This is to certify that the
thesis entitled

BTAUDIO (BLUETOOTH AUDIO PROGRAM) AND QUITE TALK PROFILE

presented by

JUN CHEN

has been accepted towards fulfillment
of the requirements for the

M.S. degree in Computer Science and Engineering

Major Professor's Signature

12/11/03

Date

BTAUDIO (BLUETOOTH AUDIO PROGRAM) AND QUITE TALK PROFILE

By

Jun Chen

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Computer Science and Engineering

2003

Abstract

BTAudio (Bluetooth Audio Program) and Quite Talk Profile

By

Jun Chen

In this thesis, we discuss the Bluetooth audio gateway design and implementation. We implement the audio gateway into a program named BTAudio. BTAudio is developed on the top of BlueZ (Official Linux Bluetooth protocol Stack) to fulfill its core functions. BTAudio also provides a graphical user interface written in GTK+.

By running BTAudio, the computer can control a Bluetooth device, which is attached to it. Then the Bluetooth device can act as the audio gateway. The audio gateway has these main functions: build connection, receive or send audio data from or to headset and release connection.

We also conducted some experiments on the Bluetooth devices by using BTAudio. From the experiments we arrived at some important results about Bluetooth technology.

In the end, we propose a new Bluetooth profile: Quite Talk Profile. This profile describes how the Bluetooth devices can provide a mobile, full duplex, and hand-free way for up to three people to communicate.

Acknowledgments

It is not easy that I finish this thesis. Here I want to thank all the people who have helped me in this work.

First, I am very grateful to my advisor: Dr. Lionel M. Ni. He helps me finish this very interesting thesis research and always gives me direction when I feel lost in my thesis research. I also acknowledge Dr. Abdol-H. Esfahanian and Dr. Matt W. Mutka. Their inspiring comments are very important to my thesis research.

I also want to thank Yunhao Liu, Hongbo Zhou, Abhishek P. Patil, Pei Zheng and other colleagues in ELANS. Their advice helps me overcome many obstacles.

At last many thanks to my family, especially my husband: yuan. He gives me great support in spirit.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

1. Introduction

1.1 Motivation

As a new technology with a history of nine years, Bluetooth [1] wireless technology has rapidly gained a lot of consumer awareness and product penetration across a range of diverse industries. More and more manufacturers plan to launch products using Bluetooth technology.

Bluetooth is the new technology using short-range frequency-hopping radio link between mobile computers, mobile phones, PDA, headset and other portable devices. Originally Bluetooth is brought out as a cable-replacement technology. After plugging a small, cheap radio chip into computers, printers, keyboards, etc, people are set free from the cable entanglement. The interest in Bluetooth is soaring because of its key features: robustness, low complexity, and low power. Later on people think about a lot of idea with using Bluetooth, such as building a dial-up networking on the laptop via a cellular phone, sending files from a PDA to a laptop, sending music from a computer to a headset and so on.

Some applications for Bluetooth are a carrier of audio data. In theory, up to three full-duplex audio channels can be provided in one Bluetooth chip. The audio quality provided by Bluetooth shall be same as that of a cellular telephone because Bluetooth uses the same audio data format as the GSM (Global System for Mobile Communication) system.

Bluetooth manufacturers produce a very nice Bluetooth device: Bluetooth headset. It is lightweight, small and convenient to use. Normally it weighs less than an ounce. We

can hang the headset on our ear to receive audio data. After a while, we will forget there is a headset on our ear. This Bluetooth device set people free from the wired headset.

The common headset usage is to connect to cell telephone. But there is one important potential function of the headset: connecting headset to Bluetooth enabled computer or PDA. With this function, we can enjoy the music on the headset without sitting close to the computer or PDA. We can hear music without carrying anything except the lightweight headset. How to accomplish this function is a big step in Bluetooth technology.

1.2 Objective

In this thesis we plan to design and implement the Bluetooth audio gateway in a computer to accomplish this function. Because the headset is very passive, it only has the button as the input. Thus the major work has to be on the computer side. The major function of the audio gateway is to build connection, receive or send audio data from or to headset and release connection.

After accomplishing the implementation, we want to perform some experiments on the Bluetooth devices to discover some properties of the Bluetooth technology.

Furthermore, We want to propose a new Bluetooth profile. If we can transmit audio data between the headset and audio gateway, we can let the two headsets communicate with help of the audio gateway. Then we can provide a new short-range, mobile, full duplex and hand-free way for communication using Bluetooth technology.

1.3 Organization

Chapter 2 covers Bluetooth specifications, which gives basic information about the Bluetooth technology. Chapter 3 talks about the Bluetooth profiles. Chapter 4 discusses the audio gateway design. Chapter 5 is mainly about how to implement the audio gateway into a program named BTAudio. Chapter 6 describes the experiments we conduct on the Bluetooth devices using BTAudio and gives the results. Chapter 7 proposes a new Bluetooth profile: Quite Talk Profile. Chapter 8 summarizes all the works carried out in this thesis research and lists the future work.

2. BLUETOOTH SPECIFICATION

2.1 Overview

Bluetooth is brought out as a replacement of the cable by Ericsson Mobile Communications in 1994. In 1998 Ericsson Mobile Communications, Intel Corp., IBM Corp., Toshiba Corp. and Nokia Mobile Phones formed the Bluetooth Special Interest Group (SIG). SIG is a group of companies that work together to promote and define the Bluetooth specification. Version 1.0 of the Bluetooth specifications came out in July 1999. The name of Bluetooth comes from a tenth-century Danish king who united Denmark and Norway. They chose this name because they expect Bluetooth to unify the telecommunications and computing industries.

A critical feature of the Bluetooth specification is that it aims to allow devices from lots of different manufacturers to work with each other. For this objective, Bluetooth does

not only define the radio system; it also defines a software stack to enable applications to find the services that can be provided in other devices.

The Bluetooth specification is made up of two parts: core specification and profiles. We also call the first part as Bluetooth specification, which is mainly about the layers in the Bluetooth stack. The second part: Bluetooth profile gives details about how applications use the Bluetooth protocol stack. We will talk about Bluetooth specification in this chapter. The Bluetooth profiles will be covered in next chapter.

2.2 Bluetooth Protocol Stack

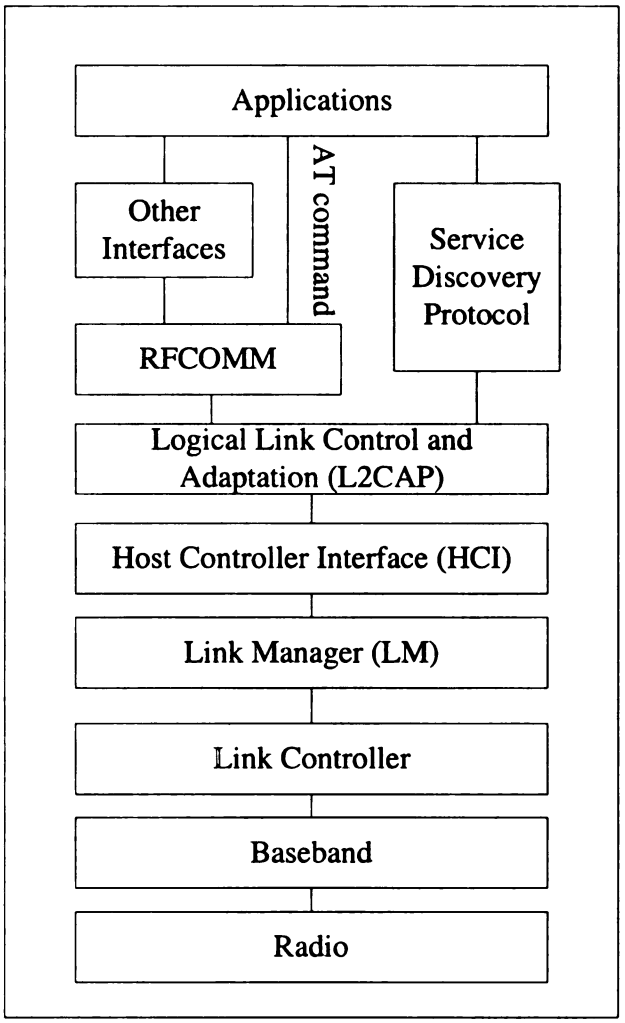

Figure 1: Bluetooth Specification Protocol Stack

Figure 1 illustrates the Bluetooth specification protocol stack. It is made up of eight layers: radio, baseband, link controller, link manager, host controller interface, logical link control and adaptation, RFCOMM or service discovery protocol, and applications.

The comparison of the Bluetooth protocol stack and OSI (Open System Interconnect) Standard reference model is displayed in Figure 2. Although there is no perfect mapping

between them, it helps people to understand the Bluetooth protocol stack. The functionality of each layer in Bluetooth protocol stack will be explained later.

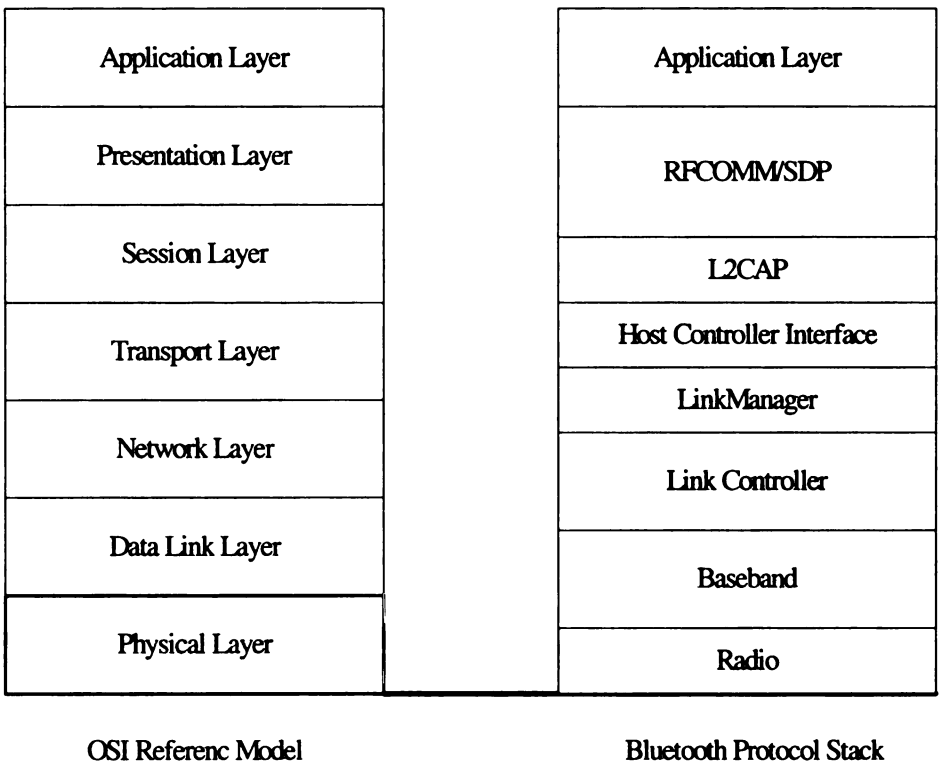

Figure 2: OSI Reference Model and Bluetooth Protocol Stack

2.3 Radio

Bluetooth Radio operates in the 2.4 GHz ISM (Industrial Scientific Medicine) band. In most of the countries around the world Bluetooth uses a frequency hop technology with 79 hops displaced by 1MHz, starting at 2,400 MHz and stopping at 2,483.5 MHz. In some countries like France, the frequency range is 2,446.5- 2,483.5 MHz and Bluetooth uses a frequency hop technology with 23 hops.

Each Bluetooth has an antenna to send out the radio signal. The Bluetooth equipment is classified into three power classes by the power levels at the antenna connector. Power

Class 1 is the equipment with maximum output power of about 100mW, which normally can communicate for a maximum of around 100 meters. Power Class 2 is the equipment with maximum output power of about 2.5mW, which may cover a range of 30 meters. Power Class 3 is the one with maximum output power of about 1mW, which can communicate within 10 meters area.

2.4 Baseband

The baseband manages physical channels and links. It also handles packets, error correction, pages and inquiries to access the Bluetooth devices within its area.

The channel in baseband is represented by a pseudo-random hopping sequence hopping through the 79 or 23 channels. The Bluetooth device takes 1600 hops in one second. The channel is divided into 625 us time slots. Figure 3 shows one frequency-hopping example. At the first time slot it uses channel 78. It jumps to channel 77 in the second time slot.

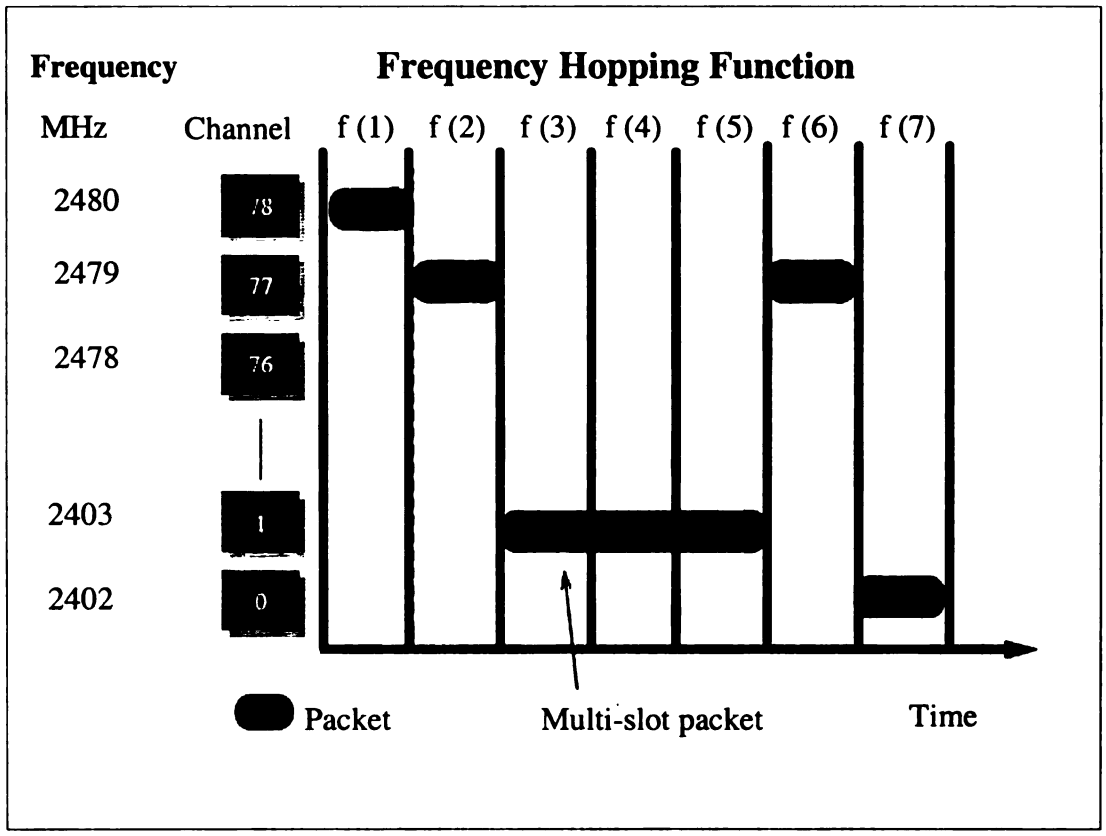

Figure 3: Frequency Hopping Example

Two or more Bluetooth devices can form a piconet by using the same frequency hopping sequence. In each piconet there is one master and one or more slave(s). The hopping sequence unique for the piconet is determined by the Bluetooth address (BD_ADDR) of the master. The phase in the hopping sequence is determined by the Bluetooth clock of the master. A scatternet is the multiple piconets with overlapping Bluetooth device, that is more than one device joins more than one piconet. But each piconet has a different master. One Bluetooth device cannot act as a master of two piconets. If one device acts as the master of two piconets, then the hopping sequence will be same for the two piconets. Then it will turn the scatternet into a piconet.

Figure 4 shows the topology of a point-to-point link between Master and Slave. The example of two piconets are displayed in the diagram (a) and (b). In diagram (a) there is

one master and one slave. In diagram (b) there is one master and two slaves. The diagram (c) gives one example of scatternet: the master of one piconet becomes a slave of another piconet.

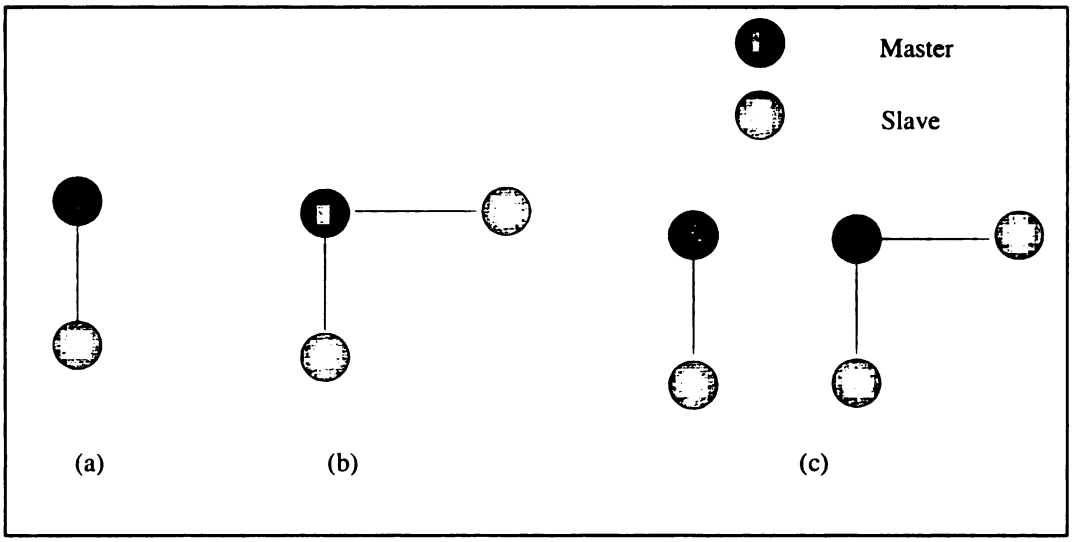

Figure 4: Piconets with a Single Slave (a), Multiple Slaves (b) and a Scatternet (c)

The baseband manages two types of links: Synchronous Connection-Oriented (SCO) link and Asynchronous Connection-Less (ACL) link. The ACL link is a point-to-multipoint link between the master and all the slaves in the piconet. The SCO link is a point-to-point link between a master and a single slave participating the piconet. The master maintains the SCO links by using reserved slots at regular intervals. When the slots are not reserved for the SCO link, the master can establish an ACL link on a per-slot basis to any slave, including the slave(s) already engaged in an SCO link.

An ACL link exists between the master and slaves as soon as a connection has been established. The ACL link provides a packet-switched connection where data is exchanged sporadically when data is available. The choice of which slave to send out to or receive from is up to the master on a slot-by-slot basis.

A SCO link provides a circuit-switched connection between the master and a slave with reserved channel bandwidth and regular periodic exchange of data in the form of reserved slots. The SCO link is mainly used to transmit time-bounded information such as audio data.

There are 13 different packet types defined for the baseband. Some are for both SCO and ACL links. Some are for ACL link only. Some are only for SCO link. Most ACL packets perform error checking and retransmission to assure data integrity. The SCO packets are never retransmitted. Baseband provides the following error correction and detection for the ACL packets: FEC (Forward Error Correction), CRC (Cyclic Redundancy Checksum) and ARQ (Automatic Repeat Request) scheme.

In baseband, five logical channels are defined to transfer different types of information. LC (Link Control) channel and LM (Link Manager) channel are used at the link control level and link manager level. The user channels: UA (User Asynchronous Data), UI (User Isochronous Data), US (User Synchronous Data) channels are used to carry asynchronous, isochronous, and synchronous user information, respectively.

2.5 Link Controller

The link controller layer performs higher-level operation such as inquiry and paging and manages multiple links between different devices.

For Bluetooth devices in link controller layer, there are two major states: STANDBY and CONNECTION and seven substates: page, page scan, inquiry, inquiry scan, master response, slave response and inquiry response. The substates are interim states that are for the devices to stay while they join a piconet. Either commands from the Bluetooth

link manager or internal signals in the link controller can make the devices move from one state to the other. Figure 5 illustrates the movement between these states.

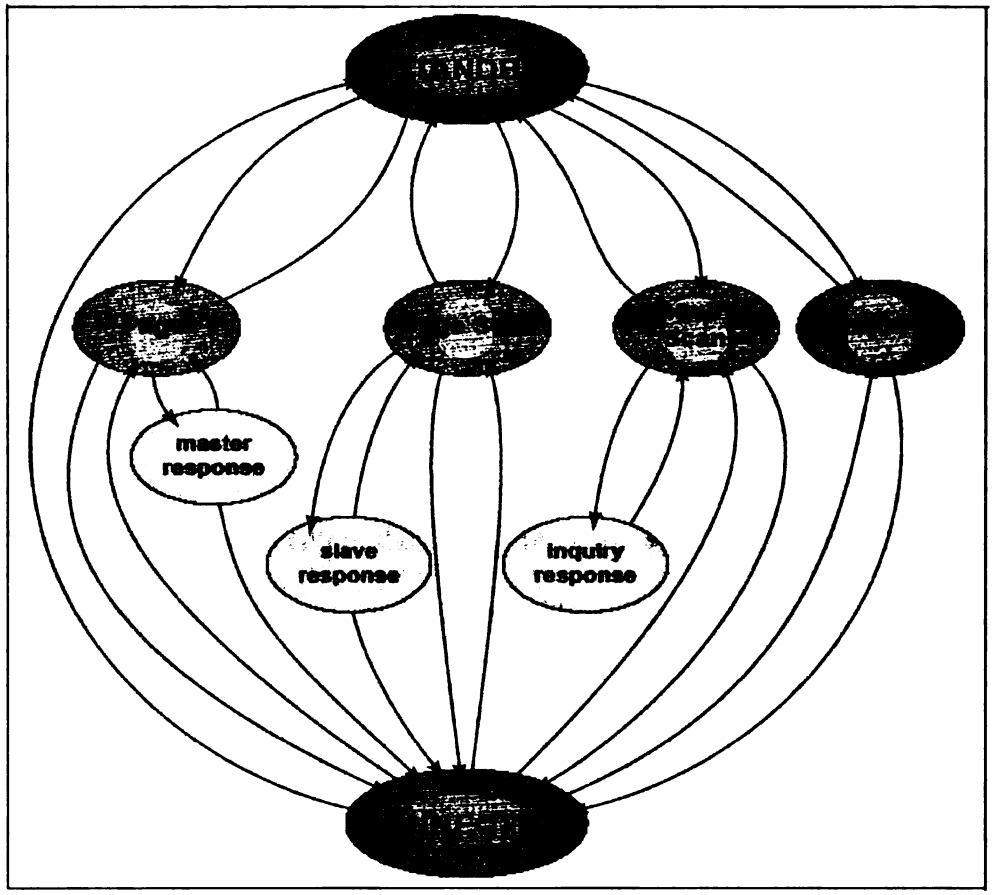

Figure 5: State Diagram of Bluetooth Link Controller [1]

In order to establish new connection between unknown devices, the inquiry procedures and access procedures must be performed sequentially. If a device knows the destination device's address, only access procedure is needed.

The inquiry procedure enables a discovering unit to collect the Bluetooth device addresses and clock of all devices that respond to the inquiry message. A normal inquiry procedure is carried out in the below way:

1) The source device enters the inquiry state and broadcasts inquiry packets.

2) The destination device in inquiry scan state receives the inquiry packets.

3) The destination device will then enter inquiry response state and send an inquiry reply packet containing its address and clock to the source device.

During the access procedure, important information: the channel access code and the channel hopping sequence are exchanged between each other. And their clocks are synchronized. Typically the access procedure occurs the following:

1) The source device enters page state to page another device.
2) The destination device in the page scan state receives the page packets.
3) The destination device enters slave response state and sends a reply to the source.
4) The source device enters the master response state and sends packets contains important information such channel hopping sequence to the destination device.
5) The destination device sends it's second reply packet to the source.
6) The destination and source devices then follow the source channel parameters.
7) The connection state starts when a POLL packet sent by the source device to verify the destination device has switched to the source's timing and channel frequency hopping sequence.

2.6 Link Manager

The major task of Link Manager is to translate the Host Controller Interface commands into operations at the baseband level. A Bluetooth Link Manager communicates with Link Manager on other Bluetooth device by using the Link Management Protocol messages. Link Manager provides the following functions:

attaching slaves to a piconet, configuring the link including controlling Master/Slave switches, setting up ACL and SCO links, and so on.

2.7 Host Controller Interface (HCI)

The HCI provides a uniform interface method of accessing the hardware capabilities of Bluetooth devices. The HCI is made up of three parts: the HCI firmware driver, HCI transport layer and HCI driver.

The HCI firmware carries out the HCI commands to the Bluetooth hardware by accessing baseband commands, link manager commands, hardware status registers, control registers and event registers. The HCI driver on the host transmits data, packets and commands with the HCI firmware on the Bluetooth hardware. The HCI transport layer exists between the HCI driver on the host system and the HCI firmware in the Bluetooth hardware to provide the ability to transfer data without intimate knowledge of the data.

There are three HCI transport layers: USB (Universal Serial Bus), RS-232 (a serial interface with error correction) and UART (Universal Asynchronous Receiver Transmitter, a serial interface without error correction). The Figure 6 illustrates the path of data exchange between two Bluetooth device enabled hosts.

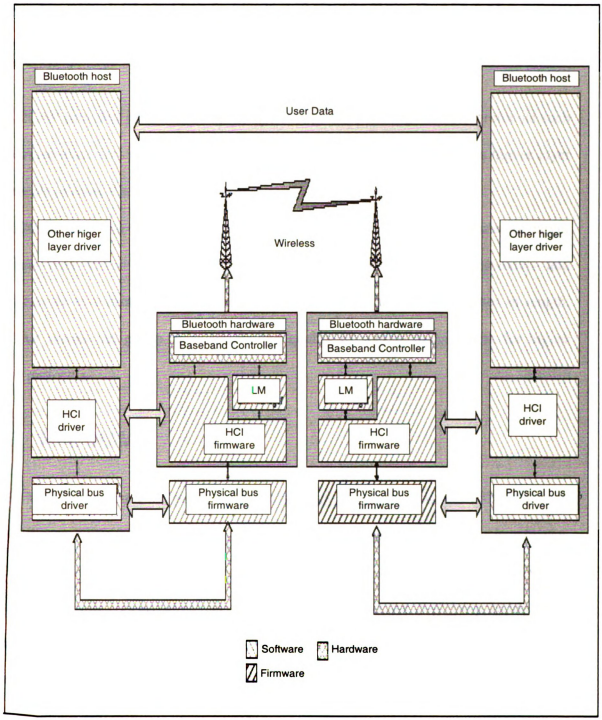

Figure 6: End-to-End Overview of Lower Software Layers to Transfer Data [1]

2.8 Logical Link Control and Adaptation Protocol (L2CAP)

L2CAP provides protocol multiplexing between different higher layer protocols such as service discovery protocol and RFCOMM. Protocol multiplexing allows higher layer protocols to share the lower layer links.

L2CAP performs segmentation and reassembly operation to allow transfer of larger packets than lower layers can support. L2CAP allows higher-level protocols and applications to transmit and receive L2CAP data packets up to 64 kilobytes in length. L2CAP carries out group management and quality of service management for higher layer protocols

2.9 RFCOMM

RFCOMM emulates the serial cable line settings and status of an RS-232 serial port. To build an RFCOMM connection, an L2CAP connection must be set up in advance. RFCOMM frames are sent in the payload field of the L2CAP packets. RFCOMM provides multiple concurrent connections by replying on L2CAP to perform multiplexing over single connections, and to provide connection to several devices. RFCOMM depends on the baseband to provide reliable in-sequence delivery of byte streams for it does not have any ability to correct errors.

RFCOMM supports two types of devices. A type 1 device is the end of a communications path and supports an application over RFCOMM.A type 2 device is an intermediate device and has a physical RS-232 serial port over RFCOMM.

2.10 Service Discovery Protocol (SDP)

SDP resides on the top of L2CAP levels. SDP employs a Client-Server interaction mechanism as shown is Figure 7. This mechanism provides a way for client applications on one Bluetooth device to discover the existence of services offered by server application in another device as well as the attributes of those services.

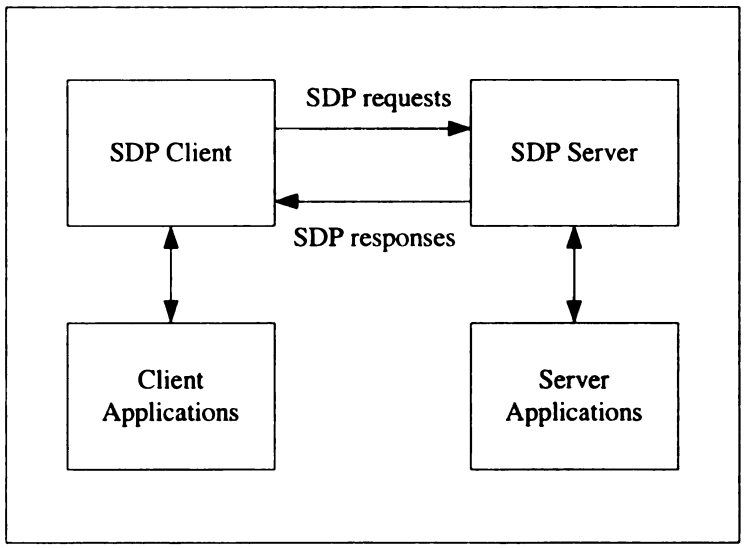

Figure 7: SDP Client-Server Interaction Mechanism

The SDP server maintains a database of service records. Each service record describes he characteristics and contains information about a single service. A SDP client can get the information about a service record maintained by the SDP server by issuing an SDP request. If the client wants to use a service, it must open a separate connection to the service provider to utilize the service. Although SDP provides a mechanism for discovering services and their attributes, it does not provide a mechanism to use those services.

3.B

3.1

The

standar

produc

each o

by pro

the lo

Each

Gene

3. BLUETOOTH PROFILE

3.1 Bluetooth Profile Relationships

The purpose of a profile is to give a clear description of how a full specification of a standard system to implement a given function. If everyone implement the function into products as describe in the profile, then each product shall be able to interoperate with each other. Bluetooth profiles have the same functionality. They ensure interoperability by providing a well-defined set of higher layer procedures and uniform means of using the lower layers. Figure 8 shows how the Bluetooth profiles are built up in layers [5]. Each profile relies upon the layers below. For example, Headset Profile depends on Generic Access Profile and Serial Port Profile.

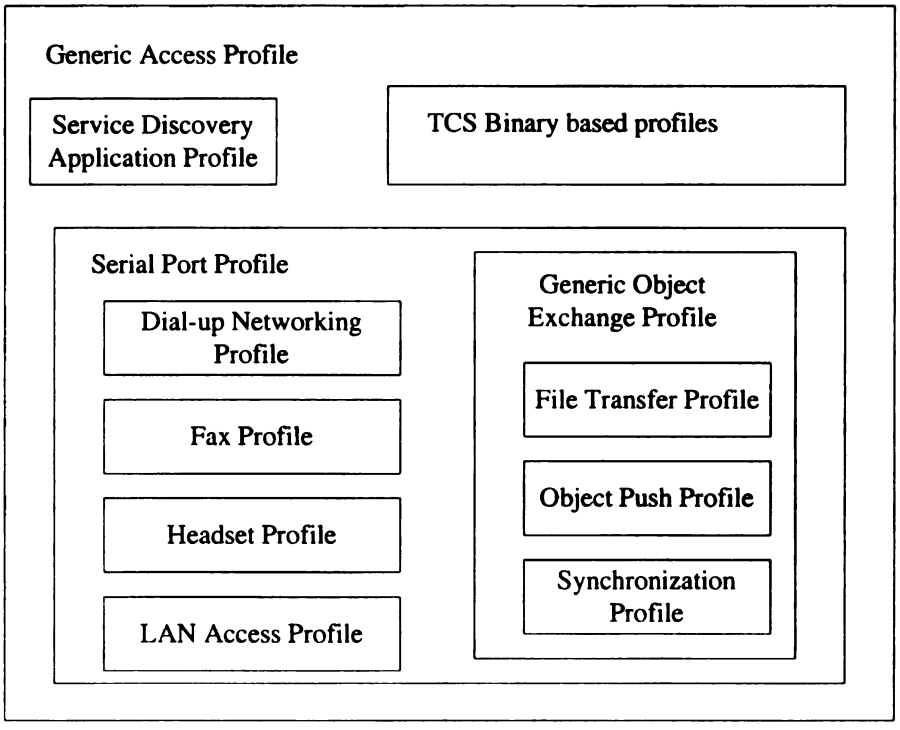

Figure 8: Bluetooth Profile

In this chapter we only briefly talked about Generic Access Profile (GAP) and Serial Port Profile. Headset profile will be covered in next chapter. For detailed information about other profiles, please look at the profile part of specification of the Bluetooth technology.

3.2 Generic Access Profile (GAP)

The GAP forms a common basis for all other Bluetooth profiles. The objective of the GAP is to make sure that all devices can successfully establish a baseband link. To obtain this objective, the GAP defines the requirements of the features which must be implemented in all devices, general procedure to discover Bluetooth devices, procedures related to the use of security in different layers and common format requirement of device parameters on the user interface level. The GAP also provides link management facilities for connecting to Bluetooth devices [6].

In Generic Access Profile two devices sharing a link key is called bonded. The procedure to create a relationship based on a common link key is called bonding. Bonding creates a link especially for the purpose of creating and exchanging a common link key. During bonding, the link managers verify that they share a secret key, which is called authentication. After authentication, the link managers will create and exchange a link key. The authentication in link level and link key generation is collectively called pairing.

The GAP defines modes of operation for Bluetooth devices and defines which one is compulsory and which one is optional. The four modes are the following:

1) Discoverability (Controls the use of the inquiry scan and whether other devices can discover a Bluetooth device when it is within their area of radio coverage.)
2) Connectivity (Controls the use of the inquiry scan and whether other devices can connect to a Bluetooth device when it is within their area of radio coverage.)
3) Pairability (Controls the use of the link manager's pairing facilities, which are used to create link keys for use in encrypted links.
4) Security (Control when and how encryption is initialed in a link.)

Three types of discoverability mode are the following: non-discoverable, limited discoverable and general discoverable. There are two connectivity modes: connectable and non-connectable. Two pairability modes are pairable and non-pairable. There are three security modes: non-secure, service level enforced security and link level enforced security.

3.3 Serial Port Profile (SPP)

The SPP defines the necessary requirements of Bluetooth devices for building emulated serial cable connections using RFCOMM between two peer devices [7]. The requirements are expressed in terms of services offered to applications and by defining the features and procedures that are compulsory for interoperability between Bluetooth devices.

The SPP is based on the GSM standard GSM 07.10. It allows multiplexing of several serial connections over one serial link. It supports two device types: a communication endpoint and an intermediate device. A computer is an example of the communication

endpoint. A modem is an example of the intermediate device, which forms part of a communication link.

Support for security is mandatory in the SPP. But security does not have to be used. Either device can request bonding, which requires the use of a shared secret PIN. The PIN can be pre-configured, or can be entered via a user interface. If the devices do not know the common PIN, users will have to exchange the PIN by way other than Bluetooth. Then either side can request the baseband to be encrypted.

The SPP describes how to set up virtual serial ports on two Bluetooth enabled devices and connect them with Bluetooth to emulate a serial cable between the two devices. The three-step application layer procedure is the following:

1) The source device establishes a link and sets up virtual serial connection.
2) The destination device then accepts the link and establishes virtual serial connection
3) Both of the devices register service record for application in local SDP database.

4. Bluetooth Audio Gateway Design

4.1 Headset Profile

Bluetooth audio gateway is just one essential part of Bluetooth headset profile. Thus it should follow all the essential protocols and procedures defined in Headset profile. In this section we will give some general information about the Headset Profile.

4.1.1 Roles in Headset Profile

There are two roles defined in headset profile: Audio Gateway (AG) and Headset (HS). AG acts as the gateway of the audio, both for input and output. Typical devices acting as AG are cellular phones, computer and PDA. HS is the device acting as the AG's remote audio input and output device.

4.1.2 Headset Profile Stack

Figure 9 illustrates the protocols and entities used in the headset profile. Headset control is responsible for handling headset specific control signaling.

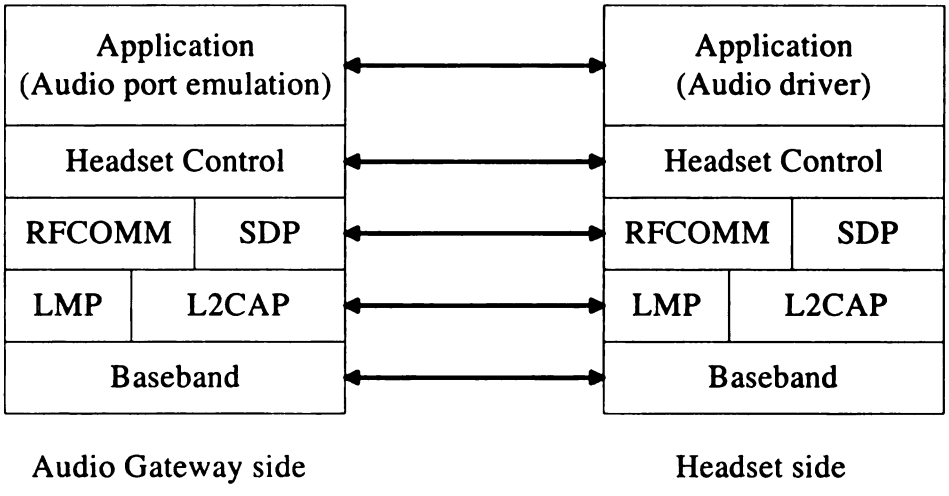

Figure 9: Headset Protocol Model

The headset control signaling is based on AT command. In the original definition, AT command is any instructions sent to a modem that begin with "AT". It is widely used in GSM. The headset profile only uses a subset of AT commands and result codes from existing standards.

The name and description of the AT commands which is mandatory in Bluetooth headset profile are shown in table 1.

Table 1: Mandatory AT Commands in Headset Profile

AT Command Name	Description
RING	The indication of the incoming call in V.250 [8]
+CKPD	The command to control keypad in GSM TS 07.07 [8]. For <keys>, the value of 200 indicates that the button of the headset is being pressed. In the headset profile, the <pause> and <time> parameters have no meaning.

The name, description, syntax and values of the AT commands which is optional in Bluetooth headset profile are shown in table 2.

Table 2: Optional At Commands in Headset Profile

AT Command Name	Description	Syntax	Values
Microphone gain level report	It is the command used by the HS to report its microphone gain level setting to the AG. <gain> is an unsigned octet, representing a particular volume level controlled by the HS.	+VGM = <gain>	<gain>: 0-15
Speaker gain level	It is the command used	+VGS=<gain>	<gain>:0-

indication report	by HS to report the current speaker gain level setting to the AG. The definition of <gain> is same as that in Microphone gain level report.		15
Microphone gain	It is the command used by the AG to set the microphone gain level of the HS. The definition of <gain> is same as above.	+VGM=<gain>	<gain>:0-15
Speaker gain	It is the command used by the AG to set the speaker gain level of the HS. The definition of <gain> is same as above	+VGS=<gain>	<gain>:0-15

4.1.3 Headset Profile Requirement

The restrictions applied to Headset Profile are the following:

1) It is assumed that the headset use case is the only one use case active between the AG and HS.
2) It is mandatory that the transmission of audio data must use CVSD for encoding and decoding.
3) At a time only one audio connection can exist between headset and the audio gateway.
4) It is the audio gateway that controls the SCO link establishment and release. The headset will directly connect (disconnect) the internal audio stream once the SCO link is established (released).
5) The profile only provides basic interoperability. It will not handle multiple calls at the same audio gateway.
6) The only assumption on the headset side is the possibility to detect a user-initiated action such as pressing the button.

4.2 Audio Data Requirements

To implement the Bluetooth audio gateway, we need to know the requirements of the audio input data for Bluetooth devices. In this section we will study the details of Bluetooth audio system and at last decide what kind of audio data we will use for the Bluetooth audio gateway.

Bluetooth doesn't define the maximum sound pressure for an audio device. It's the responsibility of each Bluetooth manufacturer to design their audio products in a safe way with regards not to injury the human ear. Audio levels are calculated as Send Loudness Rating (SLR) and Receive Loudness Rating (RLR).

Figure 10 illustrates the components to measure SLR. MRP is mouth reference point. A/D is analog/digital converter. PGA is programmable gain amplifier. PCM is pulse code modulation. CVSD is continuously variable slop data modulation. Bluetooth specifies three different audio coding techniques: Log PCM coding using either A-law or μ-law and CVSD. We will talk about these techniques in the below section. BTR stands for the binary file transfer.

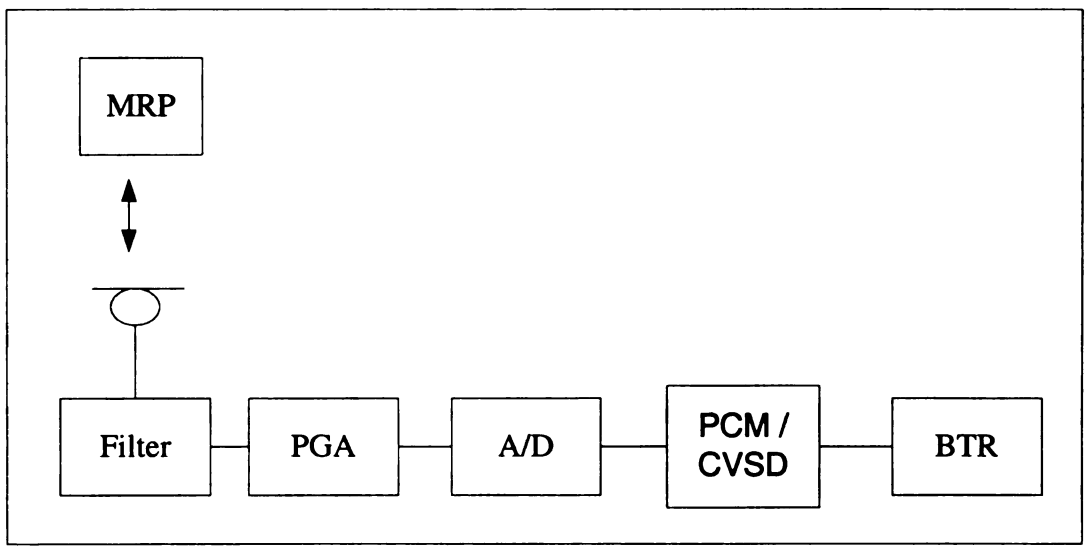

Figure 10: SLR Measurement Set-up

Figure 11 shows the components of RLR measurement. ERP stands for ear reference point.

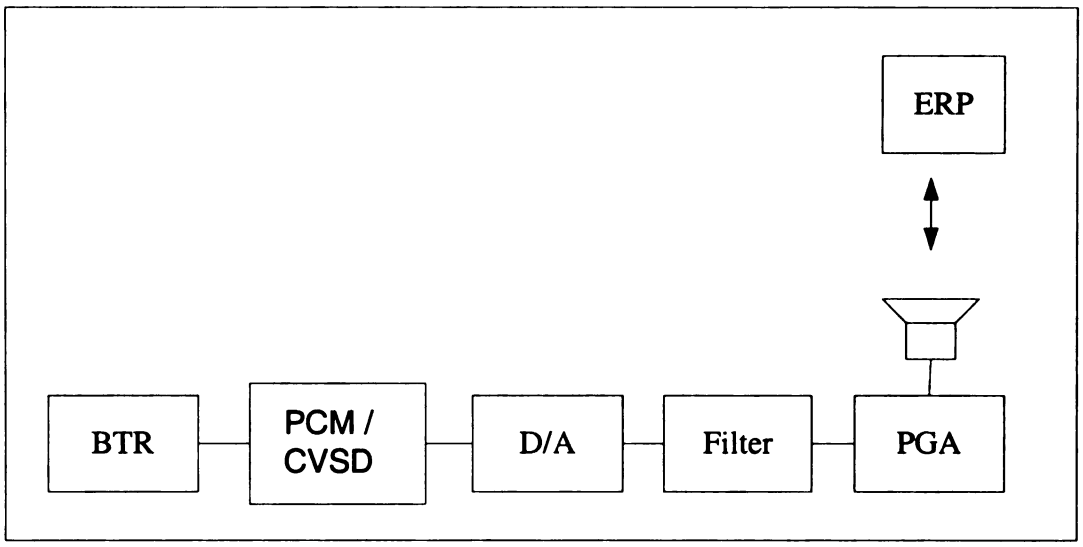

Figure 11: RLR Measurement Set-up

Log PCM coding is widely used in many existing devices such as the PTSN (Public Switched Telephone Network) and fixed-line telephone handsets. Log PCM coding compresses the input data using a logarithmic transfer function so as to present the more accurate (higher bit width) input data with a less accurate (lower bit width) output value. But the logarithmic transfer function guarantees that the effect of the compression gives rise to a minimal decrease in quality as perceived by the human ear. The specification of the exact characteristics are given in the International Telecommunications Union (ITU-T) [8], recommendation G.711, which provides conversion tables to and from linear PCM and log PCM for both A-law or μ-law compression.

The input to the log PCM encoder is up to 3 channels of 13-bit (for A-law) or 14-bit (for μ-law) linear PCM at 8kHz. The output of the log PCM is up to 3 channels of 8-bit encoded data at 8kHz.

CVSD is a more complex method than log coding. It utilizes the strong correlation between adjacent audio samples by quantizing the difference of amplitude between the

two samples as compared to the entire sample amplitude. This needs fewer quantization steps for the same signal quality, and consequently lowers bandwidth. The approach referred as Differential PCM can be revised to reduce the required bandwidth even further by making the quantization step adaptive, which is called Adaptive PCM. This method represents low-amplitude signals with acceptable accuracy without decreasing performance on large-amplitude signals. In a word, CVSD is Adaptive PCM using delta modulation.

CVSD processes 16-bit samples and single-bit symbols at 64 kHz, which are different from those for log PCM. In fact the extra information represented by the 16 bit samples at 64kHz is redundant and merely a side effect of the CVSD process. Moreover, the data rate is too high to make it sensible to pass on to another device as it is, especially if the audio is to be routed via HCI. Thus, it is necessary to interpolate and decimate to reduce the 64kHz sample rate to 8kHz rate as required for the log PCM encoder.

The speech quality of CVSD output is not very good, but is acceptable for most Bluetooth application. In fact CVSD has been used in military communications systems because of its encoded nature, low bandwidth and acceptable quality.

Figure 12 illustrates the typical Bluetooth audio subsystem. As referred before, the log PCM encoding and decoding functions share a common path to and from the PCM data at 8kHz while the CVSD encoder and decoder requires interpolation and decimation respectively. We have discussed each of the blocks shown in Figure 12.

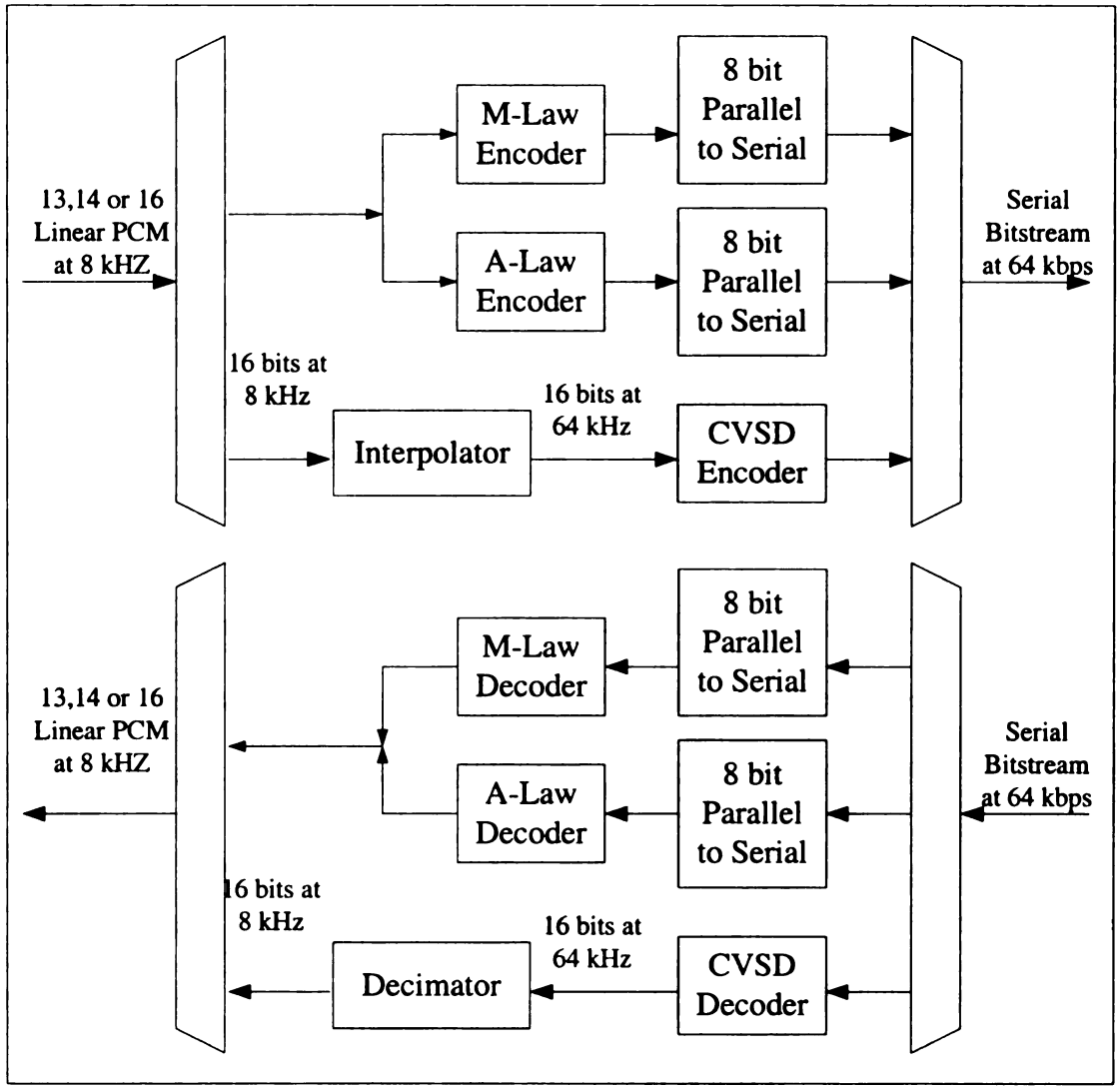

Figure 12: Bluetooth Audio System [9]

Although some SCO packets are corrected by FEC, none of the audio packets are protected by a CRC. Thus the re-transmission is nearly impossible due to the time-bounded nature of SCO data. Much work has been made to decrease the error rate of CVSD to random bit errors. But it is always possible that enough errors will occur so as to make a packet unusable. The errors may not be detected in the packet payload, but it is very possible that the access code may be rejected. In this situation, there will not be a valid packet for the decoder to decode.

Thus a mechanism for filling in or masking the missing packet is required. A simple way is to repeat the previous packet again. A more sophisticated method is to lower the following "repeat" packets with a random white noise to reduce the possibility of an audible tone due. If a new packet is not received after a predetermined time, it may be necessary to decrease the audio level to avoid the repetition becoming audible.

Since CVSD is a different method, the decoder output depends on many previous data (unlike log PCM). When a packet is lost, the information about the current status of the accumulator and the size of the step is also lost. There are various methods to restart the algorithm when a new packet arrives. The simplest one is to reset the step size and accumulator to their initial values when a new packet arrives. Thus the algorithm recovers itself quickly. Even multiple missing bursts happen, the effect on the sound quality is small.

Table 3 shows the comparison of the different type of audio data in different systems. The level of audio quality provided by Bluetooth SCO channel is almost equivalent to that of the GSM cellular telephone audio channel. It is surely not surprising considering that the origins of the Bluetooth standard is based on GSM. Although transmitting MP3 (MPEG Layer3) audio data may be feasible by using two or three SCO channels, version 1.0b of the Bluetooth specifications does not define a profile for such a service. Thus using MP3 data as audio source for Bluetooth audio gateway is not feasible for now.

The frequency shown in the column "Quality" is the sampling rate. The number of bit shown in the column "Quality" is the size of each sample. The data rate is computed by multiplying the sampling rate by the size of each sample.

Table 3: Comparison of Audio Data

System	Quality	Data Rate (kb/s)
Audio CD	Stereo 16 bit @ 44.1kHz	705.6
MP3-encoded Audio	Stereo Near-CD Quality	128
POST Telephone	Mono 8 bit @11.025kHz	88
GSM Audio	Mono 8 bit @8kHz	64
Bluetooth SCO Channel	Mono 8 bit @8kHz	64

At last, we decide to use the audio data with quality of Mono 8 bit @ 8kHz as the audio input for the Bluetooth audio gateway to fit the requirement of the SCO channel.

4.3 Functionalities of Bluetooth Audio Gateway

Bluetooth audio gateway shall have the following functionalities: outgoing audio connection, incoming audio connection, audio connection release, and remote audio volume control.

4.3.1 Outgoing Audio Connection

The procedure to build outgoing audio connection is shown in Figure 13. The steps to finish the procedure are:

1) The audio gateway initiates the connection establishment.

2) An unsolicited result code RING will be sent to headset.

3) The RING may be repeated for as long as the connection establishment is finished.

4) The user presses the button on the headset to accept this connection.

5) The headset then sends out the AT+CKPD command to the audio gateway

6) The audio gateway will establish the SCO link if it is not established.

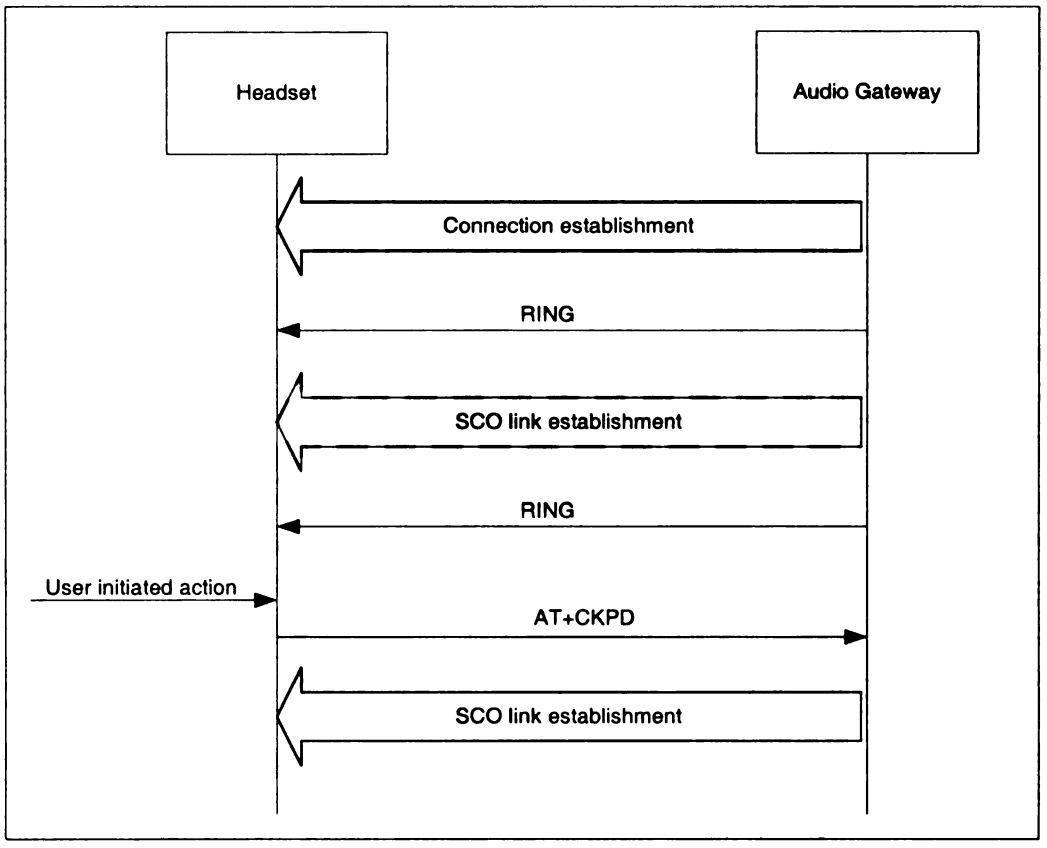

Figure 13: Outgoing Audio Connection Establishment

4.3.2 Incoming Audio Connection

The procedure to build incoming audio connection is a little simpler than the outgoing connection, which is shown in Figure 14.

The steps to finish this procedure are:

1) The user presses the button on the headset to initiate the link.

2) The headset then initiates the connection establishment.

3) The headset will then send out the AT+CKPD command to audio gateway.

4) The audio gateway establishes the SCO link.

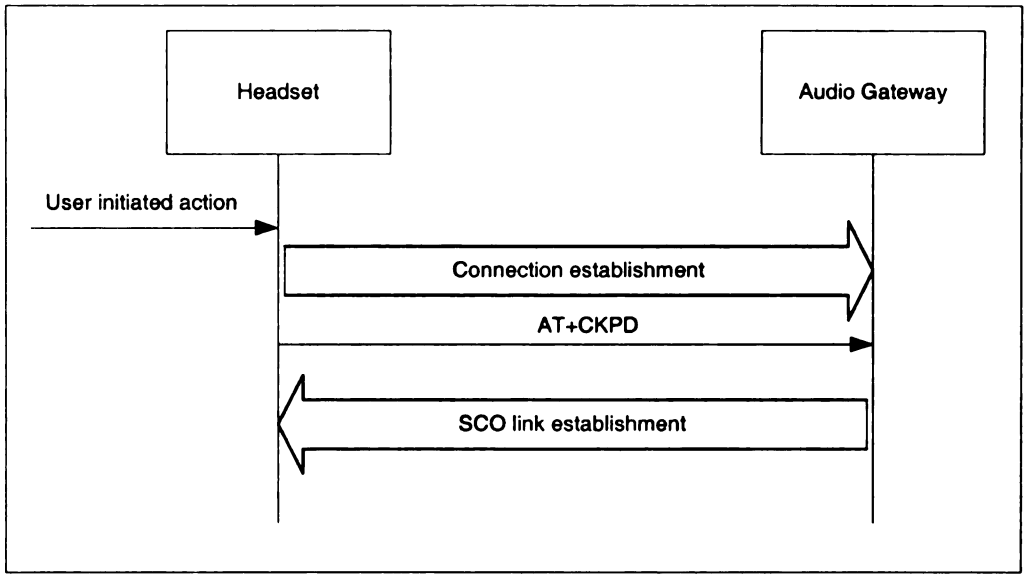

Figure 14: Incoming Audio Connection Establishment

4.3.3 Audio Connection Release

The SCO link can be ended either by the headset or by the audio gateway. Either the button on the headset be pressed or the internal action on the audio gateway can trigger this action. But no matter which initiate the release, audio gateway is responsible for releasing the connection.

Figure 15 shows the procedure to release an audio connection if audio gateway initiates the release. The steps for this procedure are:

1) Internal event or user action of the Audio Gateway takes place.

2) The audio gateway then releases the SCO link.

3) At last the audio gateway releases the connection.

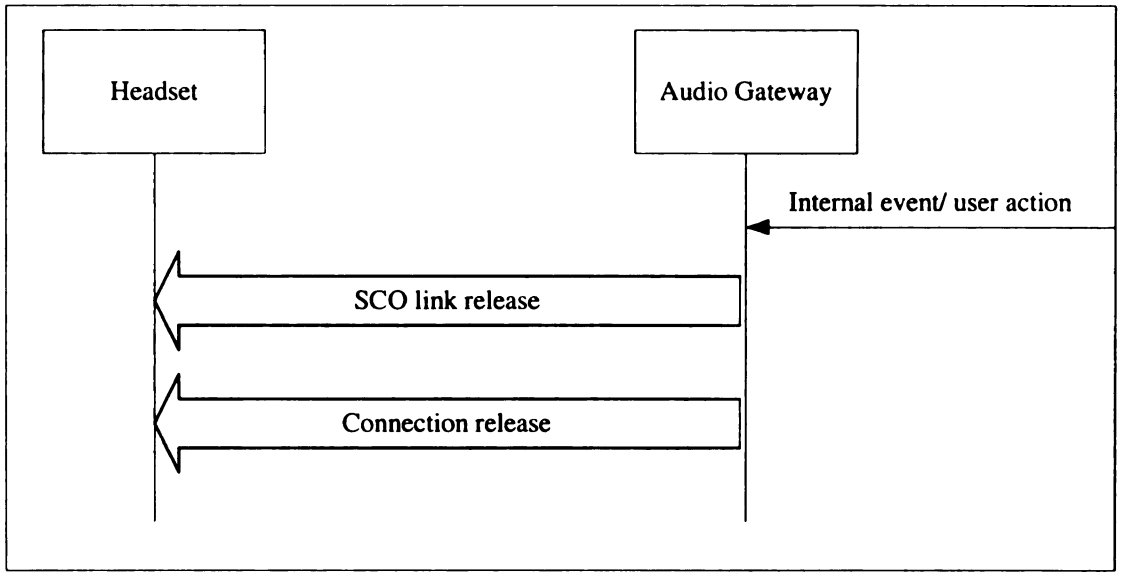

Figure 15: Audio Connection Release – AG Launched

Figure 16 illustrates the procedure to release an audio connection if headset initiates the release. The general steps for this procedure are:

1) The user presses the button on the headset.
2) The AT command "AT+CKPD" will be sent to audio gateway.
3) Then audio gateway releases the SCO link.
4) The audio gateway releases the connection.

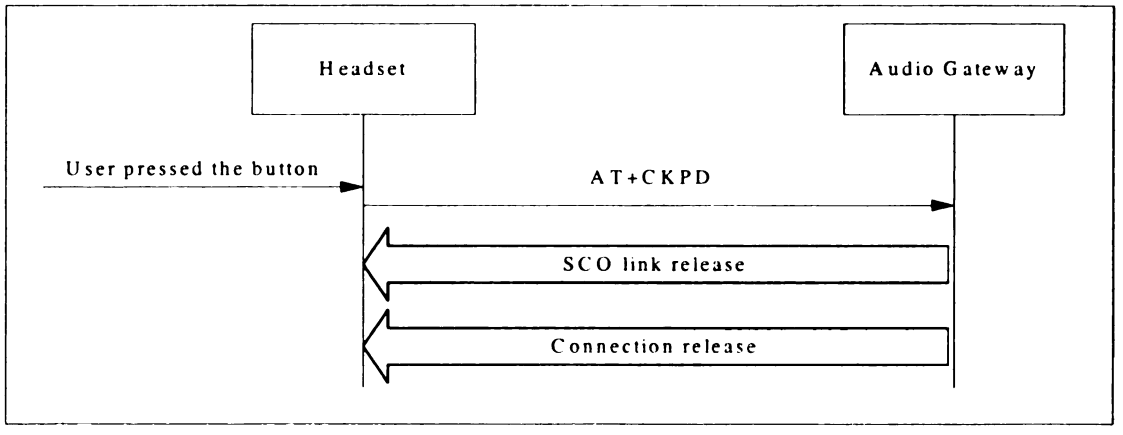

Figure 16: Audio Connection Release - HS Launched

5. Bluetooth Audio Gateway Implementation: BTAudio

5.1 System Requirements

We have implemented the Bluetooth Audio Gateway into a program named BTAudio in Linux. BTAudio can control the Bluetooth device to send or record music file to or from a headset. It is mainly developed on top of BlueZ.

BlueZ is the official Linux Bluetooth protocol stack. Originally it was developed by Qualcomm Incorporated. Now it is an open source project with many contributors all over the world. BlueZ is already part of the official Linux kernel after version 2.4.6. Figure 17 shows the overall architecture of BlueZ. BlueZ provides standard Berkeley socket interface to all Bluetooth layers, which makes it possible to program in application layer.

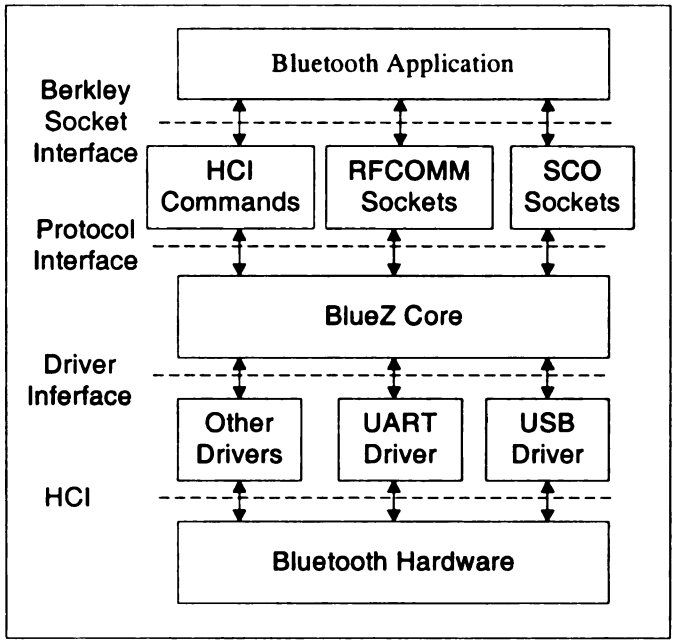

Figure 17: BlueZ Overall Architecture

BlueZ provides support for a variety of Bluetooth devices. It can support Bluetooth PCMCIA and compact flash cards, Bluetooth USB adapters, Bluetooth serial dongles and other Bluetooth devices.

Before we talk about how to design and write the BTAudio, we will give the steps how we installed BlueZ, which is mandatory to use BTAudio. If the user cannot install the BlueZ properly, BTAudio may not work as it expected. What we described is just a specific example; different people may have a different way to install BlueZ.

Because BlueZ is not a mature system, there are some problems existing in the USB SCO driver. To solve these problems we need to install ALSA (Advanced Linux Sound Architecture) packages and apply a patch to Linux kernel. The ALSA packages and patch can be found at these website: http://www.alsa-project.org/ and http://www.dcs.gla.ac.uk/~jp/snd-bluez-sco/.

First we need to build a new Linux kernel for BlueZ. Download the Linux kernel 2.4.22 source file: linux_2.4.22.tar.gz from this website http://www.kernel.org/pub/linux/kernel/v2.4/ and store the zip file into directory /usr/src/. Before compiling the kernel, we need to create one patch: hciusb.patch to fix the problem in USB SCO driver. The content of the patch is shown in Appendix A.

Then unzip the source file, apply patch, configure the kernel, compile the kernel and reboot the system. The setting for Bluetooth modules is shown below in Figure 18. One example to demonstrate all the above steps is given in Appendix B.

```
#
# Bluetooth support
#
CONFIG_BLUEZ=m
CONFIG_BLUEZ_L2CAP=m
CONFIG_BLUEZ_SCO=m
CONFIG_BLUEZ_RFCOMM=m
CONFIG_BLUEZ_RFCOMM_TTY=y
# CONFIG_BLUEZ_BNEP is not set

#
# Bluetooth device drivers
#
CONFIG_BLUEZ_HCIUSB=m
CONFIG_BLUEZ_USB_SCO=y
CONFIG_BLUEZ_USB_ZERO_PACKET=y
CONFIG_BLUEZ_HCIUART=m
CONFIG_BLUEZ_HCIUART_H4=y
CONFIG_BLUEZ_HCIUART_BCSP=y
CONFIG_BLUEZ_HCIUART_BCSP_TXCRC=y
CONFIG_BLUEZ_HCIBFUSB=m
CONFIG_BLUEZ_HCIDTL1=m
CONFIG_BLUEZ_HCIBT3C=m
CONFIG_BLUEZ_HCIBLUECARD=m
CONFIG_BLUEZ_HCIBTUART=m
CONFIG_BLUEZ_HCIVHCI=m
```

Figure 18: Bluetooth Module Setting

After entering the new Linux kernel, we download all the BlueZ packages from the website http://bluez.sourceforge.net/download/download.html. Then we unzip these

packages, install them, edit the module configure file and so on. Please see details about this part installation in Appendix C.

Every time we want to use the BTAudio, we shall make sure that all the Bluetooth modules have been loaded, the Bluetooth USB dongle has been set up and the proper setting has been applied to the device. The user can use the commands shown in Figure 19 to fulfill these functions.

```
#Load all BlueZ modules
modprobe bluez
modprobe hci_usb
modprobe l2cap
modprobe rfcomm
modprobe sco

#Start BlueZ
/etc/init.d/bluetooth start

#Get up the Bluetooth device
hciconfig hci0 up
hciconfig hci0 voice 0x0040
```

Figure 19: Commands to Load Modules and Get the Device UP

Before we begin use BTAudio, we should know the Bluetooth address of the destination headset and the number of the RFCOMM channel. We can use BlueZ service discovery command: sdptool to get the information. The result of service discovery inquiry is shown in Figure 20. From the result, we can know the address of headset is 00:0A:D9:52:1D:A6 and the RFCOMM channel is 2.

```
[root@localhost bluetooth]# sdptool search HSET
Inquiring ...
Searching for HSET on 00:0A:D9:52:1D:A6 ...
Service Name: Headset u
Service RecHandle: 0x10001
Service Class ID List:
  "Headset" (0x1108)
  "Generic Audio" (0x1203)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 2
Profile Descriptor List:
  "Headset" (0x1108)
    Version: 0x0100
```

Figure 20: Result of sdptool Inquiry Result

5.2 Supported Hardware

We use the BTAudio with 3Com Bluetooth USB dongle 3CREB96 and Sony Ericsson Bluetooth headset HBH-60. The picture of the headset, USB dongle and a US quarter is shown in Figure 21. After plugging the Bluetooth USB dongle into the USB interface of the computer, we can control the USB dongle by running the BTAudio on the computer.

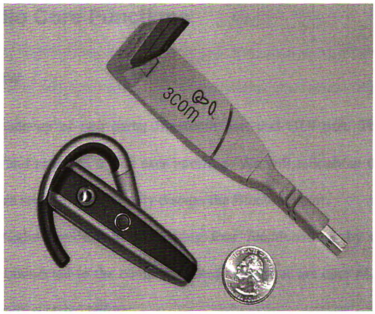

Figure 21: The Headset and USB Dongle Used in Our System

Any Bluetooth headset that complies with the Bluetooth specifications 1.1 can act as the headset. All the current headsets are not programmable. The only thing we can do on the Bluetooth headset is to press some buttons to release connection, increase or decrease the sound volume.

In theory BTAudio can work with other Bluetooth devices such as Bluetooth PCMCIA card and serial dongle if only the SCO service is provided in these devices' drivers in BlueZ. Please see the BlueZ supported hardware at this website: http://www.holtmann.org/linux/bluetooth/devices.html.

5.3 BTAudio Core Functions

5.3.1 Overview

BTAudio is made up of two parts: Bluetooth part and GUI part. The GUI part is written in GTK+ and provides a nice user interface. We will talk about the GUI part in later section. In this section, we will only discuss the Bluetooth part.

Currently BTAudio provides the following four functionalities by controlling the Bluetooth device connected to the computer. These functions are core of audio gateway functions and already referred before.

1) Given the address of the destination headset, BTAudio can send audio file to the headset.
2) Given the address of the destination headset, BTAudio can record audio into files from the headset.
3) During the local Bluetooth device is connected to the headset, BTAudio can release the connection if the button on the headset is pressed.
4) During the local Bluetooth device is connected to the headset, BTAudio can release the connection if the user on the gateway side wants to end the connection.

In fact BTAudio can send out and receive audio data at the same time because the connection between the audio gateway and headset is full duplex. But in practice people seldom record their voice while listening to the music. So the current BTAudio doesn't provide this function.

Whe

setting

parame

configu

size, a

Blueto

Table

When BTAudio starts audio data transmission, the first thing is to check the voice setting of the local Bluetooth device, which is very important to audio connection. The parameters of voice setting control the configuration for voice connections. The configuration consists of input coding, air coding format, input data format, input sample size, and linear PCM parameter. All these settings apply to all audio connections. In Bluetooth there are two bytes for voice setting. But only the last 10 bits are meaningful. Table 4 illustrates the value for voice setting and the description of parameters.

Table 4: Voice Setting Value and Parameter Description

Value	Parameter Description
00XXXXXXXX	Linear input coding
01XXXXXXXX	μ-law input coding
10XXXXXXXX	A-law input coding
11XXXXXXXX	Reserved for future use
XX00XXXXXX	Input data format: 1's complement
XX01XXXXXX	Input data format: 2's complement
XX10XXXXXX	Input data format: Sign-Magnitude
XX11XXXXXX	Reserved for future use
XXXX0XXXXX	Input sample size: 8-bit
XXXX1XXXXX	Input sample size: 16-bit
XXXXXnnnXX	Linear PCM_Bit_Pos
XXXXXXXX00	CVSD air coding format
XXXXXXXX01	μ-law air coding format

In E

coding.

air cod

Aft

the gat

the RF

comma

SCO c

the RF

SCO c

BT

created

SCO c

channe

RFCO

The

we che

XXXXXXXX10	A-law air coding format
XXXXXXXX11	Reserved for future use

In BTAudio, the voice setting should be 0x0040. That means we use linear input coding, 2's complement as input data format, 8-bit for input sample size and CVSD for air coding format.

After verifying the voice setting is correct, BTAudio creates the two channels between the gateway and the headset. One channel is called RFCOMM channel, which is built in the RFCOMM layer. The RFCOMM channel is used for signal controlling. All the AT commands are exchanged in this channel. The other channel is named SCO channel. The SCO channel is used to transmit audio data. Furthermore, the SCO channel is built after the RFCOMM channel is created. Thus if the RFCOMM channel cannot be created, the SCO channel cannot be created neither.

BTAudio will create the two channels with use of socket. If the RFCOMM channel is created successfully, BTAudio will build the SCO channel further. After setting up the SCO channel, BTAudio enter the loop to receive and send out data from or to on the SCO channel. Meanwhile BTAudio still checks and replies the AT commands on the RFCOMM channel in the loop.

The flow diagram of BTAudio is shown in Figure 22. We will describe in detail how we check the voice setting and create RFCOMM and SCO channels in the later sections.

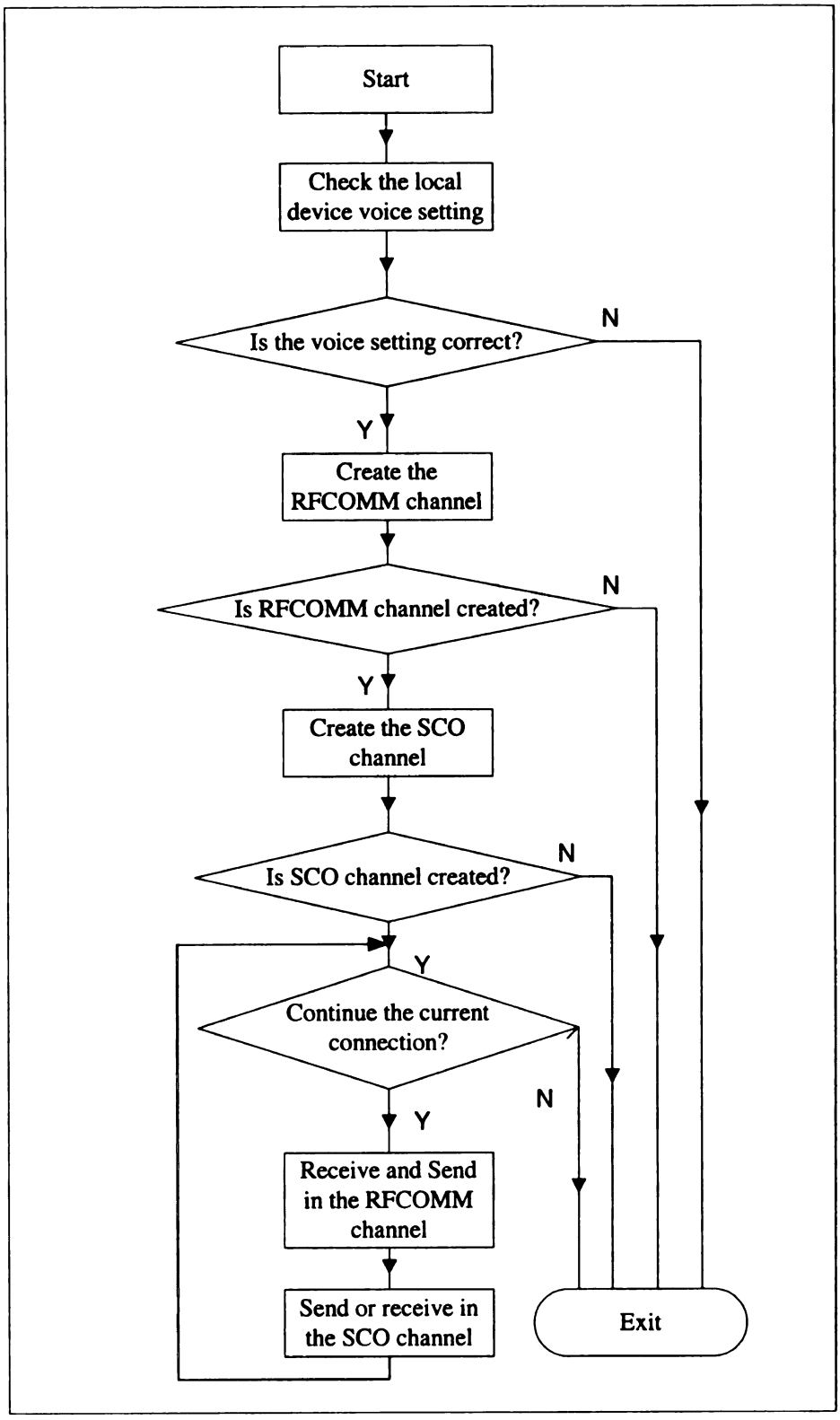

Figure 22: Flow Diagram of BTAudio Bluetooth Part

5.3.2 Check the Voice Setting

BTAudio checked the device voice setting by sending inquiry request into HCI layer. BTAudio accomplish its inquiry by calling the functions provided by BlueZ. The procedure is described below.

First BTAudio shall declare one variable rp with type of read_voice_setting_rp and another variable rq with type of hci_request. The read_voice_setting_rp is a data type for storing the voice setting. The hci_request is a structure for carrying the result of HCI inquiry request. There is one component of hci_request: rparam with type of void *. By assigning rparam with the pointer of different type, the hci_request can be used to inquiry of different information.

The function hci_send_req will fulfill the inquiry by returning the hci_request with the component rparam carrying the inquiry result. In BTAudio, read_voice_setting_rp pointer of rp will be assigned to hci_request's component rparam. After calling the function hci_send_req, BTAudio can get the voice setting from the variable rp.

5.3.3 Create RFCOMM Channel

BTAudio uses the standard Berkeley socket to create RFCOMM channel. The general procedure to create RFCOMM channel is almost same as other socket program. BTAudio uses these functions: socket, bind and connect to set up the channel.

BTAudio selects PF_BLUETOOTH as the protocol family, chooses SOCK_STREAM as the service type and uses BTPROTO_RFCOMM as the protocol. The channel number for local device is set as 0, while the channel number of the destination is set as the RFCOMM channel number of the destination headset. We can get the information of the

RFCOMM channel number in destination headset by using the service discovery inquiry, which we have talked in section 5.1. Figure 23 shows the general procedure to build the RFCOMM channel.

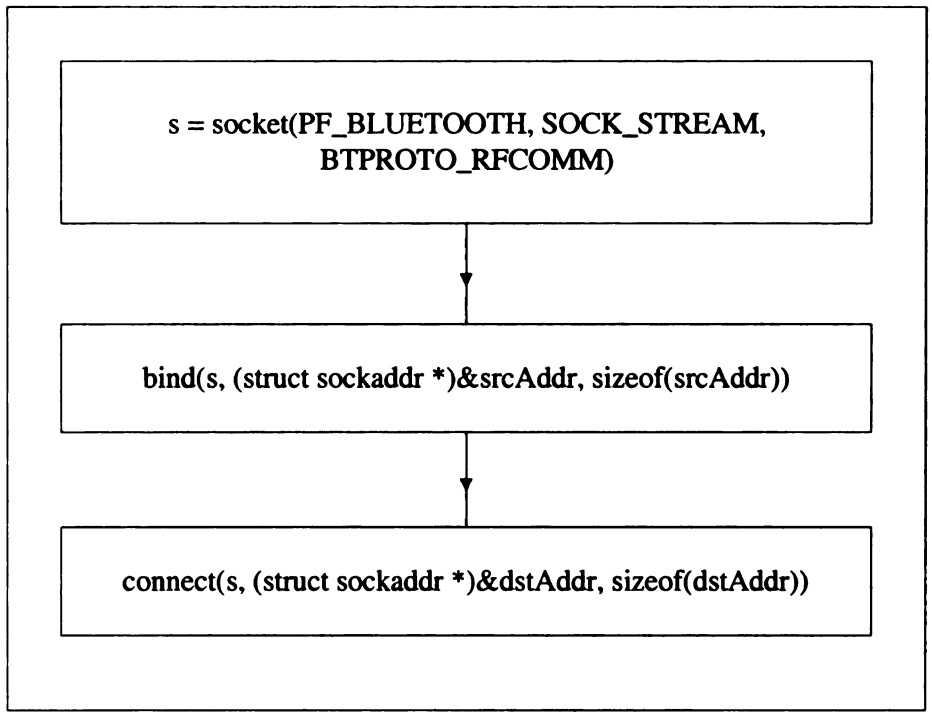

Figure 23: Procedure to Build RFCOMM Channel

5.3.4 Create SCO Channel

After the RFCOMM channel is built, BTAudio begins setting up the SCO channel. The procedure of setting up the SCO channel is very similar to that for RFCOMM channel. BTAudio just need to revise some settings when setting up the SCO channel. Figure 24 illustrates the procedure to build SCO channel.

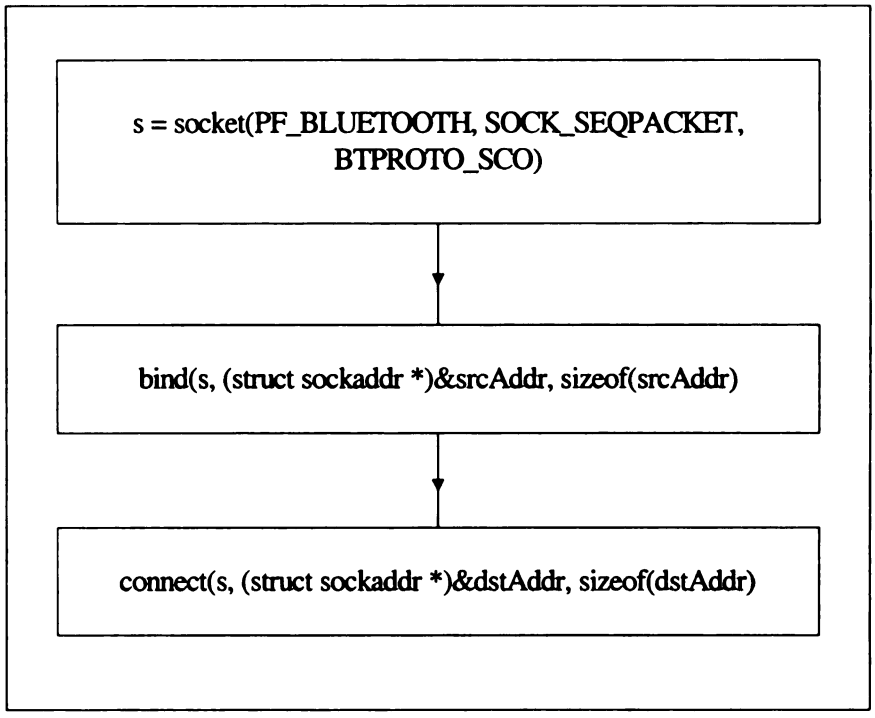

Figure 24: Procedure to Set Up the SCO Channel

Once the SCO channel is built, BTAudio invokes the function getsockopt provided by BlueZ. We can get the HCI handle number and MTU (Maximum Transmission Unit) in SCO channel by using the function getsockopt.

5.3.5 Transfer Data on the RFCOMM and SCO Channel

After the SCO channel is built, BTAudio will send or receive audio data to or from the headset in the SCO channel while it still check and reply the AT commands in the RFCOMM channel.

To transfer data on the RFCOMM and SCO channels, BTAudio use UNIX system call read and write, which are accessed from C program through two functions called read and write: "int n_read = read (int fd, char *buf, int n)" and "int n_write = write (int fd, char *buf, int n). The first argument can be a file descriptor or a socket descriptor. The second argument is a character array where the data go to or come from. The third

argument is the number of byte to be transferred. Each function returns a count of the number of byte transferred which may be less than the number requested.

Every time BTAudio reads the AT commands from the RFCOMM channel, it will judge whether it is "+CKPD =200". If yes, that means the headset wants to release the connection. Then BTAudio will release the RFCOMM and SCO channels. For each command BTAudio received, it will reply a command "OK".

When BTAudio wants to send data onto the SCO channel, it needs to receive data first. The number of byte to be sent depends on how many bytes it received from the SCO channel. This method is very flexible. When the connection between headset and audio gateway is lost, the size of data received in SCO channel will 0. In this case, sending data into the SCO channel is useless. Secondly, the size of SCO packet in the headset side may be fixed because of imbedded procedure in the headset side. Using this way it avoid sending the SCO packets bigger than the headset can handle.

To send audio data to headset, BTAudio gets the source data from music file stored in the local computer. After receiving audio data from the headset, BTAudio stores the data into music file. BTAudio uses the system calls: open, read and write to complete these functions.

Figure 25 illustrated the sequence diagram of the data transfer on the RFCOMM and SCO channels. In this example the audio gateway received the AT command "+CKPD =200" and closed the connection.

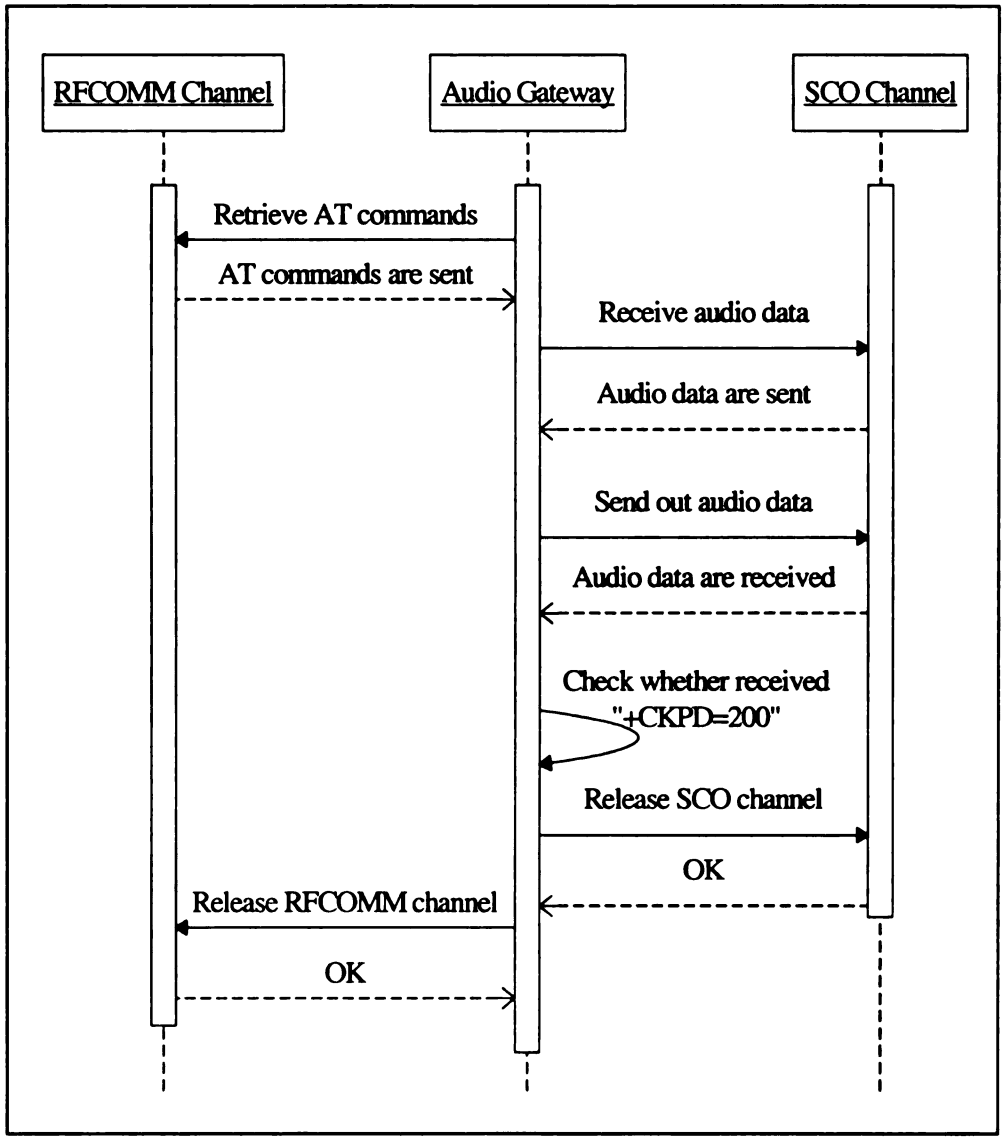

Figure 25: Sequence Diagram of the Data Transfer on the RFCOMM and SCO Channels

5.4 BTAudio GUI

5.4.1 Introduction

The GUI for the BTAudio is developed using GTK+[3]. GTK+ is a multi-platform toolkit for developing graphical user interfaces and has been designed from the ground up to support a range of languages, such as C/C++, Perl and Python.

The graphic user interface of BTAudio is shown in Figure 26. It consists of two list boxes, one text window and five buttons.

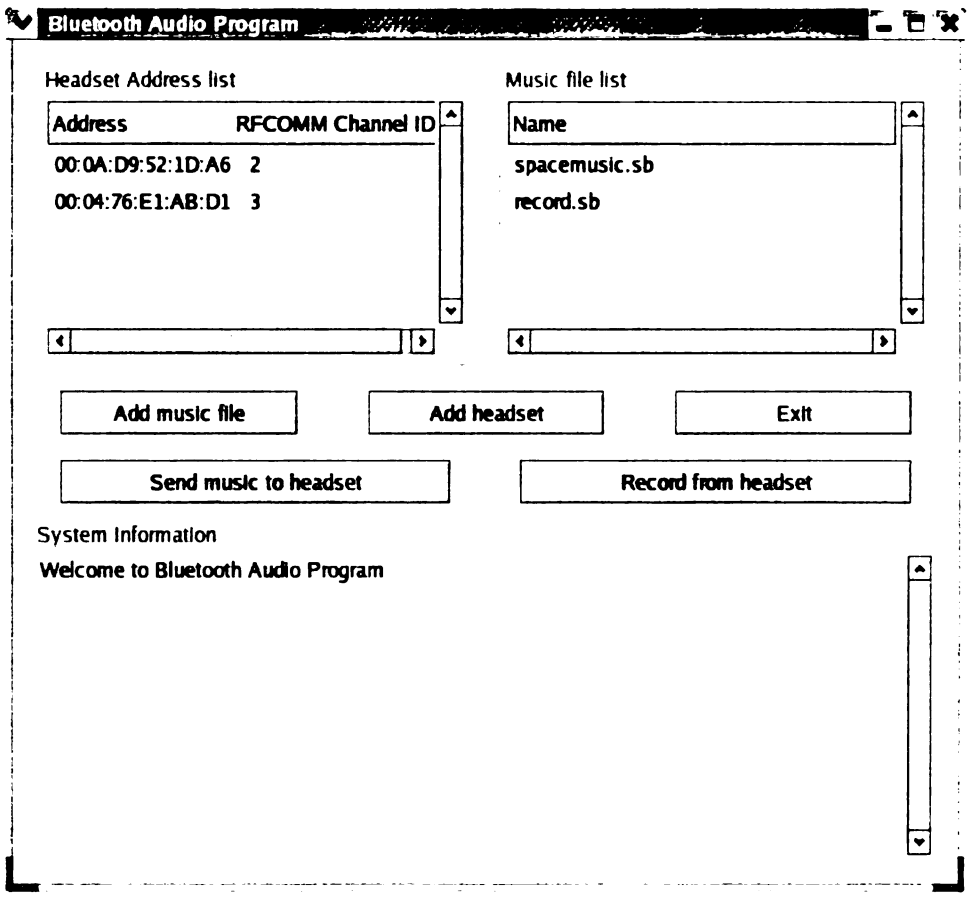

Figure 26: BTAudio User Interface

Before running the BTAudio, the user shall have prepared two text files: deviceList.txt and musicList.txt. The file deviceList.txt stores the information for destination headset. The file musicList.txt contains the information of the audio files, which may be sent to the headset. The format for deviceList.txt is one line for one device. The address should be written at the very beginning, which is followed by the RFCOMM channel number. But there should be at least one empty space to separate the address and the channel number. The format for musicList.txt is one line for one file. The name of the file should be put into each line without any character in front of it. All the audio files should be

mono files with sample rate at 8kHz. Also, the two text files and audio files should be stored under the same directory as the BTAudio executable file.

One of the list boxes displays the destination device address and its RFCOMM channel number. The other displays the name of the music files. BTAudio gets these information by reading two text files: musicList.txt and deviceList.txt.

The text window is used to display system information to the user. The button with label "Add music file" is used to add one music file into the music list box. The button labeled with "Add headset" is for adding one Bluetooth device into the device list box. The user can click the button with label "Send music to headset" and the button with label "Record from the headset" to send out or record audio file to or from the headset. The button labeled "Exit" can be used to close the program.

5.4.2 Send out music file

To send out a music file to headset, the user needs to select the destination device in device list box and the file name he wants to transmit by clicking the items in the list boxes. Then the user needs to click the button to send out the music.

BTAudio will check the voice setting before connecting to destination device. If the voice setting is not correct, the BTAudio will ask the user to change the voice setting and abort this action. BTAudio will abort the action too if the user didn't choose the device or the music file name. One example that the BTAudio aborted its action is shown in Figure 27.

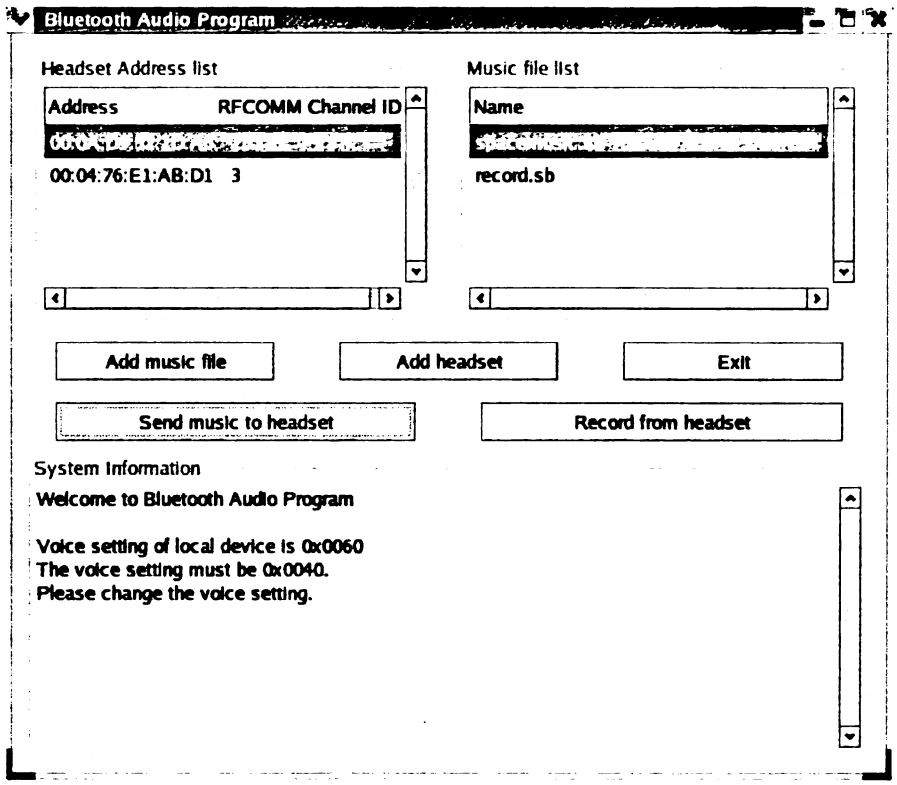

Figure 27: Checking the Voice Setting

During the process of sending out the music, the text window will display the system information such as which step BTAudio is taking and the AT commands it received. We can see one example in Figure 28. From the text window we can see that the SCO HCI handle ID is 43 and MTU size is 64. The string starts with AT+ are all AT commands. After the connection is built, BTAudio will send out AT command "AT*ECBP=?" to inquiry about the speaker's gain level. The AT command " AT+VGS=8" is the reply from the headset. The following AT commands such as "AT+VGS=10" appears because the user adjusts the speaker gain level. When the user adjusts the speaker gain level, it will automatically inform its audio gateway. At the end of the text window we can see that BTAudio informs the user about the amount of data sent, the time used for transmitting and the average transmission rate.

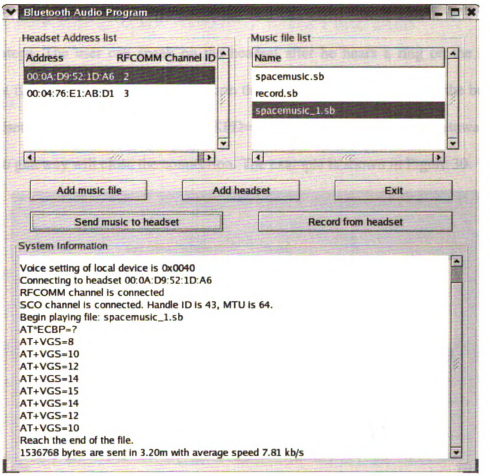

Figure 28: User Interface after Sending Out Music

5.4.3 Record voice from Headset

When the user wants to record voice from the headset, he can click the button with label "Record from headset". Then a dialog shown in Figure 29 will appear for the user to give the name for the record file.

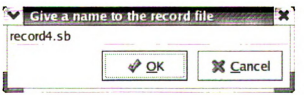

Figure 29: The Dialog to Give a Name to the Record File

After filling in the record file name, the user can click button with label "OK" for further steps. The user can speak on the headset after he hears a ring on the headset. When he finishes his talking, he can inform the audio gateway by pressing the button on the headset. The AT command "AT+CKPD=200" will be sent to audio gateway. Then the audio gateway will close the connection. The example is shown in Figure 30.

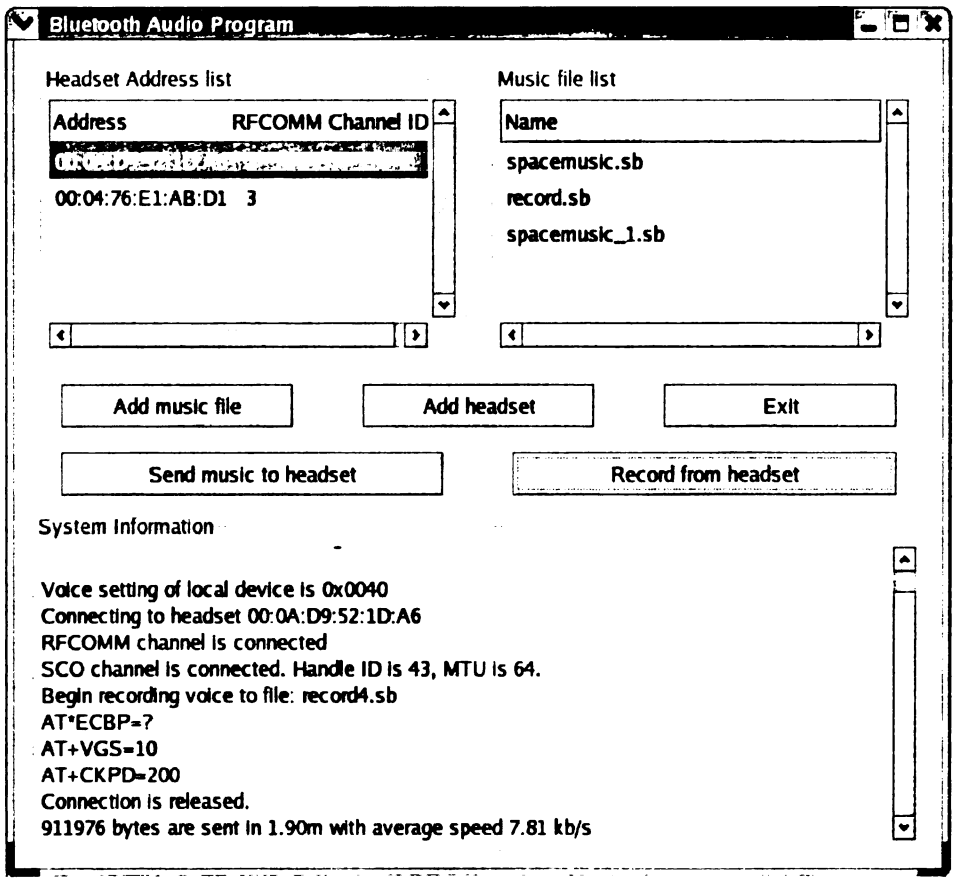

Figure 30: The Example to Record Voice from the Headset

6. BTAudio Experiments and Results

6.1 BTAudio Experiments

We conduct the experiments on Bluetooth devices mainly by sending audio data to the headset. We did not perform experiments by recording the voice from the headset. The reasons are below. First, we cannot separate the effects of the quality of the microphone on the headset. There is some noise in the audio data even the audio gate way is very close to the headset. We do not think it is caused by the Bluetooth transmission for the quality of the music heard on the headset is very good when the headset is very close to the audio gateway. Second, the difference between recording from headset and sending audio data to headset is just the difference in direction. In essential they are same because the full duplex character of the SCO channel. If we find the properties of sending audio data to headset, recording from headset will surely have the same properties too.

The main method is to compare the quality of the data transmission, the transmission rate, and the size of SCO packets when we change the distance between the audio gateway and headset. We also put the headset in different containers to test the Bluetooth device propagation ability.

Currently, we do not have a metrics to judge the quality of data transmission. We just judge it by the quality of music heard on the headset by ear. The more noise we hear, the worse the quality is.

We keep the audio gateway on a computer static while placing the headset in different areas. We put it in the office, in the hallway, and even outside the building. Figure 31 shows the map of experiment areas. The thinner black line means interior room wall. The

thicker black line stands for exterior building wall. We marked five open doors in the map. But during our experiments, all the doors are closed. The rectangle filled with white color is the office room we work in. The computer carrying the audio gateway is placed inside this office room, which is represented by the red computer. All the areas marked with slantwise lines are the areas we have performed experiments. Except the office room, all other area are either hall way or doorway.

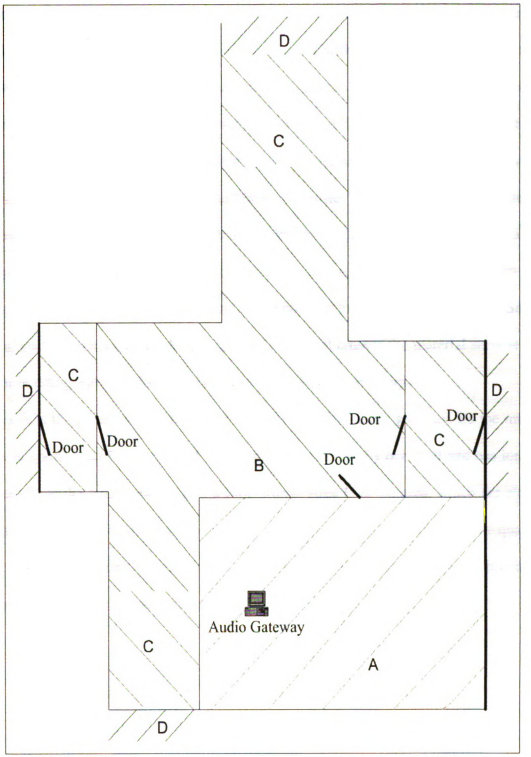

Figure 31: Map of Experiment Areas

After the experiments are finished, we divide the whole area into 4 parts: area A, area B, area C and area D according to the quality of music heard on the headset. Area A is

the area marked with slantwise green lines. In this area, we can clearly hear the music. Very little noise is heard. Area B is the area marked with slantwise blue lines. In this area, we can hear some noise accompanying the music. The rhythm of music is still continuous. Area C is the area marked with slantwise red lines. In this area, we will hear lots of noise. The rhythm of the music is not continuous. Area D is the area marked with slantwise black lines. In this area, we can't hear any music. The connection is totally lost.

These results match what we expected. We know that Bluetooth uses radio signal to communicate with each other. As the distance gets longer, more loss will occur in the propagation path. Some materials such as wood furniture can absorb signal. Some material such as metal can block the signal. If the signal crosses the material such as the exterior building wall, the distance it can reach will be decreased rapidly.

Area A is the place, which shares the same room as the audio gateway. So the music quality is good. Area B is separated from the audio gateway by one wall and has longer distance. Thus the music quality gets a little worse. Area C is the area, which are either separated with two walls or have much longer distance. It is not surprising that the quality is bad. Area D is the area, which is separated by two interior walls and one much thicker exterior wall. Thus the connection is totally lost.

6.2 Results

6.2.1 Relationship Between Quality of Transmission and Distance

Based on the above experiments, we get the approximate relationship between the quality of music heard on the headset and the distance between headset and audio

gateway, assuming there is no strong signal screen material existing between the headset and gateway. The relationship is shown in Table 5.

Table 5: the Relationship Between the Quality of Music Heard and the Distance

Distance between Headset and Audio Gateway (m)	Quality of Music Heard
0 – 10	Very clear and no noise
10 – 19	Some noise exists
19 – 26	Many noise heard
> 26	No music can be heard

6.2.2 Transmission Rate in Different Areas

As shown in Figure 32, we compared the average transmission rate on the SCO channel between the audio gateway and the headset when the distance between them is different. The transmission rate for area A, B and C are almost same. The transmit speed in area A is 7.82 Kbytes/s. The transmit speed in area B and C is 7.81 Kbytes/s. The connection cannot be built in area D. So the transmit speed in area D is 0.

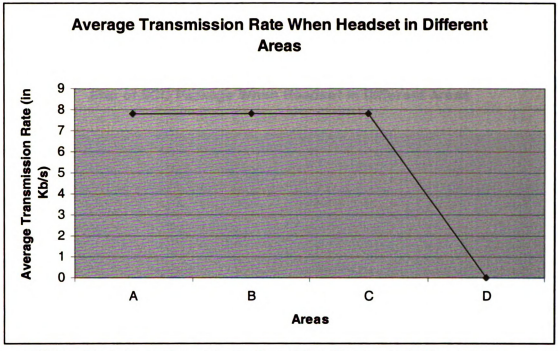

Figure 32: Average Transmission Rate When Headset in Different Areas

This result matches the theory of Bluetooth. We know that the sampling rate for Bluetooth audio input data is 8KHz. Each sample has 1 byte. So the data rate for sampling is 8Kbytes/s. In order not to distort the audio data, the transmission rate shall be 8Kbytes/s too. Thus the SCO link in Bluetooth has reserved bandwidth, which guarantees that SCO data will be transmitted at constant rate of 8Kbytes/s.

The transmission rate we got is a little smaller than 8Kbytes/s. The reason is that we need to get the audio data from the music file, which consumes some time.

6.2.3 Size of SCO Packets

After knowing that the transmission rate in area A, B and C are same, we then checked the size of the packets sent on the SCO channel when the headset are placed in different areas. We found that the size of SCO packet is always 24 if the connection exists between the headset and the audio gateway. When the connection is lost, the size of the SCO

packet will become 0. Figure 33 shows the size of the SCO packets when the headset is in different areas.

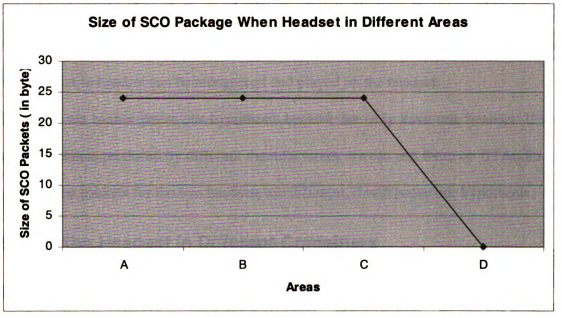

Figure 33: The Size of SCO Packet When Headset in Different Areas

We changed the source code of BTAudio to send audio data by manually defining the packet size. If we define the packet size as 24 bytes, the BTAudio works very well as before. But if we define the packet size as other values such as 25 or 23, the audio data transmission cannot continue further.

Now we do not know whether the fixed size of packet is caused by the limitation of the BlueZ or the limitation of the Bluetooth device. Only the baseband in Bluetooth defines that the fixed size of voice data is 80 bits in the SCO socket. The upper or application layer in Bluetooth does not give the limitation of the size of the SCO packets. In theory, packets of any size in the application layer can be transmitted. The lower layer will be responsible for segmenting and reassembling the packets.

We guess the reason is because of the Bluetooth headset. The Bluetooth headset is not programmable. We can only press the buttons to accept the connection, close the

connection, and adjust the speak volume. How the headset receive the audio data and send out audio data are already imbedded into the Bluetooth chip. Although the audio gateway is responsible for creating and releasing the connection, it may need to comply with the settings on the headset. Only by using the setting acceptable by the headset, the audio data sent to headset can be recognized and played on the headset.

Currently we cannot prove our hypothesis because we only have one headset. If we have two headsets produced by different manufacturers, we can test them on BTAudio. If the sizes of the packets for the two headsets are different, it will prove our hypothesis.

6.2.4 Put the Headset in Different Containers

Beside we change the distance, we put the headset inside four containers: steel drawer, icebox, steel box and microwave oven. The distance between the containers and the audio gateway is within 5 meters. The result is that the headset still can communicate with the audio gateway when inside the steel drawer, icebox and steel box. But it cannot communicate with the audio gateway when inside the microwave oven.

The first three experiments demonstrate that the signal of Bluetooth device has strong penetration ability. The result of the fourth experiment is not surprising. The microwave oven has a perfect screen system to prohibit any signal to be transmitted across its wall.

6.2.5 Summary

In conclusion, we get the below properties of the Bluetooth based on the above experiments:

1) When the connection between the headset and audio gateway exists, the average transmission rate is 7.8Kb/s, which is very close to the data rate for sampling. We proved that the SCO channel in Bluetooth has fixed bandwidth.
2) During our system, the SCO packet size is fixed when the headset is inside the area where the connection can be set up.
3) We guess the fixed SCO packet is because of the fixed setting in Headset. This needs to be proved in the future.
4) Bluetooth device have strong propagation ability. The interior wall, door, thin steel layer cannot stop its propagation.

7. New Bluetooth Profile: Quite Talk Profile

7.1 Overview

Quite Talk Profile provides a mobile, full duplex, and hand-free, quite way for people to communicate. For example, in a large room, two persons: A and B far way from each other wants to talk with each other. Person A needs to use both of his hands on the computer. Person B needs to walk around several devices and operate them. By using this profile, they can talk with each other conveniently and quietly without disturbing other people in the same room.

There are two types of roles in this profile: two or three headsets and one audio gateway. The headsets can talk to each other by connecting to one audio gateway. The audio gateway acts as the data transmit server.

We know that two headsets cannot pair with each other and cannot create SCO link. Then they cannot communicate with each other directly. Thus the audio gateway is necessary in this Quite Talk Profile.

We will talk about its system requirements, the functionality and its feature below.

7.2 System Requirements

Quite Talk Profile has the below requirements:

1) Up to three SCO link establishments can exist on the audio gateway.
2) At least two headsets connect to the audio gateway.
3) There is only one SCO link establishment on each headset.
4) Every headset connects to the audio gateway.
5) It is mandatory that the transmission of the audio data must use CVSD for encoding and decoding.
6) When the user press the button on one of the headsets, audio gateway will set up connection between itself and all the headsets.
7) After connection is built, the user can disconnect the connection by press the button. Then the audio gateway will disconnect the connection between that headset and the audio gateway.
8) If there is only one headset connected to the audio gateway, audio gateway will release this connection too.

7.3 Functionalities of Quite Talk Profile

Quite Talk Profile provides these main functionalities: audio connection, audio data transmission and audio connection release. The procedures about audio connection, audio data transmission and audio connection release are given below.

7.3.1 Audio Connection

The procedure to build audio connection between two headsets and audio gateway is shown in Figure 34. For clarity, we name the headsets as headset A and headset B.

The steps to finish the procedure are:

1) The user presses the button on the headset B to initiate the link.
2) The headset B then initiates the connection establishment.
3) The headset B will then send out the AT+CKPD command to audio gateway.
5) The audio gateway establishes the SCO link between itself and headset B.
6) The audio gateway starts the connection establishment between itself and headset A.
7) The AT commands RING will be sent to headset A.
8) The RING may be repeated for as long as the audio gateway receives a reply from the headset.
9) The user presses the button on the headset A to accept this connection.
10) The headset A then sends out the AT+CKPD command to the audio gateway.
11) The audio gateway will establish the SCO link between audio gateway and headset A if it is not established.

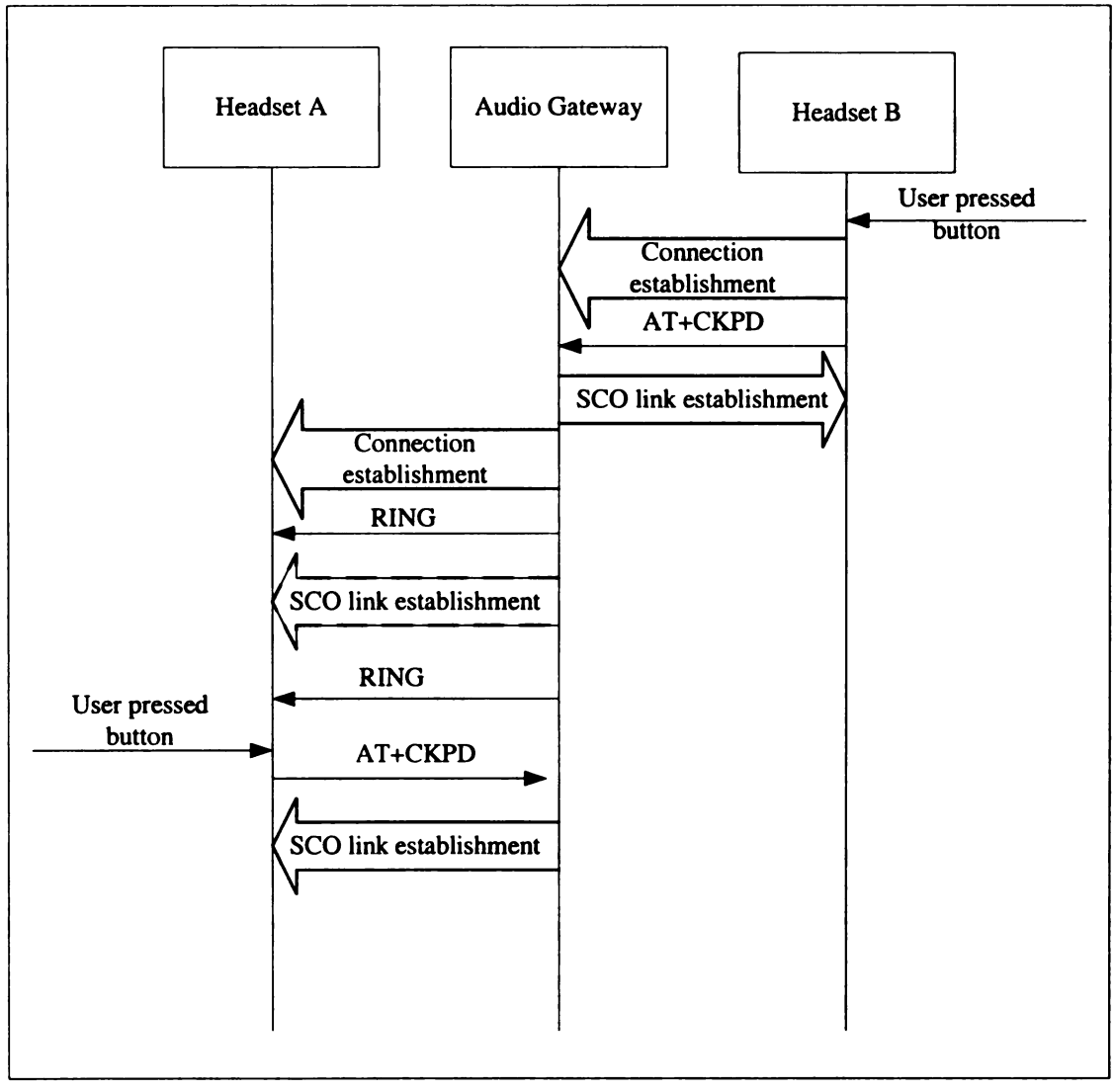

Figure 34: Audio Connection in Quite Talk Profile

The procedure for three headsets and one audio gateway connection is very similar to the above procedure. The steps for the first two headsets are same as that for the headset A and B. The step for the third headset is same as that for headset B.

7.3.2 Audio Data Transmission

For simplicity we only talk about the procedure of data transmission among two headsets and audio gateway. The procedure among three headsets and audio gateway is almost same as that among two headsets and audio gateway.

The procedure to transmit audio data among two headsets and audio gateway is shown in Figure 35. The steps for audio data transmission between audio gateway and headset A are the following:

1) The audio gateway starts to receive the audio data from the headset A.
2) The audio data from headset A is stored into Buffer_B.
3) The audio gateway sends out the audio data stored in Buffer_A to headset A.
4) The headset A finishes receiving the audio data.

The steps between headset B and audio gateway is very similar to the steps between headset A and audio gateway except that they use different buffer to store the incoming and outgoing data. The audio gateway stores the data from headset A into Buffer_B. Later the audio gateway will send the data in Buffer_B out to headset B. Thus the audio data from headset A is sent to headset B. In the same way audio data from headset B will be sent to headset A.

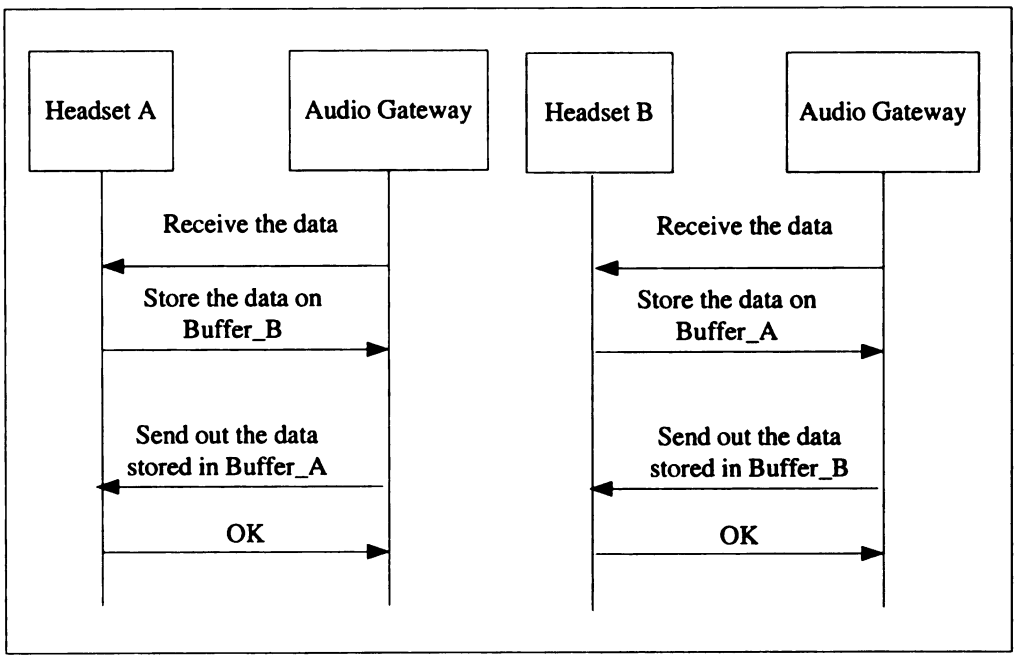

Figure 35: Audio Data Transmission in Quite Talk Profile

7.3.3 Audio Connection Release

In Quite Talk Profile the audio gateway never initiates the audio connection release. It is always one of the headsets that initiate the audio connection release. Then the audio gateway is responsible for releasing the audio connection between it and the headset. But after that the audio gateway will check whether there is only one audio connection left.

The audio gateway will release that audio connection if it is the only audio connection existing on the audio gateway. This feature is determined by the nature of the Quite Talk Feature. We know that one person can speak to himself without any tools. Thus it is required that at least two audio connections existing on the audio gateway. Otherwise the only connection will be released automatically.

Figure 36 shows the procedure that headset A initiates the audio connection release when there are two audio connections on the audio gateway. As in the example, the audio connection between audio gateway and headset B is released automatically.

The steps for the above procedure are below:

1) The user presses the button on headset A.
2) The AT+CKPD command is sent to audio gateway.
3) The audio gateway releases the SCO link between it and headset A.
4) The audio gateway releases the connection between it and headset A.
5) Later the audio gateway checks the number of the audio connection.
6) After finding that there is only one audio connection left, the audio gateway release the SCO link between it and headset B.
7) The audio gateway releases the connection between it and headset B.

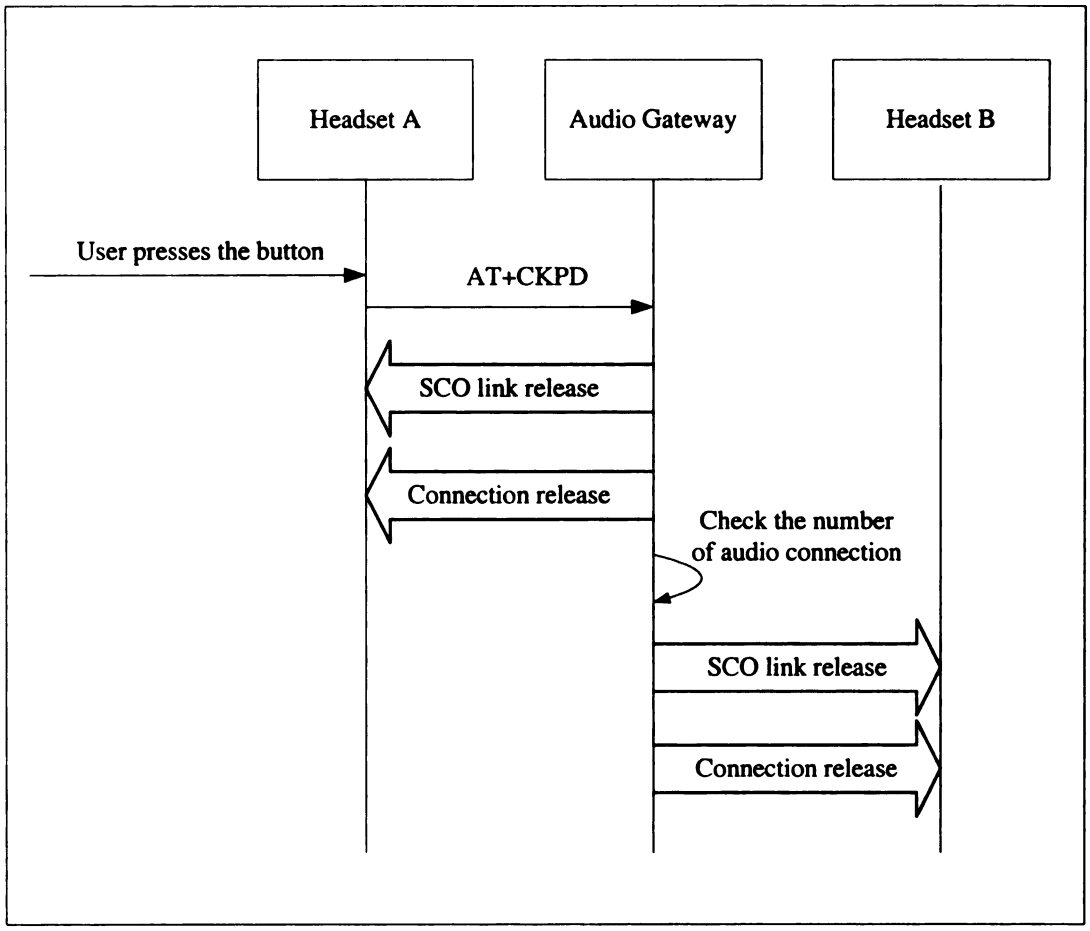

Figure 36: Audio Connection Release

When there are three audio connections existing on the audio gateway, one of three headsets wants to release the connection. The audio gateway will only release the connection between it and that headset. The procedure for this action is same as the procedure of audio connection initiated by the headset, which we described in section 4.3.3.

7.4 Feature of Quite Talk Profile

Quite Talk profile has the following features:

1) It can support up to three headsets to communicate at the same time.

2) It is full duplex, which makes the communication more convenient.

3) It is hand-free, which liberate the hands of people. A person can talk while he uses his hands with something else.

4) It is mobile. People can talk while they keep walking.

8.Conclusion and Future Work

8.1 Conclusion

This thesis discusses the Bluetooth audio gateway design and how to implement it into the program: BTAudio in Linux. It also covers experiments using the BTAudio in different conditions and get some important results about Bluetooth. Later it proposes a new Bluetooth profile: Quite Talk Profile.

The main contributions of this research are the following:

1) Finished the audio gateway design and implementation. BTAudio can be used as an application program. Also BTAudio can be used as a tool for performing future research in Bluetooth.

2) Conducted experiments using BTAudio and got some important results about Bluetooth.

3) Proposed a new Bluetooth profile: Quite Talk Profile. This Profile provides a mobile, full duplex, and hand-free, quite way for up to three people to communicate.

8.2 Future Work

Because of limited time, we only accomplish the above tasks. In fact we still have several problems left for future work.

The first one is how to transfer BTAudio into the PDA. Currently the BTAudio can only run in computers. It cannot run in PDA. The reason is that the PDA has a small memory space. We cannot install the compilation tools into it. Thus we cannot compile the source code of BTAudio on PDA. But using cross compilation can solve this problem. Cross compilation is that using a compiler running on one system to produce executable for another system. If we can cross compile the source code in the computer and get the executable for the PDA. Then this problem can be solved.

The second problem is that testing BTAudio on a headset, which is not produced by Sony Ericsson. We suspect that the fixed size of packet in the SCO channel might be caused by the fixed setting on the Bluetooth headset. But because we only have one Bluetooth headset, we cannot prove whether the hypothesis is correct or not.

The third problem is that we compared the quality of audio data transmission between the audio gateway and headset by ear. We do not have a metrics to judge the transmission quality. Finding one way to evaluate the quality of the data transmission will benefit the Bluetooth research.

The fourth problem is to implement the new Bluetooth profile: Quite Talk Profile. Although this is not very important to Bluetooth research, this nice application will attract more Bluetooth user and promote Bluetooth prevalence.

The last problem is to explore new profiles or applications for Bluetooth. Bluetooth can take effect in many areas, such as in the automotive environment. Current some of

the car manufacturers have included Bluetooth devices in the car. There can be many and various applications in the car.

Figure 37 illustrates the Bluetooth applications in the car. The MP3 player can transmit the music file to car audio system. Then the speakers in the car can play the music. If the car has installed GPS (Global Position System) devices, the PDA can get the location of the car by connecting to the GPS system, compute the direction by using the maps and send out the guidance commands to headset worn by the driver. The driver can also send an inquiry to PDA by pressing the button on the headset. Then the PDA can reply it by connecting to GPS devices.

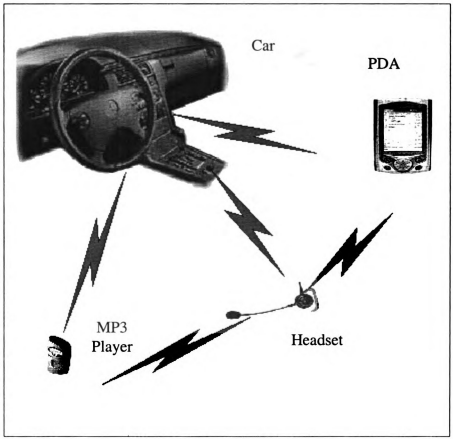

Figure 37: Bluetooth Applications in Car

There are many other applications the Bluetooth can be employed. Figure 38 shows some of the Bluetooth enabled devices. With Bluetooth device everywhere, Bluetooth promises to be a very powerful technology that will change the world.

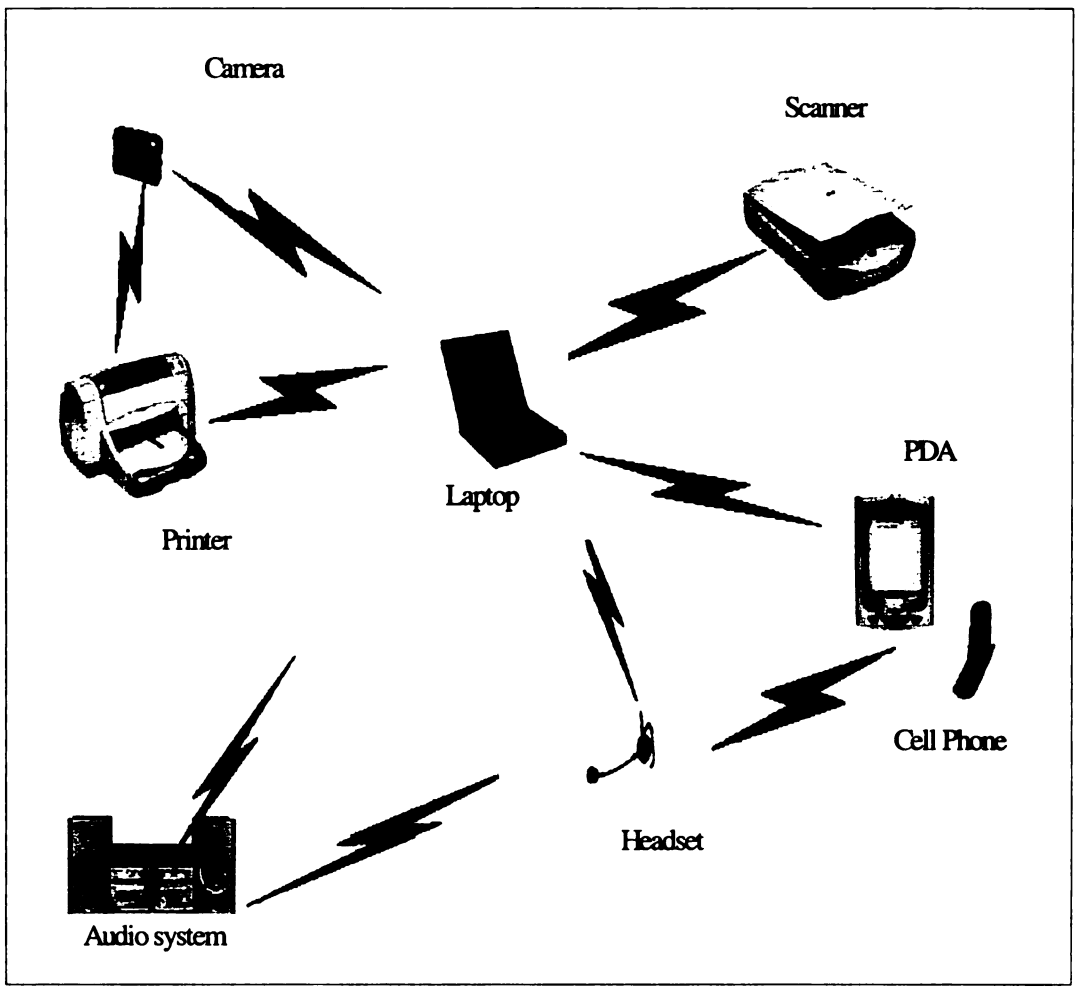

Figure 38: Bluetooth Enabled Devices

BIBLIOGRAPHY

[1]Telefonaktiebolaget LM Ericsson, International Business Machines Corporation, Intel Corporation, Nokia Corporation, Toshiba Corporation, "Specification of the Bluetooth System, Specification Volume 1, Core", Version 1.0 b, December 1999.

[2]BlueZ: Official Linux Bluetooth protocol stack, http://bluez.sourceforge.net/

[3]GTK+: The GIMP Toolkit, http://www.gtk.org

[4]Palowireless: Bluetooth Resource Center, http://www.palowireless.com/infotooth/whatis.asp

[5]Telefonaktiebolaget LM Ericsson, International Business Machines Corporation, Intel Corporation, Nokia Corporation, Toshiba Corporation, "Specification of the Bluetooth System, Specification Volume 1, Profiles", Version 1.0 b, December 1999.

[6]J. Bray and C. F. Sturman, "Bluetooth: connect without cables", Prentice Hall PTR, New Jersey, 2001

[7]B. A. Miller and C. Bisdikian, "Bluetooth revealed", Prentice Hall PTR, New Jersey, 2001

[8]The ITU Telecommunication Standardization Sector (ITU-T), "Q.931, Digital Subscriber Signaling System No. 1(DSS 1) — ISDN User-Network interface Layer 3 Specification for Basic Call Control", March 1993

[9]P. Ekstrom, F. Hoel, "Audio over Bluetooth and MOST", Master Thesis, Sweden, 2002,

Appendix A. Content of hciusb.patch

The content of the hciusb.patch is shown in Figure 39.

```
--- linux-2.4.22/drivers/bluetooth/hci_usb.c   Mon Aug 25 13:44:41 2003
+++ linux-2.4.22.patch/drivers/bluetooth/hci_usb.c    Tue Oct  7 09:00:25
2003
@@ -302,7 +302,9 @@

 #ifdef CONFIG_BLUEZ_USB_SCO
              if (husb->isoc_iface)
-                  hci_usb_isoc_rx_submit(husb);
+               for (i = 0; i < HCI_MAX_ISOC_RX; i++)
+                 hci_usb_isoc_rx_submit(husb);
+
 #endif
       } else {
              clear_bit(HCI_RUNNING, &hdev->flags);
@@ -522,10 +524,10 @@
 #ifdef CONFIG_BLUEZ_USB_SCO
              /* Process SCO queue */
              q = __transmit_q(husb, HCI_SCODATA_PKT);
-             if (!atomic_read(__pending_tx(husb, HCI_SCODATA_PKT)) &&
-                        (skb = skb_dequeue(q))) {
-                   if (hci_usb_send_isoc(husb, skb) < 0)
-                        skb_queue_head(q, skb);
+             if (atomic_read(__pending_tx(husb, HCI_SCODATA_PKT)) <
HCI_MAX_ISOC_TX &&
+                 (skb = skb_dequeue(q))) {
+               if (hci_usb_send_isoc(husb, skb) < 0)
+                 skb_queue_head(q, skb);
              }
 #endif

@@ -830,8 +832,13 @@

 #ifdef CONFIG_BLUEZ_USB_SCO
                             case USB_ENDPOINT_XFER_ISOC:
-                                  if (ep->wMaxPacketSize < size)
-                                        break;
+                               /* Use only the 9 byte
+                                  "One voice channel with 8 bit encoding"
+                                  endpoint until there is support for
changing
+                                  the endpoint dynamically. See
+                                  Bluetooth 1.1 Part H:2, section 2.1 */
+                               if (ep->wMaxPacketSize != 9)
+                                 break;
                                   size = ep->wMaxPacketSize;

                                   isoc_iface = iface;
--- linux-2.4.22/drivers/bluetooth/hci_usb.h   Fri Jun 13 16:51:32 2003
+++ linux-2.4.22.patch/drivers/bluetooth/hci_usb.h    Tue Oct  7 09:00:44
2003
@@ -41,6 +41,8 @@
 #define HCI_MAX_BULK_TX          4
 #define HCI_MAX_BULK_RX          1

+#define HCI_MAX_ISOC_RX          2
+#define HCI_MAX_ISOC_TX          2
 #define HCI_MAX_ISOC_FRAMES      10

 struct _urb_queue {
```

Figure 39: Content of hciusb.patch

Appendix B. Recompile the Linux Kernel

One example of install ALSA and recompile the Linux kernel is shown in Figure 40.

```
tar -xvf alsa-tools-0.9.6.tar
cd alsa-tools-0.9.6
cd envy24control/
./configure --disable-alsatest
make
make install
cd ../as10k1
./configure
make install
cd ../..

tar -xvf alsa-utils-0.9.6.tar
cd alsa-utils*
./configure --disable-alsatest
make
make install

cd /usr/src/linux
patch -p1 < hciusb.patch
make xconfig
make dep
make clean
make bzImage
make modules
make modules_install
make install
reboot
```

Figure 40: Install ALSA and Recompile the Linux Kernel

Appendix C. Install and Configure BlueZ

C.1 Install BlueZ

The steps to install BlueZ packages are shown in Figure 41.

```
cp /home/download/bluez*.* ./
tar -xzvf bluez-libs-2.4.tar.gz
cd bluez-libs-2.4
./configure
make
make install
cd ..
tar -xzvf bluez-utils-2.3.tar.gz
cd bluez-utils-2.3
./configure
make
make install
cd ..
tar -xzvf bluez-pan-1.1.tar.gz
cd bluez-pan-1.1
./configure
make
make install
cd ..
tar -xzvf bluez-sdp-1.5.tar.gz
cd bluez-sdp-1.5
./configure
make
make install
cd ..
tar -xzvf bluez-hciemu-1.0.tar.gz
cd bluez-hciemu-1.0
./configure
make
make install
cd ..
tar -xzvf bluez-hcidump-1.5.tar.gz
cd bluez-hcidump-1.5
./configure
make
make install
cd ..
tar -xzvf bluez-bluefw-0.9.tar.gz
cd bluez-bluefw-0.9
./configure
make
make install
```

Figure 41: One Example to Install BlueZ package

C.2 Change the Module Configuration File

Add the following lines into file module.conf under directory of /etc/:

alias net-pf-31 bluez

alias bt-proto-0 l2cap

alias bt-proto-2 sco

alias bt-proto-3 rfcomm

alias bt-proto-4 bnep

alias tty-ldisc-15 hci_uart

C.3 Change the Bluepin for Pairing

Replace the content of file bluepin under directory /bin with the following script.

!/bin/sh –e

echo "PIN:0000"

Please notice that 0000 is the PIN number. It is defined by the headset. Different headset may have different PIN number.