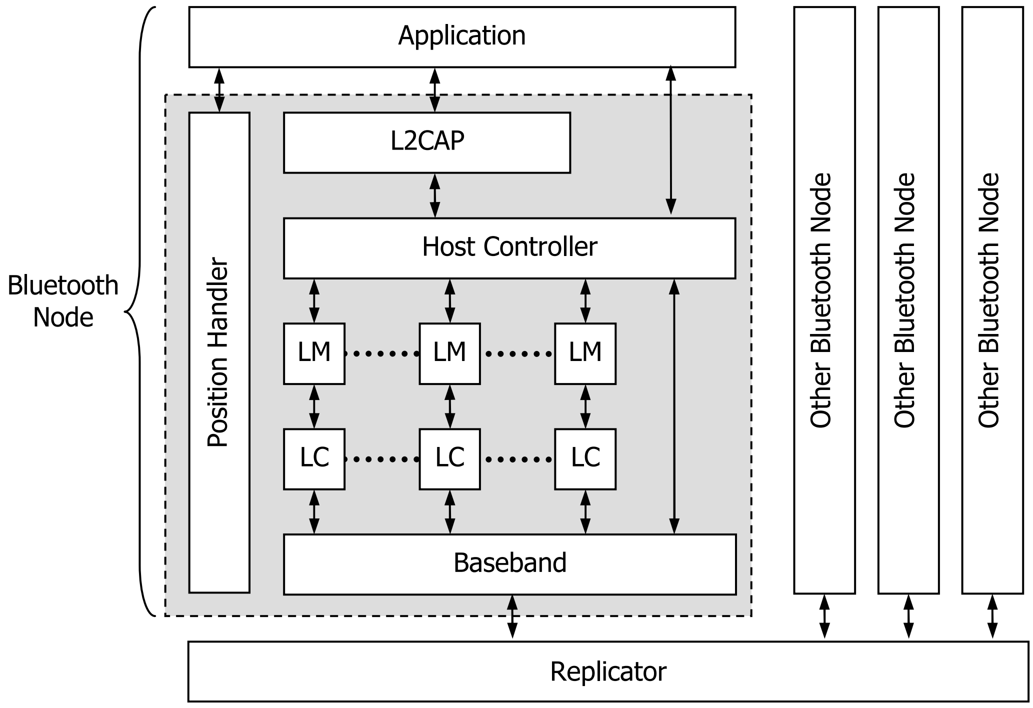


Figure 4.4 – Structure of blue-ns

In this section, the implementation of Bluetooth simulation in *ns-2* is discussed in details.

4.2.1 Bluetooth Node

Each Bluetooth device is represented as a wireless node in *blue-ns*. Each node consists of Bluetooth protocol stack from the lowest Baseband layer up to the L2CAP layer. Anything above the L2CAP layer is application specific. The application layer can issue command to L2CAP and Host Controller (using HCI commands) to control or to obtain status of the lower layers.

The Baseband of the node is attached to a Replicator. The Replicator will send all packets that it receives to all other attached nodes. Effectively, it works just like a Bluetooth device sending a packet to other Bluetooth devices.

The physical location of each node is represented by the x and y position on the map. Each node has a Position Handler which will keep track of the position. The Position Handler will constantly compare the current position with the specified destination. If the node has not reached its destination, the Position Handler will calculate the position based on the speed and time until it has reached.

4.2.2 Baseband

The Baseband module in the *blue-ns* simulator has the following functions: -

- Maintain an internal Native Clock (CLKN) that will increment every 625 μsec. With every CLKN ticked, it represents a time-slot of 625 μsec has elapsed.
- Maintain various timers used for inquiry, paging, response and connection.
- Perform frequency hop sequence selection to determine the frequency to be used for each channel.
- Receive and filter out unwanted Bluetooth packets before sending up to upper layer.
- Perform Baseband level inquiry, paging, creating connection and disconnection procedures.
- Send Bluetooth packets from the upper layers.

All upper layers that need to send packet must pass it to the Baseband. It is compliant with the Bluetooth specification as described in Section 3.3 and it is capable of performing various Baseband level operations (inquiry, paging, connection, packet exchange and disconnection). Any packet received that is not related to Baseband will be sent to its upper layer, i.e. the Link Controller.

4.2.3 Link Controller

For every link formed between the master and the slave, there will be an instance of Link Controller to manage the connection. For instance (as shown in Figure 4.5), if a master device is connected to three other slave devices, there will be three instances in the master device while there is only one instance of Link Controller in each of the slave devices.

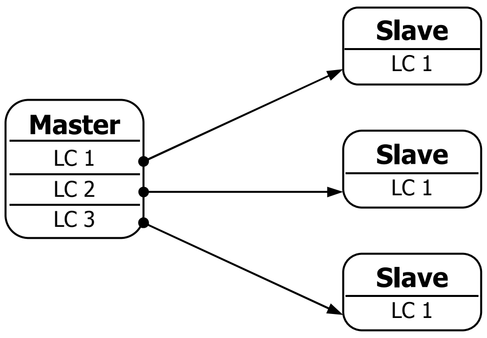


Figure 4.5 – Link Controller in a master and slave devices

The Link Controller is used to ensure reliability of a single link by implementing the Automatic Repeat Request (ARQ) and sequence number (SEQN) in the packet header. For every packet received, the recipient must send an Acknowledgement (ACK) back to the sender. If the sender does not receive any ACK, it will assume a Negative-Acknowledgement (NAK) and it will retransmit the packet again. The Link Controller will also filter out packets that have the same SEQN flags, because it means that the packet has been received previously.

When the Link Controller receives a packet from the lower layer, which is the Baseband, the Link Controller will then execute the mechanism as described. Following the execution, the packet will be sent up to the upper Link Manager layer.

The Link Controller only examines the ARQN and SEQN field in the packet header without modifying other content of the packet.

4.2.4 Link Manager

The Link Manager implements several operations that are used for link setup and control. More specifically, the Link Manager is responsible for setting up, configuring as well as shutting down a connection. Just like the Link Controller, for every connection setup, there will be an instance of Link Manager to manage that connection. Table 4.1 shows the LMP commands supported in *blue-ns*.

LMP	**Length**	**OpCode**	**Contents**
LMP_Accepted	2	3	Opcode
LMP_Detach	2	16	Reason
LMP_Host_Connection_Req	1	51	-
LMP_Hold	7	20	Hold time, hold Instant
LMP_Hold_Req	7	21	Hold time, hold instant
LMP_Not_Accepted	3	4	Opcode, reason
LMP_Quality_of_Service	4	41	Poll interval, N_{BC}
LMP_Quality_of_Service_Req	4	42	Poll interval, N_{BC}
LMP_Setup_Complete	1	49	-
LMP_Supervision_Timeout	3	55	Supervision timeout

Table 4.1 – LMP commands in blue-ns

The Link Manager handles packets that contain the LMP. An LMP packet can be distinguished by the L_CH field of the payload header (refer to Table 3.1). These LMP packets are filtered and interpreted by the Link Manager on the receiving side and are not propagated to higher layer. The LMP packets are also composed by the Link Manager when commanded to do so by the upper layer.

4.2.5 Host Controller

The Host Controller implements the Host Controller Interface (HCI). It receives command and data packet from the upper layer, particularly the L2CAP and the application layer, and translates them into appropriate actions. It sends back HCI event packets to reflect the status of lower layers (refer to Section 3.6).

Not all HCI commands specified in the specification are implemented by the *blue-ns*, but the implementation provides sufficient functions. Table 4.2 lists the HCI commands and events implemented by *blue-ns*.

Link Control Commands	
HCI_Inquiry	To discover other nearby Bluetooth devices.
HCI_Create_Connection	To create a connection link to a Bluetooth device.
HCI_Disconnect	To terminate an existing connection.
HCI_Accpet_Connection_Request	To accept a new incoming connection request.
Host Controller & Baseband Commands	
HCI_Write_Scan_Enable	To enable or disable the inquiry scan and page scan of the device.
HCI_Write_Link_Supervision_Timeout	To set the value of supervision timeout.
HCI Events	
HCI_Inquiry_Result_Event	To indicate that a Bluetooth device has responded during the Inquiry process.
HCI_Inquiry_Complete_Event	To indicate that the Inquiry process is finished.
HCI_Connection_Complete_Event	To indicate that the new connection has been established between two devices.
HCI_Connection_Request_Event	To indicate that a new incoming connection is trying to be established.
HCI_Disconnection_Complete_Event	To indicate that a connection has been terminated
HCI_Command_Complete_Event	To indicate the return status of a command issued and the other event parameters for each HCI command.
HCI_Command_Status_Event	To indicate that the command has been received and the Host Controller is currently performing the task for this command.

Table 4.2 – HCI commands and events in blue-ns

4.2.6 Logical Link Controller and Adaptation Protocol (L2CAP)

The *blue-ns* simulator implementation includes the L2CAP layer as well. It supports two main functions: - protocol multiplexing and segmentation and reassembly. The L2CAP is layered on top of the Host Controller.

Signaling Commands	Parameters
L2CAP_Connect_Req	PSM, Source CID
L2CAP_Connect_Rsp	Destination CID, Source CID, Result, Status
L2CAP_Configuration_Req	Destination CID, Flags, Options
L2CAP_Configuration_Rsp	Source CID, Flags, Result, Config
L2CAP_Disconnect_Req	Destination CID, Source CID
L2CAP_Disconnect_Rsp	Destination CID, Source CID
L2CAP_Data	Destination CID, Length, Data

Table 4.3 – L2CAP signalling commands

4.3 Basic Bluetooth Operations

We can actually simulate and understand the behaviors of a Bluetooth device in details using the *blue-ns* simulator. The results and performance under different conditions can be easily analyzed and obtained. Following this section, some basic Bluetooth operation using the *blue-ns* simulator will be examined. We will see how the command issued flows through the protocol stacks and how different Bluetooth packets are exchanged between devices.

4.3.1 Inquiry – discover other devices

The first step is the inquiry process. Due to the ad-hoc nature of Bluetooth network, highly mobile devices may come into and go out of range of other communicating devices. This causes the topology and membership to change constantly. Thus, inquiry is an important process to find out the devices within the area.

A device (inquiring device/inquirer) wishing to discover other nearby devices enters into **INQUIRY** mode requested by the higher control layer (usually by the Host Controller). During this mode, the Baseband will continuously send out an inquiry ID packet[5] consisting of the IAC at every half of the time slot, i.e. every 312.5 μsec. The process will continue until the inquiring timer reaches the inquiry timeout period specified. The default inquiring timeout is 1.28 seconds but this value can be changed to different value. Figure 4.6 shows that an Inquiry ID packet denoted as the innermost circle of node 1 (broadcast packet is represented as circle) was broadcasted at 2.271563 seconds.

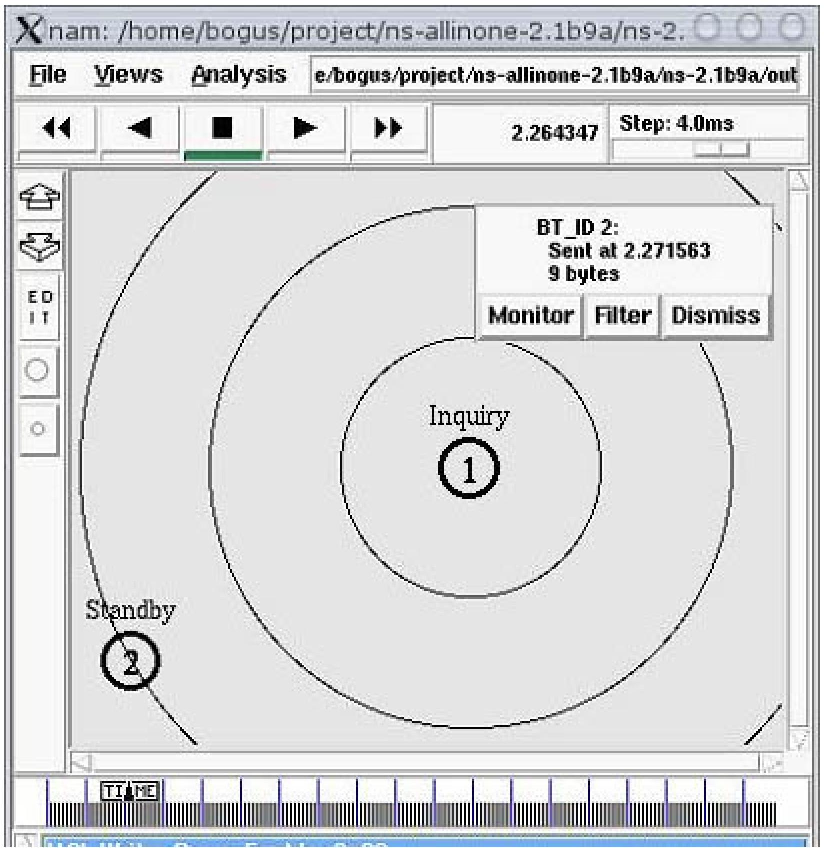
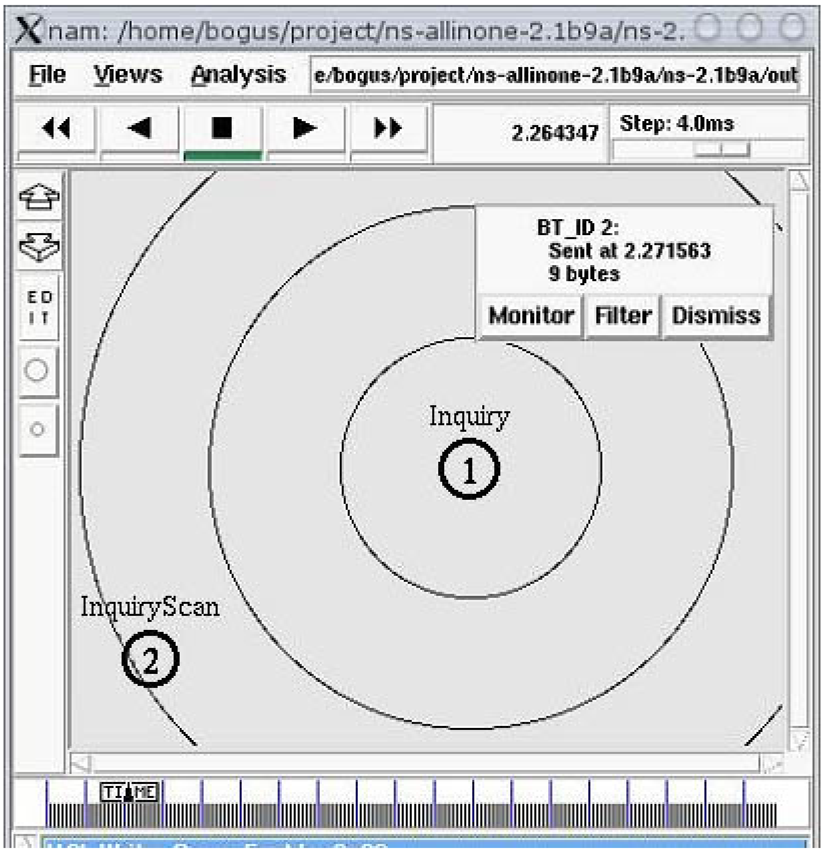

nam: /home/bogus/project/ns-allinone-2.1b9a/ns-2.
File
Views
Analysis
e/bogus/project/ns-allinone-2.1b9a/ns-2.1b9a/out
2.264347
Step: 4.0ms
Monitor
Filter
Dismiss
Inquiry
1
Standby
2
TIME
BT_ID 2:
Sent at 2.271563
9 bytes

Figure 4.6 – Node 1 is broadcasting out the inquiry ID packets

[5] An inquiry ID packet contains the Inquiry Access Code (IAC) in the access code field of the packet header. Refer to 3.3.5.1 for more details.

On the other side, a device (inquiring scanning device/inquiry scanner) wanting to be discovered issues the enable scan command to enable the **INQUIRY SCAN** mode. Instead of scanning for inquiry ID packets continuously like the **INQUIRY** mode, the Baseband only performs inquiry scan periodically over a short window. The advantage is to conserve power as continuous scanning would drain the battery quickly. Having said so, the main disadvantage of this scheme is that this device may miss an inquiry ID packet sent by the inquiring device and causes long discovery period. Figure 4.7 shows that Bluetooth node 2 enters into **INQUIRY SCAN** mode and during this period it receives an Inquiry ID packet from node 1.


nam: /home/bogus/project/ns-allinone-2.1b9a/ns-2.
File
Views
Analysis
e/bogus/project/ns-allinone-2.1b9a/ns-2.1b9a/out
2.264347
Step: 4.0ms
EDIT
Monitor
Filter
Dismiss
Inquiry
1
InquiryScan
2
TIME

BT_ID 2:
Sent at 2.271563
9 bytes

Figure 4.7 – Node 2 receives inquiry ID packet while in Inquiry Scan mode

When the inquiry-scanning device receives an inquiry ID packet during page scanning windows, it could respond immediately, but this could lead to several devices responding together. The responses would interfere with one another and eventually the inquiring may end up receiving none of the responses. Instead, the Bluetooth Specification defines a random back-off period ranging from 0 ms to 640 ms that the inquiry-scanning device must wait before responding to an inquiry ID packet. It waits for a random period, and then re-enter the scanning state, listening once more for another ID packet. If it hears the inquiry one more time, it will reply with the FHS packet. This scheme will ensure that several devices can respond to an inquirer but their responses will be spaced out randomly and thus will not interfere. Figure 4.8 illustrates the packets exchange during the inquiry process.

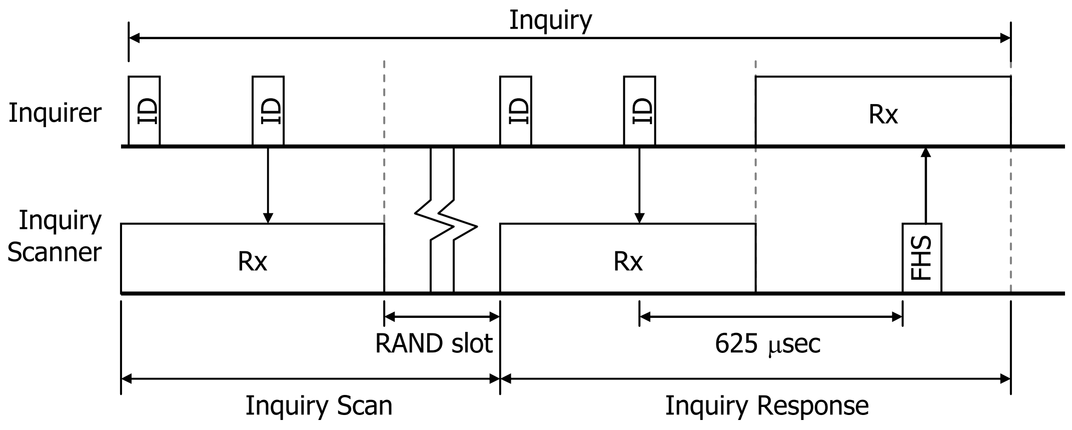


Figure 4.8 – Packets exchange during the inquiry process

4.3.2 Paging – initiating connection

Paging is a term used to describe a request to form a connection/link with a remote device. On the other hand, a page-scanning device will be listening for such request. The device (paging device/pager), which sends out the request, is usually a *master* while the device (page-scanning device/page-scanner) that is listening for such request is called *slave*.

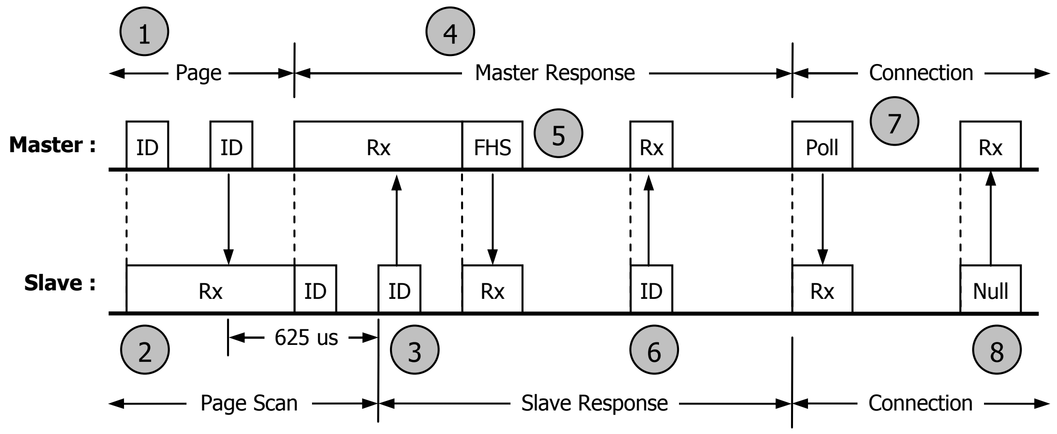


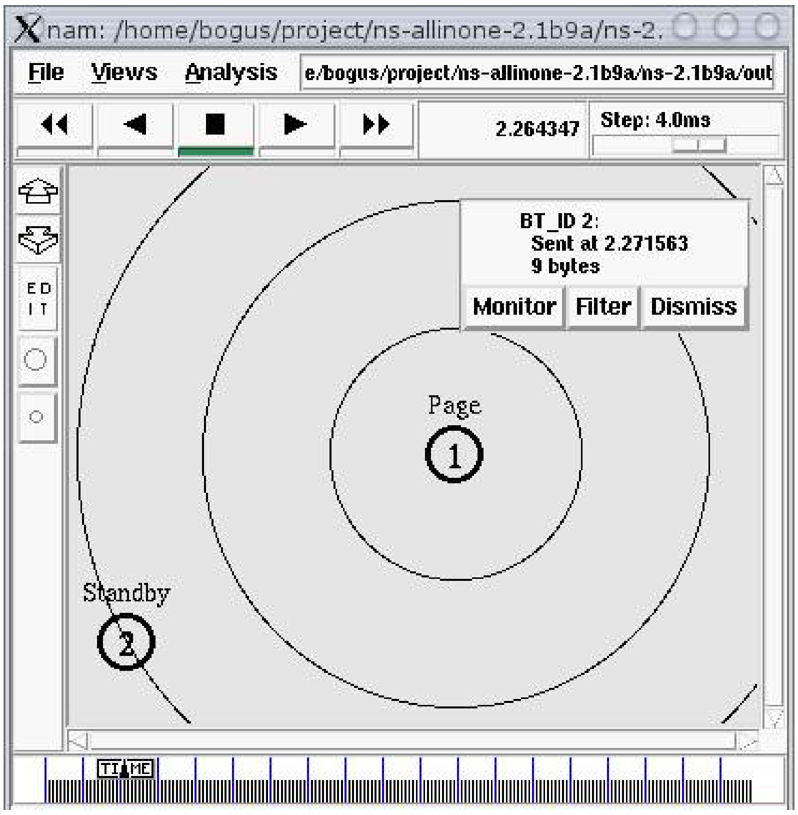
Figure 4.9 – Packets exchange during the Paging process

The steps involved in the paging process is shown in Figure 4.9 and explained as follows: -

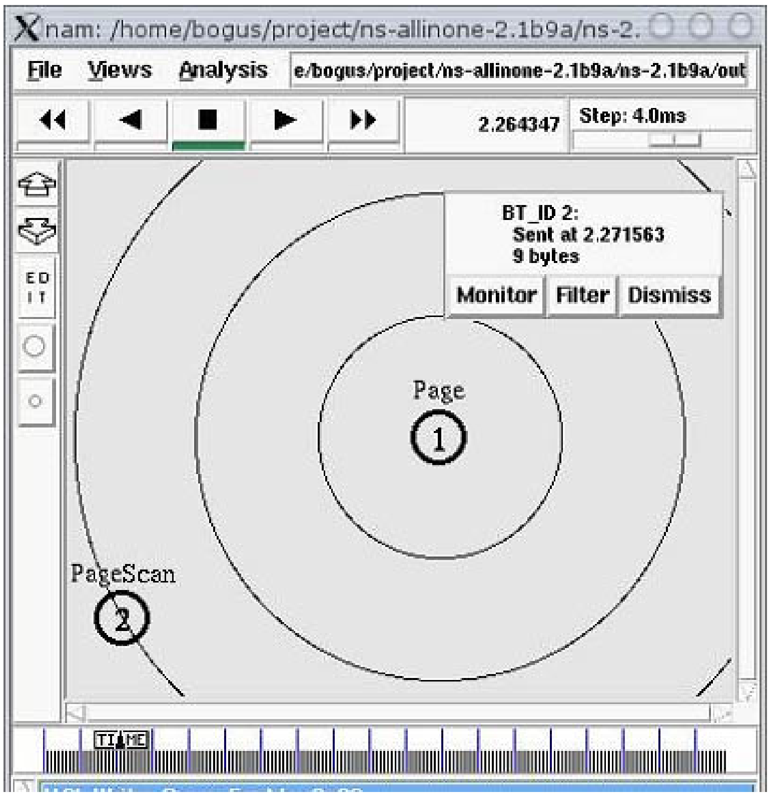
1. When the upper control layer issues a create connection command, the Baseband will enter into **PAGE** mode where it sends out a series of paging ID packets[6] at an interval of 312.5 µsec (see Figure 4.10 (a)).
2. Meanwhile, the page-scanning device is configured to carry out periodic page scans of a specified duration and at a specified interval. It is in **PAGE SCAN** mode (see Figure 4.10 (b)).
3. When the page-scanner receive an paging ID packet with its own address in the access code, it will reply with another ID packet to the paging device after 625 µsec and enter into **SLAVE RESPONSE** sub-state. There is no need for random back-off timer as in the inquiring process because only one device will respond to this ID packet (see Figure 4.10 (c)).
4. When the pager receives the responded ID packet, it enters the **MASTER RESPONSE** sub-state (see Figure 4.10 (c)).

[6] A paging ID packet contains the page scanner device's address in the access code field of the header. Refer to 3.3.5.1 for more details.

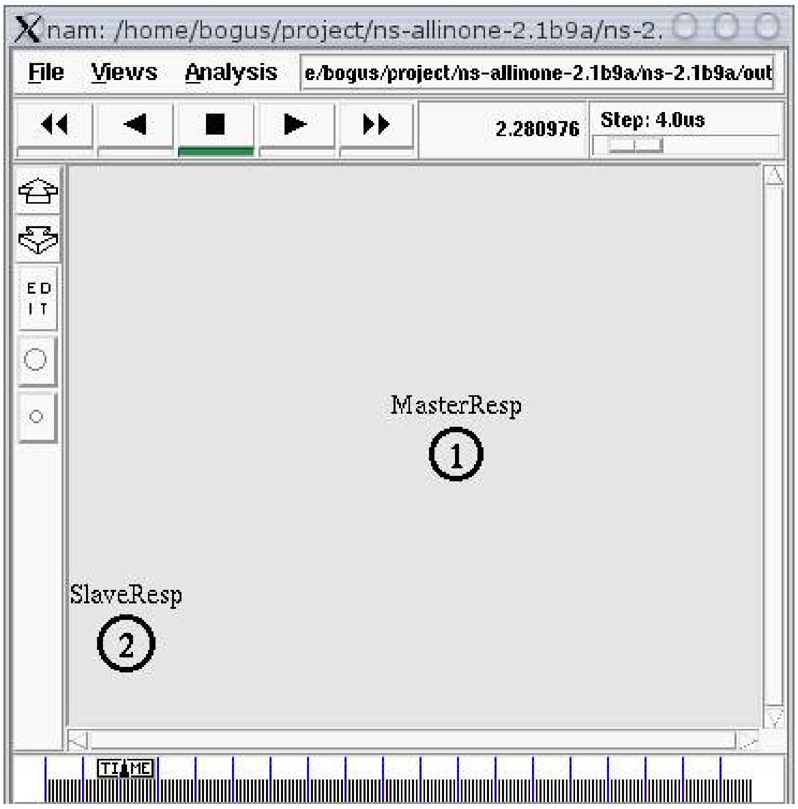
5. A series of handshaking process begins. The paging device will send a FHS packet containing essential parameters used for this new connection, including the master clock, AM_ADDR, etc. This information is sufficient to calculate the hop frequency sequence used in the transmission.
6. The page-scanner device will acknowledge the reception of the FHS packet by replying with an ID packet again and now move into the **CONNECTION** state and becomes slave of the connection.
7. On reception of this ID packet, the paging device enters into **CONNECTION** mode as well and sends a POLL packet on next time slot. This is to check that the frequency hop sequence switch has happened correctly.
8. The slave must respond with a NULL packet.
9. Following this, the link is established and ACL packet can be exchanged.



(a) Node 1 enters in Page mode and sends out paging ID packets



(b) Node 2 enters Page Scan mode at specified time and receives paging ID packet from node 1


nam: /home/bogus/project/ns-allinone-2.1b9a/ns-2.
File
Views
Analysis
e/bogus/project/ns-allinone-2.1b9a/ns-2.1b9a/out
2.280976
Step: 4.0us
MasterResp
1
SlaveResp
2
TIME

(c) Node 1 enters Master Response mode and node 2 enters Slave Response mode

Figure 4.10 – The state transition of the nodes in the paging procedure

4.3.3 Creating connection

Following the successful Baseband paging procedure, an ACL link would have been established. These raw data is further protected by the Link Controller for its reliability and integrity (see Section 3.4). The ACL link merely provides the transmission and reception of raw data over the air. Thus, higher layer protocol must be defined so that the raw data can be understood. The protocol messages are carries in packet through the ACL link.

4.3.3.1 Link Manager level connection

First of which is the Link Manager Protocol (LMP) used by the Link Manager. LMP is mainly used for link setup and control. The messages intended for Link Manager are interpreted and filtered out by the Link Manager on the receiving side and are not propagated to higher layer. All the LMP messages are flowing through the ACL link that the Baseband has already setup previously.

When the paging device wishes to create a connection involving layers at the Link Manager, it sends the LMP_host_connection_req message to the slave. When the other side receives this message, the host will be informed of this incoming connection. The upper layer may choose to reject or accept this connection. To accept the connection, it needs to send back a LMP_accepted message to the master.

At this point, the master may choose to exchange other optional configuration messages such as security procedures, quality of service and etc. After all the configuration has been completed, the master will send a LMP_setup_complete message to the slave to confirm the completion of the link setup and the slave will send back LMP_setup_complete message to acknowledge the master as well. Figure 4.11 shows the protocol exchange between the Link Managers.

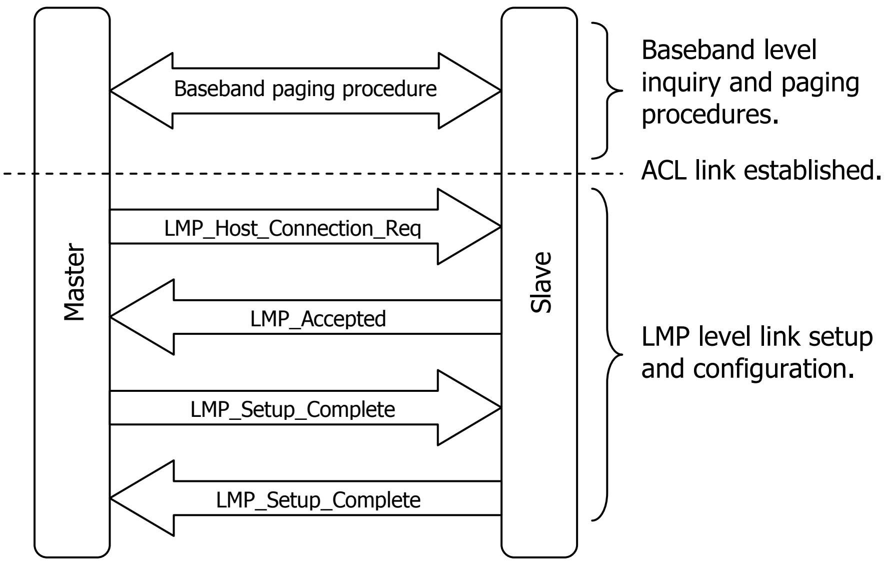

Master
Slave
Baseband paging procedure
Baseband level inquiry and paging procedures.
ACL link established.
LMP_Host_Connection_Req
LMP_Accepted
LMP_Setup_Complete
LMP_Setup_Complete
LMP level link setup and configuration.

Figure 4.11 – Link Manager level connection establishment

4.3.3.2 L2CAP level connection

A layer up the Link Manager is the L2CAP layer. L2CAP uses also ACL lnks to reliably pass data without errors across the lower layers of the Bluetooth stack. The connection procedure starts when the higher layer send the L2CAP a L2CA_Connect_Req command. This causes the L2CAP to send L2CAP_Connect_Req packet over the air to the slave device.

When the slave receives this L2CAP packet, the Baseband sends this packet straight up to the L2CAP level where L2CAP will inform the upper layer with a L2CA_Connect_Ind event. To accept this connection, the application/upper layer send the L2CA_Connect_Rsp command to the L2CAP layer. Upon receiving this command, the L2CAP will compose a L2CAP_Connect_Rsp packet to send over the air back to the master. This packet contains the response of the slave of whether it accept or reject the connection. The master receives this response packet and forward this packet to the L2CAP layer to interpret the result. If the packet indicates negative response, the application/upper layer will receive a L2CA_Connect_Neg, else it will receive a L2CA_Connect_Cfm event. The procedure is illustrated in Figure 4.12.

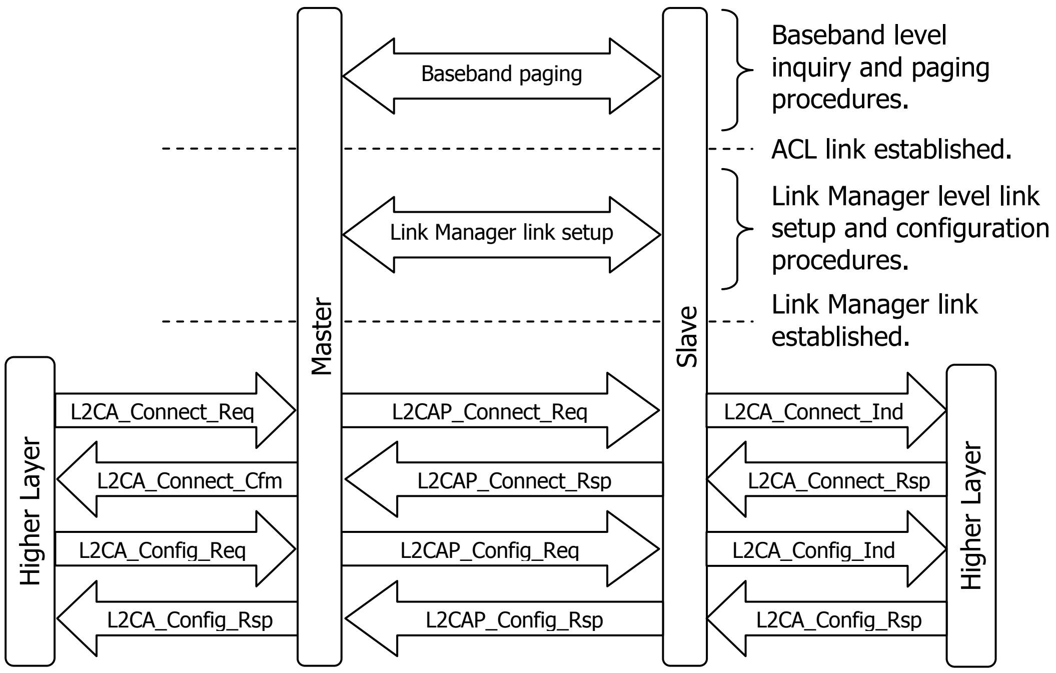


Figure 4.12 – L2CAP level connection establishment

At this point, assuming that both devices agreed to established connection, the master then sends a L2CA_Config_Req command to the L2CAP and the L2CAP will send out the L2CAP_Config_Req packet to the slave. This packet contains configuration information required to further configure link properties, such as Maximum Transfer Unit (MTU), Quality of Service (QoS) and etc. The slave, upon received this configuration packet, will adjust according the configuration packet and reply with a L2CAP_Config_Rsp packet. The L2CAP, at the master side, considers this response packet as an agreement from the slave to accept the configuration setting. From this point onwards, the L2CAP level connection is established.

4.3.4 Data transfer

Once the connections have been setup on various layers, the data is now able to flow between the devices. The application could send the data through the HCI data packets (refer to Section 3.6). The Host Controller uses smaller HCI data packets. The Bluetooth Specification defines that all implementations must support packets carrying up to 255 bytes of data. Thus, for the application data that is larger than 255 bytes, it has to route through the L2CAP layer using the L2CA_Data_Write and L2CA_Data_Read as shown in Figure 4.13. The L2CAP packets are very large, carrying up to 65,535 bytes of data. The L2CAP still uses the HCI data packets to send the large size data, but in chunks of smaller size packets that the HCI allows.

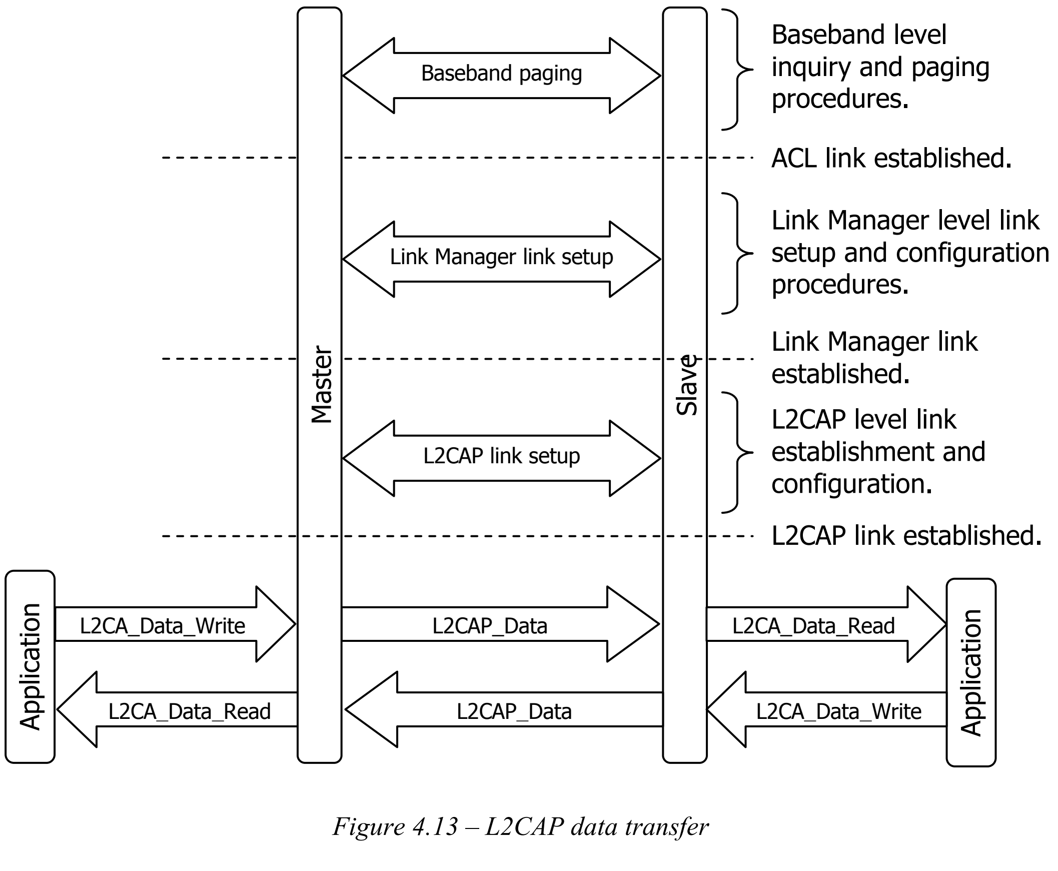
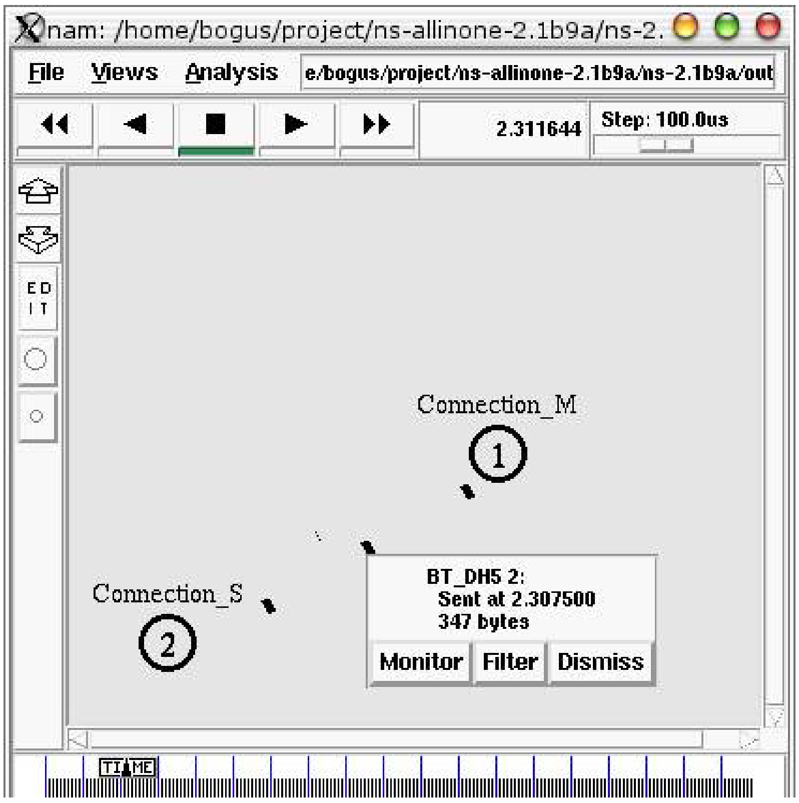


Figure 4.13 – L2CAP data transfer

The larger L2CAP packets may have to be segmented into data portions of several HCI data packets. When the packets come to its destinations, they have to be reassembled to form the large data packet as the originated packet. The first packet of the segmented L2CAP packet is marked in the L_CH flag of the L2CAP packet header to ease the reassembly process. Although each packet may have to be retransmitted several times, the Link Controller will reliably transfer packets in order and any repeats received will be filtered out by the Link Controller. So even though L2CAP packets may be split up and reassembled several times in passing through the Bluetooth protocol stack, they can always be delivered reliably and reassembled in the right order, with the start of each L2CAP packet easily identifiable. See Figure 4.14.


nam: /home/bogus/project/ns-allinone-2.1b9a/ns-2.
File
Views
Analysis
e/bogus/project/ns-allinone-2.1b9a/ns-2.1b9a/out
2.311644
Step: 100.0us
EDIT
Connection_M
1
Connection_S
2
Monitor
Filter
Dismiss
TIME

BT_DH5 2:
Sent at 2.307500
347 bytes

Figure 4.14 – Node 1 and 2 are connected and transmitting data over the connection

4.3.5 Disconnection

As with connection established, disconnection has to be carried out layer by layer. It starts with the L2CAP disconnection, followed by Link Manager link shutdown and finally the Baseband layer disconnection.

4.3.5.1 L2CAP disconnection

Once the data transfer has finished, the protocol or service using the L2CAP channel can send an L2CA_DisconnectReq to request disconnection. When the L2CAP receives the L2CA_DisconnectReq, it causes an L2CAP_DisconnectReq packet to be sent across the Baseband link to the remote Bluetooth device. At the same time, L2CAP stops sending and receiving data on the channel. Any queues of data for transmission are emptied and any data received is just discarded.

The device receiving the L2CA_DisconnectReq discards all data queued for transmission on that channel, since the device that sent this request is discarding all data it receives anyway. The recipient replies with an L2CAP_DisconnectRsp. When the L2CAP that sent the L2CA_DisconnectReq receives the L2CAP_DisconnectRsp, it can inform the upper protocol layer of the result of its disconnect request. This is shown in Figure 4.15.

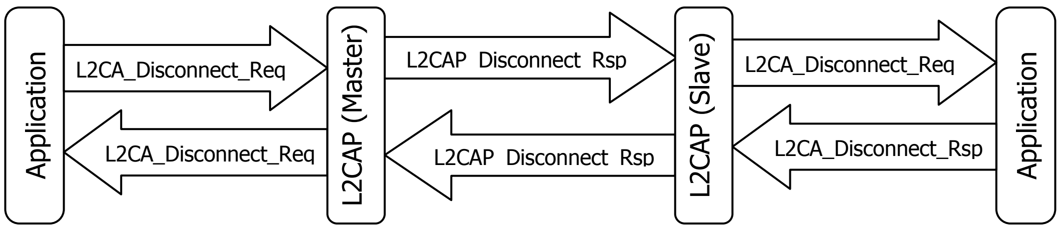


Figure 4.15 – L2CAP disconnection procedure

4.3.5.2 Link Manager Disconnection

At any time the master or slave that wishes to disconnect a link/connection, it can issue a detach command to Link Manager to send the LMP_detach message to the remote device. Three possible reasons for link shutdown are: - user ended connection, low resource, about to power off. Unlike other LMP message, the receiver (may be slave or master) does NOT need to acknowledge the reception of a LMP_detach message.

CHAPTER 5 Roaming for Bluetooth

One of the useful features of Bluetooth is its use as an interface for mobile devices to access network system. There is no mention about how roaming is going to be implemented in the Bluetooth Specification. However, two profiles in the Bluetooth Specification are defined to support network access - LAN Access Profile (LAP) and Personal Area Network (PAN).

Roaming is very important for *MEMSwear* application because the wearer would not want to be constrained by the short range of wireless communication channel provided by Bluetooth. However, this can be solved by implementing roaming for Bluetooth devices for use in *MEMSwear* application. Together with the Bluetooth simulator, *blue-ns*, the roaming mechanism can be implemented into the software. The simulator is a very useful tool in evaluating the performance of different roaming schemes suggested. In this section, the detailed implementation of roaming for Bluetooth is explained. The roaming implementation is incorporated into the *blue-ns* simulator for performance evaluation and analysis.

5.1 Needs for roaming

Accessing the network requires Access Points which behave like base stations in the cellular network. Bluetooth devices have limited range 10m to 100m. To be able to cover a large area, many access points need to be deployed in a similar fashion as cellular network. Most Bluetooth devices are personal device and therefore move around with their owner. When Bluetooth devices are attached to the Smart Shirt, the devices must be able to support high mobility. Thus, there is a definite need to support the mobility of the user to move freely from one access point to another without the need for user intervention and in a timely fashion. It must also be able to migrate network connections transparently.

5.2 Scenarios

Roaming is usually required only if the Bluetooth wants to be constantly connected to a backend network, which could be either the Internet or just a home network. Access Points are used as the gateway to the backend network. Thus, the Bluetooth would connect to the Access Point to gain access to the network. To be able to provide continuous quality services while the user is moving around at walking speed among different Access Points, an efficient roaming technique would become evidently necessary. The structure is illustrated in Figure 5.1.

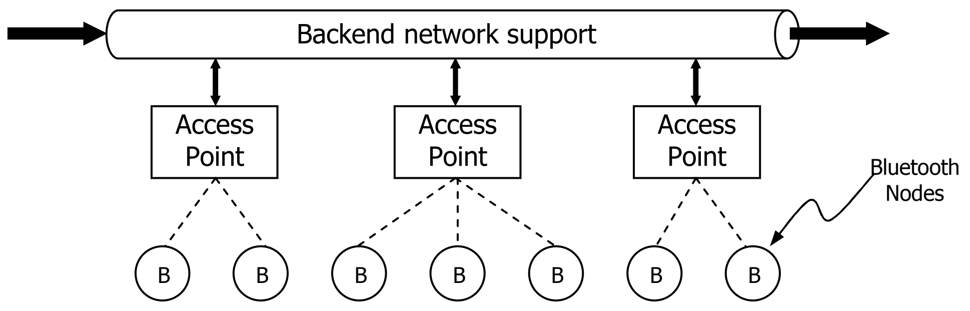


Figure 5.1 – Devices access to backend network through Access Points

5.2.1 Without roaming

A device is connected to an Access Point. Due to un-anticipated events such as power failure of either side or a device moving out of range, the Access Point has no way to determine whether the device is still alive. Bluetooth specified that if there is no response from other device longer than a supervision timer (default is 20 seconds), it could assume a disconnection from the other device.

In this type of unforeseen disconnection, there is no indication from the lower layer except the disconnection event itself. It will be too late to react upon receiving this event because the link has already been broken. The application has to start the whole process of inquiry, paging and connection again. More importantly, large data packet would have to be retransferred after reconnection is established. This results not only in resource wastage and may cause data loss during the disconnection. The reconnection could take a very long time as it needs to repeat the process of inquiry, paging and connection again.

5.2.2 With roaming

In situation where roaming is supported in Bluetooth, the user needs not do anything if it goes out of range, because the roaming support would automatically reconnect to any available access point. The device starts the procedure of inquiry, paging even before it gets disconnected (when the link quality goes below unacceptable level). Data flows to the backend network smoothly throughout the process without break. The transition to different access points is carried out smoothly (refer to Figure 5.2).

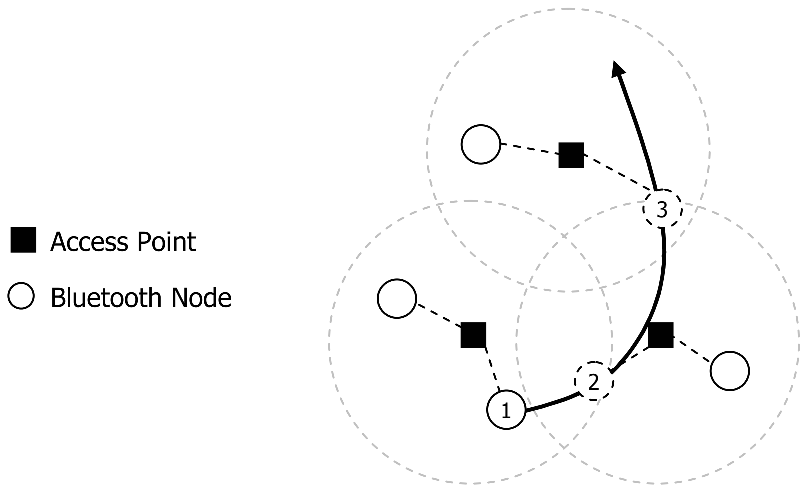


Figure 5.2 – A Bluetooth node is switching to different access point with roaming support

As the roaming process takes relatively shorter time delay compared to without roaming support, the data coming from the upper layer or application could be buffered at the roaming layer to prevent data loss during the roaming procedure.

5.3 Requirements

There are four requirements to be considered: -

- **Without user intervention**
  The users of a Bluetooth device with roaming support need not do anything when the device switches from one access point to another.
- **Short delay time (shorter than without roaming)**
  The time taken for switching access points should not be more than that is required as if it is done manually.
- **Easily fit on top of the stack**
  The proposed solution should be easily implemented on top of the existing Bluetooth protocol stack.
- **No data loss during the process**
  As the application does not realize that it has been disconnected from the network, it would continue to send data over the connection. Thus, roaming support needs to ensure that no data loss occurs during the roaming process.

5.4 Basic Implementation

It is proposed that all the roaming mechanisms are performed at an additional layer in the Bluetooth stack called Roaming layer, sitting on top of the L2CAP (see Figure 5.3). To maintain the compatibility with any upper layer above the L2CAP, this Roaming layer accepts all the L2CAP commands issued by the upper layer and send up L2CAP events as well. With this compatibility, there is no need to change any codes of existing applications.

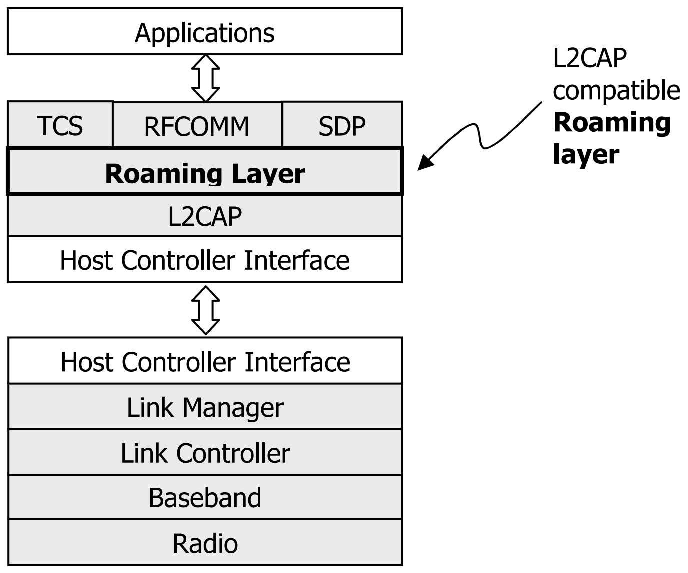

Applications
TCS
RFCOMM
SDP
Roaming Layer
L2CAP
Host Controller Interface
Host Controller Interface
Link Manager
Link Controller
Baseband
Radio
L2CAP compatible **Roaming layer**

Figure 5.3 – Addition of a Roaming layer in Bluetooth protocol stack

Assuming a Bluetooth device is connected with an Access Point and moves out of range. After some time longer than the supervision timeout, the Bluetooth at the Baseband level declares the link to be dead and closes the Baseband level connection. Because of this, the Baseband sends the disconnect event to all the upper layers, from the Link Controller to the L2CAP. When the L2CAP layer receives the disconnect event (L2CA_Disconnect_Ind), it sends this disconnect event up to the Roaming layer. When the Roaming receives this event, it does not send this event further up to the application layer. Instead, it keeps the higher layer in suspended mode and starts the roaming procedure until the procedure is completed.

Figure 5.4 shows the internal structure of the Roaming layer. It consists of a L2CAP commands and data buffer. As the application does not know that the connection has been broken it keeps sending command or data down as if it is still connected. The buffer is used to store all the L2CAP commands or data during the Roaming procedure. The Roaming layer also filters out event from the lower layer (see Figure

5.4). If it receives a disconnect event (L2CA_Disconnect_Ind), it starts the roaming procedure. Otherwise, it sends up all other events to the application layer.

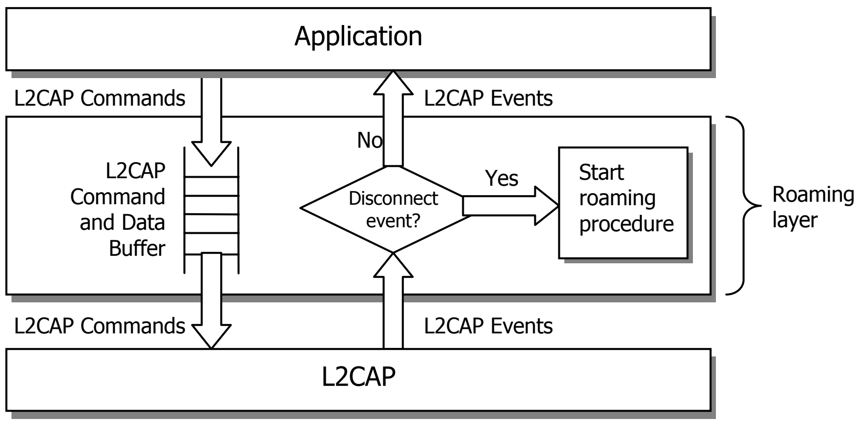


Figure 5.4 – Internal structure of Roaming layer

5.4.1 Roaming Process

The roaming procedure starts with the Bluetooth device performing inquiry process. If the inquiry does not find any Access Point, the node continues with the inquiry process, up to the point where it timeout and closes down the higher layer, which means that there is no Access Point nearby. If the Bluetooth device finds an Access Point, it connects to the Access Point. It registers to the Access Point and opens a L2CAP connection with this Access Point and then reconnects the previously suspended higher layer on top of this L2CAP connection.

In order to alleviate the impact of link loss during the roaming process, the Roaming layer has a buffer. It tries to prevent packet loss by buffering all packet recently sent from the upper layer. After reconnection is successful, the stored packets and newly arrived packets are sent down to the lower layer. Thus, there is no packet loss, if the buffer size is sufficient to store all incoming packets during the link loss. Hence, smaller buffer is required if the roaming delay is short and larger buffer if the roaming delay takes longer time.

5.4.2 Basic Roaming Simulation

The Bluetooth simulator, *blue-ns*, provides an environment for which the roaming mechanism can be implemented to further evaluate the performance of the proposed algorithm. The roaming simulation shows various situations how the nodes change from one access point to another as they wonder around the environment. Based on the output trace file, the roaming delay can be found easily.

5.4.2.1 Basic Setup

The simplest setup to observe the effect of roaming process involves one mobile Bluetooth node and two Access Points. This is shown in Figure 5.5 and Figure 5.6. The Bluetooth Node (N) is shown as a circle while the Access Point (AP) is shown as square. At first, the mobile node, N3 is connected to Access Point, AP2. As it moves away from AP2 towards AP3, it loses the connection with AP2. It then starts inquiry for any nearby Access Point. When the inquiry result shows that there is an Access Point (AP1) around, the node N3 then starts paging for AP1 and eventually connects to AP1. The application layer does not know that there has been a disconnection occurs at all. It is kept suspended by the roaming layer during the switching of Access Point. Once it has been reconnected to a new Access Point, the data from the application continues to transmit to the backend network through the new Access Point.

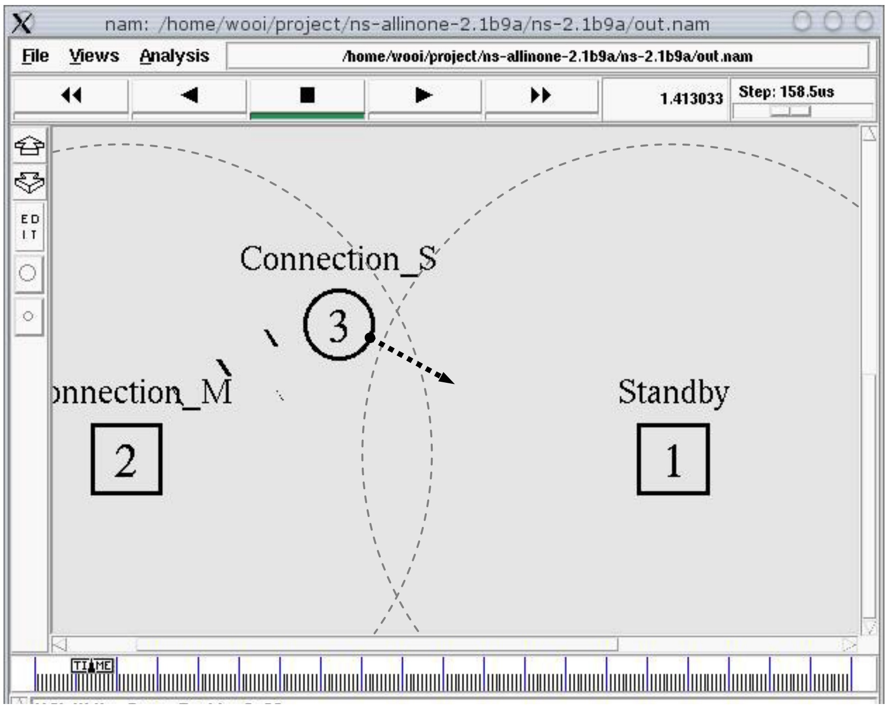
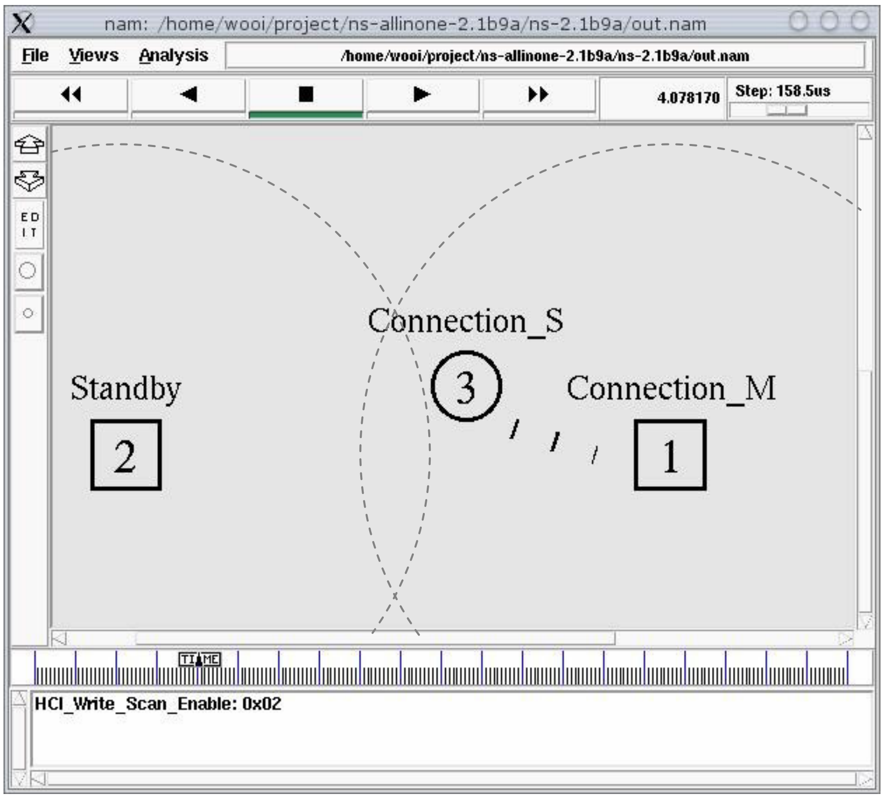

nam: /home/wooi/project/ns-allinone-2.1b9a/ns-2.1b9a/out.nam
File Views Analysis
/home/wooi/project/ns-allinone-2.1b9a/ns-2.1b9a/out.nam
1.413033
Step: 158.5us
Connection_S
3
onnection_M
2
Standby
1

Figure 5.5 – A Bluetooth node [3] connected to access point [2].


nam: /home/wooi/project/ns-allinone-2.1b9a/ns-2.1b9a/out.nam
File Views Analysis
/home/wooi/project/ns-allinone-2.1b9a/ns-2.1b9a/out.nam
4.078170
Step: 158.5us
Connection_S
Standby
3
Connection_M
2
1

```
HCI_Write_Scan_Enable: 0x02
```

Figure 5.6 – Node [3] switches to access point [1] as it leaves access point [2]

Based on this setup, several sets of simulation are run, each with different settings as follows: -

- The nodes are placed in different location randomly
- The nodes move in and out of the Access Points randomly at different time
- The Access Points are placed at fixed locations.

The results show that the algorithm for the basic implementation still incurs large amount of time upon switching Access Point.

5.4.3 Results and Analysis

Roaming is already something that other technology implements (for example, the 802.11 has strong support for roaming). However, the Bluetooth has a certain number of features making roaming more challenging and interesting. The result from the simulation shows that the roaming process takes up approximately **3.5 seconds** to complete which is unsatisfactory. Further analysis from the results of the roaming simulation gives a clearer picture of the causes for deficiency in the algorithm.

5.4.3.1 Discovery Latency

The whole discovery process takes a long time. The discovery process is defined as the steps required by the Bluetooth device to know the presence of a device, and consequently connect to it. Hence, it usually consists of 3 phases: -

1. Inquiry process
2. Paging process
3. Higher layer connection process

Inquiry process

The Inquiry process takes up the longest time. Inquiry involves synchronization of timing of both the inquirer and the inquiry scanner (see 4.3.1). The inquiry process does not guarantee that all devices that need to be discovered are discovered (because of fading or interferences). The timing is dependent of the inquiry time of the Bluetooth device and the inquiries scan time of the access point. For the access point, there is clearly a tradeoff between throughput and discovery latency. If the inquiry scanning is very frequent (more chances of being discovered by other devices), lesser time is allocated for data traffic. If high data traffic is required, less time is allocated for the inquiry scanning and thus, lesser chance of being discovered.

Paging process

The paging process is relatively fast. This is mainly because much information required for paging is acquired during the inquiry process. This process usually takes up only approximately 100ms.

Higher layer connection process

This process involves any layer higher than the Link Controller. This includes the Link Manager layer, the L2CAP layer and above. Based on the Baseband layer connection, the higher layer connection will take place, firstly the Link Manager connection followed by the L2CAP connection. If there is any application on top of the L2CAP, it needs to be connected also. These higher layer connections require the exchange of messages using the Baseband connection and it takes up a short time as well.

5.4.3.2 Connection for Identity

Unlike other wireless technology, Bluetooth Specifications does not include the device identity in the inquiry result. At the end of the inquiry process, the Bluetooth device merely has a list of devices in the vicinity. To know whether the found device is an access point, the inquirer needs to connect to the device to identify the identity of the device. Hence the device has to connect to all the devices that it finds during the inquiry process and find out whether the device is an access point and select the access point that gives the best connection quality.

5.4.3.3 Lack of RSSI Support

Received Signal Strength Indicator (RSSI) measures the power of the incoming radio signal. The problem with Bluetooth device is that the Bluetooth Specification does not explicitly specify that all Bluetooth devices need to provide RSSI support (though some manufacturers may include RSSI in their Bluetooth products). Implementing a reliable and sufficiently accurate RSSI in the radio is expensive. RSSI support is very important for roaming, because a node would like to connect to the access point that provides the strongest radio strength. Besides, RSSI also provides a way to determine whether the node should switch to other access point with stronger radio strength.

5.5 Improved Implementation

The basic scheme is very simple to implement but unfortunately not very efficient. The improved implementation attempts to address the problems of roaming for Bluetooth by proposing a set of optimizations to improve the performance of roaming in terms of shortening the roaming time.

5.5.1 Optimizations

The optimizations described in the following sections may deal with different aspects of the basic roaming mechanism. These optimizations may be used alone or can be combined to improve the overall performance and some implementations may actually depend on others.

5.5.1.1 Access Point to Access Point Communication

This optimization involves changing the wired structure of the Access Points. The Access Points in vicinity may be connected together with the wired backbone. There are two configurations where such communication can take place as shown in Figure 5.7 (as compared to Figure 5.1). One way is to provide wired connection between Access Points. Thus, the Access Point may communicate directly or indirectly (routed through other Access Point) with other Access Point. The other way is to connect all the Access Points to a Master Access Point. All communication (including the data traffic) will be routed through the Master Access Point.

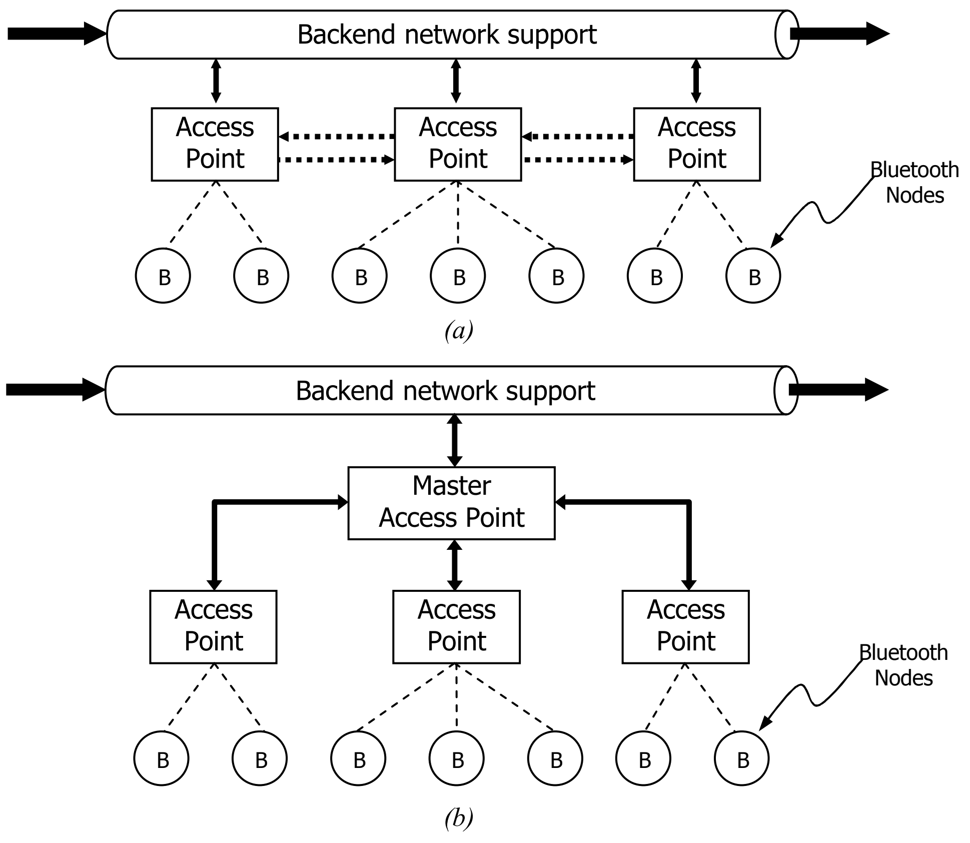


Figure 5.7 – Improved structure for Access Points (a) without master and (b) with master

In the wireless environment, because of interference and fading, a wireless link can be blocked for quite some time before data can flow normally again. For this reason, an Access Point assumes that a node is no longer connected to it only after a long timeout, to avoid disrupting a connection that is already potentially weak.

On the other hand, every node connected to an Access Point consumes some resources in that Access Point. The Access Point has to keep a record of every connected node, but the maximum number of connection allowed is only 7 nodes. It would be wasteful to keep a record for a node that has already roamed out the Access Point. When the node starts the roaming process, it does not know if it will be successful and thus, it does not terminate its existing connection with the Access Point (in case it needs to come back). In most cases, the node will have already lost contact with the Access Point when starting the roaming procedure. Therefore, the node does not inform the Access Point when it is roaming away.

Communication between Access Points becomes important because once the node is connected to the new Access Point, the new Access Point can broadcast this information to other Access Points on the wired backbone, so that the old Access Point can learn about it and flush all the resources associated with that node. This optimization will result in significant resource saving in the Access Point and thus increasing the number of nodes per Access Point. The advantage of having the Access Points to communicate with each other is to flush the resource that were used by the roaming node in the old Access Point.

5.5.1.2 Neighboring Access Point List

When the node needs to roam, it has no way to identify which Access Point in its vicinity offers the same service. As explained in Section 5.4.3.2, the node needs to connect to the device found from inquiry process to determine whether it is an Access Point.

To solve this problem, the current Access Point could send to the node a list of Access Points in the vicinity that offer the same service (including the timing information related to inquiry and paging). When the node performs the basic handover procedure, it simply compares the result of the inquiry with the list of Access Point and picks an Access Point. This scheme would save all the trials and errors to find a right Access Point.

The list consists of two levels down the hierarchical tree of neighboring Access Point. In other words, the Access Point would send the list of its immediate neighbors and for each neighbor, the list of its immediate neighbors. Having neighboring Access Point one hop and two hops away should be good enough to cover most roaming speed and environment.

5.5.1.3 Access Point Probing

Together with the list of neighboring Access Points, the timing information, such as the inquiry and paging timing, for each Access Point is also available. As the roaming node has all the necessary information, it can probe the Access Point only when it knows it is performing inquiry and paging and thus can find it very quickly.

By doing some regular probing of the various Access Points synchronized to the inquiry and paging patterns (i.e. probing at the right time), the node can determine which Access Point it can reach and which it cannot with minimal overhead. The advantage of this scheme is that the node does not need to perform a full inquiry and can connect directly with the Access Point it desires. In other words, with the information from the Access Point list, the node is able to perform continuously some targeted fast inquiries, so that a full inquiry does not need to be performed when disconnected.

5.5.2 Simulation Results

The suggested improvement is implemented and simulated using the *blue-ns*. The result shows significant improvement. The roaming delay time reduces from 3.5 seconds to merely 200 milliseconds. This is mainly because the implementation of these optimizations has eliminated the inquiry process which has taken up most of the time required for roaming. The node knows beforehand which Access Point that it wants to connect from the list of Access Point sent by the currently connected Access Point Thus, it does not need to perform the most time-consuming inquiry process.

CHAPTER 6 Conclusions

MEMSwear is a vital signs monitoring wearable system for elderly at home. The vital signs are collected from various sensors attached to the Smart-shirt. To enhance the convenience and comfortability of the wearer, the data is then transmitted to a desktop computer wirelessly using Bluetooth wireless technology.

The main advantages of Bluetooth technology are its low cost, small size and low power features. However, the low power consumption has limited its transmission to only 10 meters. Together with interference from the surrounding, it may have even shorter range. This is insufficient for normal use at home.

It is thus proposed to implement roaming for Bluetooth for this application. In evaluating the performance and feasibility of Bluetooth wireless technology for *MEMSwear*, tests and analysis are simulated first in a simulation environment. Based on the current Bluetooth simulation (*BlueHoc*) extension of the Network Simulator 2 (*ns-2*), a new and improved Bluetooth simulator (*blue-ns*) is developed. Compared with the *BlueHoc*, the blue-ns provides support for dynamics node movement and graphical simulation output (these features are not available in *BlueHoc*).

By further extending the *blue-ns* to include the proposed roaming support, the performance can be evaluated without using expensive hardware implementation. It is observed that the basic roaming algorithm is not optimised. Hence, it is proposed to polish the roaming algorithm by adding access point optimisations to address the shortcomings of the basic algorithm. It is shown that the performance and reliability of the optimised algorithm for roaming has greatly improved.

6.6 Recommendations and Future Works

The *blue-ns* simulator has opened up more opportunities of research for both the developers and users Bluetooth technology. However, there are some features that are not included which the author wished to implement in the current version of *blue-ns* due to time constraints.

The *blue-ns* does not incorporate all the features specified in the Bluetooth Specifications. Only specific set of features of Bluetooth commands and events are implemented which is more than sufficient to achieve the objective. Thus, it is recommended to further enhance the *blue-ns* simulator by adding more features specified in the Bluetooth Specifications.

The roaming scheme for Bluetooth suggested in this thesis is only simulated by the blue-ns. It has not yet been tested using actual Bluetooth modules. The performance of the roaming scheme under actual environment may not tally with the simulation result. This may be due to factors such as noise and interference from other electronics devices. Thus, it is important to test and further refine the scheme before it is used for final implementation.

In short, all the objectives of this project have been achieved. A new and improved Bluetooth simulator, *blue-ns*, has been developed. A roaming mechanism for

Bluetooth is proposed and simulated under the *blue-ns* environment. The performance of the proposed roaming mechanism has been found satisfactory based on the result of the simulation.

Bibliography

[1] United Nations Department of Economic and Social Affairs (Population Division), The Sex and age distribution of the world populations (The 1996 Revision), 1996.

[2] The Georgia Tech Wearable Motherboard™: The Intelligent Garment for the 21st Century, http://www.gtwm.gatech.edu.

[3] S. Park, K. Mackenzie, S. Jayaraman, The Wearable Motherboard: A Framework for Personalized Mobile Information Processing (PMIP), Proceedings of the 39th Design Automation Conference, June 10-14, 2002.

[4] Bluetooth Special Interest Group, Specification of the Bluetooth System 1.1, Volume 1: Core Specification, http://www.bluetooth.com, Dec 2000.

[5] J. Haartsen, " The Bluetooth Radio System", IEEE Personal Communications, Vol. 7, No. 1, pp.28-36, Feb 2000.

[6] Bluetooth Special Interest Group, Specification of the Bluetooth System 1.1, Volume 2: Profiles Definitions, http://www.bluetooth.com, Dec 2000.

[7] A. Capone, M. Gerla, and R. Kapoor, Efficient polling schemes for Bluetooth picocells, Proceedings of IEEE International Conference on Communications, volume 7, pages 1990–1994, June 2001.

[8] A. Das, A. Ghose, A. Razdan, H. Saran, and R. Shorey, Enhancing performance of asynchronous data traffic over the Bluetooth wireless ad-hoc network, Proceedings Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies IEEE INFOCOM 2001, volume 1, pages 591–600, Apr. 2001.

[9] C. de Morais Cordeiro, D. Sadok, and D. P. Agrawal, Piconet interference modeling and performance evaluation of Bluetooth MAC protocol, Proceedings Global Telecommunications Conference 2001 GLOBECOM '01, volume 5, pages 2870–2874, Nov. 2001.

[10] S. Garg, M. Kalia, and R. Shorey, MAC scheduling policies for power optimization in Bluetooth: a master driven TDD wireless system, Proceedings VTC2000-Spring IEEE 51st Vehicular Technology Conference, volume 1, pages 196–200, May 2000.

[11] N. Golmie, R. E. Van Dyck, and A. Soltanian, Interference of Bluetooth and IEEE 802.11: simulation modeling and performance evaluation, Proceedings 4th ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems, pages 11–18, July 2001.

[12] N. Johansson, U. Korner, and P. Johansson, Performance evaluation of scheduling algorithms for Bluetooth, Proceedings of BC'99 IFIP TC 6 Fifth International Conference on Broadband Communications, pages 139–150, Nov. 1999.

[13] E. Kail, G. Nemeth, and Z. R. Turanyi, Throughput of ideality routed wireless ad hoc networks, Proceedings 2001 ACM International Symposium on Mobile ad hoc networking & computing, pages 271–274, Oct. 2001.

[14] M. Kalia, D. Bansal, and R. Shorey, MAC scheduling and SAR policies for Bluetooth: A master driven TDD pico-cellular wireless system, Proceedings Sixth IEEE International Workshop on Mobile Multimedia Communications (MOMUC'99), pages 384–388, Nov. 1999.

[15] S. Zurbes, Considerations on link and system throughput of Bluetooth networks, Proceedings of the 11th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications PIMRC 2000, volume 2, pages 1315–1319, Sept. 2000.

[16] P. Bhagwat and A. Segall, A routing vector method (RVM) for routing in Bluetooth scatternets, Proceedings Sixth IEEE International Workshop on Mobile Multimedia Communications (MOMUC'99), pages 375–379, Nov. 1999.

[17] C.-C. Chiang and M. Gerla, Routing and multicast in multihop, mobile wireless networks, Proceedings 1997 IEEE 6th International Conference on Universal Personal Communications, volume 2, pages 546–551, Oct. 1997.

[18] A. Kumar, L. Ramachandran, and R. Shorey, Performance of network formation and scheduling algorithms in the Bluetooth wireless ad-hoc network, Journal of High Speed Networks, 10:59–76, Oct. 2001.

[19] C. Law, A. K. Mehta, and K.-Y. Siu, Performance of a new Bluetooth scatternet formation protocol, Proceedings 2001 ACM International Symposium on Mobile ad hoc networking & computing, pages 183–192, Oct. 2001.

[20] C. Law and K.-Y. Siu, A Bluetooth scatternet formation algorithm, Proceedings Global Telecommunications Conference 2001 GLOBECOM '01, volume 5, pages 2864–2869, Nov. 2001.

[21] A. Racz, G. Miklos, F. Kubinszky, and A. Valko, A pseudo random coordinated scheduling algorithm for Bluetooth scatternets, Proceedings 2001 ACM International Symposium on Mobile ad hoc networking & computing, pages 193–203, Oct. 2001.

[22] G. V. Zaruba, S. Basagni, and I. Chlamtac, Bluetrees – scatternet formation to enable Bluetooth based ad hoc networks, Proceedings of IEEE International Conference on Communications, volume 1, pages 273–277, June 2001.

[23] G. Aggelou and R. Tafazolli, RDMAR: A Bandwidth-efficient Routing Protocol for Mobile Ad Hoc Networks, Proceeding 2nd ACM International Workshop on Wireless Mobile Multimedia Workshop, Sept. 1999.

[24] Z. J. Haas, A New Routing Protocol for the Reconfigurable Wireless Networks, Proceeding ICUPC'97, Oct. 1997.

[25] J. J. Garcia-Luna-Aceves and M. Spohn, Source Tree Adaptive Routing in Wireless Networks, Proceeding IEEE ICNP'99, Sept. 1999.

[26] V. Park and S. Corson, A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks, Proceeding IEEE INFOCOM, Apr. 1997.

[27] D. J. Y. Lee and W. C. Y. Lee, Ricocheting Bluetooth, Proceeding IEEE 2nd International Conference Microwave and Milimeter Wave Technology 2000, pages 432–435, 2000.

[28] D. Lee and W. Lee, Integrating Bluetooth with Wireless and Ricocheting, Proceeding 11th IEEE PIMRC 2000, volume 2, pages 1310–1314, 2000.

[29] M. Albrecht, M. Frank, P. Martini, M. Schetelig, A. Vilavaara, A. Wenzel, IP Services over Bluetooth: Leading the Way to a New Mobility, Proceedings of the 24th IEEE Conference on Local Computer Networks, pages 2–11, Oct. 1999.

[30] S. Baatz, M. Frank, R. Gopffarth, D. Kassatkine, P. Martini, M. Schetelig, A. Vilavaara, Handoff Support for Mobility with IP over Bluetooth, Proceedings of the 25th IEEE Conference on Local Computer Networks, pages 143–154, Nov. 2000.

[31] A. Kansal and U. B. Desai, Mobility Support for Bluetooth Public Access, IEEE, International Symposium on Circuits and System (ISCAS), volume 5, pages 725–728, May 2002.

[32] ns-2 Network Simulator, http://www.isi.edu/vint/nsnam/.

[33] A. Kumar, BlueHoc - Bluetooth Ad-hoc Network Simulator, IBM, http://www-124.ibm.com/developerworks/opensource/bluehoc/.