

BLUETOOTHTM 1.1

**Connect
Without
Cables**

SECOND EDITION

Bluetooth 1.1's ◀
dramatic improvements in
interoperability and reliability

Includes thoroughly revised ◀
coverage of Bluetooth security
and power conservation

New Bluetooth profiles— ◀
*including the long-awaited
Personal Area Networking profile!*

**Jennifer Bray and
Charles F Sturman**

Foreword by Tom Siep, General Manager of Bluetooth SIG, Inc.

Geotab Exhibit 1116
Geotab v. Omega
IPR2023-00504

Bluetooth™

Connect Without Cables

Second Edition

Jennifer Bray
Charles F. Sturman



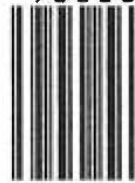
Prentice Hall PTR
Upper Saddle River, New Jersey 07458
www.phptr.com

ISBN 0-13-066106-6



9 790130 661066

9 0000



Library of Congress Cataloging-in-Publication Data

Bray, Jennifer.

Bluetooth 1.1: connect without cables/Jennifer Bray, Charles Sturman.—2nd ed.

p.cm.

Includes bibliographical references and index.

ISBN 0-13-066106-6

1. Bluetooth technology. 2. Telecommunication—Equipment and supplies. 3. Computer network protocols. I. Sturman, Charles F. II. Title.

TK5103.3 .B72 2002

004.6'2—dc21

2001054573

Publisher: Bernard Goodwin

Editorial Assistant: Michelle Vincenti

Marketing Manager: Dan DePasquale

International Product Manager: Mike Vaccaro

Manufacturing Buyer: Alexis Heydt-Long

Cover Design: Talar Agasyan-Boorju

Cover Design Director: Jerry Votta

Cover Photograph: Charles F. Sturman

Compositor/Production Services: Pine Tree Composition, Inc.



© 2002 by Prentice Hall PTR

Prentice-Hall, Inc.

Upper Saddle River, NJ 07458

BLUETOOTH is a trademark owned by Bluetooth SIG, Inc., USA.

All products or services mentioned in this book are the trademarks or service marks of their respective companies or organizations.

Some material in Chapter 10 © ETSI 1999, TS 101 369 v3 is the property of ETSI. Further use, modification, redistribution is strictly prohibited and must be the subject of another Copyright Authorisation.

The above-mentioned standard may be obtained from the ETSI Publication Office, publications@etsi.fr,

Tel: +33 (0)4 92 94 42 41 or downloaded from the website at <http://www.etsi.org/eds/eds.htm>

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed in the United States of America

ISBN 0-13-066106-6

Pearson Education Ltd., *London*

Pearson Education Australia Pty, Limited, *Sydney*

Pearson Education Singapore, Pte. Ltd,

Pearson Education North Asia Ltd, *Hong Kong*

Pearson Education Canada, Ltd., *Toronto*

Pearson Educación de México, S.A. de C.V.

Pearson Education—Japan, *Tokyo*

Pearson Education Malaysia, Pte. Ltd.

Contents

Foreward to the Second Edition	xvii
Foreward to the First Edition	xix
Preface to the Second Edition	xxi
Preface to the First Edition	xxiii
Acknowledgments	xxv
Introduction	xxvii
1 Overview	1
1.1 Bluetooth's Origins	2
1.2 The Bluetooth SIG	2
1.3 Aims	4
1.4 The Protocol Stack	5
1.5 Security	12

1.6	Applications and Profiles	12
1.7	Using Bluetooth	13
1.8	Management	17
1.9	Test and Qualification	18
1.10	Bluetooth in Context	19
1.11	Summary	22

Protocol Stack Part 1—The Bluetooth Module

2	Antennas	25
2.1	Radiation Pattern	25
2.2	Gains and Losses	27
2.3	Types of Antennas	28
2.4	Ceramic Antennas	30
2.5	On-chip Antennas	30
2.6	Antenna Placement	31
2.7	Summary	31
3	Radio	32
3.1	Introduction	32
3.2	Frequency Hopping	33
3.3	Modulation	33
3.4	Symbol Timing	34
3.5	Power Emission and Control	34
3.6	Radio Performance Parameters	35
3.7	Simple RF Architecture	36
3.8	RF System Timing	39
3.9	Blue RF	39
3.10	Summary	40
4	Baseband	41
4.1	Introduction	41
4.2	Bluetooth Device Address	42

4.3	Masters, Slaves, and Piconets	42
4.4	System Timing	45
4.5	Physical Links: SCO and ACL	48
4.6	Bluetooth Packet Structure	49
4.7	Packet Types and Packet Construction	57
4.8	Logical Channels	58
4.9	Channel Coding and Bitstream Processing	59
4.10	Timebase Synchronisation and Receive Correlation	62
4.11	Frequency Hopping	63
4.12	Summary	64
5	The Link Controller	65
5.1	Introduction	65
5.2	Link Control Protocol	65
5.3	Link Controller States	67
5.4	Link Controller Operation	70
5.5	Piconet Operation	79
5.6	Scatternet Operation	82
5.7	Master / Slave Role Switching	84
5.8	Low-Power Operation	88
5.9	Baseband / Link Controller Architectural Overview	88
5.10	Summary	90
6	Audio	93
6.1	Introduction	93
6.2	Audio Transports in the Protocol Stack	94
6.3	Quality and Bandwidth	94
6.4	SCO Links	95
6.5	Audio CODECs	96
6.6	Audio Subsystem	99
6.7	Audio Data Formats and HCI	101
6.8	Implementation	102
6.9	Summary	102

7	The Link Manager	103
7.1	LMP Protocol Data Units (PDUs)	104
7.2	The Link Management Channel	104
7.3	Link Setup	105
7.4	LMP Link Shutdown	108
7.5	Role Change	109
7.6	Control of Multi-Slot Packets	111
7.7	Security	112
7.8	Low-Power Modes	112
7.9	Power Control	112
7.10	Quality of Service	114
7.11	Information Messages	114
7.12	Supported Features	114
7.13	LMP Version	115
7.14	Name Request	116
7.15	Test Mode	117
7.16	Summary	117
8	The Host Controller Interface	119
8.1	HCI Packet Types	121
8.2	The HCI Transport Layer	125
8.3	Flow Control	128
8.4	Configuring Modules	130
8.5	Inquiring: Discovering Other Bluetooth Devices	134
8.6	Inquiry Scan: Becoming Discoverable	137
8.7	Paging: Initiating Connections	140
8.8	Page Scan: Receiving Connections	141
8.9	Sending and Receiving Data	145
8.10	Switching Roles	147
8.11	Power Control	147
8.12	Summary	148

Protocol Stack Part 2—The Bluetooth Host

9	Logical Link Control and Adaptation Protocol	151
9.1	Multiplexing Using Channels	152
9.2	L2CAP Signalling	153
9.3	Establishing a Connection	156
9.4	Configuring a Connection	160
9.5	Transferring Data	162
9.6	Disconnecting and Timeouts	164
9.7	Connectionless Data Channels	166
9.8	Enabling and Disabling Incoming Connectionless Traffic	167
9.9	Handling Groups	168
9.10	Echo and Ping	170
9.11	Get Information	171
9.12	L2CAP State Machine	172
9.13	Implementation-Dependent Issues	173
9.14	Summary	175
10	RFCOMM	176
10.1	Serial Ports and UARTs	177
10.2	Types of RFCOMM Devices	177
10.3	RFCOMM Frame Types	178
10.4	Connecting and Disconnecting	179
10.5	Structure of RFCOMM Frames	180
10.6	Multiplexer Frames	186
10.7	Service Records	194
10.8	Summary	195
11	The Service Discovery Protocol	196
11.1	SDP Client/Server Model	197
11.2	The SDP Database	198
11.3	Browsing SDP Records	204

11.4	Universally Unique Identifiers (UUIDs)	205
11.5	SDP Messages	208
11.6	Service Discovery Profile	221
11.7	Summary	223
12	The Wireless Application Protocol	224
12.1	The WAP Forum	225
12.2	The WAP Stack	226
12.3	PPP Links	227
12.4	WAP Clients and Servers	231
12.5	Suspend and Resume	233
12.6	Service Discovery	234
12.7	WAP Interoperability	234
12.8	Using WAP	234
12.9	Summary	236
13	OBEX and IrDA	237
13.1	OBEX in the Bluetooth Stack	238
13.2	Object Model	240
13.3	Session Protocol	242
13.4	Summary	249
14	Telephony Control Protocol	251
14.1	TCS Signalling	253
14.2	Call Establishment Signalling	256
14.3	Call Clearing Signalling	262
14.4	DTMF Signalling	264
14.5	Wireless User Group (WUG) Signalling	265
14.6	Connectionless Signalling	269
14.7	TCS Call States	270
14.8	Summary	271

Protocol Stack Part 3—Cross Layer Functions

15	Encryption and Security	275
15.1	Key Generation and the Encryption Engine	277
15.2	Secret Keys and PINs	282
15.3	Pairing and Bonding	283
15.4	Starting Encryption	292
15.5	Security Modes	294
15.6	Security Architecture	295
15.7	Summary	298
16	Low-Power Operation	300
16.1	Controlling Low-Power Modes	301
16.2	Hold Mode	301
16.3	Sniff Mode	304
16.4	Park Mode	308
16.5	Low-Power Oscillator	318
16.6	Summary	318
17	Quality of Service	320
17.1	Requesting QOS	323
17.2	QOS Violations	328
17.3	Flushing and Delays	329
17.4	Link Supervision	331
17.5	Broadcast Channel Reliability	332
17.6	Data Rates and Packet Types	333
17.7	Summary	338
18	Managing Bluetooth Devices	339
18.1	Link Configuration and Management	340
18.2	Device Manager Architecture	351

18.3	Security Management	355
18.4	Integrating Applications	356
18.5	Accounting Management	357
18.6	Capacity	358
18.7	User Interface Design	358
18.8	Summary	360

Applications—The Bluetooth Profiles

19 Foundation Profiles

19.1	Structure of Profiles	365
19.2	The Generic Access Profile	365
19.3	The Serial Port Profile	367
19.4	Dial Up Networking	369
19.5	FAX Profile	372
19.6	Headset Profile	373
19.7	LAN Access Point Profile	375
19.8	Generic Object Exchange Profile	378
19.9	Object Push Profile	379
19.10	File Transfer Profile	380
19.11	Synchronisation Profile	381
19.12	Intercom Profile	382
19.13	The Cordless Telephony Profile	383
19.14	Benefits of Profiles	385
19.15	Summary	386

20 Draft Post – Foundation Profiles 388

20.1	The Human Interface Device Profile	389
20.2	The Hands-Free Profile	397
20.3	The Basic Imaging Profile	405
20.4	The Basic Printing Profile	415
20.5	The Hard Copy Cable Replacement Profile	423
20.6	Summary	427

21	Personal Area Networking	428
21.1	The PAN Profile	428
21.2	Bluetooth Network Encapsulation Profile	435
21.3	Summary	443
22	ESDP for UPnP	444
22.1	Universal Plug and Play Device Architecture	444
22.2	L2CAP Based Solutions	447
22.3	IP Based Solutions	449
22.4	Summary	451

Test and Qualification

23	Test Mode	455
23.1	Activating Test Mode	455
23.2	Controlling Test Mode	458
23.3	Radio Transmitter Test	460
23.4	Loopback Test	463
23.5	Summary	464
24	Qualification and Type Approval	465
24.1	Bluetooth Qualification	466
24.2	Bluetooth Interoperability Testing	474
24.3	Regulatory Type Approval	475
24.4	Summary	476

Bluetooth in Context

25	Implementation	479
25.1	Introduction	479
25.2	System Partitioning	480
25.3	Hardware Integration Options	488
25.4	Bluetooth as an IP Core	492

25.5	ASIC Prototyping and FPGAs	497
25.6	Making the Right Design Choices	497
25.7	Radio Implementation	499
25.8	Summary	501
26	Related Standards and Technologies	502
26.1	Introduction	502
26.2	What Are the Requirements?	502
26.3	Infrared Data Association (IrDA)	503
26.4	Digital Enhanced Cordless Telecommunications (DECT)	504
26.5	IEEE 802.11	506
26.6	The HomeRF™ Working Group (HRFWG)	509
26.7	IEEE 802.15 and the Wireless Personal Area Network (WPAN)	511
26.8	HIPERLAN	512
26.9	MMAC	514
26.10	The Future	514
26.11	Summary	516
26.12	Useful Web Addresses	517
27	The Bluetooth Market	518
27.1	Introduction	518
27.2	Market Pull and Technology Push	518
27.3	Market Segments	521
27.4	Success in the Marketplace	523
27.5	Enabling Technologies and Components	525
27.6	Consumer Products	526
27.7	The Bluetooth Brand	527
27.8	Summary	529
28	Future Developments	530
28.1	Working Groups and New Bluetooth Profiles	531
28.2	Profile Working Groups	532
28.3	Future Bluetooth Core Specifications	535
28.4	Summary	537

Contents

Appendix—Bluetooth 1.1 Updates	538
Glossary	550
References	577
Index	581

Overview

Bluetooth has been the subject of much hype and media attention over the last couple of years. As various manufacturers prepare to launch products using Bluetooth technology, an unsuspecting public is about to be catapulted into the next stage of the information technology revolution.

Bluetooth is a low cost, low power, short range radio technology, originally developed as a cable replacement to connect devices such as mobile phone handsets, headsets, and portable computers. This in itself sounds relatively innocuous; however, by enabling standardised wireless communications between any electrical devices, Bluetooth has created the notion of a Personal Area Network (PAN), a kind of close range wireless network that looks set to revolutionise the way people interact with the information technology landscape around them.

No longer do people need to connect, plug into, install, enable, or configure anything to anything else. Through a ubiquitous standardised communications subsystem, de-

vices will communicate seamlessly. One does not need to know where one's cellular phone is, or even if it is switched on. As soon as the Web browser appears on the mobile computer screen, a link is established with the phone the Internet Service Provider is connected to, and the user is surfing the Web.

The Bluetooth specification is an open, global specification defining the complete system from the radio right up to the application level. The protocol stack is usually implemented partly in hardware and partly as software running on a microprocessor, with different implementations partitioning the functionality between hardware and software in different ways.

1.1 BLUETOOTH'S ORIGINS

Version 1.0 of the Bluetooth specification came out in 1999, but Bluetooth started five years earlier, in 1994, when Ericsson Mobile Communications began a study to examine alternatives to the cables that linked its mobile phones with accessories. The study looked at using radio links. Radio isn't directional, and it doesn't need line of sight, so it has obvious advantages over the infra-red links previously used between handsets and devices. There were many requirements for the study, including handling both speech and data, so that it could connect phones to both headsets and computing devices.

Out of this study was born the specification for Bluetooth wireless technology. The specification is named after Harald Blatand (Blatand is Danish for Bluetooth), Harald was a tenth-century Danish Viking king who united and controlled Denmark and Norway. The name was adopted because Bluetooth wireless technology is expected to unify the telecommunications and computing industries.

1.2 THE BLUETOOTH SIG

The Bluetooth Special Interest Group (SIG) is a group of companies working together to promote and define the Bluetooth specification. The Bluetooth SIG was founded in February 1998 by the following group of core promoters:

- Ericsson Mobile Communications AB.
- Intel Corp.
- IBM Corp.
- Toshiba Corp.
- Nokia Mobile Phones.

In May 1998, the core promoters publicly announced the global SIG and invited other companies to join the SIG as Bluetooth adopters in return for a commitment to support the Bluetooth specification. The core promoters published version 1.0 of the Bluetooth speci-

fication in July 1999, on the Bluetooth Web site, <http://www.bluetooth.com>. In December 1999, the Bluetooth core promoters group enlarged with the addition of four more major companies:

- Microsoft.
- Lucent.
- 3COM.
- Motorola.

1.2.1 Joining the Bluetooth SIG

Any incorporated company willing to sign the Bluetooth SIG membership agreement can join the SIG as a Bluetooth adopter company. To join the SIG, companies simply fill in a form on the Bluetooth Web site, www.bluetooth.com. This form commits SIG members to contributing any key technologies which are needed to implement Bluetooth.

This commitment to share technology means that Bluetooth SIG member companies who put their products through Bluetooth qualification are granted a free license to build products using the Bluetooth wireless technology. The license is important because there are patents required to implement Bluetooth; companies that do not sign the Bluetooth adopter's agreement will not be entitled to use the technology. This offer proved so attractive that by April 2000, the SIG membership had grown to 1,790 members.

In addition to getting a free license to patents needed to implement Bluetooth wireless technology, Bluetooth SIG members also have permission to use the Bluetooth brand. There are restrictions on the use of the brand, and these are set out in the Bluetooth brand book. The trademark may only be used on products which prove they are correctly following the Bluetooth specification by completing the Bluetooth qualification program (a testing process).

To get the Bluetooth figure mark and instructions on how to use it, companies sign the Bluetooth trademark agreement, also available on www.bluetooth.com. Questions on the Bluetooth trademark can be emailed to brand.manager@Bluetooth.com.

1.2.2 Bluetooth SIG Organisation

At the head of the Bluetooth SIG is the program management board. This board oversees the operations of a number of other groups as shown in Figure 1-1.

The main work of defining the specification is done by the technical working groups. Adopter companies can apply to become associate members of the SIG; they may then apply to join working groups and hence contribute directly to the forming of Bluetooth specifications.

Sitting on the technical working groups is quite time-consuming, and so many companies with valid comments on the specification do not have the resources to sit on the working groups. These companies can pass comments via email to the writers of the standard and can also participate in an online discussion forum on the Bluetooth Web site.

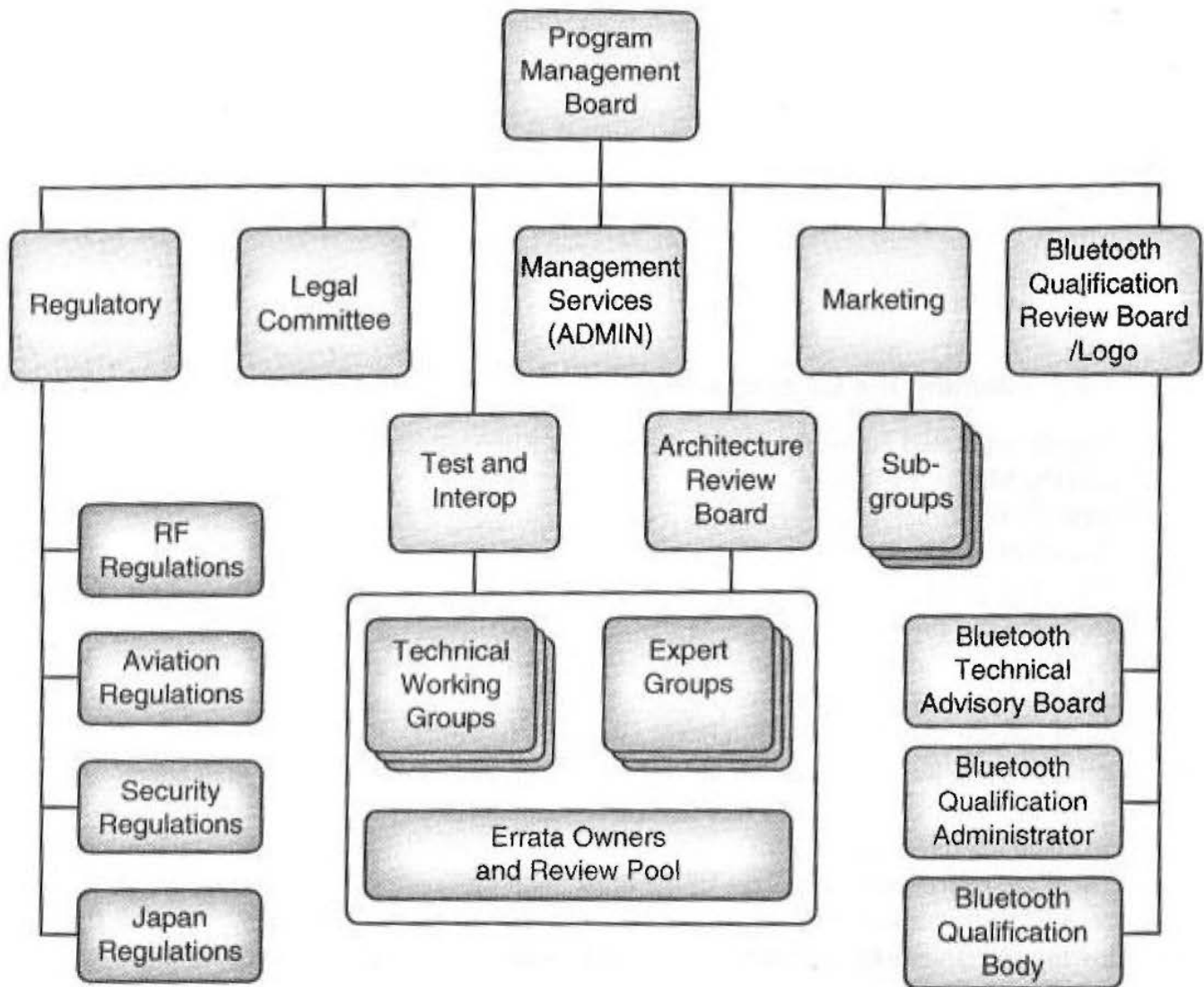


Figure 1-1 Organisation of the Bluetooth SIG.

1.3 AIMS

Why should a group with such diverse interests as the Bluetooth promoters cooperate? Basically, because it's good for their businesses. The members of the Bluetooth promoters group all stand to gain something from mobile devices communicating better, whether by selling devices that have enhanced functionality or by selling the extra software that people will need once they can more easily access information on the move.

The reasons for making the Bluetooth specification freely available to anyone who cares to sign an adopter's agreement are basically the same. The Bluetooth promoters group has made Bluetooth an open specification, rather than keeping it restricted and proprietary, because consumers are more likely to adopt a technology which can be bought from many manufacturers than one which is just limited to a select few. Wide acceptance

among consumers is likely to lead to a larger overall market for Bluetooth devices. So the promoters will gain from more companies becoming involved in the Bluetooth SIG.

The aim of the Bluetooth specification is basically to sell more of the core promoters' products. This will happen because Bluetooth will make their products more useful by improving communications between them. Before the advent of Bluetooth, telecommunications and computing devices were usually connected by cables, which were easily broken or lost. Cables are also awkward to carry around. The Bluetooth specification aimed to ease communication between mobile devices by providing a cable replacement.

Being a cable replacement technology imposes several requirements. If Bluetooth technology is to replace cables, it can't be much more expensive than a cable or nobody will buy it. At the time of writing, a data cable for a cellular mobile phone was about \$10. Allocate half the cost of the cable to each end of the link and it's obvious that for a cable replacement technology to be attractive on purely financial grounds, each unit should cost no more than \$5. So, the two ends of the link should cost the same as the cable they replace.

Because Bluetooth technology is designed for mobile devices, it must be able to run on batteries. So, it must be very low power, and should run on low voltages. It must also be lightweight and small enough not to intrude on the design of compact mobile devices such as cellular phones, headsets, and PDAs.

It must be as easy and convenient to use as plugging in a cable, and it must be as reliable as the cable it replaces. Because it is a wireless technology, to be reliable, Bluetooth must also be resilient. Reliability means it works overall; resilience means that it can cope with errors.

So, Bluetooth aims to be widely available, inexpensive, convenient, easy to use, reliable, small, and low power. If Bluetooth achieves all these goals, it will be incredibly good for the businesses involved with it.

1.4 THE PROTOCOL STACK

A key feature of the Bluetooth specification is that it aims to allow devices from lots of different manufacturers to work with one another. To this end, Bluetooth does not just define a radio system, it also defines a software stack to enable applications to find other Bluetooth devices in the area, discover what services they can offer, and use those services.

The Bluetooth stack is defined as a series of layers, though there are some features which cross several layers.

Every block in Figure 1-2 corresponds to a chapter in the core Bluetooth specification. The core specification also has three chapters on test and qualification:

- Bluetooth Test Mode.
- Bluetooth Compliance Requirements.
- Test Control Interface.

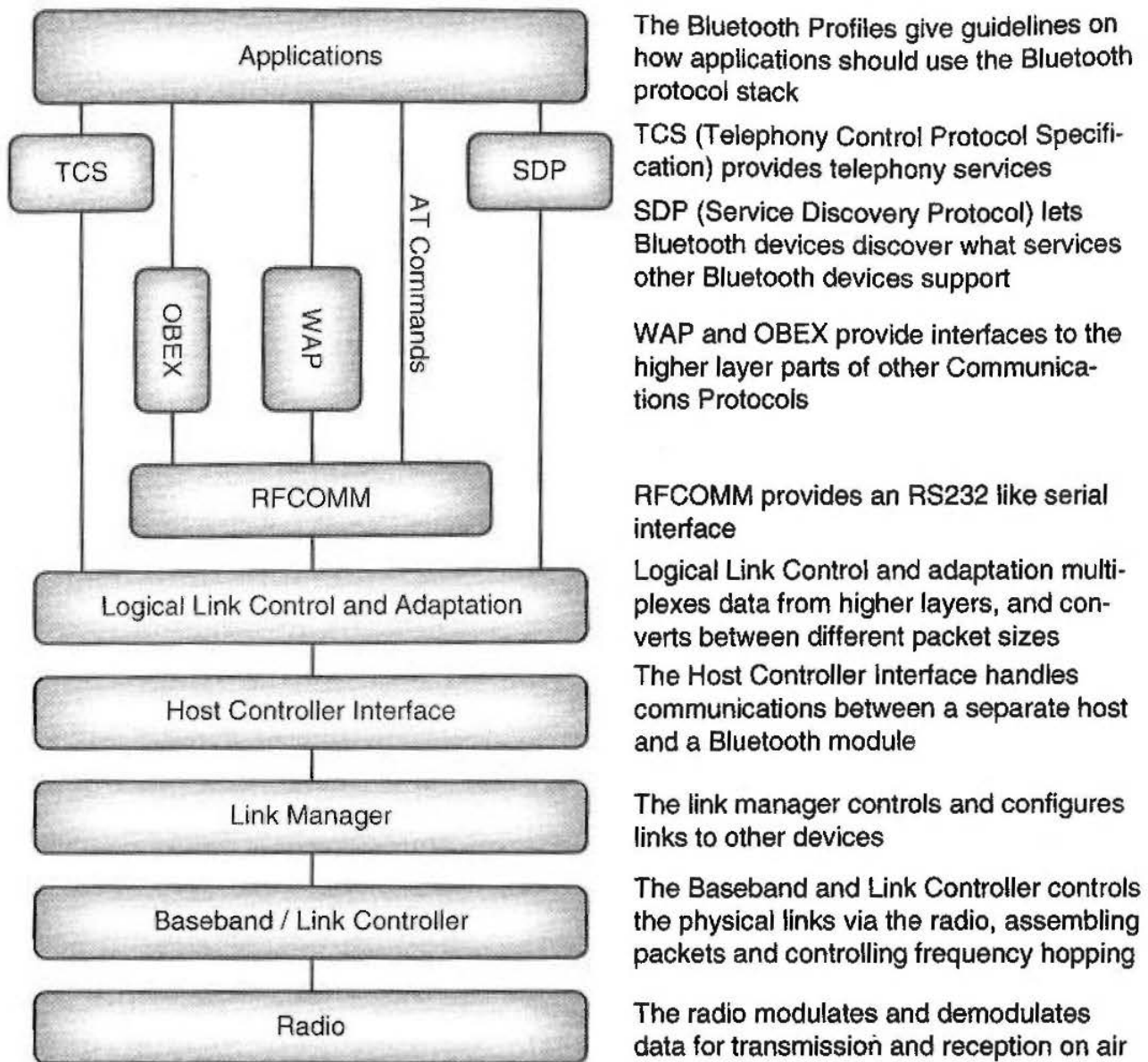


Figure 1-2 The Bluetooth protocol stack.

The Bluetooth specification encompasses more than just the core specification. There are also profiles which give details of how applications should use the Bluetooth protocol stack, and a brand book which explains how the Bluetooth brand should be used.

1.4.1 The OSI Reference Model

Figure 1-3 shows the familiar Open Systems Interconnect (OSI) standard reference model for communications protocol stacks. Although Bluetooth does not exactly match the model, it is a useful exercise to relate the different parts of the Bluetooth stack to the vari-

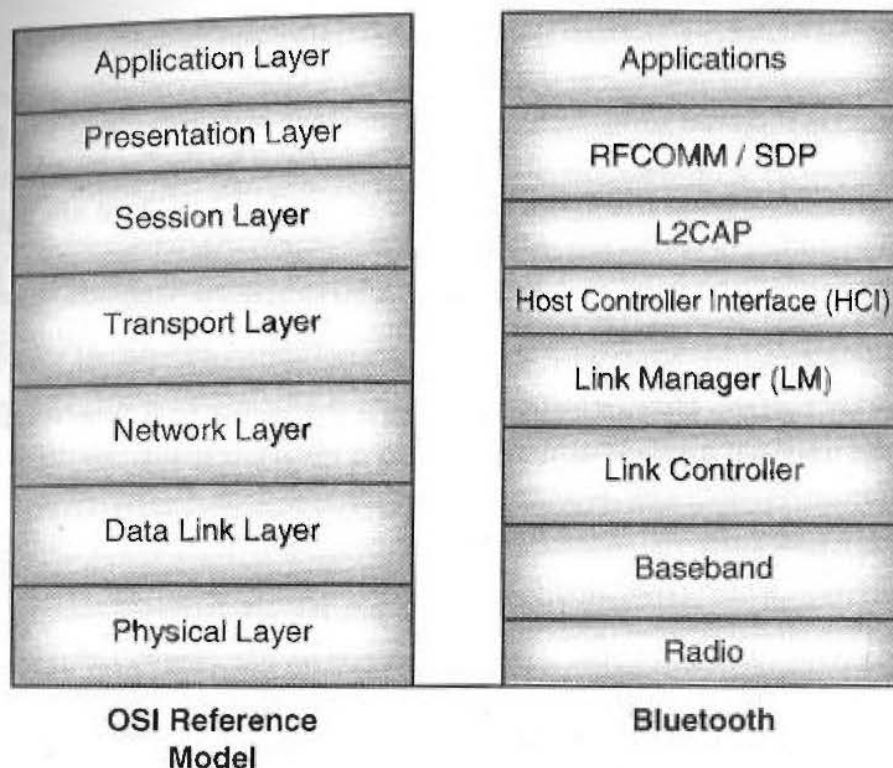


Figure 1-3 OSI reference model and Bluetooth.

ous parts of the model. Since the reference model is an ideal, well-partitioned stack, the comparison serves to highlight the division of responsibility in the Bluetooth stack.

The Physical Layer is responsible for the electrical interface to the communications media, including modulation and channel coding. It thus covers the radio and part of the baseband.

The Data Link Layer is responsible for transmission, framing, and error control over a particular link, and as such, overlaps the link controller task and the control end of the baseband, including error checking and correction.

From now on, it gets a little less clear. The Network Layer is responsible for data transfer across the network, independent of the media and specific topology of the network. This encompasses the higher end of the link controller, setting up and maintaining multiple links, and also covering most of the Link Manager (LM) task. The Transport Layer is responsible for the reliability and multiplexing of data transfer across the network to the level provided by the application, and thus overlaps at the high end of the LM and covers the Host Controller Interface (HCI), which provides the actual data transport mechanisms.

The Session Layer provides the management and data flow control services, which are covered by L2CAP and the lower ends of RFCOMM/SDP. The Presentation Layer provides a common representation for Application Layer data by adding service structure to the units of data, which is the main task of RFCOMM/SDP. Finally, the Application Layer is responsible for managing communications between host applications.

1.4.2 The Physical Layer

Bluetooth devices operate at 2.4 GHz in the globally available, licence-free ISM band. This band is reserved for general use by Industrial, Scientific, and Medical (ISM) applications, which obey a basic set of power and spectral emission and interference specifications. This means that Bluetooth has to be very robust, as there are a great many existing users and polluters of this shared spectrum.

The operating band is divided into 1 MHz-spaced channels, each signalling data at 1 Megasymbol per second so as to obtain the maximum available channel bandwidth. With the chosen modulation scheme of GFSK (Gaussian Frequency Shift Keying), this equates to 1 Mb/s. Using GFSK, a binary 1 gives rise to a positive frequency deviation from the nominal carrier frequency, while a binary 0 gives rise to a negative frequency deviation.

After each packet, both devices retune their radio to a different frequency, effectively hopping from radio channel to radio channel (FHSS—frequency hopping spread spectrum). In this way, Bluetooth devices use the whole of the available ISM band and if a transmission is compromised by interference on one channel, the retransmission will always be on a different (hopefully clear) channel. Each Bluetooth time slot lasts 625 microseconds, and, generally, devices hop once per packet, which will be every slot, every 3 slots, or every 5 slots.

Designed for low-powered portable applications, the radio power must be minimised. Three different power classes are defined, which provide operation ranges of approximately 10 m, 20 m, and 100 m; the lowest power gives up to 10 m range, the highest up to 100 m.

1.4.3 Masters, Slaves, Slots, and Frequency Hopping

If devices are to hop to new frequencies after each packet, they must all agree on the sequence of frequencies they will use. Bluetooth devices can operate in two modes: as a Master or as a Slave. It is the Master that sets the frequency hopping sequence. Slaves synchronise to the Master in time and frequency by following the Master's hopping sequence.

Every Bluetooth device has a unique Bluetooth device address and a Bluetooth clock. The baseband part of the Bluetooth specification describes an algorithm which can calculate a frequency hop sequence from a Bluetooth device address and a Bluetooth clock. When Slaves connect to a Master, they are told the Bluetooth device address and clock of the Master. They then use this to calculate the frequency hop sequence. Because all Slaves use the Master's clock and address, all are synchronised to the Master's frequency hop sequence.

In addition to controlling the frequency hop sequence, the Master controls when devices are allowed to transmit. The Master allows Slaves to transmit by allocating slots for voice traffic or data traffic. In data traffic slots, the Slaves are only allowed to transmit when replying to a transmission to them by the Master. In voice traffic slots, Slaves are required to transmit regularly in reserved slots whether or not they are replying to the Master.



Figure 1-4 Point to point and point to multipoint piconets.

The Master controls how the total available bandwidth is divided among the Slaves by deciding when and how often to communicate with each Slave. The number of time slots each device gets depends on its data transfer requirements. The system of dividing time slots among multiple devices is called Time Division Multiplexing (TDM).

1.4.4 Piconets and Scatternets

A collection of Slave devices operating together with one common Master is referred to as a piconet (see Figure 1-4). All devices on a piconet follow the frequency hopping sequence and timing of the Master.

In Figure 1-4, the piconet on the left with only one Slave illustrates a point to point connection. The piconet on the right with three Slaves talking to the Master illustrates a point to multipoint connection. The Slaves in a piconet only have links to the Master; there are no direct links between Slaves in a piconet.

The specification limits the number of Slaves in a piconet to seven, with each Slave only communicating with the shared Master. However, a larger coverage area or a greater number of network members may be realized by linking piconets into a scatternet, where some devices are members of more than one piconet (see Figure 1-5).

When a device is present in more than one piconet, it must time-share, spending a few slots on one piconet and a few slots on the other. On the left in Figure 1-5 is a

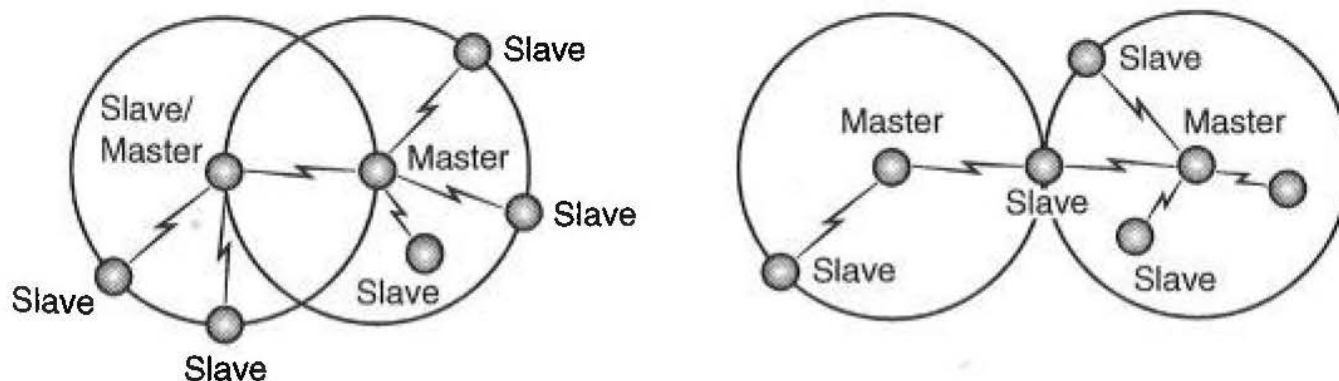


Figure 1-5 Scatternets.

scatternet where one device is a Slave in one piconet and a Master in another. On the right is a scatternet where one device is a Slave in two piconets. It is not possible to have a device which is a Master of two different piconets, since all Slaves in a piconet are synchronised to the Master's hop sequence. By definition, all devices with the same Master must be on the same piconet.

In addition to the various sources of interference mentioned already, a major source of interference for Bluetooth devices will clearly be other Bluetooth devices. Although devices sharing a piconet will be synchronised to avoid each other, other unsynchronised piconets in the area will randomly collide on the same frequency. If there is a collision on a particular channel, those packets will be lost and subsequently retransmitted, or if voice, ignored. So, the more piconets in an area, the more retransmissions will be needed, causing data rates to fall. This is like having a conversation in a noisy room: the more people talking, the noisier it gets, and you have to start repeating yourself to get the point across.

This effect will happen if there are many independent piconets in one area, and it will also happen to scatternets, since the piconets making up the scatternet do not coordinate their frequency hopping.

1.4.5 Radio Power Classes

The Bluetooth specification allows for three different types of radio powers:

- Class 1 = 100mW (20 dBm).
- Class 2 = 2.5mW (4 dBm).
- Class 3 = 1mW (0 dBm).

These power classes allow Bluetooth devices to connect at different ranges. At the time of writing, most manufacturers are producing Class 3, low power, 1 mW radios. These can communicate for a maximum of around 30 feet (10 m). However, because things like bodies and furniture absorb microwaves, reception may not be reliable at the limit of this range. So, when using 1 mW radios, a more realistic figure for reliable operation in a normal room will probably be 5 m. This provides a low cost, low power communications solution which has plenty of range for a cable replacement technology.

Obviously, higher power radios have longer ranges. The maximum range for a Class 1, 100 mW radio is about 100 metres. There is also a minimum range for a Bluetooth connection. If radios are put too close together, some receivers may saturate, so a few Bluetooth radios may be unreliable on short link lengths (under 10 cm).

A 100 m link needs a high power Class 1 device at both ends, but it is possible to create piconets with a mixture of high and low power devices at different ranges. Figure 1-6 shows a mixture of high and low power devices in different piconets occupying an area.

This figure shows piconets which overlap each other. This is possible because each Master has its own frequency hopping sequence, so two piconets are unlikely to be on the same frequency at the same time. If they do meet on the same frequency, after the next

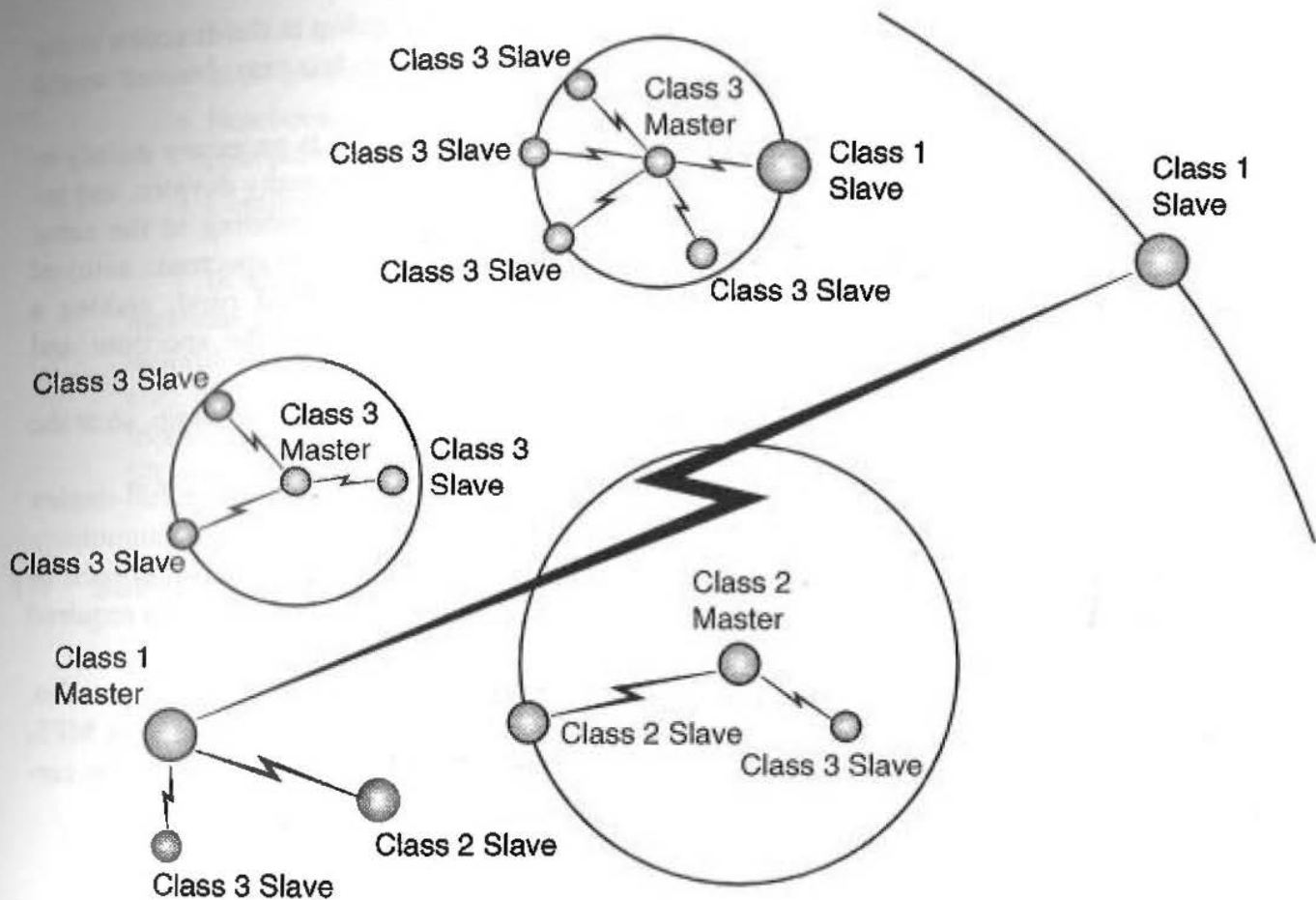


Figure 1-6 Piconets made up of different power class devices.

frequency hop they will not still be on the same frequency, so the data which may have been lost when the two piconets were on the same frequency can be resent.

1.4.6 Voice and Data Links

Bluetooth allows both time critical data communication such as that required for voice or audio, as well as high speed, time insensitive packet data communication. To carry such data, two different types of links are defined between any two devices. These are SCO (Synchronous Connection Oriented) links for voice communication and ACL (Asynchronous Connectionless) links for data communication.

ACL data packets are constructed from a 72-bit access code, a 54-bit packet header and a 16-bit CRC code, in addition to the payload data. There are a variety of packet types allowing different amounts of data to be sent. The packet which carries the largest data payload is a DH5 packet, which stretches over five slots. A DH5 packet can carry 339 bytes, or 2,712 bits of data. So, 2,858 bits are sent on air for 2,712 bits of information.

A DH5 packet uses up five slots, and the minimum length reply is one slot. Thus, the maximum baseband data rate in one direction is 723.2 kb/s. In this case, with 5-slot packets sent in one direction, the 1-slot packets sent in the other direction will only carry

57.6 kb/s, so this would be an asymmetric link with more data going in the direction using 5-slot packets. If 5-slot packets were sent in both directions, the data rate obtained would be 433.9 kb/s, quite a reduction from the 1 Mb/s data rate on air.

This overhead in both data encoding and frequency hopping is necessary mainly to provide a robust link since the ISM band is a shared resource with many devices, and indeed other communications standards and even noise sources, cohabiting in the same spectrum. In addition, to further reduce the interference problem in the spectrum, national radio regulations limit the power emission per unit time in the ISM band, making a frequency hopping scheme necessary to spread transmissions over the spectrum and over time.

The higher layers of the protocol stack also use up some of the bandwidth, so at the application level, the maximum data rate could be around 650 kb/s.

The SCO links work at 64 kb/s, and it is possible to have up to three full-duplex voice links at once or to mix voice and data. These voice channels give audio communication of a quality one would expect from a modern mobile cellular phone system such as GSM. As such, SCO links are not really suitable for delivering audio of a quality required for music listening.

One alternative to support music delivery is to use an ACL channel to carry audio. Raw CD-quality audio requires 1411.2 kb/s, but with suitable compression, such as MP3, which reduces this bit rate to around 128 kb/s, near CD-quality audio could easily be carried providing the time criticality of the audio was maintained.

1.5 SECURITY

The high speed, pseudo-random frequency hopping algorithm employed in Bluetooth makes it very difficult to listen in on a Bluetooth connection. In fact, the U.S. military considers a communications link using frequency hopping over 79 channels to be secure in itself.

For link encryption and authentication, Bluetooth uses a strong contemporary cipher algorithm available in the public domain called SAFER+, which generates 128-bit cipher keys from a 128-bit plain text input.

1.6 APPLICATIONS AND PROFILES

Looking back to Section 1.3, the Bluetooth specification aimed to produce convenient, reliable, resilient, cost effective, low power, short range voice and data communications. It has achieved all these, but what sort of applications does this enable?

At its most basic, Bluetooth wireless technology replaces a cable and untethers devices. This makes it suitable for short range connections between a variety of mobile devices such as:

- Mobile cellular phone to Public Switched Telephone Network (PSTN) through an access point.
- Mobile cellular phone to a notebook PC.

- Mobile cellular phone to a headset.
- LAN access points for laptops or palmtops.
- Notebook, palmtop, or other Internet access device to the Internet via a PSTN access point or access module.
- Communication between laptops and palmtops.

In addition to the core specification, which defines the Bluetooth wireless communications protocol, the Bluetooth specification includes a profiles document. Each profile describes how a particular application can be implemented, including which parts of the core Bluetooth protocol should be used to support the profile. Version 1.0 of the Bluetooth specification provides profiles for all of the connections listed above. The Bluetooth SIG continues to define profiles to support further applications.

1.7 USING BLUETOOTH

Bluetooth is unlike any wired network, as there is no need to physically attach a cable to the devices you are communicating with; indeed, you may not know exactly what devices you are talking to and what their capabilities are. To cope with this, Bluetooth provides inquiry and paging mechanisms and a Service Discovery Protocol (SDP).

This section examines how these mechanisms are used to allow Bluetooth devices to link up and use one another's services.

1.7.1 Discovering Bluetooth Devices

Imagine two Bluetooth enabled devices, say, a cell phone and a laptop computer. The cell phone is capable of acting as a modem using the dial up networking profile, and it periodically scans to see if anyone wants to use it.

The user of the laptop opens up an application that needs a Bluetooth dial up networking connection. To use this application, the laptop knows it needs to establish a Bluetooth link to a device supporting the dial up networking profile. The first stage in establishing such a connection is finding out what Bluetooth enabled devices are in the area, so the laptop performs an inquiry to look for devices in the neighbourhood.

To do this the laptop transmits a series of inquiry packets, and eventually the cell phone replies with a Frequency Hop Synchronisation (FHS) packet. The FHS packet contains all the information that the laptop needs to create a connection to the cell phone. It also contains the device class of the cell phone, which consists of major and minor parts. The major device class tells the laptop that it has found a phone; the minor part says that the type of phone is a cellular phone. This exchange of messages is illustrated in Figure 1-7.

In the same way, every Bluetooth-enabled device in the area that is scanning for inquiries will respond with an FHS packet, so the laptop accumulates a list of devices.

What happens next is up to the designer of the application. The laptop could present the user with a list of all the devices it has found and let the user choose what to do next; but if it did that at this stage, all it could do is tell the user about the types of devices it has found. Instead of telling the user about the devices it has found, the application could au-

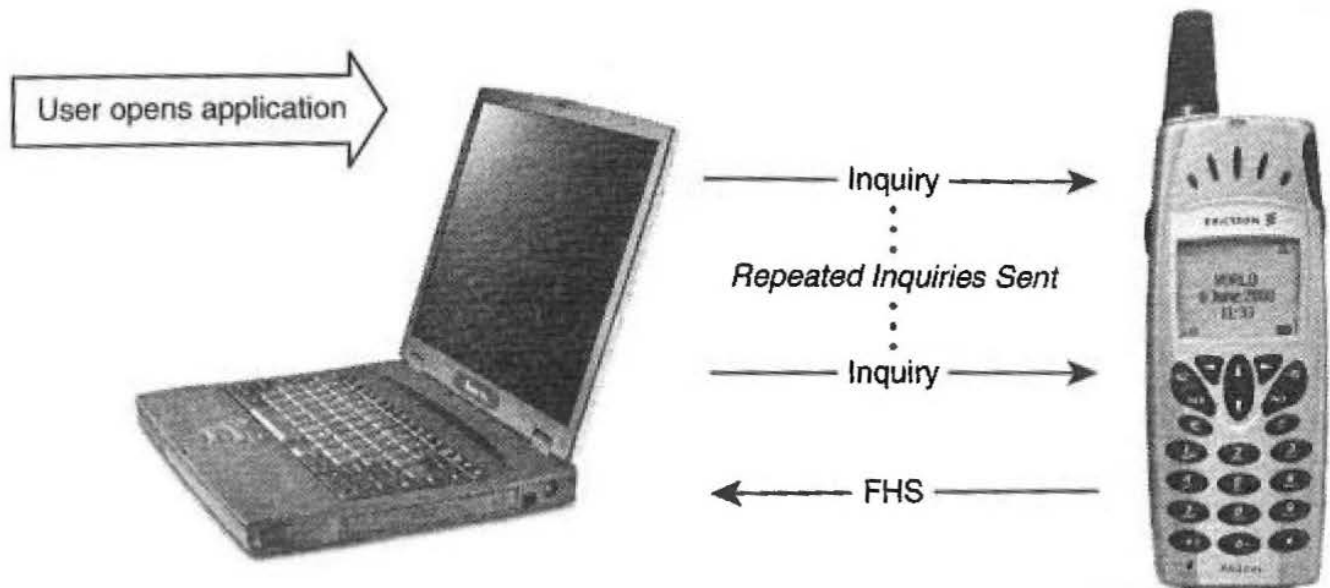


Figure 1-7 Discovering a Bluetooth device.

tomatically go on to the next stage and find out which devices in the area support the dial up networking profile.

1.7.2 Connecting to a Service Discovery Database

To find out whether a device supports a particular service, the application needs to connect to the device and use the service discovery protocol (SDP). Figure 1-8 illustrates how this is done. First the laptop pages the cellular phone, using the information it gath-

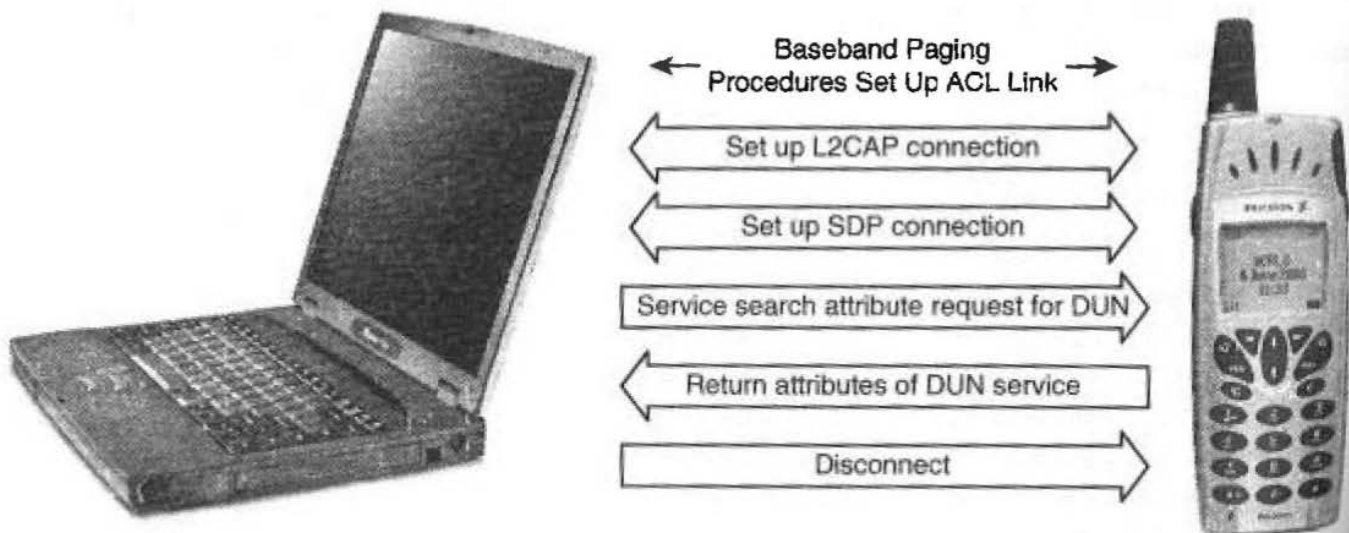


Figure 1-8 Retrieving information on services.

ered during inquiry. If the phone is scanning for pages, it responds, and an ACL baseband connection can be set up to transfer data between the two devices.

Once an ACL connection has been established, a Logical Link Control and Adaptation Protocol (L2CAP) connection can be set up across it. An L2CAP connection is used whenever data has to be transferred between Bluetooth devices. L2CAP allows many protocols and services to use one baseband ACL link. L2CAP distinguishes between different protocols and services using an ACL connection by adding a Protocol and Service Multiplexor (PSM) to every L2CAP packet. The PSM is different for every protocol or service that uses the link. Since this connection will be used for service discovery, its PSM = 0x0001, a special value that is always used for service discovery.

The laptop uses the L2CAP channel to set up a connection to the service discovery server on the cellular phone. The laptop's service discovery client can then ask the cellular phone's service discovery server to send it all the information it has relating to the dial up networking profile. The service discovery server on the cellular phone searches through its database and returns the attributes (characteristics) relating to dial up networking.

Once the service discovery information has been retrieved, the laptop may decide to shut down the connection to the cellular phone. If the laptop wants to collect service discovery information from many devices in the area, then it makes sense to shut down the links after using them, since one device can only use a limited number of links at a time, and keeping the links alive will consume battery power unnecessarily.

After the laptop has collected service discovery information from devices in the area, what happens next is again up to the application. It could display the information on all devices it has found which support the dial up networking profile and let the user decide which one to connect to. Alternatively, the application could decide for itself which device to use without bothering the user.

Either way, the service discovery information tells the laptop everything it needs to know to connect to the dial up networking service on the cellular phone.

1.7.3 Connecting to a Bluetooth Service

The process of actually making a connection is shown in Figure 1-9. The paging process which establishes a baseband ACL link is the same as was used when connecting for service discovery.

This time the link is being set up for a protocol which may have particular quality of service requirements, so the application running on the laptop may wish to configure the link to meet its requirements. This is done by the application sending its requirements to the Bluetooth module using the Host Controller Interface. Next, the module's link manager configures the link using the link management protocol.

Once the ACL connection is set up to the laptop's satisfaction, an L2CAP connection is set up. The dial up networking profile uses RFCOMM, an RS-232 emulation layer, so the L2CAP connection uses the Protocol Stack Multiplexor for RFCOMM (PSM = 0x0003).

After the L2CAP link has been set up, an RFCOMM connection can be set up across it. RFCOMM, like L2CAP, can multiplex several protocols or services across one connection. Each protocol or service is given its own channel number. The cellular

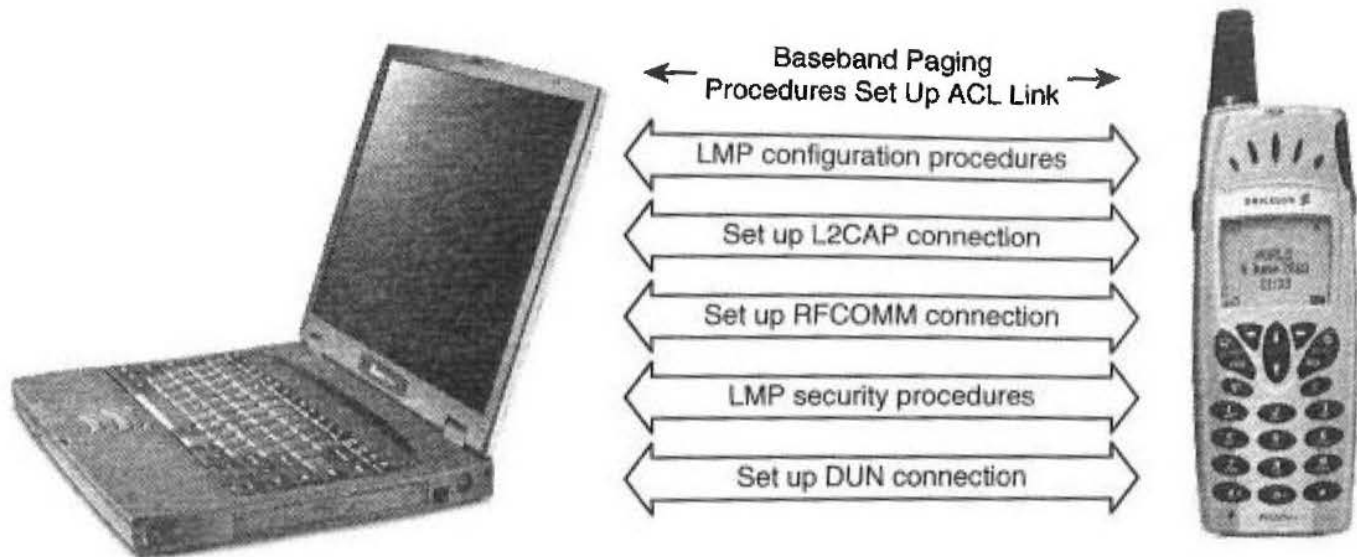


Figure 1-9 Connecting to a Dial Up Networking service.

phone's channel number for Dial Up Networking was sent to the laptop in the service discovery information, so the laptop knows which channel number it should use when setting up the RFCOMM connection.

Finally, the Dial Up Networking (DUN) connection is set up using the RFCOMM connection, and the laptop can start to use the dial up networking services of the cellular phone. Now, the laptop can use the cellular phone to make connections across the phone network without the two needing to be joined together by a data cable.

If the cellular phone is picked up and taken out of the range of the laptop, the laptop will have to repeat the procedure and find another device to connect to. Meanwhile, the cellular phone is still scanning and might be connected to another device elsewhere. The process of connecting is ad hoc and arbitrary with Bluetooth connections, possibly only lasting for a short period of time as devices move around.

1.7.4 Discoverability and Connectability Modes

It is important to realise that for a connection to be established using Bluetooth wireless technology, both ends of the link have to be willing to connect.

Some devices may be set so that they will not scan for inquiries; in this case, other devices cannot discover them, and they will effectively be invisible. Similarly, some devices may be set so that they do not perform page scans. In these cases, they can still initiate connections, but they will not hear other devices trying to connect to them.

Applications can choose whether to make devices connectable or discoverable. A connection cannot be forced on a device which is not in the correct mode to accept it.

1.8 MANAGEMENT

Some parts of the Bluetooth system have to manage the links, establishing ACL links as needed and disconnecting when they are finished with them. L2CAP could fulfill this function, but since links must also be managed at the RFCOMM and SDP levels, it makes sense to have a separate device manager.

The core specification of Bluetooth does not say how the connections should be managed, although the security white paper gives some hints. The reason for this lack of mention in the specification is that device management does not affect end-to-end interoperation, so it is safe to leave it up to individual implementers to find their own solutions. Furthermore, the most appropriate solution is likely to be different for different devices. For instance, a headset only has to handle a single link, which is used for SDP and the headset service, so management is likely to be pretty simple, whereas a LAN access point has to juggle multiple links and balance bandwidth between them, so management of the links will be much more complex.

Bearing in mind that no single solution will be optimal for all possible devices, Figure 1-10 shows a possible protocol stack with a device and security manager that can handle establishment and configuration of the links.

The device manager interfaces to the HCI layer, SDP, RFCOMM, L2CAP, and to applications. It can provide the following facilities:

- Fault management—Detects, isolates, and corrects abnormal operation.
- Accounting management—Enables charging for use of managed objects and services.
- Configuration and name management—Controls, identifies, collects data from, and provides data to managed objects to assist in interconnection services.
- Performance management—Evaluates the behaviour of managed objects and the effectiveness of communication activities.
- Security management—Provides security management to protect managed objects and services.

Those familiar with the OSI model will recognise these facilities as those provided by OSI management.

Applications will have to register with the Bluetooth device manager to use Bluetooth links. They may then ask the device manager for connections, and request particular security and quality of service levels. If all applications go through the device manager, this allows it to create a database of higher layer protocols, applications, and services using links.

The device manager can also handle timeouts, shutting down links when they are no longer needed.

Bluetooth's management facilities do not have to be handled by a separate device management entity as described above. This function could be built into L2CAP, or into a

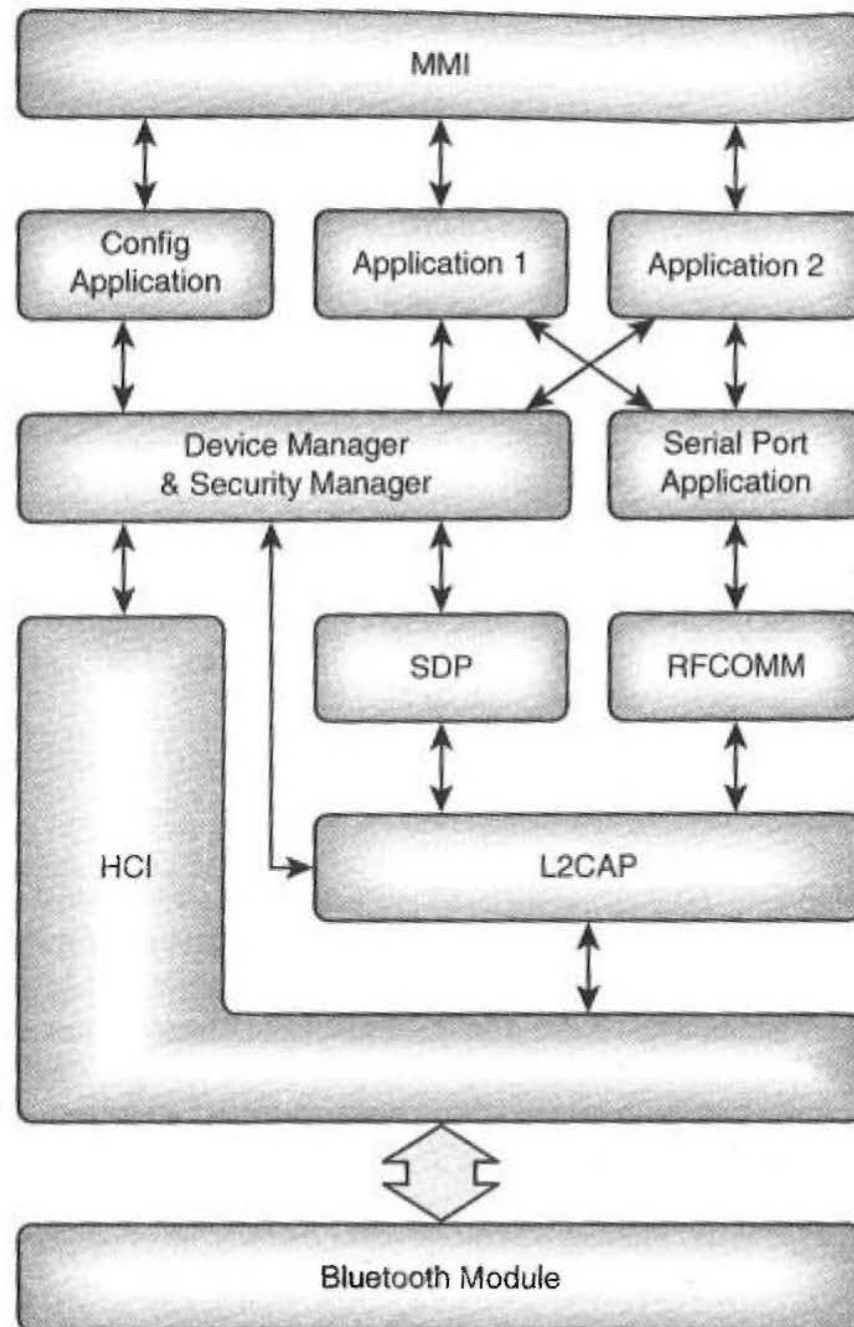


Figure 1-10 Management of the Bluetooth protocol stack.

profile application. However it is done, there must be some part of every Bluetooth implementation which handles management of the links.

1.9 TEST AND QUALIFICATION

The terms of the Bluetooth adopters agreement entitle all signatories to use the essential Intellectual Property Rights (IPR) underpinning Bluetooth free of charge. However, this

entitlement for a particular product is conditional on that product passing successfully through the Bluetooth qualification program. To facilitate recognition in the marketplace of approved Bluetooth devices, qualification also entitles a product to use the Bluetooth brand. This mechanism makes it possible for the SIG to take legal proceedings against any nonconformant products which purport to offer Bluetooth functionality but fall short of the specification. The result is that the market should see only products which work reliably with all other Bluetooth products supporting the same profiles.

The requirements for qualification are split into four categories:

- Bluetooth radio link requirements.
- Bluetooth protocol requirements.
- Bluetooth profile requirements.
- Bluetooth information requirements.

In addition to the preceding, manufacturers must also ensure that the radio is tested appropriately to meet local radio emissions standards in the countries where it will be sold.

Interoperability tests must also be carried out to ensure that Bluetooth devices correctly implement the Bluetooth profiles, and can interwork with other devices implementing the same profiles. Devices can then be sold with a statement concerning which profiles they support so that it is perfectly clear to consumers which products they will work with and in what ways.

1.10 BLUETOOTH IN CONTEXT

Bluetooth does not exist in a vacuum. This section summarises some of the issues that affect Bluetooth devices which are not covered by the Bluetooth specification itself: how to implement Bluetooth, related technologies, the market for Bluetooth, health concerns, and the future of the specification.

1.10.1 Implementing the Technology

The key issue faced by implementers is that of partitioning. The hardware/software partition inside the Bluetooth subsystem trades off performance, cost, and power consumption against risk and time to market. The partition between the host system and the Bluetooth subsystem is where (usually in software) the stack is partitioned. This will either add loading to the host's processing resources or require more performance and resources in the highly cost sensitive Bluetooth subsystem. Ultimately, the demand for a full implementation as a self contained, embedded Bluetooth solution will require very careful design and optimisation to avoid creating an uncommercial product.

The quality of the "user experience" is very important for any Bluetooth product and will also require careful design and a good understanding and specification of the target application. Bluetooth is going to be a very high volume system and demand very low cost with high levels of optimisation being crucial.

1.10.2 Related Technologies and Standards

Bluetooth, like most innovations throughout history, does not have the field to itself. There are many other initiatives and standards for wired and wireless data communications, either already deployed or under development. They vary between overlapping with Bluetooth's sphere of operation, while exhibiting clear differentiators, to potential head-on competitors of Bluetooth. The two most active areas of work at the current time are the distribution of data and voice in a personal sphere of influence, the so called Personal Area Network (PAN), which appears to be Bluetooth's home ground, and the emerging demand for high speed wireless multimedia data distribution.

1.10.3 The Bluetooth Market

Most commentators agree that the Bluetooth market is going to be huge, with forecasts putting the installed base at half a billion devices by 2004, with a total market for Bluetooth components worth \$2 billion in the same year. It seems that, for once, the technology push provided by Bluetooth is a good match for the market pull in terms of consumer needs and wants at this time.

There are a great many opportunities for Bluetooth-enabled products which exploit the various features of the technology to add value. However, there are many issues which are yet to be resolved. Potentially competing technologies could cause consumer confusion and at worst push Bluetooth into a niche corner. For manufacturers, the cost of the technology is paramount, and for Bluetooth to become ubiquitous, it must be built into all products, not just the high-end models.

For consumers, poor interoperability and/or poor user experience would be a major problem for Bluetooth and cause it to falter. The well-discussed "Out of Box Experience" has to be seamless and simple. The hope is that the strength of the Bluetooth brand will promote the notion of reliability and ease of use.

1.10.4 Health

Bluetooth uses frequency spectrum in the range of 2400 MHz to 2483.5 MHz. This range encompasses the natural frequency of H₂O molecular oscillation at 2450 MHz, which is also used by microwave ovens specifically to excite water molecules inside food in order to cook it.

Sharing the same frequency range as microwave ovens has led to some concerns that Bluetooth devices might *cook* their users. Some microwave radiation will be absorbed in flesh. It will be absorbed by field-induced rotation of polarized water molecules, which is converted to heat through molecular friction. Basically, the microwaves shake the water in flesh, and it heats up as it shakes. But, as the radiated output power of Bluetooth devices is incredibly low and spread in spectrum in time, experts concur that Bluetooth radiation does not pose a risk to health.

A 1 mW Bluetooth radio emits 1/1,000,000 the amount of power in a 1 KW microwave oven. Also, in a microwave oven, all the power is directed inward at the food, whereas in a Bluetooth device, the power is radiated outward, so the user only ever intercepts the smallest fraction of the radio waves which are heading in their direction.

It is interesting to compare Bluetooth devices with other popular communications devices. Bluetooth operates at 2.4 GHz and uses 1 mW (0 dBm) for most applications, with a maximum of 100 mW (20 dBm) for extended range. This means that Bluetooth signals have a penetration depth of only 1.5 cm into flesh. In comparison, cellular handsets have a power of 10 mW to 2 W peak, using 450 MHz to 2200 MHz, and exhibit a penetration depth of 2.5 cm in the middle of their range at 900 MHz. So, mobile cellular handsets give rise to a measurable heating effect of 0.1°C, compared with no measurable increase for Bluetooth devices. Although studies have shown this small heating effect, it is too low to be noticed by the user. Most of the temperature increases that mobile phone users feel when holding a handset next to their ears is caused by an insulating effect. Since the head radiates a lot of heat, if a handset blocks that radiation, then the head heats up. Getting a hot ear from a mobile phone is not necessarily a sign that you are absorbing radiation!

There has already been some controversy regarding cellular handsets and whether they have a negative impact on health. Although scientific opinion is pretty conclusive that there are no risks, to be safe, various organisations have undertaken studies and research and have laid down guidelines for exposure to radio frequencies.

The WHO, ICNIRP, and IEEE have developed Radio Frequency (RF) exposure recommendations and these guidelines have been adopted by many national authorities. In the usual way of health and safety guidelines, they incorporate large safety margins. The guidelines specify near-field¹ restrictions (referred to as SAR) between 10 MHz to 10 GHz, which devices with an output power of less than 1.6 mW are incapable of exceeding. So, all low power Bluetooth devices will fall within these restrictions. Higher power Bluetooth devices may need to be tested for SAR limits, and this will be done as part of radio regulatory testing.

The guidelines also specify a standard for total RF exposure. This is given as a power density of 10 W/m². This level of spectral density would require an unrealistic number of Bluetooth devices to operate continuously in a very small space, which would actually not be possible due to the limited spectrum in the ISM band.

Several expert panels formed from organisations such as WHO, ICNIRP, EC, and the Royal Society of Canada have debated the topic of health in the context of existing higher power cellular technology in recent years. They have all concluded that there is no credible or convincing evidence that RF exposure from wireless devices operating within accepted exposure limits causes adverse human health effects. They did, however, recommend additional research to clarify some areas and fill gaps in existing knowledge.

In conclusion, experts agree that Bluetooth devices are too low in power to have any negative health consequences. Even the higher power devices are an order of magnitude lower in power than existing cellular devices, which, based on existing research and official guidelines, have already been proven to be safe.

¹Near field are those radio waves found close to an emitting device.

RFCOMM

RS-232 serial ports have nine circuits, which can be used for transferring data and signalling. RFCOMM can emulate the serial cable line settings and status of an RS-232 serial port. RFCOMM provides multiple concurrent connections by relying on L2CAP to handle multiplexing over single connections, and to provide connections to multiple devices.

RFCOMM relies on the Bluetooth baseband to provide reliable in-sequence delivery of byte streams. It does not have any ability to correct errors. Up to version 1.0b RFCOMM's flow control relied entirely on the baseband's capabilities, but in version 1.1 RFCOMM acquired a credit based flow control system which allows individual RFCOMM channels to maintain separate flow control.

RFCOMM data rates will be limited in devices where there is a physical serial port involved (Type 2 devices). Implementations may optionally pace data on virtual serial ports (in Type 1 devices). In the absence of pacing, RFCOMM will deliver the highest possible data rate, although what the highest data rate is can be a complicated issue in the presence of multiple connections (see Chapter 18).

RFCOMM is a simple, reliable transport protocol with framing, multiplexing, and the following additional provisions:

- Modem status—RTS/ CTS, DSR/ DTR, DCD, ring.
- Remote line status—Break, overrun, parity.

- Remote port settings—Baud rate, parity, number of data bits, etc.
- Parameter negotiation (frame size).
- Optional credit based flow control.

10.1 SERIAL PORTS AND UARTs

Typically, serial port transmit and receive data lines are connected to a UART (Universal Asynchronous Receiver Transmitter). The job of the UART is to convert between the serial data sent down cables and the parallel data processing which devices use. UARTs use buffers to convert between serial and parallel data. This allows them to reduce the load on the processor. Instead of the processor having to be interrupted for every single bit that is sent down the cables, the UART transfers the bits between the cables and buffers, then the processor only has to get involved when there is a whole buffer to deal with.

The signals from a UART are connected, so they appear in the system address map. Some processors reserve a special range of addresses for I/O; other systems can map them into any part of normal memory. Because UARTs look like areas of memory to a microprocessor, it is possible to emulate a serial port in software by taking an area of memory and setting values as they would appear if they were set by a UART.

The Bluetooth RFCOMM specification talks about emulating the nine circuits of an RS-232 serial port, and specifies how a serial data stream can be emulated. But because the serial stream from an RS-232 port is viewed by the microprocessor after it has been through a UART, software dealing with serial ports is actually handling parallel data. Similarly, RFCOMM software deals with parallel data delivered by the lower layers of the Bluetooth stack.

A UART connects to some piece of hardware: wires or buffers. RFCOMM connects up to the lower layers of the software stack via L2CAP.

10.2 TYPES OF RFCOMM DEVICES

RFCOMM supports two types of devices:

- Type 1—Internal emulated serial port (or equivalent).
- Type 2—Intermediate device with physical serial port.

A protocol stack for a Type 1 RFCOMM device is shown on the left in Figure 10-1. The port emulation entity maps a system specific communication interface (API) to the RFCOMM services. This can be used to connect to legacy applications as shown, or it can be used to connect to applications specifically written for Bluetooth. A Type 1 device would usually be the end of a communication path, for example, a PC or printer.

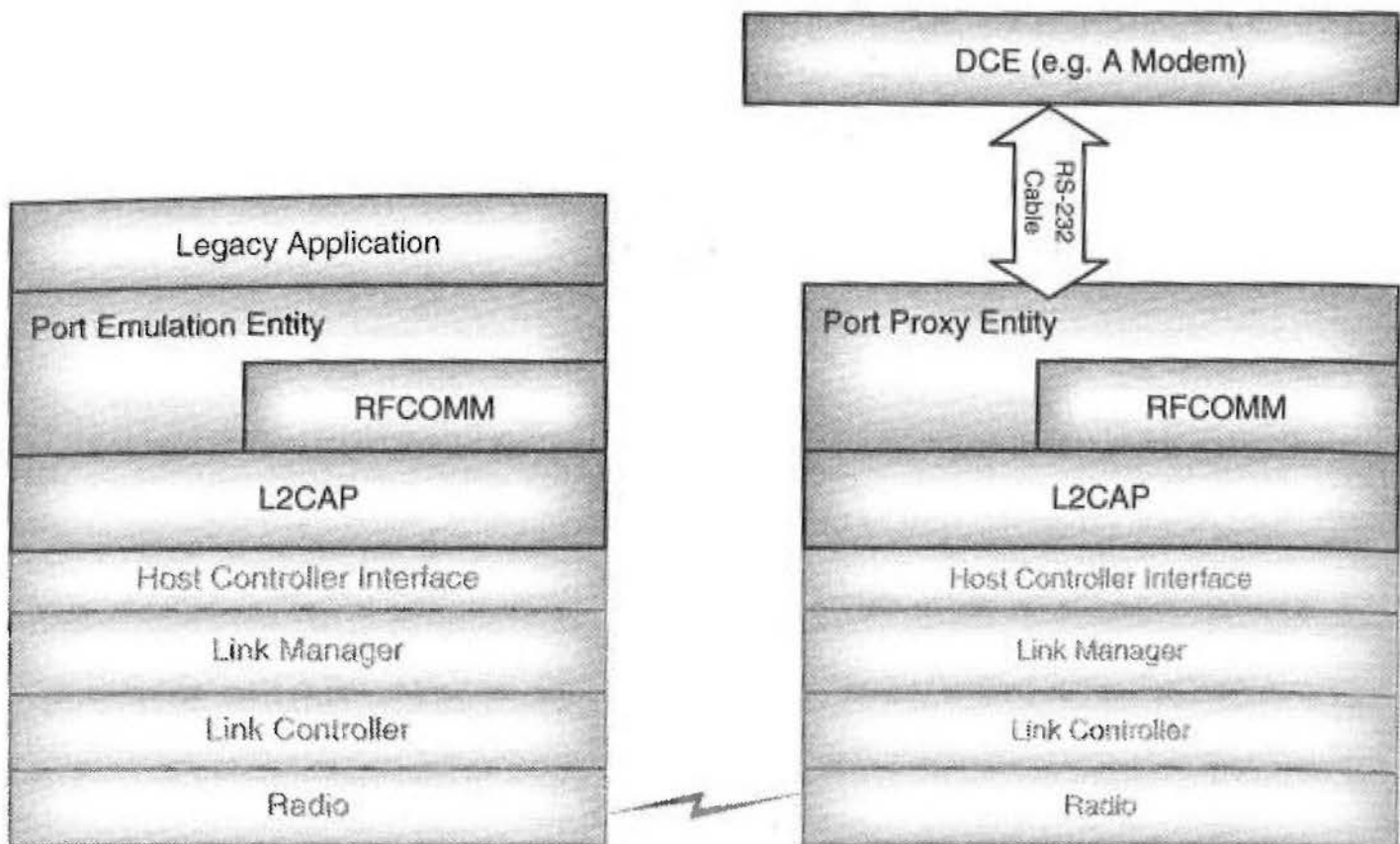


Figure 10-1 Type 1 and Type 2 RFCOMM devices.

A protocol stack for a Type 2 RFCOMM device is shown on the right in Figure 10-1. The port proxy entity relays data from RFCOMM to an external RS-232 interface linked to another device. Type 2 devices are intermediate devices which sit in the middle of a communication path. A modem is an example of a Type 2 device.

10.3 RFCOMM FRAME TYPES

RFCOMM is based on GSM TS 07.10, which is an asymmetric protocol used by GSM cellular phones to multiplex several streams of data onto one physical serial cable. RFCOMM is symmetric, and sends TS 07.10 frames over L2CAP using a subset of TS 07.10 feature frames and commands. Some of TS 07.10's features are adapted for Bluetooth.

RFCOMM communicates with frames. The RFCOMM frames become the data payload in L2CAP packets. There are five different frame types:

- SABM—Start Asynchronous Balanced Mode (startup command).
- UA—Unnumbered Acknowledgement (response when connected).
- DISC—Disconnect (disconnect command).

- DM—Disconnected Mode (response to a command when disconnected).
- UIH—Unnumbered Information with Header check.

SABM, UA, DM, and DISC are “low-level” control frames. RFCOMM uses channels, each of which has a Data Link Connection Identifier (DLCI). UIH frames on $DLCI = 0$ are used to send control messages. UIH frames on $DLCIs \neq 0$ are used to send data.

GSM TS 07.10 also has optional UI (Unnumbered Information) frames; RFCOMM doesn't use these.

10.4 CONNECTING AND DISCONNECTING

Because RFCOMM frames are carried in the payload of L2CAP packets, an L2CAP connection must be set up before an RFCOMM connection can be set up.

RFCOMM has a reserved Protocol and Service Multiplexer (PSM) value which is used by L2CAP to identify RFCOMM traffic. The RFCOMM PSM is defined in the Bluetooth core specification as 0x0003. Any L2CAP frames received with this value in the PSM field will be sent to RFCOMM for processing.

The first frame to be sent on an RFCOMM channel is a SABM frame; this is a Start Asynchronous Balanced Mode command. If the responding device's RFCOMM is willing to connect, it goes into Asynchronous Balanced Mode (ABM) and sends back an UA frame. If the responding device's RFCOMM doesn't want to connect, it refuses the connection by sending a DM frame. Figure 10-2 shows an RFCOMM channel setup being refused in this way.

RFCOMM has a 60 s timer which is started when a command is sent. If an acknowledgement isn't received when the timer elapses, the connection will be shut down. This is different from GSM 07.10, which resends the command when the timer elapses. In the case of RFCOMM, the Bluetooth baseband provides a reliable link, so if the command wasn't acknowledged first time, it is not likely to be acknowledged a second time. For the

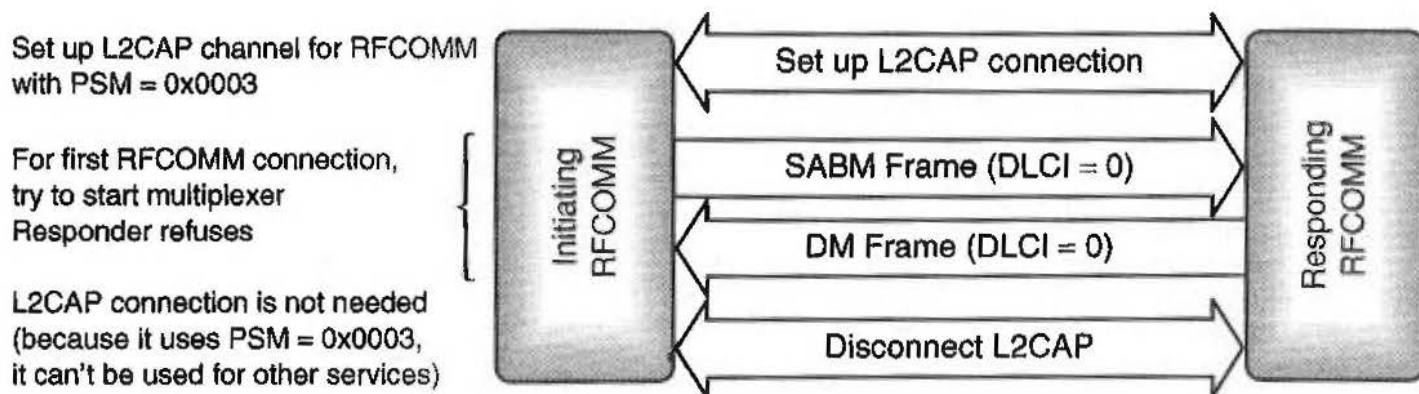


Figure 10-2 Refusing an RFCOMM connection.

SABM command, the timeout can be extended because security procedures might mean that this command takes longer to process than others. If RFCOMM ever times out and disconnects, it must send a DISC (disconnect) command frame on the same DLCI as the original SABM frame, just in case the other side has come back into range and thinks the connection is still active. Figure 10-3 shows a channel being closed down because the initiator timed out.

If the connection succeeds, the responder replies to the SABM frame with a UA (un-numbered acknowledgement) frame. This is followed by a Parameter Negotiation (PN) command from the initiator and PN response from the responder, as shown in Figure 10-4.

Once a connection with DLCI = 0 has been set up, this is available for RFCOMM signalling. To transfer data, other RFCOMM channels must be set up. Figure 10-4 shows a second RFCOMM channel being set up to transfer data. In this case, the channel requires authentication, so there is a pause for LMP authentication and encryption between the SABM command frame and the UA response frame. Once the UA frame has been received, modem status commands are exchanged to communicate the state of the control signals. Data can then be transferred immediately as shown, or there could be an exchange of PN command and response to configure the new connection's parameters.

The user data should include MSCs (Modem Status Commands), which communicate the state of serial port control signals.

To shut down an RFCOMM connection, a DISC command is sent. When the last data link has been shut down, a DISC should be sent on DLCI=0 to shut down the multiplexer. Then, whichever device shut down the multiplexer is responsible for disconnecting the L2CAP channel.

10.5 STRUCTURE OF RFCOMM FRAMES

RFCOMM borrows its frame structure from the GSM 07.10 standard. Figure 10-5 shows the frame structure for the GSM 07.10 basic option. (There is also an advanced option, which lacks the length field; RFCOMM always uses the basic option.)

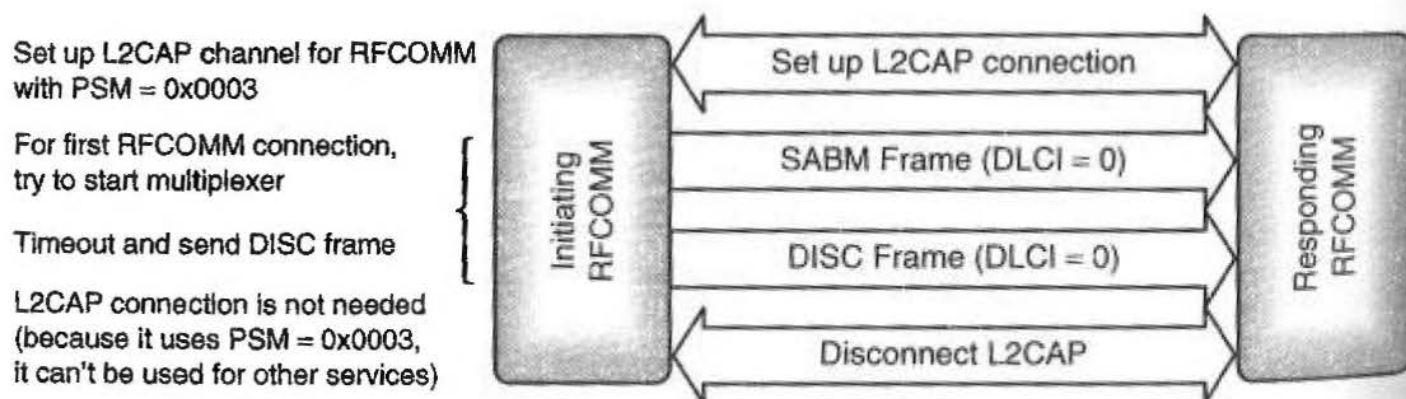


Figure 10-3 RFCOMM channel timeout.

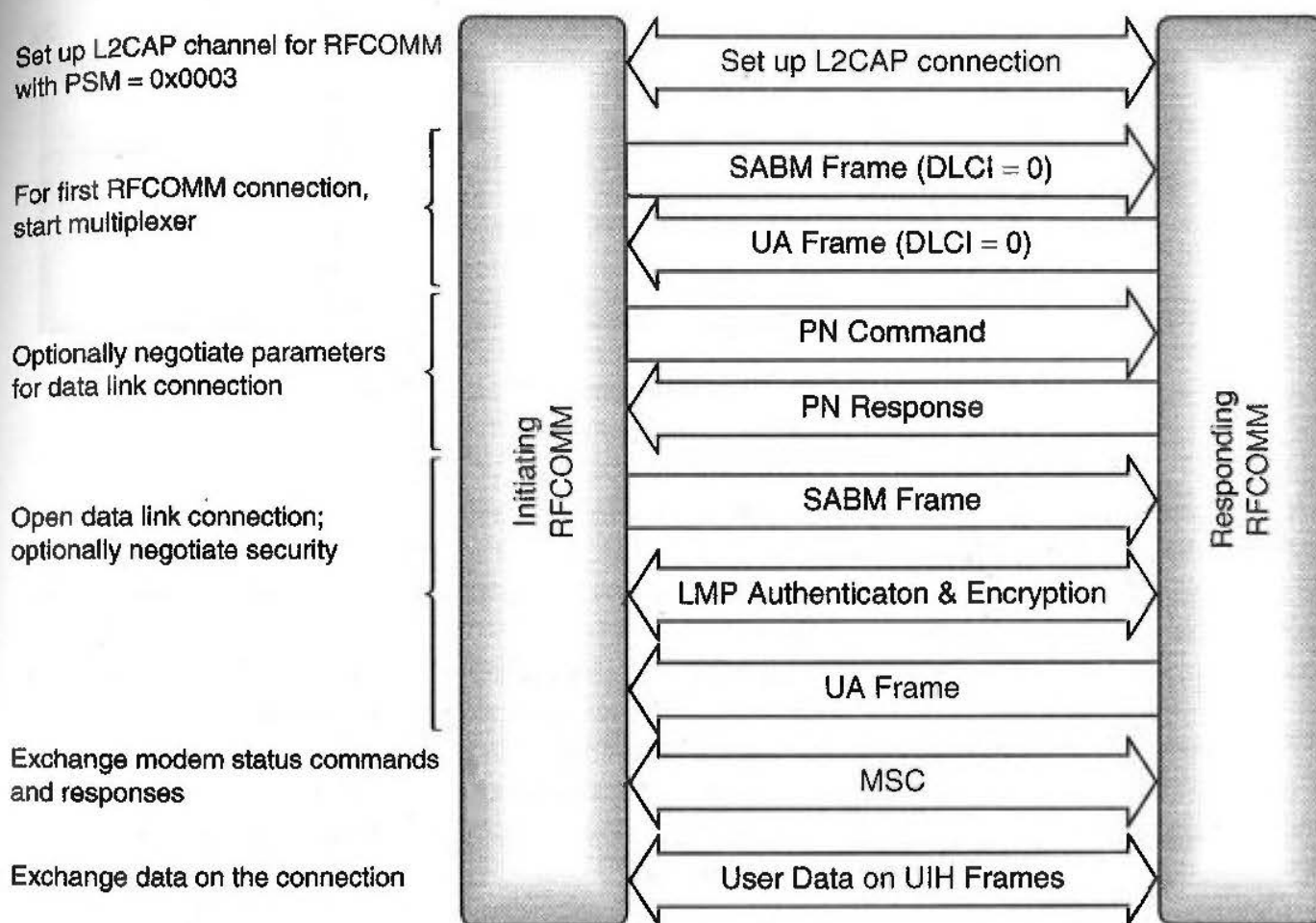


Figure 10-4 Message sequence chart for establishing an RFCOMM connection.

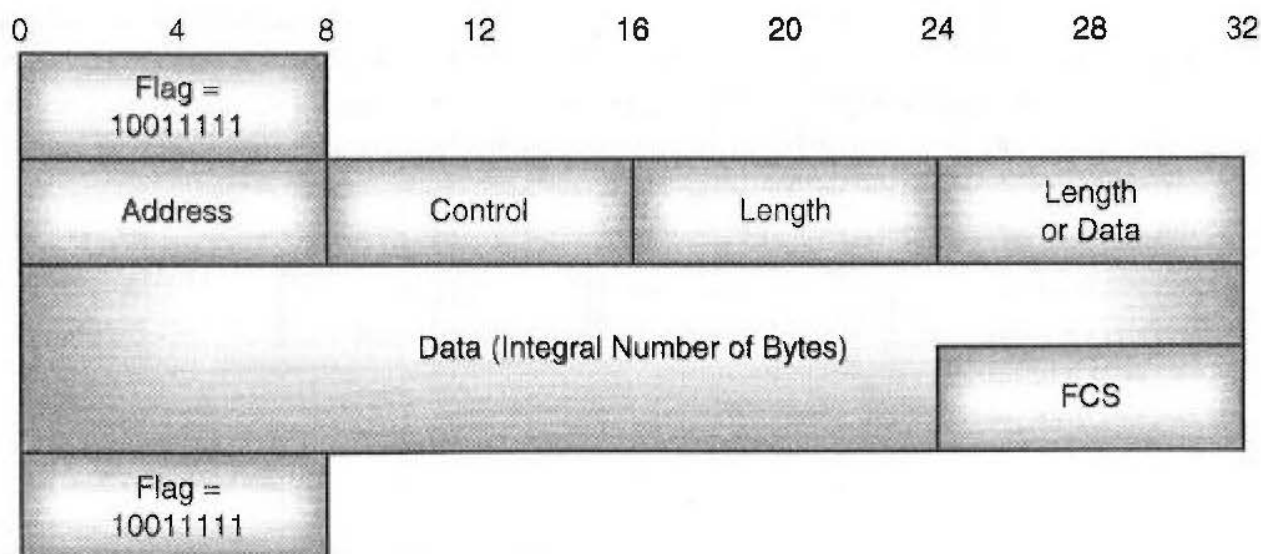


Figure 10-5 Structure of a GSM07.10 basic option frame.

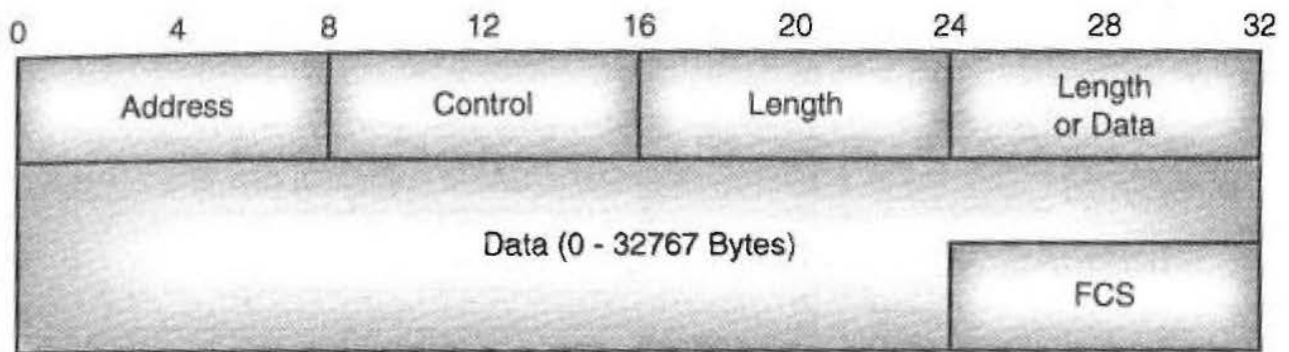


Figure 10-6 Structure of an RFCOMM frame.

Figure 10-6 shows the structure of an RFCOMM frame. This is identical to the GSM07.10 basic option frame except that RFCOMM misses out the start and end flags, and RFCOMM has to limit the number of bytes in a packet because of the limit on the size of L2CAP packets.

RFCOMM doesn't need the start and end flags because each RFCOMM frame is carried in a single L2CAP packet. There is no need to pick RFCOMM frames out of a data stream, so there is no need for flag bits to mark where the frames start and end.

Figure 10-7 shows how the RFCOMM frame structure changes when credit based flow control is in use. The length field is followed by a credit field which is used to give the remote device permission to send frames.

10.5.1 Address Field

An RFCOMM frame begins with an address field. This identifies which of the many multiplexed channels the frame belongs to. The address field is split up as shown in Figure 10-8.

The EA (Extend Address) field can be used to extend an address. If EA=0, then more address octets follow; if EA=1, then this is the last octet of the address. Since the Bluetooth specification says that server applications are assigned a server channel number in the range 1 to 30 and an RFCOMM address frame has five bits for the server channel,

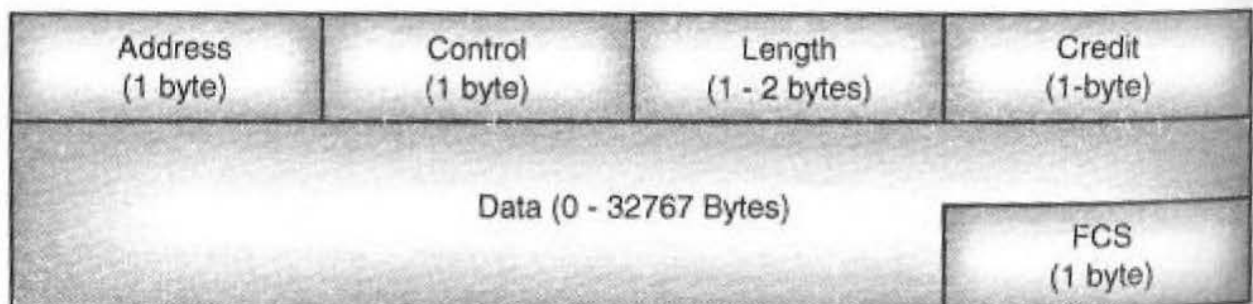


Figure 10-7 Structure of an RFCOMM frame with credit based flow control.

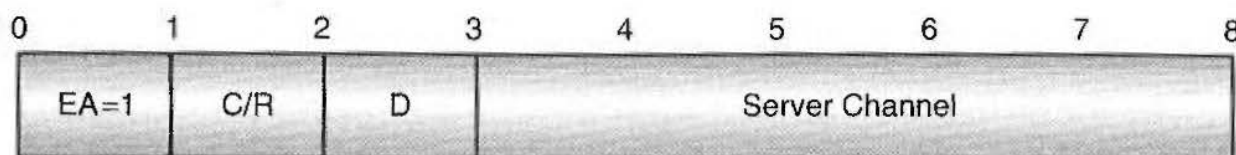


Figure 10-8 Address field.

there will never be any need to use extended addressing, so the EA bit will always be set to 1 in an RFCOMM address field.

The C/R (Command/Response) bit says whether the frame is a command or a response. Its value depends not only on whether the frame carries a command or a response, but also on which end of the channel is sending the frame. The device which set up the connection (by sending a SABM command on DLCI 0) is called the initiator. The device which responded (by sending the UA response on DLCI 0) is called the responder. As long as the traffic follows this original pattern, the C/R bit is 1, so commands from the initiator and responses from the responder have C/R = 1. For exchanges in the opposite direction, the C/R bit is 0, so commands from the responder and responses from the initiator have C/R = 0. When sending data, the initiator sets C/R = 1 and the responder sets C/R = 0. Figure 10-9 illustrates how the C/R bit is set.

After the C/R bit comes the Data Link Connection Identifier (DLCI). In GSM TS 0.10, this is one undivided field, but in RFCOMM, it is split up into a direction bit and a server channel number. The initiator always sets the direction bit to 1 (D=1); the responder always sets the direction bit to 0 (D=0). As with the C/R bits, who is the initiator and responder is defined by which device sent the SABM frame to start up the connection.

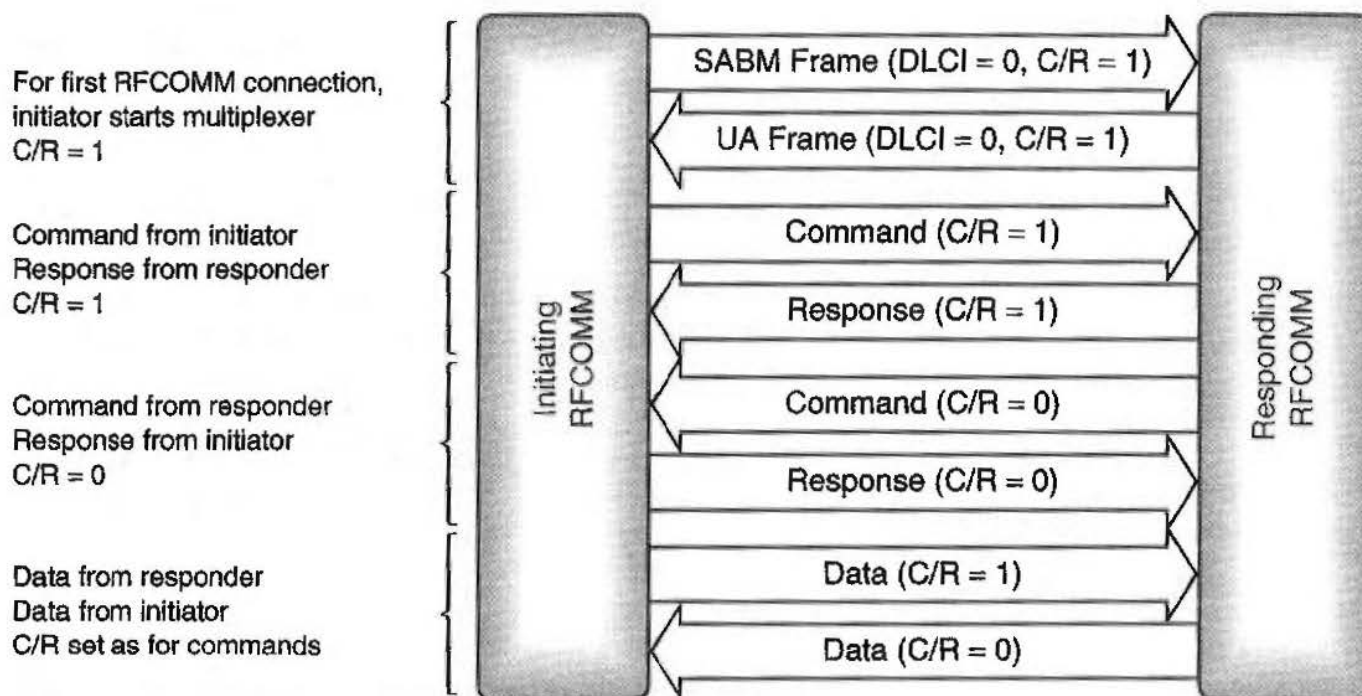


Figure 10-9 Settings of C/R bit in RFCOMM exchanges.

Table 10-1 Control Field Values

Control Bit Number	1	2	3	4	5	6	7	8
SABM (Set Asynchronous Balanced Mode)	1	1	1	1	P/F	1	0	0
UA (Unnumbered Acknowledgement)	1	1	0	0	P/F	1	1	0
DM (Disconnect Mode)	1	1	1	1	P/F	0	0	0
DISC (Disconnect)	1	1	0	0	P/F	0	1	1
UIH (Unnumbered Information with Header Check)	1	1	1	1	P/F	1	1	1

The server channel number has five bits; this would give it a range from 0 to 31, but 0 and 31 are reserved, so only 1 to 30 can be allocated as server channel numbers for services. Channel 0 is used for sending control information; channel 31 is reserved by TS 07.10. Bluetooth avoids using channels which are reserved by TS 07.10 to preserve compatibility with TS 07.10 applications.

The DLCI is calculated once before the data link connection is established. The RFCOMM server channel number in the responding device is used for the DLCI. Because server channel numbers 1 to 30 are available, one device can have up to 30 services using RFCOMM.

10.5.2 Control Field

Referring back to Figure 10-6, the next field in an RFCOMM frame is the control field. This is used to identify the type of the frame. Table 10-1 shows the control field values used in RFCOMM frames. (These match the values used in the corresponding GSM TS 07.10 Frames.)

P/F is the Poll/Final bit. In commands, it is called the P (Poll) bit; in responses, it is called the F (Final) bit.

A command with its P bit set to 1 is used when a response or a series of responses is wanted from the device at the far end of the link. The responding device should send back its responses with the F bit set to 1. There should only ever be one command frame with P bit set to 1 waiting for a response.

DM packets are processed regardless of the state of the P/F bit, but SABM or DISC commands and UA responses are thrown away if the P/F bit is set to zero.

10.5.3 Length Field

The length field begins with an EA bit as shown in Figure 10-10. If this is 1, then it is followed by seven bits of length, so the length field is one byte long. If the EA bit is 0, then it is followed by fifteen bits of length, so the length field is two bytes long.

The default length of an RFCOMM frame is 32 bytes; the maximum length is 32,767.

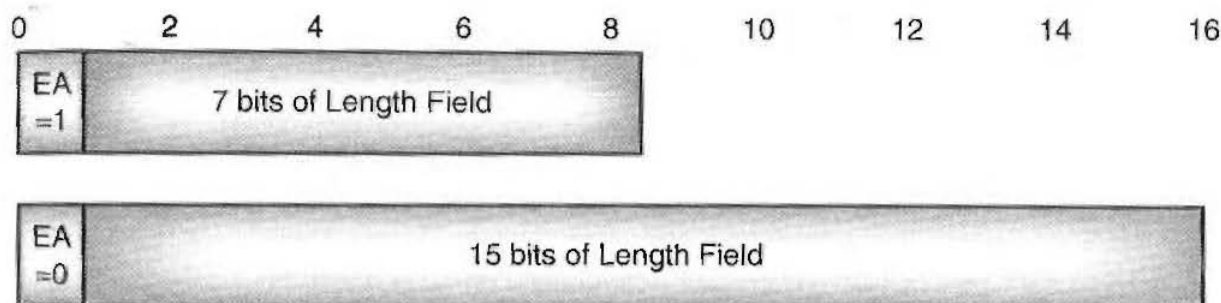


Figure 10-10 Structure of RFCOMM length fields.

10.5.4 Credit Field

The credit field only exists when the use of credit based flow control has been negotiated. An initial number of credits is granted using three bits in the parameter negotiation command. Each unit then keeps track of how many credits it has. Each time an RFCOMM frame is sent one credit is used up, each time a frame with a non-zero credit field is received, the value in the credit field is added onto the receiving units total credits.

10.5.5 Data

The data field is only present in UIH frames. There must be an integral number of bytes in the data up to 32,767. The size limit is set by the Maximum Transmission Unit (MTU) on L2CAP packets, so if a system has a smaller L2CAP MTU, the size of RFCOMM data will also be restricted.

10.5.6 Frame Check Sequence

To calculate the FCS:

Count up k , the number of bits the FCS will be calculated on. For SABM, DISC, UA, and DM frames, the frame check sequence is calculated on the address control and length fields. For UIH frames, it is calculated on the address and control fields.

Then:

- Calculate the remainder of $x^k (x^7 + x^6 x^5 + x^4 + x^3 + x^2 + x^1 + 1)$ divided modulo 2 by the generator polynomial $(x^8 + x^2 + x + 1)$.
- Take the contents of the frame that the FCS is calculated over before any start and stop elements have been inserted and before any other extra bits have been inserted. Multiply by x^8 and divide by the generator polynomial $(x^8 + x^2 + x + 1)$.
- Add the results of (a) and (b) modulo 2, and take the 1's complement to get the FCS.

Because UIH frames only calculate FCS on the address and control fields, their data field is not protected by the FCS. This might be a drawback for reliable data transmission, but it does have the advantage that the FCS patterns can be precalculated for all the DLCIs that are in use. This precalculation could be done when the channel is set up.

10.6 MULTIPLEXER FRAMES

Multiplexer commands and responses are sent on DLCI = 0. They are used to control the RFCOMM link. There are seven types of commands or responses:

- PN—DLC parameter negotiation.
- Test—Checks communication link.
- FCon / FCoff—Aggregate flow control on all connections.
- MSC—Modem status, used for flow control per connection.
- RPN—Remote Port Negotiation.
- RLS—Remote Line Status.
- NSC—Non-Supported Command (response only).

The multiplexer commands and responses are carried as messages inside an RFCOMM UIH frame as shown in Figure 10-11. It is possible to send several multiplexer command messages in one RFCOMM frame or split a multiplexer command message over more than one frame.

10.6.1 PN—DLC Parameter Negotiation

The PN command is used to negotiate the parameters of a data link connection. PN commands are exchanged before a new data link connection is opened. In version 1.0b this was not compulsory; in version 1.1 parameter negotiation became mandatory.

In pre-version 1.1 systems, if no PN commands are sent, default parameters will be used for the connection. This means that pre-version 1.1 systems must be able to support the default RFCOMM frame length.

A PN command is identified by the type field shown in Figure 10-12. This type field is the first information byte in the UIH frame carrying the PN command (see Figure 10-12). The EA bit is 1 because the type field occupies one byte. The C/R bit is used to indicate whether the message is a command or a response.

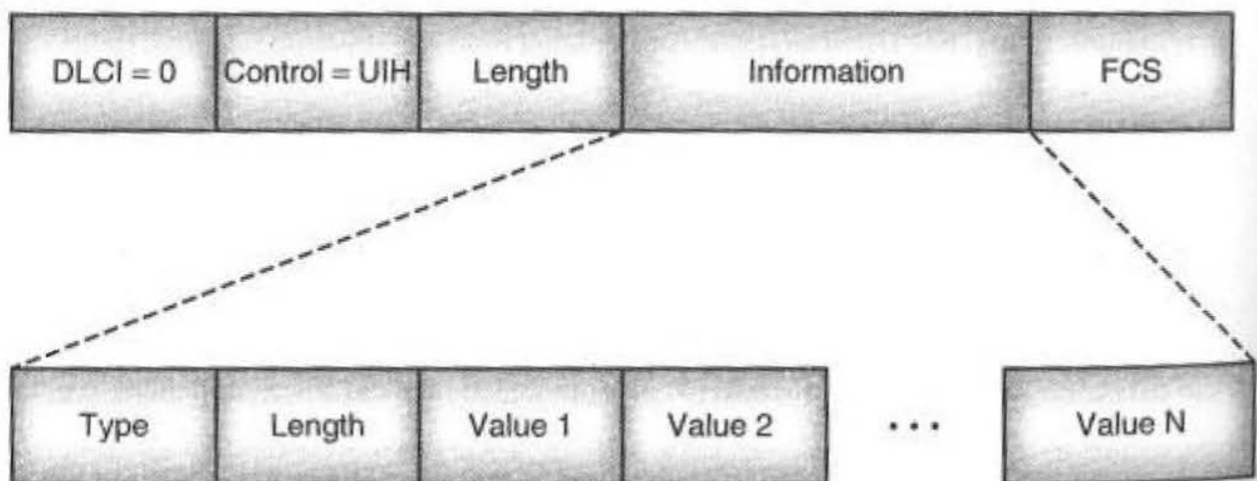


Figure 10-11 Structure of an RFCOMM multiplexer control frame.

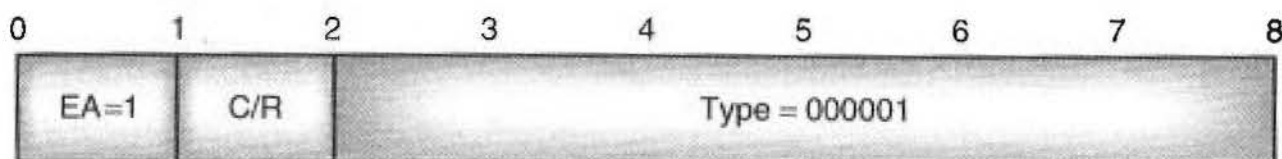


Figure 10-12 Type field for Parameter Negotiation (PN).

The length field in a PN message is always set to 8, and the value field contains 8 bytes as shown in Figure 10-13. These bytes define the parameters which will be used on a data link connection as follows:

- Six DLCI bits identify the data link connection for which parameters are being negotiated.
- Two padding bits which are always set to zero follow the DLCI; these are inserted to avoid splitting parameters across bytes.
- Four I bits give the type of frames used to carry information on the channel. In RFCOMM UIH frames indicated by the value 0b1000 are used.

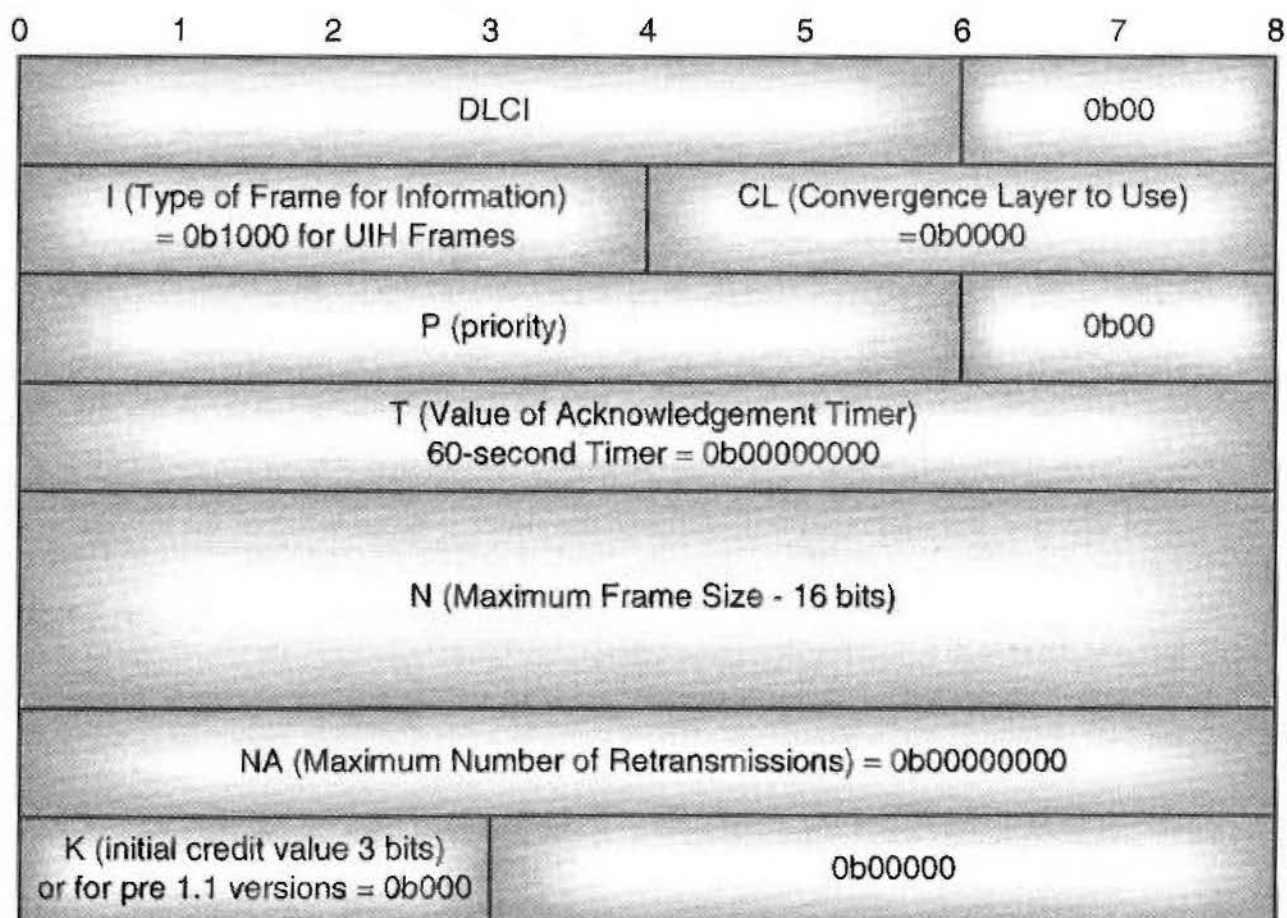


Figure 10-13 Content of value bytes in a PN message.

- Four CL bits give the convergence layer to be used. RFCOMM uses Type 1 (unstructured octet stream) = 0b0000 in versions after 1.0b this may also be set to 0x0F to enable credit based flow control.
- Six P bits assign a priority to the data link connection: 0 is the lowest priority, 63 is the highest.
- Two padding bits which are always set to zero follow the P bits; these are inserted to avoid splitting parameters across bytes.
- Eight T bits give the value of the acknowledgement timer, in GSM 07.10, this would be used to trigger a retransmit; in RFCOMM, if the timer elapses, the connection is closed down. The timer's value is not negotiable, but is fixed at 60 s. This field is set to 0 to indicate that the timer is not negotiable.
- Sixteen N bits give the maximum size of the frame.
- Eight NA bits give the maximum number of retransmissions. Because the Bluetooth baseband gives RFCOMM a reliable transport layer, RFCOMM will not retransmit, so this value is set to zero.
- Three K bits are used to indicate the initial number of credits issued to a peer; the initial credit value can range from 0 to 7. If credit based flow control is not used, the k1-k3 bits are set to zero.
- Five padding bits set to zero fill the rest of the last value field to round up the values to an integral number of bytes.

All parameters being negotiated are sent with the LSB in bit 1; this is the general rule for RFCOMM information.

It is worth noting that RFCOMM follows the conventions of the GSM 07.10 standard and numbers the bits in a frame from 1 for the least significant bit to 8 for the most significant bit. This may be confusing to people who are used to seeing the bits in a byte numbered from 0 to 7.

One device sends a PN message, and the other responds with another PN message. The response may not change the DLCI, the priority, the convergence layer, or the timer value. The response may send back a different timer value. In this case, the device which sent the first PN messages will still use the timer it proposed, but the device at the other end of the connection will use the value it sent in its message.

The response may have a smaller value for the maximum frame size, but it may not propose a larger value for this parameter.

In GSM 07.10, PN messages are optional. In RFCOMM, support of the PN message and its response are compulsory. Up to version 1.0b if default parameters are satisfactory, PN messages do not have to be sent. Version 1.1 made PN messages mandatory, even if the default parameters are satisfactory. PN messages may be exchanged until the device which sent the first message is happy with the parameters it gets sent back to it. Once it has a satisfactory set of parameters in the reply, it can go on to set up the connection.

One reason for parameter negotiation mandatory was to enable negotiation of credit based flow control. Without mandatory parameter negotiation the responding unit could be left with no opportunity to negotiate. When not using credit based flow control,

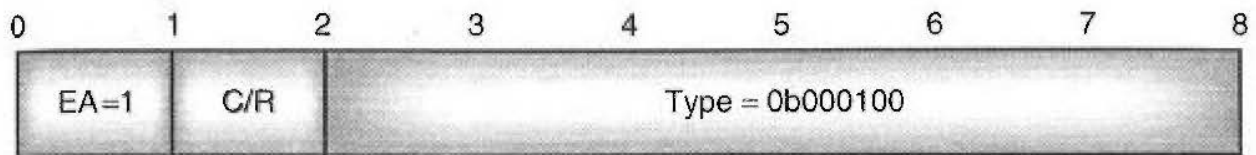


Figure 10-14 Type field for test.

RFCOMM channels used FCON and FCOFF messages for flow control on individual channels. Because Bluetooth channels are unreliable, these messages could require several retransmissions to switch off the flow of packets. This could lead to RFCOMM layer buffers overflowing and data being lost. Therefore, applications which would suffer if data buffers overflow must negotiate credit based flow control.

10.6.2 Test

The test command is used to check the RFCOMM connection. As is normal, the length byte gives the number of value bytes which follow. The number of value bytes is not fixed and is used to hold a test pattern. The remote end of the link echoes the same value bytes back.

The type field for the test command is shown in Figure 10-14. Because only a single byte is used, the EA bit is set to 1. The C/R bit is used to indicate whether the message is a command or a response.

10.6.3 FCon / FCoff—Aggregate Flow Control on All Connections

RFCOMM has a flow control mechanism which applies to all channels between two RFCOMM entities. When either RFCOMM entity can't receive RFCOMM information, it sends a Flow Control off (FCoff) command. When it is able to receive data again, it sends a Flow Control on (FCon) command.

The structure of the type field for the two flow control commands is shown in Figure 10-15. Both begin with an EA bit, which is 1 to indicate that there is only one byte in the command type. The length field in the frame carrying the command is set to zero because there is no other data in the frame. The C/R bit is used to indicate whether the message is a command or a response.

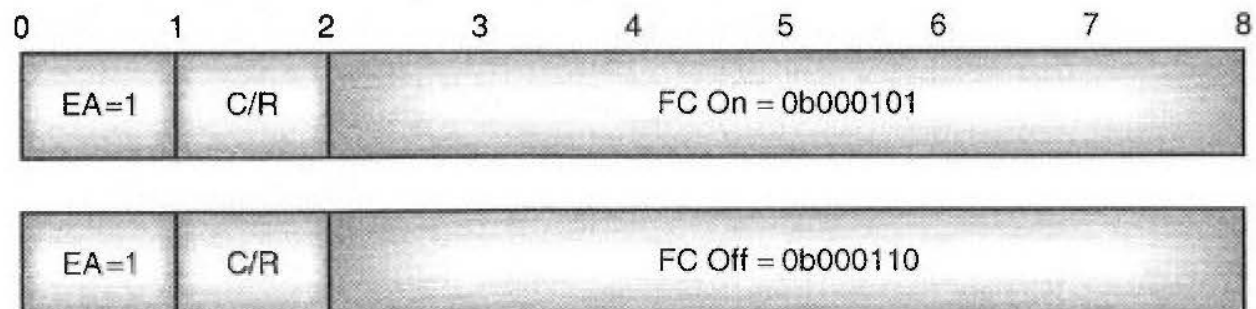


Figure 10-15 Type fields for FCon and FCoff commands.

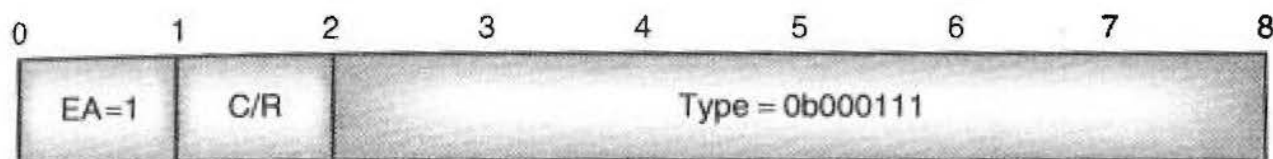


Figure 10-16 Type field for MSC command.

10.6.4 MSC—Modem Status Command

RFCOMM also has a flow control mechanism which can be applied to just one channel at a time. This is the Modem Status Command (MSC), and it is indicated by the type field shown in Figure 10-16. The EA bit is 1 because the type field occupies one byte. The C/R bit is used to indicate whether the message is a command or a response.

The command field of the MSC contains virtual V.24 control signals—that is to say, the settings that the RS 232 control wires would have if the RFCOMM data were being transferred across wires rather than across a Bluetooth connection. The signals in the command field are as follows:

- EA—Extended Address, set to 1 to indicate there is only 1 byte of command.
- FC—Flow Control bit, set to 1 when a device is unable to accept any RFCOMM frames. When the device is able to receive again, it sends another MSC with the flow control bit set to 0.
- RTC—Ready To Communicate bit, set to 1 when the device is ready to communicate.
- RTR—Ready To Receive bit, set to 0 when the device cannot receive data and 1 when it can receive data.
- IC—Incoming Call, 1 indicates an incoming call.
- DV—Data Valid, set to 1 to indicate that valid data is being sent.

These values may not seem to make sense when sent in a packet, but that is because they map onto the lines of an RS 232 interface. It might be obvious when a packet is sent that it is going to have valid data; who would bother to send a packet with invalid data? However, when dealing with physical serial port lines, such signals make more sense. A signal to say that valid data is being sent can be used to activate circuits to handle the data. The MSC command just mimics the values of the V.24 signals, which would be used on a wired RS 232 interface.

The signals from an MSC map onto RS 232 signals as follows:

- RTC maps onto DSR (Data Set Ready) and DTR (Data Terminal Ready).
- RTR maps onto RTS (Request To Send) and CTS (Clear To Send).
- IC maps onto RI (Ring Indication).
- DV maps onto DCD (Data Carrier Detect).

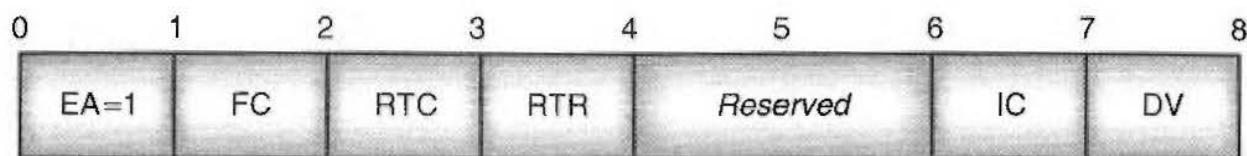


Figure 10-17 MSC control signal field.

Figure 10-17 shows how the signals are carried in the control field of the command.

The MSC is sent on a connection before any data is sent to establish the state of the RS 232 control signals. It should also be sent whenever the signals need to be changed.

In the MSC, the state of the signals in the device sending the command is sent. The response just carries a copy of the signals from the command.

10.6.5 RPN—Remote Port Negotiation

The Remote Port Negotiation (RPN) command is used to set communication settings at the remote end of a data link connection. If any of the communication settings need to be changed during a connection, the RPN command can be resent to change them.

The RPN type field is shown in Figure 10-18.

The EA bit is 1 because the type field occupies one byte. The C/R bit is used to indicate whether the message is a command or a response.

The length byte in an RPN command is either 1 or 8. If the length is 1, then there is a single value byte which contains the DLCI for the connection, and the message is interpreted as a request for the link's parameters. In this case, the remote end replies with the current parameters on the link. If the length byte is set to 8, then eight bytes of link parameters follow. If they are sent in a command, then they are a request to set up the link's parameters.

Figure 10-19 shows the order of values within an RPN command with length byte set to 8.

The parameter mask defines which parameters are being changed by the message. Figure 10-20 shows the position of bits in the RPN parameter mask.

In an RPN command, if a parameter mask bit is set to 1, it indicates a particular parameter that should be changed. If it is set to 0, then the parameter is not being changed and the value can be ignored.

In an RPN response, if a parameter mask bit is set to 0, then it means the proposal sent in an RLS has not been accepted. Conversely, a parameter mask bit set to 1 means

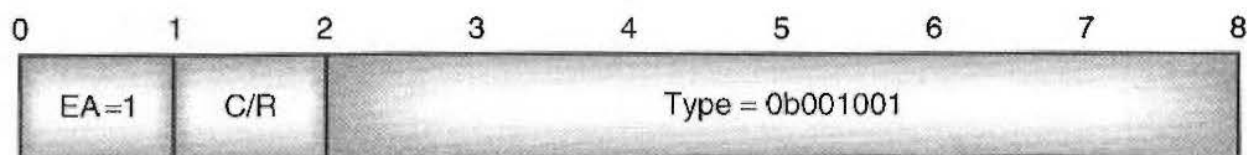


Figure 10-18 Type field for RPN.

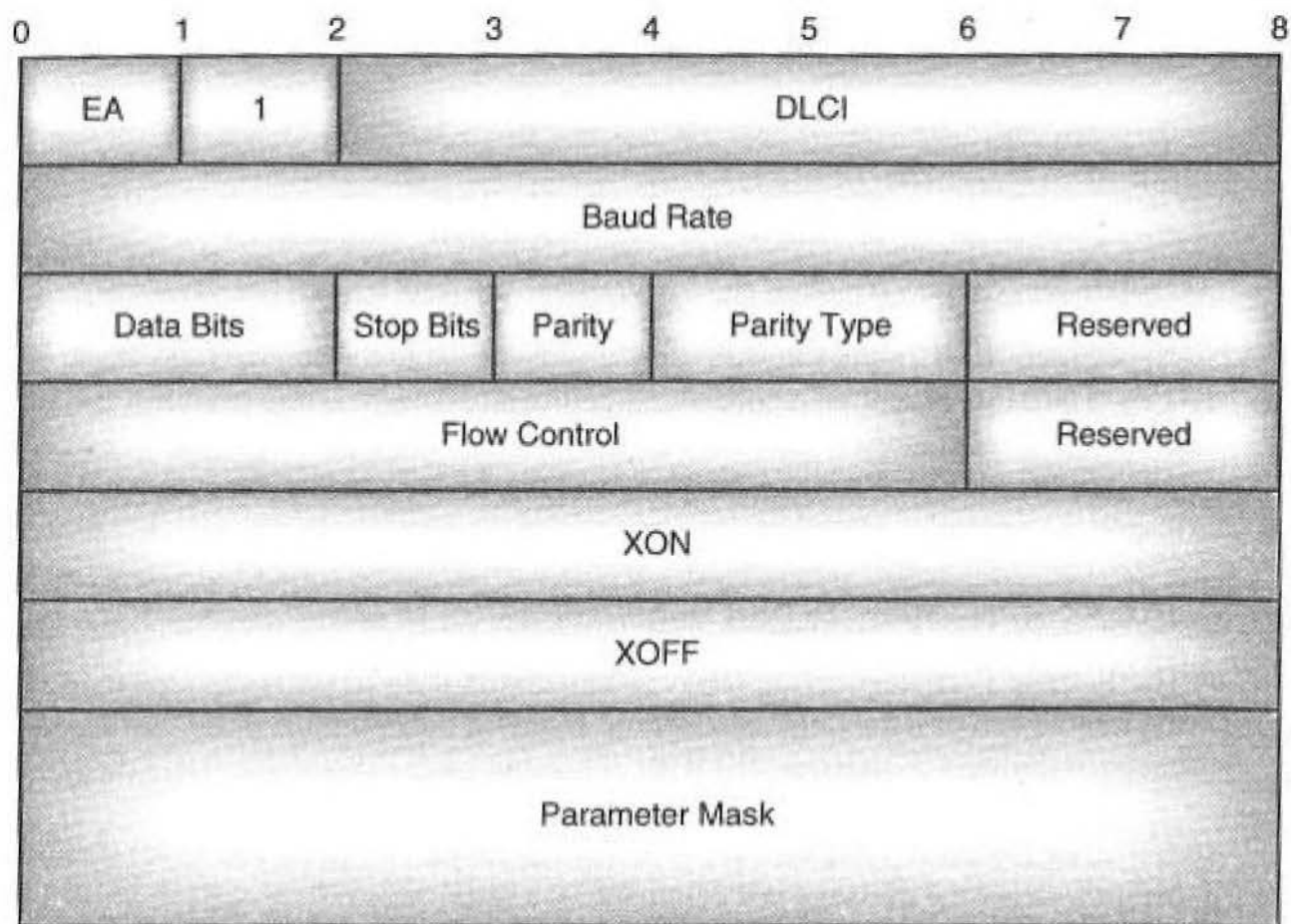


Figure 10-19 Value bytes in a RPN command.

the new value has been accepted, and the sender of the response is now using the new value.

The values of the various fields in an RLS command have the same meanings as in GSM 07.10.

10.6.6 RLS—Remote Line Status

A device sends a Remote Line Status (RLS) command when it needs to tell the other end of the link about an error.

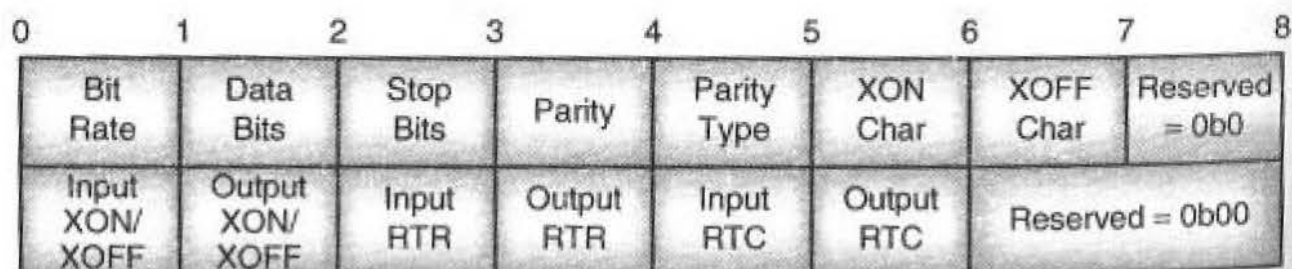


Figure 10-20 Bits in an RPN parameter mask.

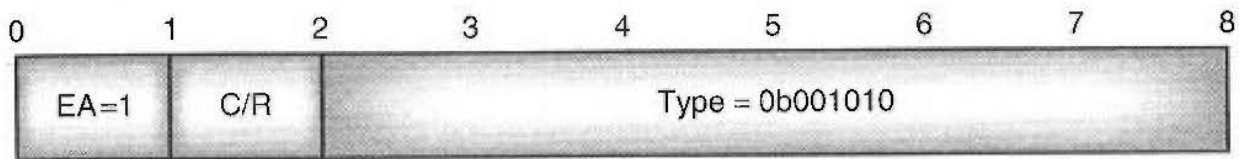


Figure 10-21 Type field for RLS.

The RLS command is identified by the type field shown in Figure 10-21. The EA bit is 1 because the type field occupies one byte. The C/R bit is used to indicate whether the message is a command or a response.

The length byte is set to 2, as there are two value bytes: the first value byte carries an EA bit, C/R bit, and the data link connection identifier common to all multiplexer command messages; the second value byte carries the error status in its first four bytes as shown in Figure 10-22.

The Line (L) status bits can signal three different errors as follows:

- 0b1100—Overrun error, a received character has overwritten a character which had not yet been read.
- 0b1010—Parity error, the parity was wrong on a received character.
- 0b1001—Framing error, a character did not end with a stop bit.

If the first line status bit is set to 0, then the RLS command is simply reporting that there are no errors on the line.

When an RLS command is received, a response is sent back with the line status copied from the command into the response.

RFCOMM implementations must recognise and respond to RLS commands, but how they deal with the line status information is up to the implementor to decide.

10.6.7 NSC—Non-Supported Command (Response Only)

The Non-Supported Command (NSC) Response is sent whenever a device receives a command it does not support.

The type field used to identify a message containing an NSC is shown in Figure 10-23. The EA bit is 1 because the type field occupies one byte. The C/R bit is used to indicate the message is a response. If the message comes from the device which initiated the connection by sending a SABM, then C/R = 0. If the message comes from the device which responded to the initial SABM, then C/R = 1.

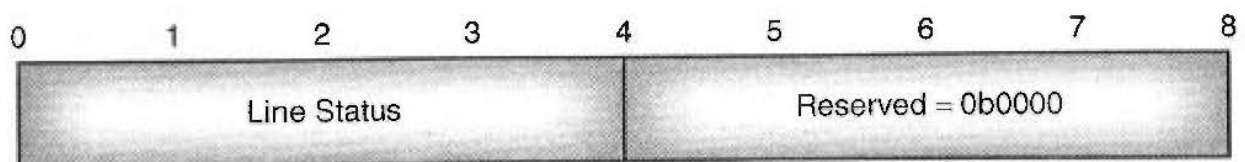


Figure 10-22 Second value byte of RLS message.

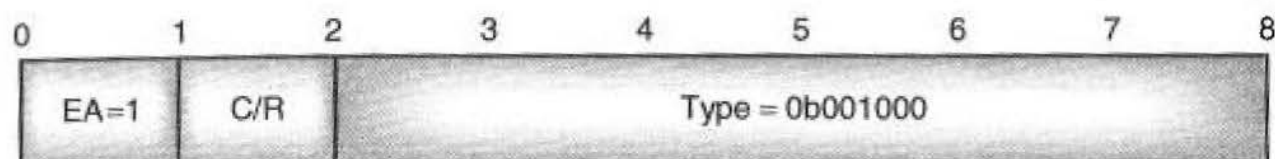


Figure 10-23 Type field for NSC response.

10.7 SERVICE RECORDS

Bluetooth devices offering services supported by RFCOMM must have an entry in their service discovery database which gives information on how to connect over RFCOMM.

RFCOMM server channel numbers are dynamic. That is to say, a service's channel number can change. Although a service's channel number doesn't change while the service is in use, it can be reallocated when the service is not in use.

The minimum information needed to connect to a service over RFCOMM is a service name (to identify the type of service) and a channel number on which to transfer data. Many services will have other additional parameters which are also needed to connect to the service. By querying SDP records, a device can find out all the information it needs to connect to a service via RFCOMM.

Table 10-2 shows a minimal service record which might be used to provide the information needed to connect to a service across RFCOMM. The ServiceClassIDList gives the name of the service. The ProtocolDescriptorList gives the supported protocols. Since RFCOMM rests on L2CAP, the L2CAP service must be present whenever RFCOMM is present. This service also has a text name which can be used to represent it on a user interface.

For more information, see Chapter 11, Table 11-1, which shows how RFCOMM information is presented for a headset application. The headset application also uses RFCOMM services to set up and control headset connections.

Table 10-2 Service Attributes Needed to Connect to an RFCOMM Service

Item	Type	Value	Attribute ID
ServiceRecordHandle	UInt32	Assigned by Server	0x0000
ServiceClassIDList			0x0001
ServiceClass0	UUID	<i>SERVICE NAME</i>	Service UUID
ProtocolDescriptorList			0x0004
Protocol0	UUID	L2CAP	0x0100
Protocol1	UUID	RFCOMM	0x0003
ProtocolSpecificParameter0	UInt8	<i>Server Channel #</i>	
ServiceName	String	<i>"TEXT NAME"</i>	0x0000 +Language Offset

10.8 SUMMARY

RFCOMM provides serial port emulation, which can be used to connect to legacy applications. It is also used for data transfer in several of the Bluetooth profiles.

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

To set up an RFCOMM connection, an L2CAP connection must first be set up. RFCOMM frames are sent in the payload field of L2CAP packets. Once the L2CAP connection is set up, RFCOMM control frames are sent back and forth to establish a signalling channel with a Data Link Connection Identifier (DLCI) set to 0. After this is set up, subsequent channels are established for transferring data. Up to 30 data channels can be set up, so RFCOMM can theoretically support 30 different services at once. (In practice, most Bluetooth devices will not have the resources to support 30 different services.)

RFCOMM is based on the GSM 07.10 standard with a few minor differences to allow for the differences between a Bluetooth connection and a GSM cellular phone connection.