Mail Application Development

For Support of the

Bluetooth Protocol

On Windows Operation System

A Report Submitted
By
Zen-Jerr Hong

In Partial Fulfillment of the Requirements
For the Degree of

Master of Science
In
Electrical Engineering

Tufts University
Dec 2001

Advisor: Dr. C. Hwa Chang

List of Contents

List of Figures

List of Tables

Abstract

"Mail Application Development For Support of the Bluetooth Protocol On Windows Operation System"

by

Zen-Jerr Hong

Advisor: Dr. C. Hwa Chang

The Bluetooth technology is a short-range, low-power wireless communication link, operating in the unlicensed band at 2.4 GHz using a frequency hopping transceiver. It allows real-time voice and data communications between Bluetooth enabled devices. The objective of this project is to develop an email application on top of the Bluetooth protocol based on client and server architecture. The devices used are two Ericsson Bluetooth Application and Training Tool Kits that include an USB-interfaced Bluetooth module and the Ericsson Bluetooth PC Reference Stack. The Bluetooth PC reference Stack is a complete function library that provides developers a convenient and easy access to higher-level Bluetooth protocols in the Windows NT, 98 or 2000 environment. The final delivery of this project is two sets of Win32 applications developed under MS Visual-C++ 6.0 and MFC framework. The mail client application can accept user data from its GUI, make a connection to the server and then transmit the user data to the server application through the Bluetooth link wirelessly. Then the server application can communicate with any SMTP server through its Internet connection and send out the mail to the destination.

Mail Application Development For Support of the Bluetooth Protocol On Windows Operation System

1. Introduction

1.1 Project Overview

This is a Bluetooth application development project. The software developing process is based on the Waterfall model. The devices used are two Ericsson Bluetooth Application and Training Tool Kits that each of them includes an USB-interfaced Bluetooth module and the Ericsson Bluetooth PC Reference Stack. The Bluetooth PC reference Stack is a complete software library that provides developers a convenient and easy access to higher-level Bluetooth protocols in the Windows NT, 98 or 2000 environment hereafter referred as Windows or Win32 environment. Because current version of Microsoft Winodws Operating System does not have direct support of the Bluetooth technology, the Bluetooth link can't be used as the physical layer and the data link layer of the Internet connection directly. In order to use enormous amount of Internet services with a Bluetooth enabled device, we need to build a bridge to transmit the data to an Internet access point. The objective of this project is to implement the high-level Internet user protocols like TELNET, FTP, HTTP, SMTP, etc, on top of the Bluetooth link. Here we selected the SMTP protocol in this project to demonstrate this concept. Using the client and server structure, the software prototype realized the wireless email solution. We started this project by examining and modifying the source codes of the sample chat application that come with the Bluetooth training kit. The final delivery of this project is two sets of Win32 prototyped applications developed under MS Visual C++ 6.0. The mail client application, hereafter referred as BTMailClient, can accept user data from its graphical user interface, make a connection to the server and then transmit the user data to the server application through the Bluetooth link wirelessly. Then the server application, hereafter referred as BTMailServer, can communicate with any SMTP server through its Internet connection and send out the mail to the destination.

1.2 Bluetooth Technology Overview

The Bluetooth technology is a short-range, low-power wireless communication link, operating in the unlicensed band at 2.4 GHz using a frequency hopping transceiver. The specification of this technology is provided by the Bluetooth Special Interest Group (SIG). The Bluetooth SIG promoters include 3Com, Agere, Ericsson, IBM, Intel, Microsoft, Motorola, Nokia and Toshiba, and hundreds of Associate and Adopter member companies. All the hardware that complies with the Bluetooth wireless specification can enable users to connect a wide range of computing and telecommunications devices easily and simply, without the need to buy, carry, or connect cables. It delivers opportunities for rapid ad hoc connections, and the possibility of automatic, unconscious, connections between devices. Within the working range, it can virtually eliminate the need to purchase additional or proprietary cabling to connect individual devices. Each Bluetooth device can connect to up to 7 other Bluetooth enabled devices simultaneously.

Currently, there are hundreds of new Bluetooth products available, including computer add-on adapter, notebook computer, handheld devices, cellular phone, headset, etc. Because Bluetooth can be used for a variety of purposes, it will potentially replace multiple cable connections via a single radio link. It creates the possibility of using mobile data in a different way. We can expect more new Bluetooth products coming out in the near future. See appendix A for the technical summary of Bluetooth technology.

1.3 Project Devices Overview

The devices used in this project are two PCs with Windows 2000 system and two Ericsson Bluetooth Application and Training Tool Kits. The Bluetooth Tool Kit is an educational product designed for schools and universities, provides low cost, convenient hands-on training involving Bluetooth wireless technology. The hardware included in the tool kit consists of a Bluetooth module, which can be connected to the host computer with a USB connection. The software included in the tool kit consists of the Ericsson Bluetooth PC Reference Stack and 2 sample applications with their source code. It also contains the full Bluetooth Specification Version1.1 and the full documentation of the Bluetooth PC Reference Stack. The PC Reference Stack is a complete set of Application Programming Interface (API) that provides access to the different layers of the Bluetooth protocol stack in WIN32 environment. The API is in the form of C function library.

We used two sets of the training kit in this project. There is one difficulty we met that these two modules were not purchased from Ericsson at the same time. The second set was obtained more than three months after the first one. So they come with different version of Bluetooth PC References Stacks. After some testing, we found that in order to function properly, the older module has to work with the old version of APIs. And the new module has to work with the newer version of APIs as well. Hence we setup the developing environment on two PCs. One is set particularly with the newer version of APIs to develop BTMailServer application and another is setup with the old version of APIs to develop BTMailClient application. So the two Bluetooth modules are not exchangeable in this project.

1.4 Development Platform Overview

The development platform of this project is the Win32 platform. All the source codes are written and maintained under the Microsoft Visual C++ 6.0 projects. All the executables can be complied from the source code files by the Visual C++ 6.0 compiler. Microsoft Visual C++ 6.0 is an integrated development environment for C++ software developing in Win32 environment. It provides whole set of visualized tools to simplify the procedure of creating the GUI and also includes a rich variety of professional tools to help the developers working on C, C++, and many other professional technologies such as MFC, OLE, ODBC, ActiveX, and COM. In the Ericsson Bluetooth Training Tool Kit, the Bluetooth PC Reference Stack is implemented as a COM server object and all the APIs are provided as C functions. The application we developed works as a COM client and it gets required services from the PC Reference Stack (COM server). In order to simplify the windows programming process, we used MFC framework to build up the structure of the application. The MFC is a set of well-defined classes that encapsulate a large portion of the Win32 API. Some of the classes encapsulate application concepts such as documents, views, and the application itself. By using this Visual C++ integrated environment and MFC, we can create and edit the resources like dialogues, bitmap images and icons very easily and also isolate the handling of lower level Win32 API calls. Therefore our focus of this project can be put on the implementation of Bluetooth communication and the target Internet protocol. See appendix B for the coding standard of this project.

1.5 Software Life-Cycle Models and Documents Overview

Because of the sample files provided by the training tool kit, this project is based on a build and fix model with prototype. The sample chat application suite that came with the Bluetooth module is our first prototype. The chapter 2: Requirement Specifications and Verification Plan is an independent document that addresses all the desired requirements and the restrictions on this project before the work began. Chapter 3 is a macro-level description of the system architecture. Chap 4 and Chap 5 describe the design and the implementation of the two most important functionalities of this application, the Bluetooth connection and the email service. Chapter 6 is a complete user manual with all the screen shots for the final delivery. Chapter 7 is the conclusion of this Bluetooth project.

2 Requirement Specifications and Verification Plan

2.1 Introduction

2.1.1 Document Objective

The purpose of the requirement specifications and verification plan is to provide a clarification of ideas and an agreement between the software developers and the client regarding the application that is to be developed. Since this project is a research oriented project that no actual products are provided to an end client, there won't be too much confusion or any miscommunication between the developers and the client. Hence, the objective of this document is to provide the members of the development group a common guideline of the project itself.

2.1.2 Audience

The function of this technical document is to provide a clear explanation of the system that we will develop. So, the document should be accessible to any potential reader, with any skill level.

2.1.3 Description of System

The system includes two PCs and two Bluetooth modules. Each Bluetooth Module connects to a PC via the USB interface. One PC that works as the BTMail server should have a stable Internet connection. The other PC that works as the BTMail client can be a desktop or a laptop computer without Internet connection. The client will communicate with the server by Bluetooth connection and the server will request SMTP service from the Internet. The BTMail server itself works like a network gateway.

2.1.4 System Objectives

The objective of this system is to build a wireless email solution on top of the Bluetooth protocols. Using a wireless link between two Bluetooth devices, the mail client can send out the user message to the server without cable connection. Then the mail server would forward the user message to a SMTP server via its Internet connection.

2.1.5 The User

As mentioned before, the project is experimental oriented. So the potential users will be the developers themselves and other members in the EECS department.

2.2 Core Requirements

2.2.1 Bluetooth Connection

Within 10-meter range, the BTMail client should have the ability to interact with the BTMail server by the Bluetooth link So the user data can be transmitted to the server without any cable connection.

2.2.2 SMTP Connection

The BTMail server application should be able to communicate with any valid SMTP server on the Internet and send out the user message to the destination.

2.2.3 User Friendly

The email service is an important communication media in our daily life. More and more people use it as a major communication tool. The system should be able to be used by any user of even the most minimal technical skill. The User Interface should be easy to understand and easy to use.

2.2.4 Message Log

Considering the security issues and the troubleshooting, keeping a system log of any sending or receiving activities on the BTMail server is an important feature for the system administrator.

2.3 Gray Requirements

2.3.1 Instant Message From Server

This is a feature that enables the operator of the BTMail server to send any instant message to the BTMail client. So the operator of the server can help users on any syntax error in the required data field or notice the users any kind of information they may need.

2.3.2 Multiple Recipients

The ability to send out a message to multiple recipients will be handy under some situations.

2.3.3 Attachment Handling

Not only the text message can be sent through the Bluetooth link, but also any size of attached binary files can be handled by the application.

2.4 Restrictions

2.4.1 Software

The Ericsson Bluetooth PC Reference stack is implemented as a COM object and all the APIs to the Bluetooth protocols are in form of C library. The existing source codes were written and compiled in Visual C++ 6.0. So the software development environment has been fixed on the Microsoft Visual C++ upon the Win32 operation system.

2.4.2 Hardware

The hardware available in this project are 2 PCs and 2 Ericsson Bluetooth Application and Training Kits. Each Bluetooth module connects to a PC via the USB connection. One PC that works as the BTMail server should has a reliable Internet connection. Another PC that wokrs as the BTMail client can be a desktop or a labtop without Internet connection. The limited hardware also constraints the implementation of the multi-tasking and multi-thread support. In the definition of the Bluetooth specification, a Bluetooth device should be able to connect to 7 different Buetooth enabled devices simultaneously. Because we only have two modules available, without the additional devices for testing, the application can only handle one connection at the same time.

2.4.3 Time

Just like any project with a deadline, the number of features one can implement and the extent to which these features can be tested is limited by the amount of time one has to work on the project.

2.5 Verification Plan

2.5.1. Purpose

Normally, the verification plan is used as a guideline to examine the software requirements with the client's need. In this project, the client is the development team itself, so the purpose of this plan is mainly to make sure the progress of the application development can follow the predetermined schedule. So this section of documentation will highlight our plans to verify that our intentions match the desires, so that we will be able to provide the final delivery with the highest quality possible.

2.5.2 Meeting Schedule and Communication Channel

The development team members and the team advisor will meet once a week to discuss current progress on the project and also share experiences and problems we meet during the work. Except the weekly meeting, the team members should use the email as a communication channel to exchange any information as needed. It's very important to have a good communication between all developers so that everyone can be synchronized to the most current state of the project.

2.5.3 Required Resources

As mentioned in chapter 1, the project required two PCs with Windows operating systems and Visual C++ 6.0 installed, and two Ericsson Bluetooth modules. The devices are located in lab room 229 in EECS department. Other resources might be located while needed from the department too.

2.5.4. Timeline

The deadline of the project is due on the end of December 2001. All the software development, documentations and presentation should be done by the deadline. The development team should put all the required effort on the project to meet the timeline.

3. System Architecture

3.1 System Overview

The system includes two PCs and two Bluetooth modules. Each Bluetooth Module connects to a PC via the USB interface. One PC that works as the BTMail server should have a stable Internet connection. The other PC that works as the BTMail client can be a desktop or a laptop computer without Internet connection. The client will communicate with the server by Bluetooth connection. The server will request SMTP service from the Internet and send the user message to an Internet SMTP server. The BTMail server itself works like a network gateway. Figure 3.1 is the overview of the system

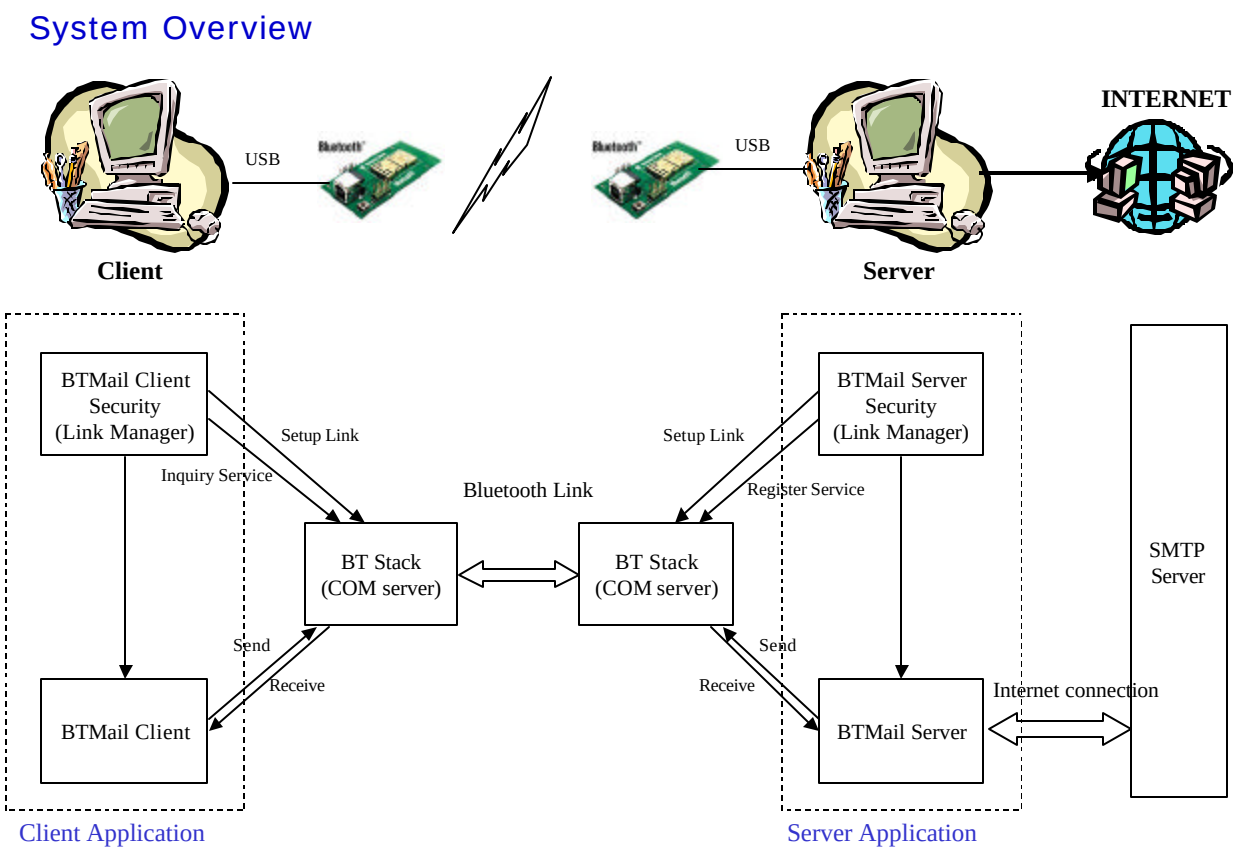

Figure 3.1 System overview

In the above figure 3.1, we can see that both BtMail Server. And BTMail Client have two programs, the security program handles the setting up of the Bluetooth link, and the other program is in charge of the specific functionality we want to implement, in this project, the email service. The reason why we separate the Bluetooth link setting up in the security program is to make the system very expandable. We can add any other services in this architecture without rewriting the Bluetooth connection setting code. We know that Bluetooth devices can connect to up to 7 devices at the same time. It is also possible that the Bluetooth server can provide more than one service concurrently. In this configuration, the security program itself works as a link manager and other services, like FTP, HTTP, TELNET, etc can also be added to this system. To add a new service under this architecture, the only thing we need to do is adding a new service-specific program to handle the user interface. Figure 3.2 on the next page shows the system configuration after adding a possible HTTP feature. We can see that only one new programs need to be added to the server side and the client side. The architecture still remains the same. In the same manner, other higher level Internet protocols can be added into this application.

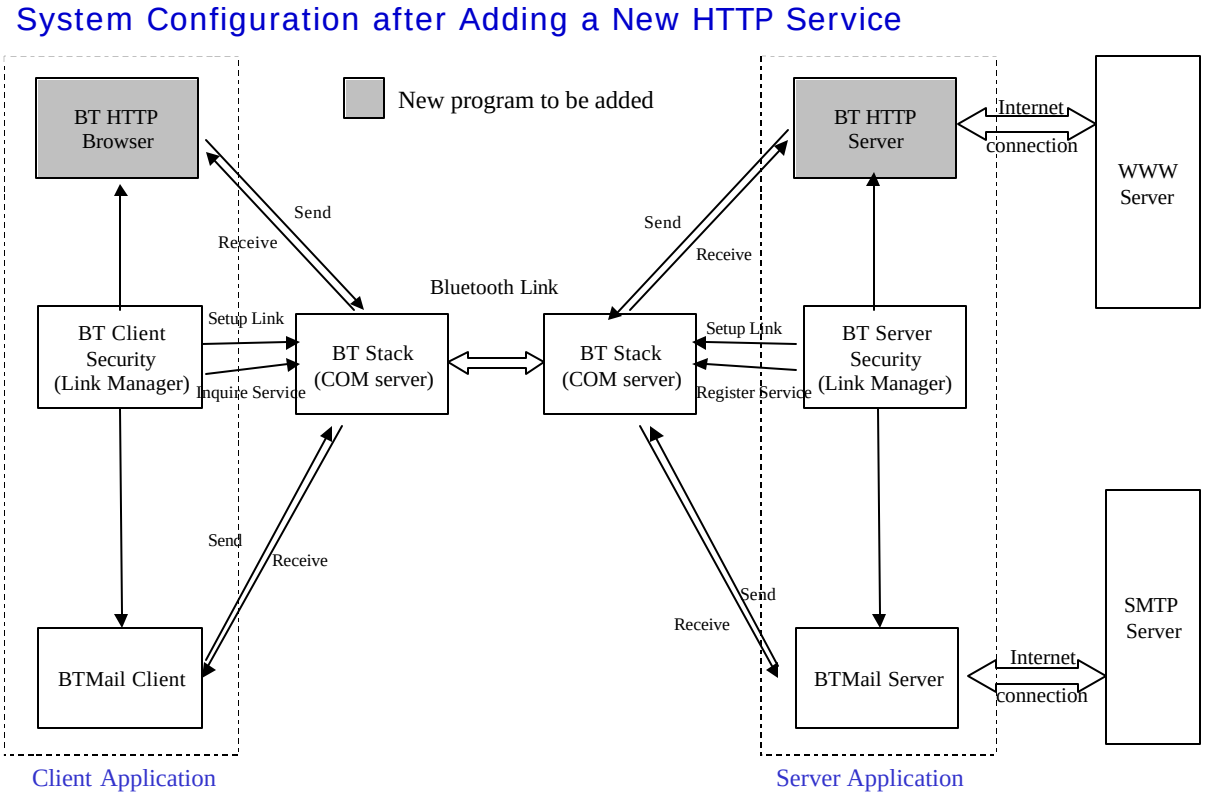

Figure 3.2 System configuration after adding a new service

3.2 System Level Description

Figure 3.3 shows the system level diagram of the Bluetooth application developed on top of the Ericsson Bluetooth PC Reference Stack. The blue block in the diagram is the user application and the yellow block is the Bluetooth PC Reference Stack, which is implemented as a COM server by Ericsson. The user application, works as a COM client, requests the service via the COM server interface to access to the API provided by the stack. Then the reference stack can control or modify the behavior of the Bluetooth Host on the module thorough the USB connection.

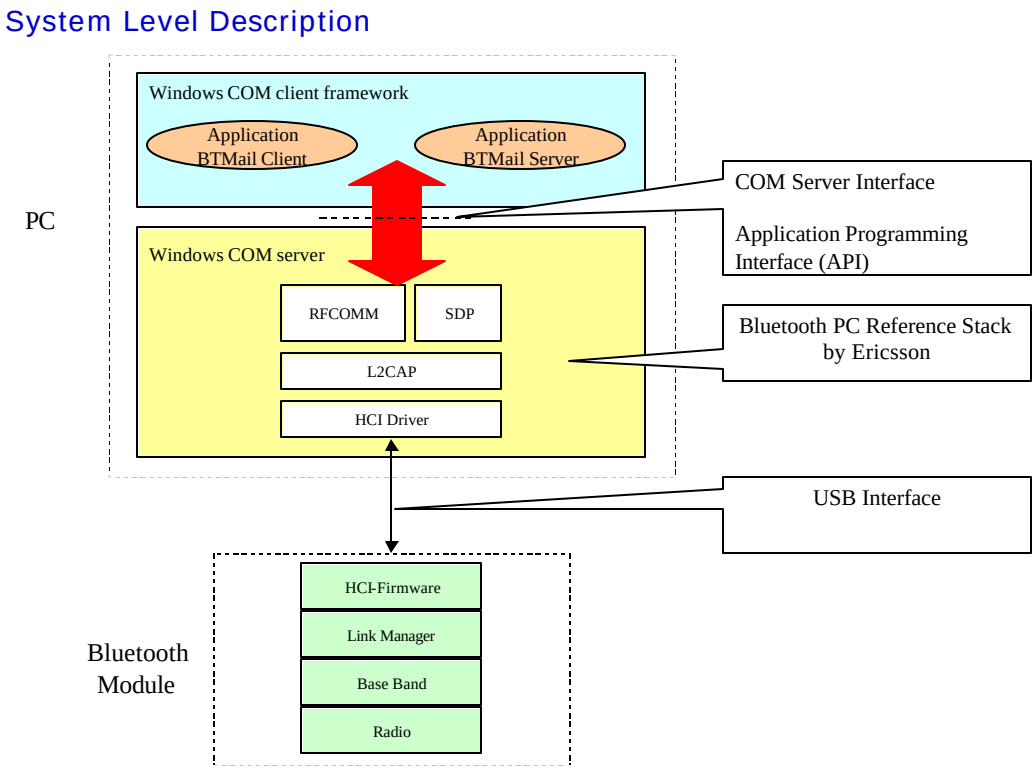

Figure 3.3 System level description

4. Bluetooth Connection

This chapter describes how a Bluetooth connection is made inside the BTMail application. The original chat sample application that came within the Bluetooth training kit has a shared chat security program to setup a server connection or a client connection. We separated these two functionalities of the program into two different programs. One for BTMail server and another for BTMail client. So the new BTMailServerSecurity program for BTMail server can only setup a connection as a server and the new BTMailClientSecurity program for BTMail client can only setup a connection as a client. Although we changed the structure of the system, all the required connection procedures are remained the same as in the sample Chat application. Hence, most contents of the first two sections 4.1 and 4.2 are quoted from the Ericsson Bluetooth PC Reference Stack User Manual Chapter 7.4. This is one of the most important parts of this project, so I kept this inside the contents of this documentation but not in the appendix.

The first two sections give an overview of the calls made to establish a Bluetooth connection. All calls will be described in sequence. All function calls are also commented in the source file with "CONNECT PROCEDURE: n", where n is the number ([n]) in front of each function depicted in this chapter (e.g. [1]SIL_SetDevice). All the details of the functions mentioned here can be found in the complete Ericsson Bluetooth PC Reference Stack User Manual Chapter 5:API.

The third section of this chapter describes the data packet format we used to transfer the user messages through the Bluetooth link and how it was implemented.

4.1 The BTMail Server Connection Procedure

Figure 4.1 is the complete Bluetooth connection procedure in the BTmail Server. The required function calls can be categorized into 4 steps as shown in different 4 colors in figure 4.1.

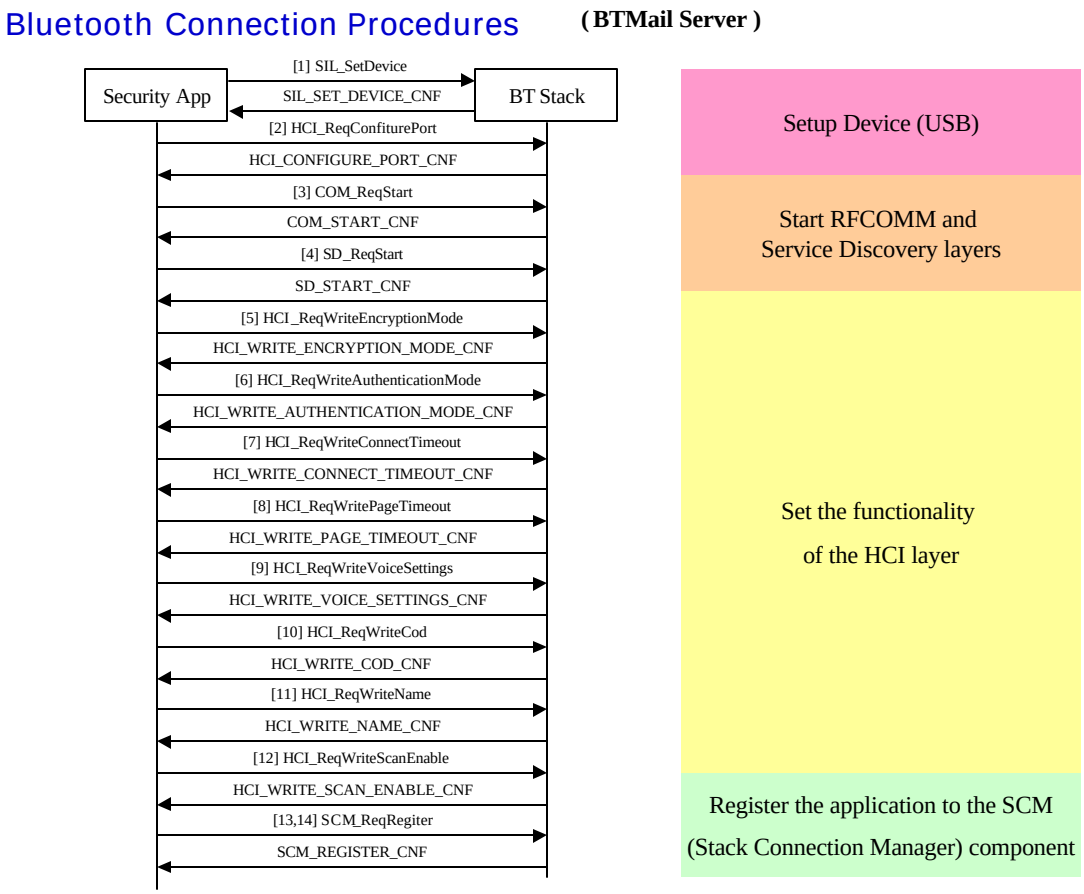

Figure 4.1 Bluetooth connection procedures – BTMail Server

4.1.1 Step 1: Setup Device (USB)

Figure 4.2 on the next page shows the function calls in step 1. The first call the security application makes is *SIL_SetDevice*. This is to set the type of device to be used, serial port or USB. If no call to *SIL_SetDevice* is made the default device is serial port. The next function call is *HCI_ReqConfiturePort*. This is to set the port configuration to be used. This function call has to be made even if the device to be used is USB.

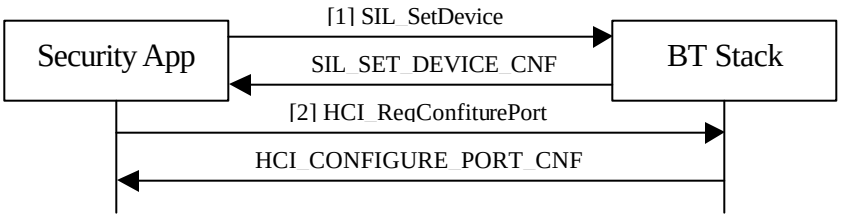

Figure 4.2 Bluetooth connection procedures – BTMail Server: Step 1

4.1.2 Step 2: Start RFCOMM and SD layers

Figure 4.3 shows the function calls in step 2. The next thing to do is to start the RFCOMM and Service Discovery layers by calling *COM_ReqStart* and *SD_ReqStart*, respectively. The RFCOMM layer is needed for the communication and the SD layer is needed in order for other Bluetooth devices to find this server.

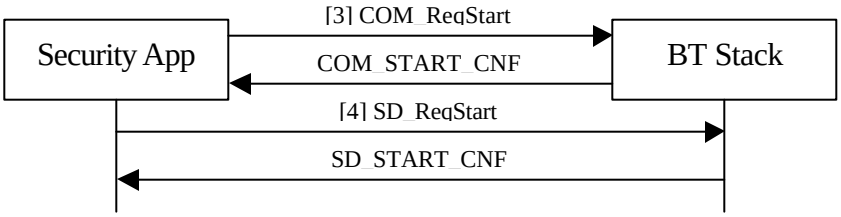

Figure 4.3 Bluetooth connection procedures – BTMail Server: Step 2

4.1.3 Step 3: Set the functionality of the HCI layer

At this point the security application will set the functionality of the HCI layer. This step 3 is done by calling several HCI layer functions as shown in Figure 4.4 on the next page.

The *HCI_ReqWriteEncryptionMode* function will set the encryption enabled or disabled (disabled in the BTMail application).

The *HCI_ReqWriteAuthentication* Mode function specifies whether authentication required or not (not required in the BTMail application).

The *HCI_ReqWriteConnectTimeout* function writes the connect timeout parameter (5 seconds in the BTMail application).

The *HCI_ReqWritePageTImeout* function writes the page timeout value (8 seconds in the BTMail application).

The *HCI_ReqWriteVoiceSettings* function write the voice settings (required function, but voice not used in the BTMail application).

The *HCI_ReqWriteCod* function writes the COD of the local device *(_tCod* in the BTMail application)

The *HCI_ReqWriteName* function changes the local name of the Bluetooth device ("BT Mail" in the BTMail application and the *HCI_WriteScanEnable* function writes the scan enable setting. (page and inquiry enabled in the BTMail application).

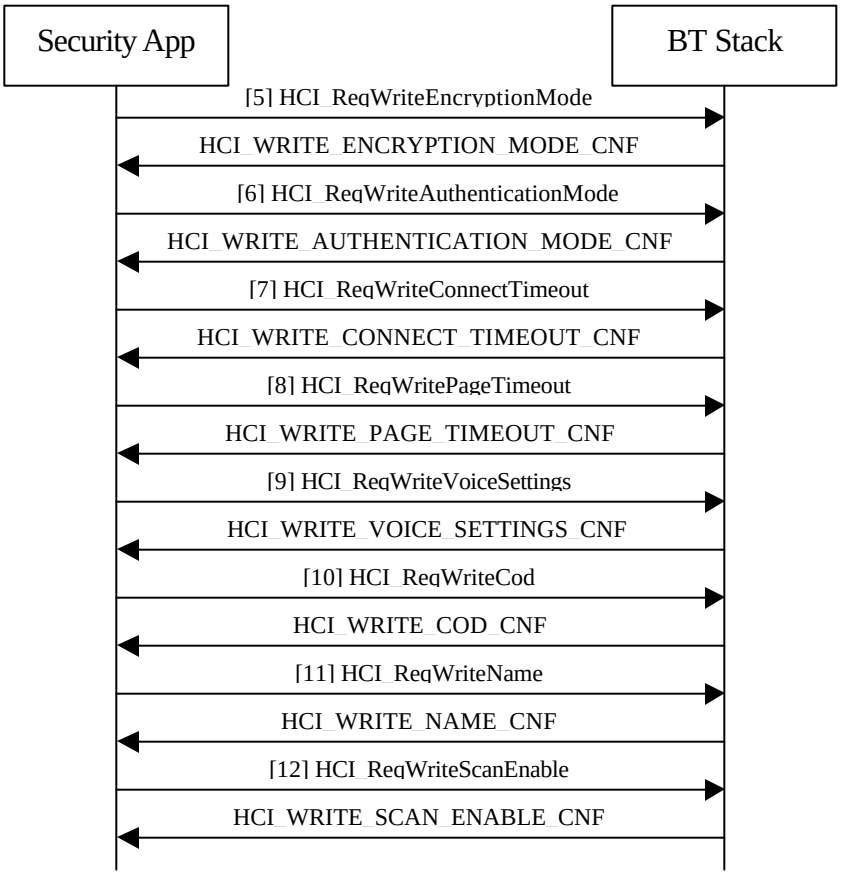

Figure 4.4 Bluetooth connection procedures – BTMail Server: Step 3

4.1.4 Step 4: Register the application to the SCM component

Figure 4.5 shows the function calls in step 4 of the connection procedures. In order to be able to respond to events, the applicaion needs to register with the SCM component. This is done with two calls to *SCM_ReqRegister*. The first call is to register the application as SCM_SEQURITY_HANDLER. The security handler accepts or rejects data links and handles the pin codes during pairing.

The second call is to register the applicaion as SCM_MONITOR_GROUP. This is to ensure that the application will be notified about created or destroyed links.

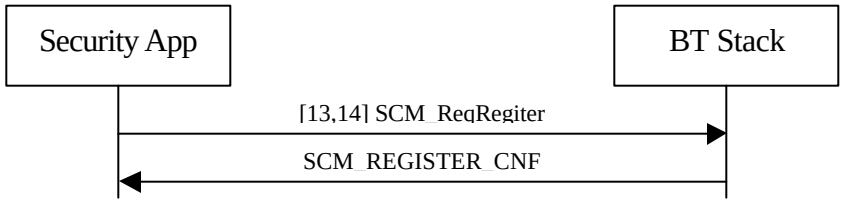

Figure 4.5 Bluetooth connection procedures – BTMail Server: Step 4

At this point the stack is ready for the BTMail server application, which will accept a connection from a Chat Client application upon request.

4.2 The BTMail Client Connection Procedure

Figure 4.6 is the complete Bluetooth connection procedure in the BTMail Client. In addition to the same connection procedure as in the Btmail Server, the BTMail Client needs to inquire and search the Bluetooth devices and services in the neighborhood. The required function calls to make a connection from the client can be categorized into 5 steps as shown in different 5 colors in figure 4.2.

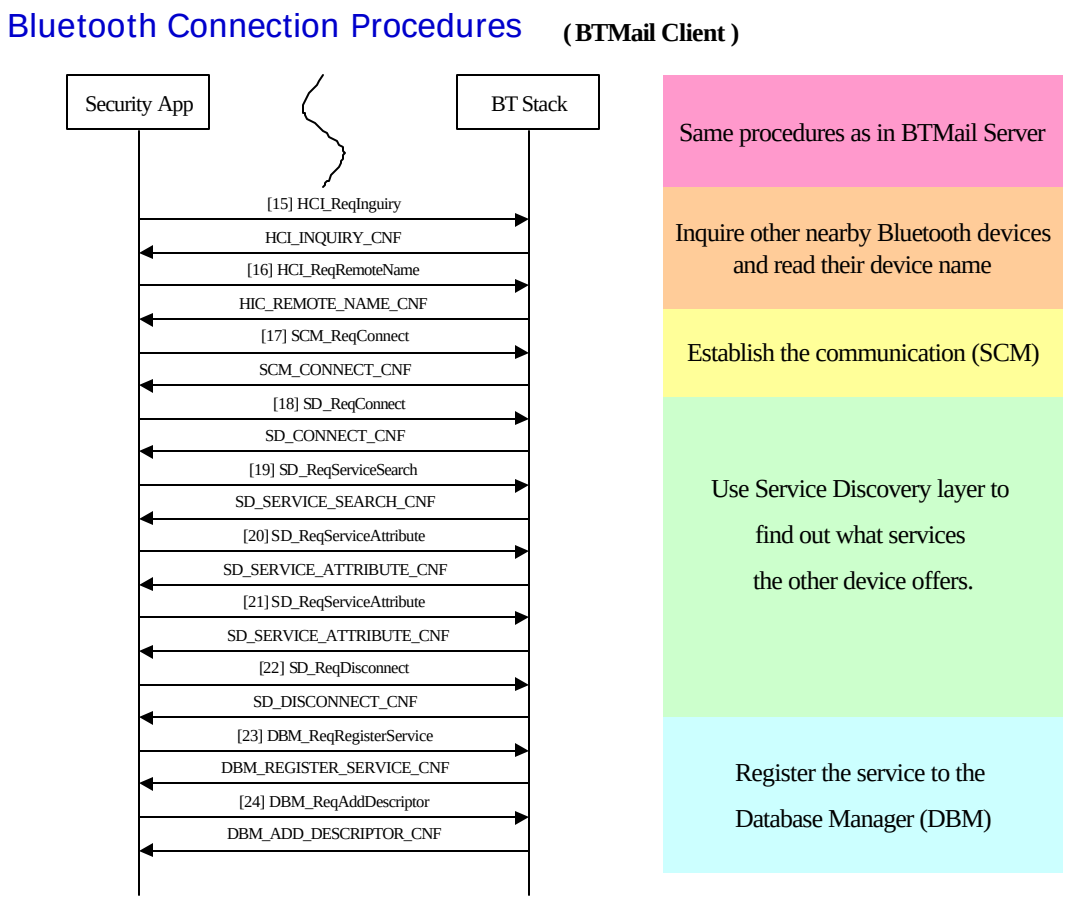

Figure 4.6 Bluetooth connection procedures – BTMail Client

4.2.1 Step 1: Same procedures as in BTMail server applicaion

All the function calls described in chapter 4.1 still needs to be called in the BTMail client to setup the Bluetooth link.

4.2.2 Step 2: Inquire nearby Bluetooth devices

In step 2, there are two more functions to be called for the HCI layer, *HCI_ReqInquiry* and *HCI_ReqRemoteName* as shown in Figure 4.7..

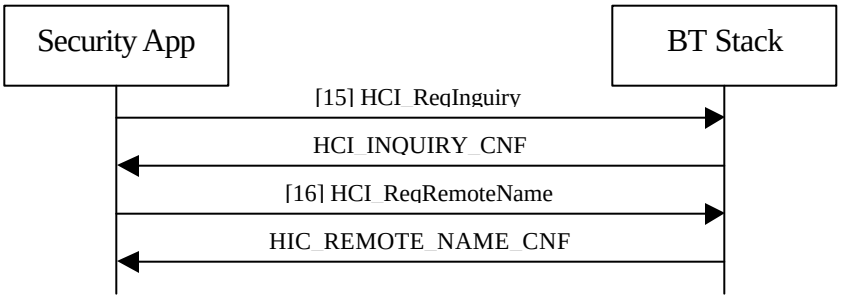

Figure 4.7 Bluetooth connection procedures – BTMail Client: Step 2

The *HCI_ReqInquiry* function is for discovering other nearby Bluetooth radios. The *HCI_ReqRemoteName* function is for reading the name of any other nearby Bluetooth devices.

4.2.3 Step 3: Establish the communication (SCM)

Figure 4.8 shows the function calls in step 3. In order to set up a data connection with another Bluetooth device, the function *SCM_ReqConnect* has to be called. This function establishes the communication between two devices.

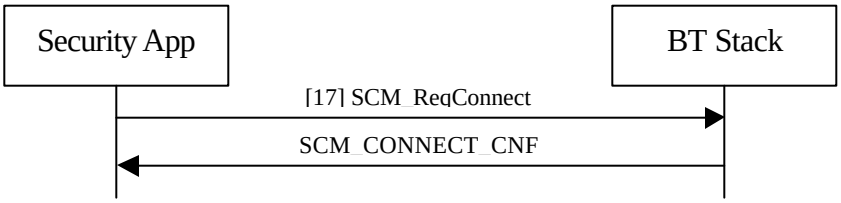

Figure 4.8 Bluetooth connection procedures – BTMail Client: Step 3

4.2.4 Step 4: Use Service Discovery layer to inquire services

Now when the connection is made, the BTMail client needs to find out what services the other device offers. The client security application will use the Service Discovery layer for this purpose. Figure 4.9 shows the function calls in this SDP session step.

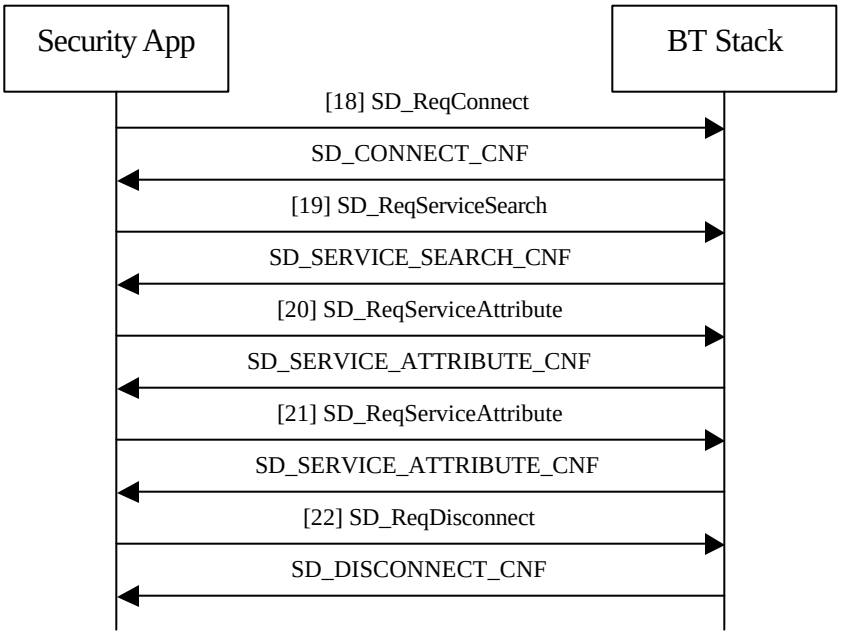

Figure 4.9 Bluetooth connection procedures – BTMail Client: Step 4

The *SD_ReqConnect* function will start an SDP instance for obtaining SDP Server Database values from a specific Bluetooth device.

The *SD_ReqServiceSearch* function is used to locate service records for a specific Bluetooth device.

The *SD_ReqServiceAttributes* function is called twice. The first call is to retrieve the name of the other Bluetooth device and the second is to retrieve a handle to that device.

The *SD_ReqDisconnect* function is called to disconnect from the Service Discovery protocol.

4.2.5 Step 5: Register the service to the Database Manager

Figure 4.10 shows the function calls in step 5. The remaining thing for the BTMail client security program to do before the stack is ready for the BTMail client application is to register this service to the Database Manager (DBM). This is done by functions *DBM_ReqRegisterService*, which will add this service to the database and the *DBM_ReqAddDescriptor*, which will add a service record for this service to the database.

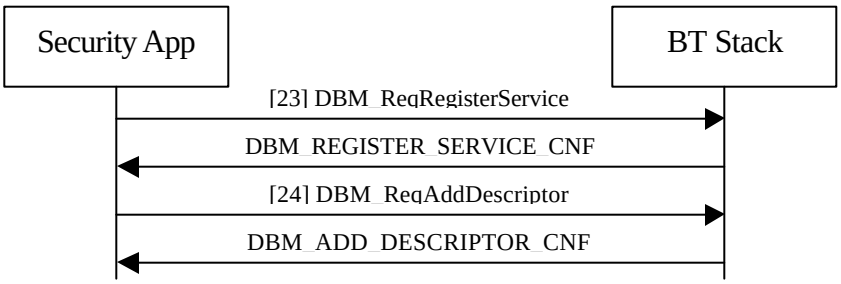

Figure 4.10 Bluetooth connection procedures – BTMail Client: Step 5

4.3 The BTMail Data Packet Format

The BTMail Client can get the email data from the user through its user interface as shown in Figure 4.11. After the Bluetooth connection is established between the BTMail Client and the BTMail Server, the BTMail Client then can send the user data to the BTMail server. There are two alternatives for transmitting the mail data. One is to send the whole data including all fields as a single data packet. Another way is to send each data field as an individual data packet.

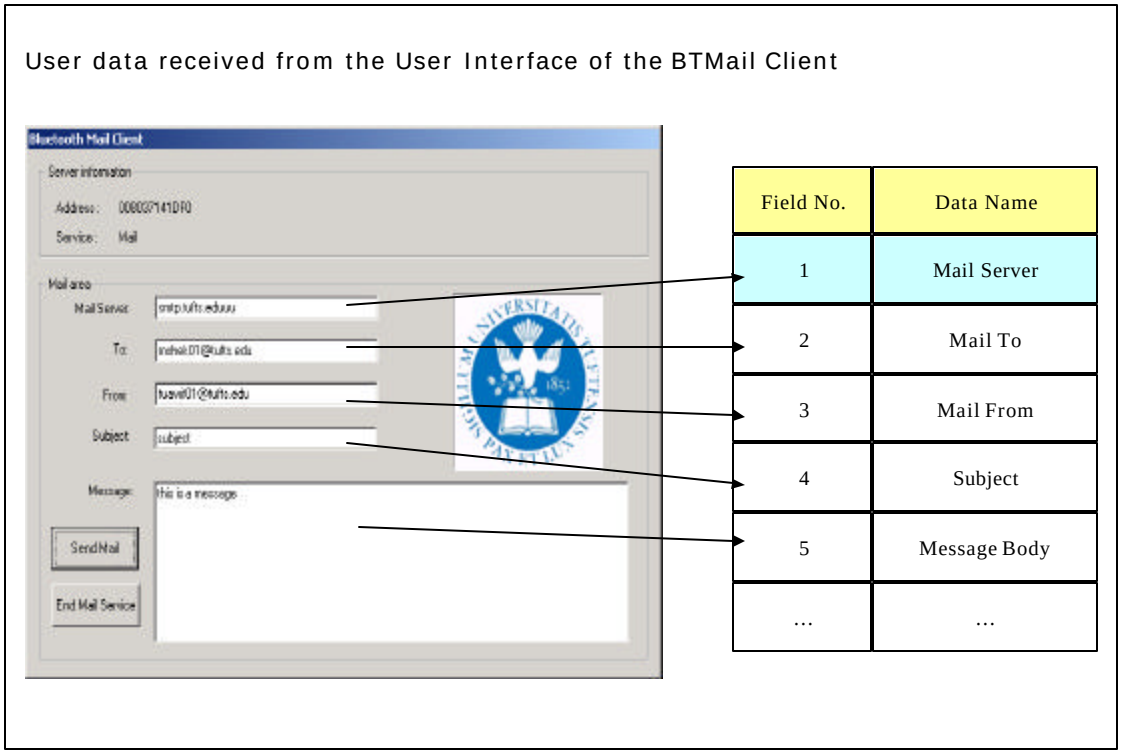

Figure 4.11 User data received from the GUI of BTMail Client

We selected the second method to send the user data, because it is more flexible and more expandable than sending the whole data as a single packet. In this schema, the server will receive several packets through the Bluetooth connection. In order to let the server recognize what data is inside each different packet, we need to provide a mechanism to carry the packet information. Here are the details about how we implemented it: first, we defined five different data packet types and each packet type is assigned to carry a different data field of the mail data as shown in Table 4.1.

Packet Type	Data Name	Data Type
1	Mail Server	String
2	Mail To	String
3	Mail From	String
4	Subject	String
5	Message Body	String

Table 4.1 Data packet type definition and carrying data assignment

Then we added an additional byte before the actual data to indicate what packet type it is. For example, if the data of the mail server field input from the user is “smtp.eecs.tufts.edu”, the data packet send from the client will be “1smtp.eecs.tufts.edu” as shown in Figure 4.12.

Ex. To send the data of a SMTP server address, an additional ‘1’ is added before the actual string.

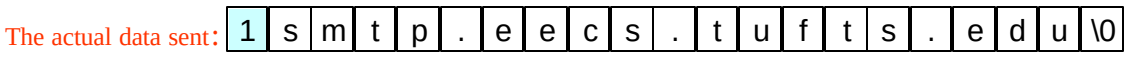

Figure 4.12 The actual data packet to send the SMTP server data

After the server receives the packet, it can look at the first byte of the packet and determine which data field the packet is carrying. Then remove the first byte and store the string as the mail server data. The advantage to use such a schema is that it’s easier to add other new data field into the application. There are totally 256 possible different packet types can be represented by using one byte. For example, to add the ability to handle attached files in the email application, we can add two new data packet types as shown in Figure 4.13 on the next page.

- For future enhancement, ‘6’ can be assigned to the file ID and the file name of the attachment and ‘7’ can be assigned to the actual data of the attached file.

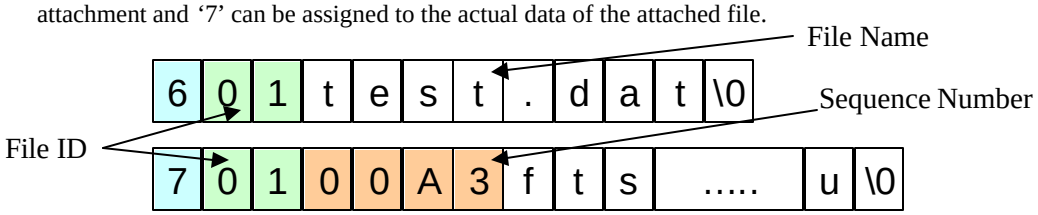

Figure 4.13 The data packet definition for attached files

To handle the attached files, one packet type ‘6’ can be added to carry the file ID and the filename and another packet type ‘7’ can be added to carry the actual file data. Using two bytes for identifying the file can represent at most 65536 files in a single mail. Because the attached file can be very large in size, it is necessary to split the file into small pieces before sending. The 4-byte sequence number in packet type ‘7’ is key to tell the server how to combine the receiving packets into a file in the correct order. The example shows how easy it is to add any additional data field without changing the existing data fields.

5. Email Handleing at BTMail Server

5.1 The SMTP Protocol Overview

The Simple Mail Transfer Protocol (SMTP) is a widely used Internet protocol defined in Internet RFC.821. The objective of the SMTP is to transfer mail reliably and efficiently. The protocol itself is independent of any particular transmission subsystem and requires only a reliable ordered data stream channel, usually a TCP connection is used.

The SMTP design is based on the following model of communication: as the result of a user mail request, the sender-SMTP establishes a bi-direction transmission channel to a receiver-SMTP. Normally, the receiver-SMTP listens for a TCP connection on a well-known port (25), and the sender-SMTP process initiates a connection on that port. The receiver-SMTP may be either the final destination or an intermediate. After the connection is established, SMTP commands are generated by the sender-SMTP and sent to the receiver-SMTP. SMTP replies are sent from the receiver-SMTP to the sender-SMTP in response to the commands. Table 5.1 lists the major commands in SMTP. More detail command syntax and information can be found in RFC 821. Basically, the sender-SMTP sends over the mail data to the receiver-SMTP by using the SMTP command in particular order defined in the SMTP specification.

Command	Description
HELO (HELLO)	This command is used to identify the sender-SMTP to the receiver-SMTP. The argument field contains the host name of the sender-SMTP
MAIL (MAIL)	This command is used to initiate a mail transaction in which the mail data is delivered to one or more mailboxes. The argument field contains a reverse-path.
RCPT (RECIPIENT)	This command is used to identify an individual recipient of the mail data; multiple recipients are specified by multiple use of this command.
DATA (DATA)	The receiver treats the lines following the command as mail data from the sender. This command causes the mail data from this command to be appended to the mail data buffer. The mail data may contain any of the 128 ASCII character codes.
RSET (RESET)	This command specifies that the current mail transaction is to be aborted. Any stored sender, recipients, and mail data must be discarded, and all buffers and state tables cleared. The receiver must send an OK reply.
VRFY (VERIFY)	This command asks the receiver to confirm that the argument identifies a user. If it is a user name, the full name of the user (if known) and the fully specified mailbox are returned.
EXPN (EXPAND)	This command asks the receiver to confirm that the argument identifies a mailing list, and if so, to return the membership of that list. The full name of the users (if known) and the fully specified mailboxes are returned in a multi-line reply.
HELP (HELP)	This command causes the receiver to send helpful information to the sender of the HELP command. The command may take an argument (e.g., any command name) and return more specific information as a response.
QUIT (QUIT)	This command specifies that the receiver must send an OK reply, and then close the transmission channel.
TURN (TURN)	This command specifies that the receiver must either (1) send an OK reply and then take on the role of the sender-SMTP, or (2) send a refusal reply and retain the role of the receiver-SMTP.

Table 5.1 Major SMTP commands

5.2 The SMTP Connection Procedure

After the BTMail server received the user data from the BTMail client through the Bluetooth link, the BTMail server will launch the SMTP connection procedures. Figure 5.1 shows the SMTP connection procedures in the BTMail server application.

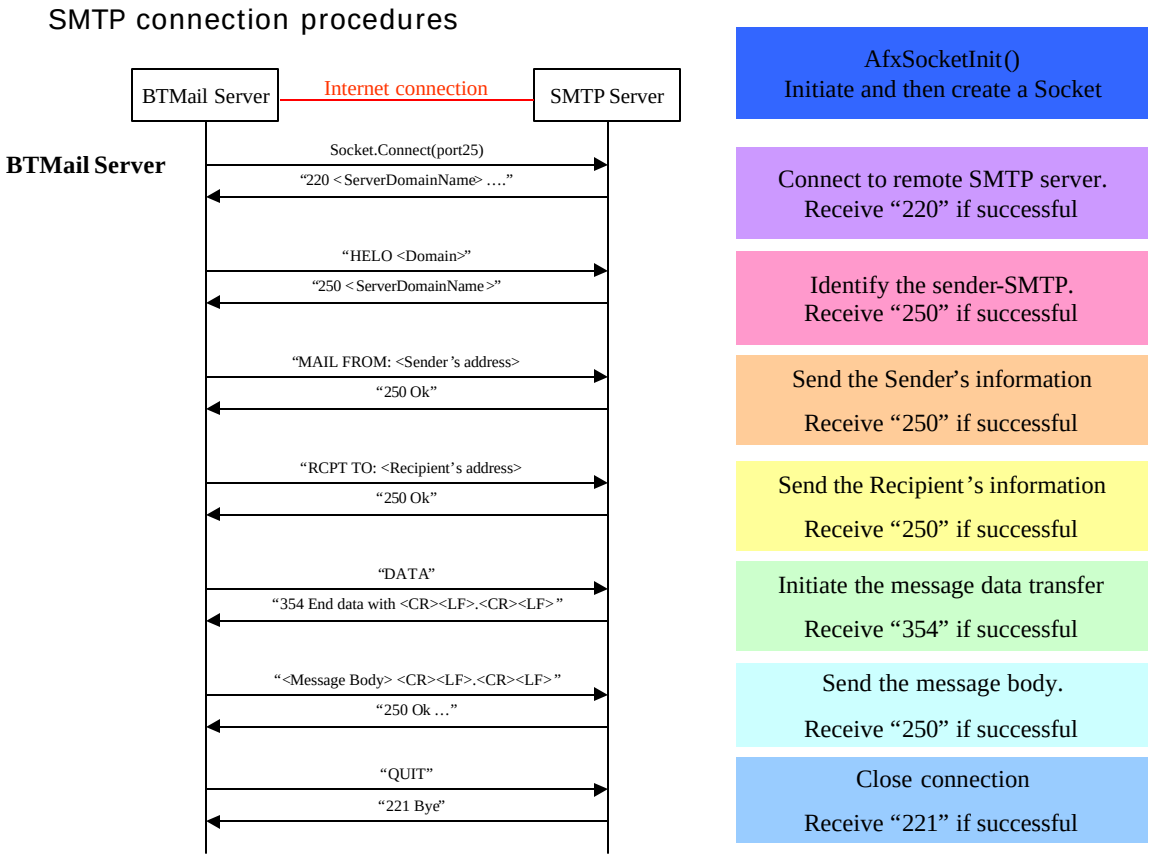

Figure 5.1 SMTP connection procedures

We implemented the SMTP connection procedures in a separate C++ function called Email(). Following is the detail description of the implementation in the SMTP procedures:

i. Call *AfxSocketinit()* to initiate the socket

ii. Create a socket object to handle the Internet connection

iii. Call *socket.open()* to Open the socket connection to the SMTP server at port 25

iv. Receive message from the SMTP server and compare it with the string "220".

v. Send the greeting message "HELO <BTMail Server Domain>" to the SMTP server

vi. Receive message from the SMTP server and compare it with the string "250".

vii. Send the sender's address to the SMTP server.

viii. Receive message from the SMTP server and compare it with the string "250"

ix. Send the recipient's address to the SMTP server.

x. Receive message from the SMTP server and compare it with the string "250"

xi. Send the command "DATA" to initiate the message data transfer

xii. Receive message from the SMTP server and compare it with the string "354"

xiii. Send the message body to the SMTP server. The message ends up with a special escape sequence "<CR><LF>.<CR><LF>"

xiv. Receive message from the SMTP server and compare it with the string "250"

xv. Send the disconnect request message "QUIT" to the SMTP server

xvi. Receive message from the SMTP server and compare it with the string "221"

There are several comparisons in the SMTP procedure. We use a data variable called "Error" to store the result of each comparison. By examining the value of the "Error" variable, the BTMail server can send back a status response to the client after the SMTP procedure.

5.3 SMTP Error Response and Message Log

As mentioned in Chapter 5.2, the BTMail server program can keep track of any connecting activities about its Bluetooth connection and the SMTP connection. The error status of the SMTP connection will be stored and the corresponding error message will be sent back to the client as a warning. We integrated a message area into the BTMail server's UI so that all the activities can be shown on the screen for monitoring. This feature can help the system administrator to notice any unusual connecting activities and also help the developers on debugging. The BTMail server not only prints the log message to the screen output, but it also writes the message into a log file as a permanent record for future references.

6. User Manual

The whole BTMail application suite includes two sets of applications, the BTServer set and the BTMail client set. Each set has two separate programs on top of the Bluetooth stack. The security program handles the bluetooth connection and the BTMail server or BTMail client program processes the high-level email functionality and the user interface.

Because the application works as a COM client on top of the Bluetooth PC Reference Stack which is implemented as a COM server, before executing the BTMail server program and the BTMail client program, the COM object (BT_comserver.exe) has to be executed on both PCs first. Then the two security programs can be launched on both PCs.

For information about the delivery file structure and the compiler settings for the BTMail Application Suite see appendix C.

6.1 BTMail Server Application Set

6.1.1 BTMail Server Security Program

Figure 6.1 is the screenshot of the BTMail ServerSecurity program. This program acts as the 'Link Manager' on top of the stack. It is in charge of setting up the Bluetooth connection at the server side. When the 'ServerSecurity' is started, the user has to click on the "USB port" check box to tell the program to start the Bluetooth connection procedure via the USB interface. After the stack has successfully connected to the Bluetooth device through the USB interface, the title bar should read 'CONNECTED TO DEVICE:' and the Bluetooth address for the device on the USB interface.

Right now, only one "mail service" is implemented in this application so only one 'Mail Service' button will be enabled. After the 'Mail Service' button is pushed, the mail server is registered to the Bluetooth stack and can be discovered by another Bluetooth device in the neighborhood, also the BTMail server program will be launched.

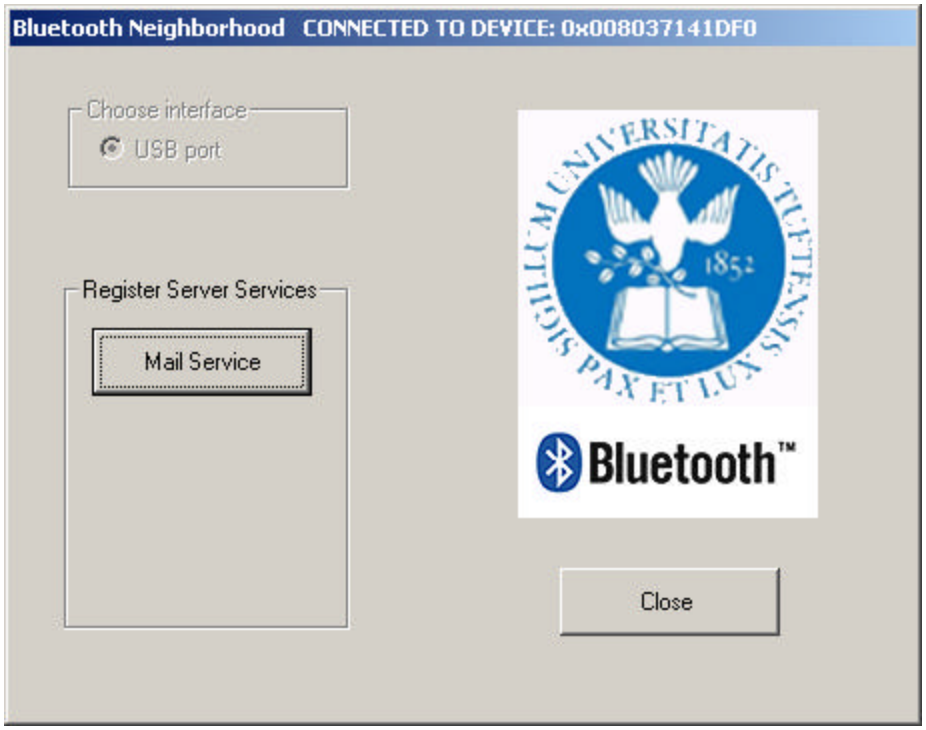

Figure 6.1 BTMail ServerSecurity User Interface

6.1.2 BTMail Server Program

Figure 6.2 is the screenshot of the BTMail server program. This program is lunched by the BTMail ServerSecurity program. The top of the window shows the client address and the service name. The middle portion is a message area that prints out all the connecting activities between the server and the client. It also prints out the result of the email sending procedure. The bottom is a input box for instant message feature. The server operator can type any message in the input area, and press 'enter' to send out the message to the client instantly. The feature is added to enable the server operator to help the client user online.

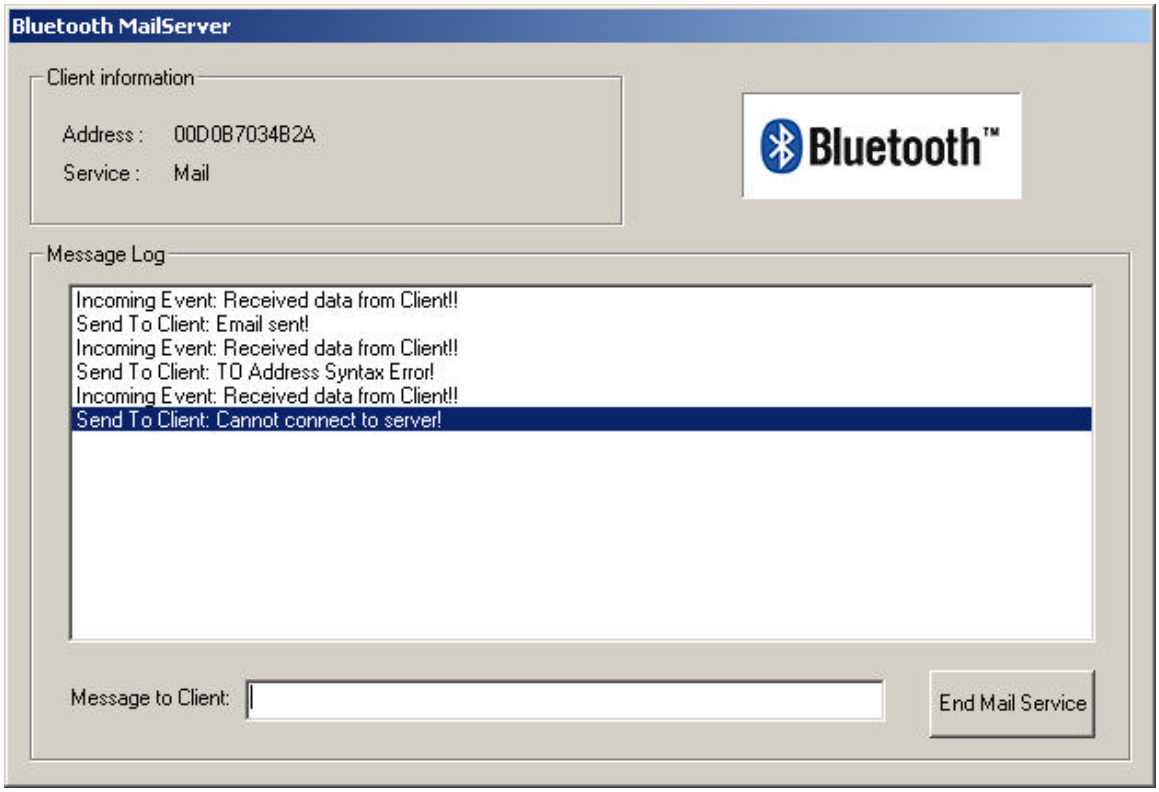

Figure 6.2 BTMail Server User Interface

6.1.3 BTMail Server Log File

Figure 6.3 is the screenshot of the BTMail server log file. The message log is written automatically by the BTMail server. All the connection activities and the messages between the server and the client are recorded alone with the happening time of the event.

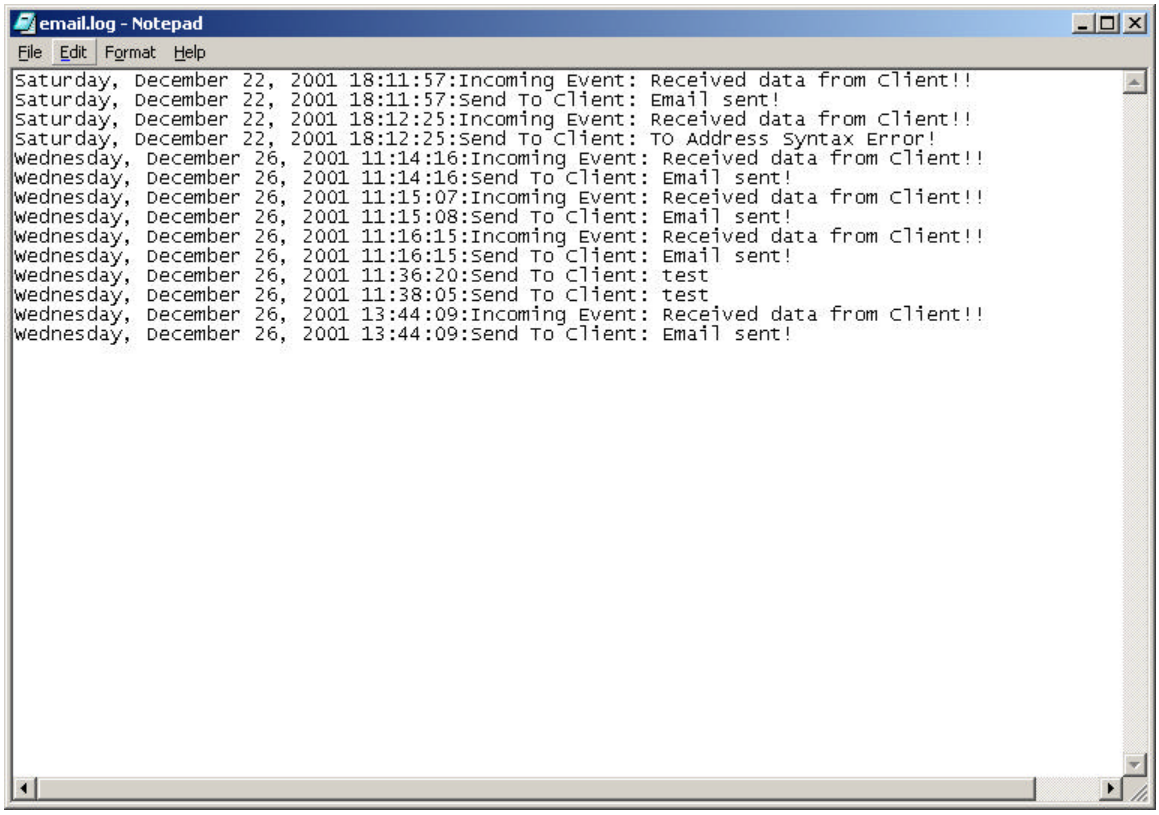

Figure 6.3 The message log of BTMail Server

6.2 BTMail Client Application Set

6.2.1 BTMail Client Sercurity Program

Figure 6.4 is the screenshot of the BTMail ClientSecurity program. This program acts as the 'Link Manager' on top of the stack. It is in charge of setting up the Bluetooth connection at the client side. When the 'ClientSecurity' is started, the program will register the application to the stack automatically and the stack will connect to the Bluetooth device through the USB interface.

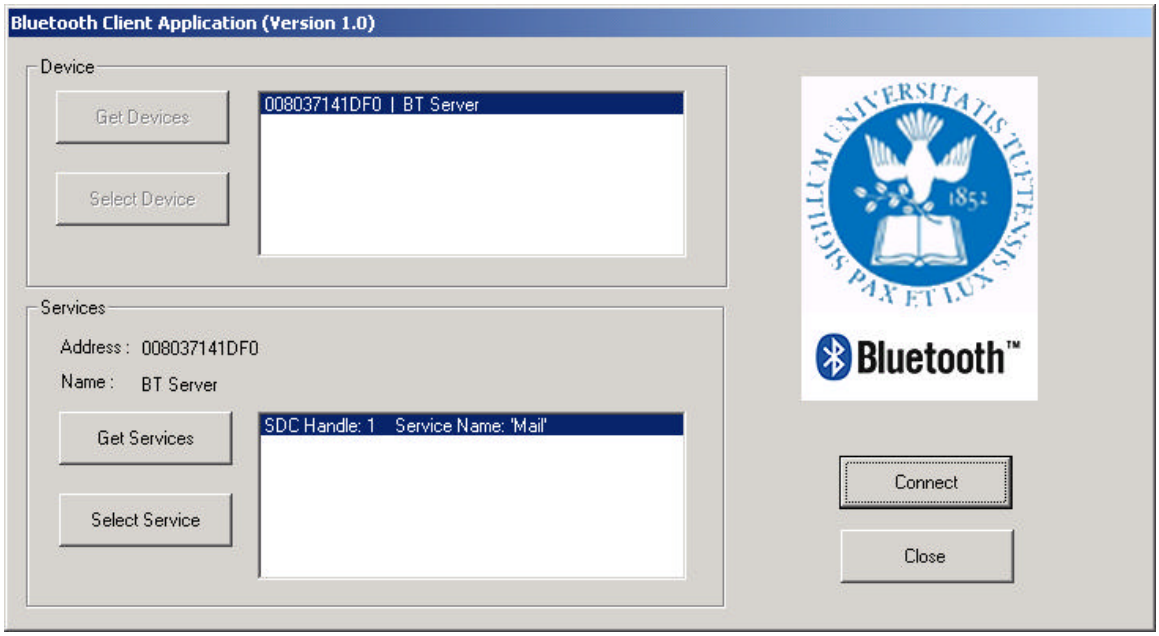

Figure 6.4 BTMail ClientSecurity User Interface

The window is divided into 2 main parts. The left part contains functionality that is necessary to initiate the Bluetooth connection at the client side. The upper part of it provides the functionality to discover all Bluetooth devices in the neighborhood via the 'Get Devices' button. The button is enabled automatically if the stack connects to the Bluetooth device successfully after the program launches. After the 'Get Devices' button is pushed by the user, all the Bluetooth devices detected in the neighborhood will be displayed in the list box and the 'Select Device' button will be

enabled.. As shown in Figure 6.4, both the detected device's name and address will be displayed. Then the user can select any one of the detected devices by highlight it and push the 'Select Device' button. Once a device is selected, the 'Get Services' button on the bottom part will be enabled.

The bottom part of the left portion of the windows provides the functionality to discover all the services on the selected device. The user can push the 'Get Services' buttons to inquire the services information form the remote device by starting an SDP session. The retrieved services information will be listed on the screen and the 'Select Service' button will be enabled. Then the user can select any one of the services by highlight it and push the 'Select Service' button. Once a service is selected, the 'Connect' button at the bottom right of the window will be enabled. Finally, the 'Connect' button can be pushed to launch the BTMail client program. The BTMail client program will pop up the user interface window on the screen.

6.2.2 BTMail Client Program

Figure 6.5 is the screenshot of the BTMail client program. This program is lunched by the BTMail ClientSecurity program. The top of the window shows the server address and the service name. The bottom of the window is the email user interface. Four input boxes and one text area are set on the UI to let the user input required information to send out an email. After the user fills out the form, the user can press the 'Send Mail' button to initiate the data transmission to the BTMail server. The BTMail.server will send back a response after the SMTP procedure. The result from the server will be shown in a pop-up dialogue. Any instant message sent by the server operator will be displayed in a pop-up window immediately.

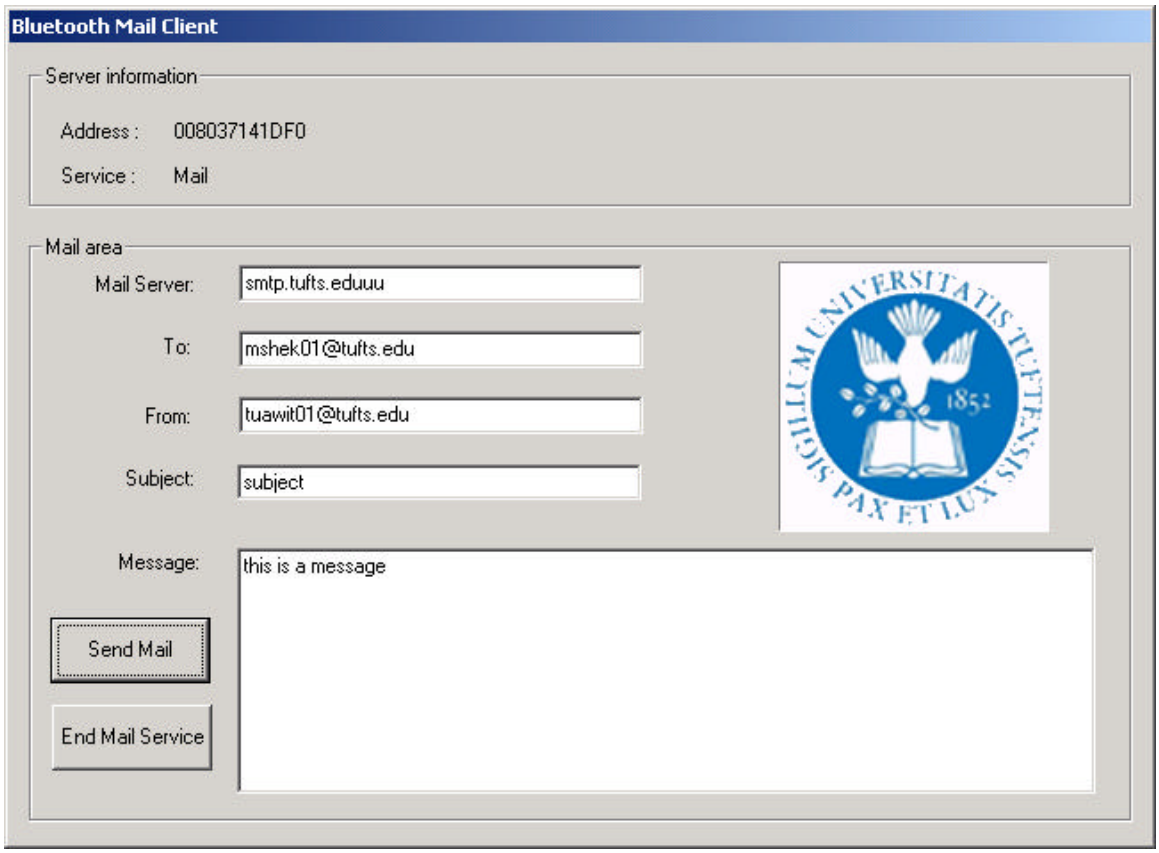

Figure 6.5 BTMail Client User Interface

7.Conclusion

7.1 Summary

This software project is a very good practice for both understanding the Bluetooth protocols and learning the windows programming. For the study of Bluetooth technology, the project demonstrates the idea of using the Bluetooth link as an Internet bridge. The final application works very well as expected. The maximum gross data transfer rate of the Bluetooth technology is 1Mbps and it's fast enough to transfer the plain text messages. The email message can be sent out to the server within one second and the server response time depends on the bandwidth of the Internet connection. The ideal utilization of Bluetooth technology is using it on mobile devices to replace the cable connection. Normally, a mobile device doesn't need a very high-speed data transfer rate because of the limitation of the memory size and the CPU power on the device. Also, the low power consumption is a very important strength of Bluetooth using on a mobile device because the battery life is a very critical issue on mobile devices. Another important feature of the Bluetooth is the device discovery protocol and the service discovery protocol. We used SDP in the application and it works perfectly in the devices detection and the services discovery utilization. The device and service at the server side usually can be detected by the remote client within 3 seconds. The Bluetooth Specification version 1.1 Volume 2 has defined many standard profiles as the common usage models. The standard profiles can make sure that Bluetooth devices can communicate with other different Bluetooth devices provided by different vendors. For example, the Bluetooth enabled headset should follow the headset profile defined in the specification, so that it can be used on any other Bluetooth enabled devices like cellular phones, CD walkmans or home stereos from other different manufacturers, as long as they are also follows the profile standard. In this BTMail project, we used a user defined profile for the mail service, and it can also be transplanted to other platforms as long as the implementation follows the same profile.

For the learning of win32 programming, this project provides a good opportunity to explore the Microsoft Visual C++ developing environment. The application is implemented by using MFC. Using MFC really simplified many development tasks because all the low level Win32 API were already encapsulated in it and it provides very convenient high level function for programmers. For example, the CSocket class is used in the SMTP connection procedures, and it really reduced the work to handle the Internet connection process. Another technology used in this software development is the COM server and client framework. Because the Ericsson Bluetooth PC Reference Stack that came within the toolkit is implemented as a COM server, In order to access the API provided by the Bluetooth Reference Stack, we need to know and understand how to deal with the COM server interface. This is a very useful technique to build software components for code reuse.

7.2 Problems and Future Improvement

Although the project itself was very successful and the final products actually met our expectation, there are still some issues need to be discussed. Most of the problems we encountered are because of the hardware availability. First, because we only have two Bluetooth modules available, we didn't have the chance to work on multiple links between Bluetooth modules. It's a very important feature of the Bluetooth technology that at most seven connections can be made at the same time on the same device. So in theory, the server application can be implemented as a multi-tasking program by using multi-thread programming technique. For example, the server can be connected by seven different clients at the same time and still can handle the email transmission.

Another Bluetooth feature we didn't implement on our project is the auto-detection. The service discovery protocol (SDP) is designed to be able to make a rapid ad hoc connection, so that the user doesn't need to initiate the connection manually. If we apply this feature into the BTMail application, the email can be written anytime and once the client detects any available mail service in its working range, it will make a Bluetooth connection and send out the email to the server automatically. An extra mobile computer is needed in order to test on the auto-detection features.

It will also be a good subject to examine the actual working range and the power consumption of the Bluetooth modules. For example, we can measure the power consumption rate within different working ranges, to see if the working distance is a major concern for the Bluetooth devices. Determining the maximum working range with a reasonable power consumption for a specific device is also very important when you want to develop any services on the device. Again, a mobile Bluetooth device and the power measurement tool are needed for doing this research.

This BTMail application is just a prototype, so we put the expandability in our consideration while designing and implementing. For the email service itself, there are still many enhanced features can be added to the application, like multiple recipients or file attachments. Also we can integrated the POP3 protocol into the application so that the email can be retrieved from the Internet to the mobile devices too.

The project is also an experiment of the concept of using Bluetooth link as an Internet bridge for the mobile devices. The usage model can apply to other high-level Internet protocols under the same concept. For example, the HTTP protocol support can be added to implement a wireless browser solution, the TELNET protocol service can be added for remote access, or the FTP protocol can be added for file sharing. These Bluetooth specific applications might not be very important for a PC or a laptop in the future because once the operating system has direct support of Bluetooth technology, the Bluetooth connections can work as the physical layer and data link layer for the PPP connections, then all the current Internet applications can run directly through the PPP over Bluetooth as they run under a dialup link. But for other non-PC mobile devices like PDAs, cellular phones, etc, this wireless Internet bridge will be very useful when the applications are transplanted to other Bluetooth enabled platforms.

List of Acronyms and Terms

ACK	acknowledge (for communication between devices).
API	Application Programming Interface
BT	Bluetooth.
COM	Component Object Model
DLL	Dynamic Link Library
FTP	File Transfer Protocol
GUI	Graphical User Interface
HCI	Host Control Interface
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
ISM	Industrial, Scientific, Medical.
LAN	Local Area Network
MFC	Microsoft Foundation Classes
ODBC	Open Database Connectivity
OLE	Object Linking and Embedding
RFC	Request for Comments
SDP	Service Discovery Protocol
SMTP	Simple Mail Transfer Protocol
TCP	Transport Control Protocol
USB	Universal Serial Bus
PPP	Point to Point Protocol

Acknowledgments

The prototype “Chat Application Suite” is provided by

Ericsson Bluetooth Application and Training Tool Kit

All the software modification and development is done by

Tufts EECS Bluetooth Application Development Team

Team Member:

Martin Shek

Tul Tul Uawithya

Zen-Jerr Hong

Project Advisor:

Dr. C. Hwa Chang

References

[1] Bluetooth SIG, Specification of Bluetooth System (Volume 1:Core), Feb 2001

[2] Bluetooth SIG, Specification of Bluetooth System (Volume 2:Profile), Feb 2001

[3] Ericsson, User Manual – Bluetooth PC Reference Stack, Jan 2001

[4] Jonathan B. Postel, RFC 821 SIMPLE MAIL TRANSFER PROTOCOL, Aug 1982

[5] David H. Crocker, RFC 822 STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES, Aug 1982

[6] Richard C. Leinecker, Visual C++ 6 Bible

[7] Chris H. Pappas, Visual C++ 6: The Complete Reference

[8] Jeff Prosise, Programming Windows with MFC second Edition

Appendix A: Technical summary of Bluetooth technology

Technical summary of Bluetooth technology

- Normal range 10m (0 dBm)
- Optional range 100m (+20 dBm)
- Normal transmitting power 0 dBm (1 mW)
- Optional transmitting power -30 to +20 dBm (100mW)
- Receiver sensitivity -70 dBm
- Frequency band 2.4 GHz
- Gross data rate 1 Mbit/s
- Max. data transfer 721+56kbit/3 voice channels
- Power consumption, hold/park ~50µA
- Power consumption, standby 300µA
- Power consumption, max. 30µA

Packet switching protocol based on a frequency hop scheme with 1600 hops/s.

Appendix B: Coding Standards

Coding Standard

The major coding standard in this project is to follow the original standard in the sample application. Although this is not a commercial software product, keeping a good coding standards is still a good practice to provide more readable codes for future references.

a. Typography

All functions or data members generated by Visual C++ class wizard or resource editor, like Windows system message handling functions or GUI event driven functions should follow the original Visual C++ naming convention. Example is OnButtonClick(), m_closebutton

Other manually added codes should follow the rules below:

- Methods -- Start with verb, first letter of each word uppercase. Example is SendMessageToClient ( )
- Data Members -- First letter uppercase and are nouns. Example is MailServer
- Temporary Variables -- All lowercase, keep to one word. Example is swap or temp.
- Constants -- All uppercase letters, spaces are underscores. Example is MAX_LINE.

b. Length of Functions

Functions should be kept to less than 50 lines, if this is unavoidable then the maximum is 110. Functions should not have to be created at line 110 in order to satisfy this requirement.

c. Whitespace

For every level of nesting, there should be one more tab than the parent level of nesting. The purpose is to allow quick determination of the level of nesting of the code, and to allow for easier tracing by different developers.

d. Comments

Comments should be placed at the top of files and before each function. At the top of a file, the following information should appear: Module Name, Developer Name, Versions, Dates, and Changes made to each version. Before each function, a brief description of the procedure should be explained in plain English, in order to allow any developer to quickly determine the purpose of the function.

Appendix C: Delivery Structure and Compiler Setting

a. Delivery Structure:

BTMail Server Application set:

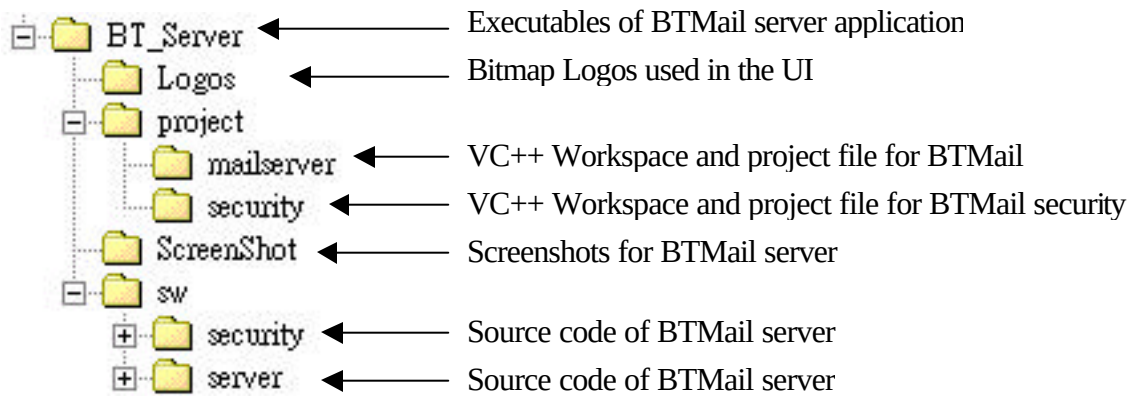

Figure C.1 The delivery file structure of the BTMail server application

BTMail Client Application set:

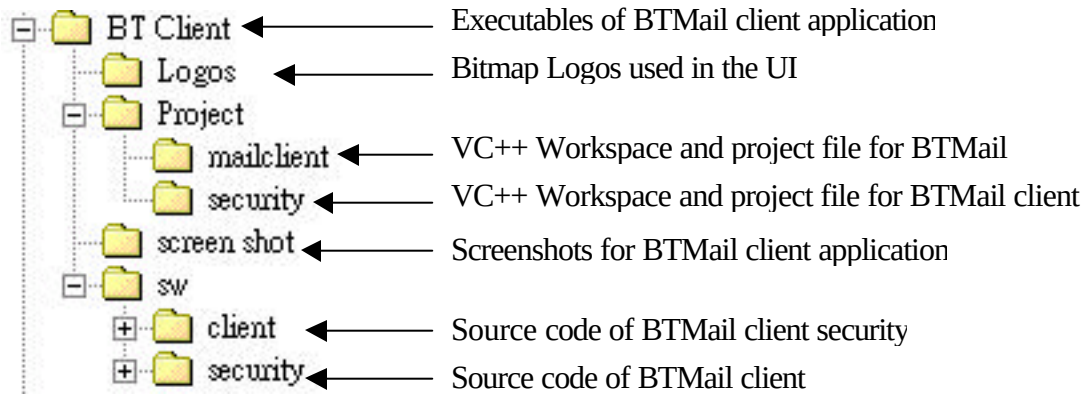

Figure C.2 The delivery file structure of the BTMail client application

b. Compiler Settings

The compiler setting is the same as in the sample chat application of the Ericsson Bluetooth Application and Training Tool Kit. Please see the User Manual of the Ericsson PC Reference Stack at Chapter 12-3 on page 264.