**译者序**

为了解除日益增多的连接线缆的烦恼，爱立信的一位工程师在1994年发明了一种名为蓝牙的短距离无线通信技术，时至今日，蓝牙已成为业界公认的连接智能移动设备与周边其他人机交互设备（如键鼠、耳机等）的最主要无线通信手段之一。自蓝牙技术联盟于2001年发布蓝牙1.1版本技术规范起，一群诺基亚的工程师就开始畅想，在若干年后，如何能够让智能移动设备与周边的嵌入式设备的连接时间更长，特别是各种传感器与执行器设备，从而使得智能移动设备成为与周边10m范围内各种电子产品通信的中枢。经过10年的努力，蓝牙技术联盟终于在2011年发布了里程碑版本的蓝牙4.0技术规范，其中最具吸引力的就是低功耗蓝牙技术规范。

与经典蓝牙相比，低功耗蓝牙不仅极大地降低了芯片的功耗，还以其连接的灵活性、良好的抗干扰性、简单的应用开发等特性，很快得到了智能手机与个人电脑厂商的认可，以及众多外设厂商和应用开发者的青睐，特别是自2012年以来，围绕智能手机打造的各种应用外设(appcessory)呈现出爆发式增长的态势，例如智能家居和车载电子设备，它们还可借助智能手机将数据发送至互联网，在云端得到进一步的分析处理，从而为消费者带来更加个性化的服务。许多互联网企业也加人可穿戴设备这样一轮新的移动互联网创新浪潮之中，为消费者带来各种集成了多种传感器的智能手环、手表等面向运动与健康的应用外设，并引发了一种称为“量化自我”的生活方式。

本书的作者Robin Heydon是CSR公司负责蓝牙技术标准化的标准架构师，蓝牙技求规范的每一个版本都有他的贡献，同时他也是制定低功耗蓝牙技术标准的超低功耗(ULP)工作组的联合主席，自2007年开始推动低功耗蓝牙技术标准的制定工作，井因其出色的工作于2010年入选蓝牙技术联盟名人堂。

译者所在的研发团队自2009年开始接触最早一批的低功耗蓝牙的工程样片，并陆续开发了基于低功耗蓝牙的端到端IPv6通信演示系统，以及基于低功耗蓝牙的高精度室内定位系统，井参与了其技术标准化工作。译者自接触低功耗蓝牙技术以来，就一直很希望能有机会与更多的同行分享这一设计出色的面向移动物联网的无线技术。就在译者翻译此书的过程中，蓝牙技术联盟叉在2013年底发布了蓝牙4.1版，其中主要包含了面向物联网(IoT)的一些新技术特性，如允许建立面向连接的信道以及未来可以承载IPv6报文的传输。另外，蓝牙4.1也进一步优化了连接的建立与保持过程，如允许两部已经配对的设备基于定向广播延长广播的周期，从而实现设备更长的待机时间。

低功耗蓝牙技术硬其未来的演进版本将继续拓展蓝牙现有的生态系统，为移动互联网的应用及设备的创新创造出更多的机会，满足消费者从家里到手提包里，从汽车上到高尔夫球场上无处不在的无线连接需求。

陈灿峰 刘嘉

2014年3月10日

**前言**

有时候，一项难能可贵的新技术的出现能够改变世界，AM广播、电视和无线互联网便是这样的例子。作为一项可广泛植入各类产品、围绕微型电池设计、可持续工作数年的技术，低功耗蓝牙已然站在了下一次无线技术革命的最前沿。

本书详尽地解释了这项技术是如何产生、如何设计以厦如何工作的。作者Robin Heydon是低功耗蓝牙领域的顶尖专家，曾参与低功耗蓝牙的标准规范制定、互操作性测试以及培训工作。

本书适合从事低功耗蓝牙产品开发工作的各类读者，例如工程师、应用程序开发人员、设计师或者市场营销人员。

对工程师来说，这本书涵盖了系统工作的完整细节，从物理无线电到设备的发现、连接和提供接口。

对应用程序开发人员来说，这本书有助于其理解低功耗蓝牙对应用程序所施加的限制，此外本书还提供了关于设计目标和实现具体需求的详细介绍，

对设计师来说，本书包含的信息将让你了解关于设计低功耗蓝牙无线产品的特殊问题，比如产品需要如何工作，选择多大的电池才能实现你的想法等。

对其他想了解该技术的人来说，这本书提供了低功耗蓝牙的许多背景材料，包括为什么要设计该技术，试图实现的设计目标是什么，以及一些将会颠覆你对无线技术及其实现方式的看法的相关知识。

全书共分为四个部分：

第一部分为技术概述，包括用于指导低功耗蓝牙开发的基本概念，系统的体系结构（无线电、各种协议层、应用层），以及新技术带来的新使用模式。

第二部分详细介绍了控制器（无线电芯片）的工作原理。产品设计人员需要将该硅芯片植入自己的终端产品中。该部分涵盖了无线电、直接测试模式和链路层，此外还介绍了如何使用上层协议栈（或称主机）与控制器进行交互。 .

第三部分深人阐述了主机（软件栈）的工作原理，涉及各主要协议背后的概念和细节，这些协议主要用于公开设备的属性。该部分还介绍了安全模型，以殛两台设备之间如何进行连接、绑定或相互关联。

第四部分讨论了产品或应用程序开发人员需要考虑的所有设计因素。首先着眼于与中央设备有关的问题，接着是外围设备的问题，最后讨论了任何蓝牙产品上市前的最后一步，即测试和认证。

如果阅读本书后，读者还想了解更多关于低功耗蓝牙的相关知识，以下列出一些可供参考的资料，标准规范可以在蓝牙技术联盟网站www.bluetooth.org中找到。如果想查找有关低能耗蓝牙的开发信息，可以访问开发者网站developer.bluetooth.org，上面有关于特性(characteristic)的详细信息。本书作者也有一个网站www.3khanncis.com．在那里可以看到针对本书或者低功耗蓝牙的一些常见问题．

**致谢**

感谢以下各位的帮助和支持，使得本书得以顺利完成。Katherine Heydon多次从头到尾翻阅全文，并就内容提出了建设性的批评意见。Jennifer Bray第一个鼓励我撰写本书，井给予了我时间和空间来完成此项工作。感谢Addison-Wesley出版团队的所有成员，尤其是Bemard Goodwin、Elizabeth Ryan，Michelle Housley和Gary Adair．感谢文字编辑Bob Rusrelnl以厦其他所有促成本书的幕后人员。感谢Nick Hunn多次与我讨论如何才能更好地交流低功耗技术的想法。Zoe Hunn为本书设计了梦幻般的封面。Andy Galass不断询问本书的完成时间，并提供了很好的审查意见。Steve Wemham长期忍受我各种天马行空般的低功耗蓝牙的改进想法。英国航空公司几乎总是给我一个前排带有隔板的座位，让我得以在许多的长途航班中使用我的蓝牙键盘和鼠标。这本书可能是在30000英尺的平均高度撰写完成的。感谢蓝牙技术联盟社区，他们在全员会议、蓝牙测试活动（UnPlugFest）和各种工作组会议中提出了许多问题，正是这些问题让我明确了哪些概念是最难解释的，本书的基本框架和内容也由此受益。

**目录**

译者序

前言

第1章什么是低功耗蓝牙技术-2

1.1设备类型…………………-4

1 2设计目标…………………-5

1 3术语………………………-6

第2章基本概念

2 1纽扣电池……………………-8

2 2时间即能量…………………………．9

2.3昂贵的内存一………“………………10

2 4非对称设计……………………10

2 5为成功而设计……………………l1

2 6凡事皆有状态………”…………l2

2.7客户端服务器架构……………12

2 8模块化架构………………………I3

2 9十亿只是个小数目………………14

2 10无连接模型……………………“l4

2 11范式………………………………t 14

目 录

2.11.1客户端一服务器架构……l4

2.11.2面向服务的架构………”15

第3章低功耗蓝牙的体系结构……l9

3．I控制器……………………- 20

311物理层……”…………………．20

3.1.2直接测试模式……- 20

31 3链路层……………-21

3.1.4主机，控制器接口- 22

3 2主机………………………- 22

3.2.1逻辑链路控制和适配协议- 22

3 2 2安全管理罂协议…- 23

3.2.3属性协议…………- 23

3.2.4通用属性规范………- 24

3.2.5逦用访日规范……-25

3.3应用层…………………………．25

3.3.1特性……………- 26

3 3 2服务…………………- 26

3 33规范……………………-26

3 4协议栈划分

3.4.1单芯片解决方案……-27

3.4.2藏芯片解决方案……………28

3 4 3三芯片解嵌方案……-28

第4章新的使用模型………- 30

4.1存在检测……

4 2广播数据…………………-31

4 3无连接模式………………-31

4.4网关…………

第5章物理层”

- 36

5.1背景………………………- 36

5 2模拟调制…………………- 36

5 3数字调制…………………-38

5 4频段……………………- 39

5 5调制………

5 6射频信道…………………-41

5 7发射功率…………………-41

5 8容限………………………-42

5 9接收机灵敏度……………-42

5.10通信距离………………- 42

第6章直接测试模式……- 45

6.1背景………………………- 45

6 2收发机测试

6.2.1测试报文格式……- 46

6 2.2发射机测试………- 47

6 2 3接收机测试………- 47

6.3硬件接口…………………- 48

6.3.1 串口……--……………………48

6 3 2命令与事件………- 48

6 4使用HCI的直接测试模式- 50

第7章链路层…………………- 51

7 1链路层状态机……………- 51

7.1.1就绪态……………- 52

7.1.2广播鸯………………- 53

7 1 3相描态…………………- 53

7 1 4发起态……………- 54

7 1 5连接态……………- 55

7.1\_6多状态机”………………．．55

7 2报文……………………- 57

7.2.1广播与数据报文……- 57

7.2.2白化……………………- 57

7 3报文结构…………………- 59

7.3.1比特序与字节………- 60

7 3 2前导……………………- 60

7 3 3接八地址…………- 60

7 3 4报头……………………- 61

7 3 5长度……”………………6l

7 3 6净荷…………………………．63

7 3 7循环冗余校验……- 63

7 4信道……………………- 63

7.4.1跳频………………………66

7.4.2自适应跳频…………- 66

7 5设备发现…………………- 68

7.5.1通用广播……………- 69

7 5 2定向广播……………- 69

7 5 3不可连接广播……- 70

7 5 4可发现广播………………70

7.6广播………………………………70

7 7建立连接…………………………．71

7.7.1接八地址………………72

7.7.2 CRC初始化………………72

7 7 3发送窗口…………………72

7.7.4连接事件…………………73

7.7.5信道图”…………………．74

7.7.6睡眠时钟精度……- 74

7 8发送数据…………………………74

7 8 1数据报头………………………75

7.8.2逻辑链路标识符……………75

7.8.3序列号…………………………76

7.8.4确认………………………．76

7.8.5更多数据…………………77

7.8.6使用序列号和更多数据的

例子……………………，77

7.9加密………………………………79

7.9.1 AES……………………………-.79

7.9.2加密净荷数据………- 80

7.9.3消，电完整性棱验…- SI

7 10管理连接………………………．82

7.10.1连接参数更新……- 83

7:10.2自适应跳频……- 84

710 3启动加密……………- 85

7.10.4重启加密……………- 87

7.10.5版本交换…………- 87

7.10.6功能变换………- 89

7 10 7终止连接………………．90

7.11鲁棒性…………………- 90

7.11.1 自适应跳叛…………- 90

711 2强CRC～……………………92

7.12为低功耗优化…………- 93

7.12.1短报丈………………- 93

7 12 2高比特率……………- 94

7 12 3低开销………………- 95

7 12 4确认机制……………- 96

7.12.5单信道连接事件…- 96

7 12 6亚速率连接事件…- 97

712 7高线加密…………。……~，．98

第8章主机，控制器接口…- 99

8.1介绍………………………- 99

8 2物理接口…………………- 99

8.2.1 UART…一………一…………99

8 2 2 3线UART-．100

8.2.3 USB……………………………IOI

8.2.4 SDIO…n…………一………102

8 3逻辑接口………………………．102

8.3.1 HCI信道……………”-102

8 3 2命令数据包……………．102

8.3，3事件数据包………………，．103

8.3.4数据包…………………．104

8 3 5命令流控………………105

8.3.6数据流拄…………………．106

84控制器的配置…………………．106

8，4.1重置控制器为已知状态…106

8 4 2读取设备地址…………．107

8.4.3设置事件掩码……………107

8 4 4读取缓冲区大小…………．-108

8.4.5诶取控制嚣支持的功能…109

8.4.6读取控制嚣支持的状态…109

8 4 7随机数………………………l10

8 4 8加密数据……………………l10

8 4 9设置随机地址……………．．111

8.4.10白名单……………………112

8 5广播和观察……………………．112

8.5.1广播…………………………．112

8 5 2被动扫描…………………．114

8 5 3主动扫描………“…………l15

8 6发起连接………………………．．116

8.6.1与白名单设备发起连接…116

8 6 2与单一设备发起连接…．118

8 6 3取消连接请求……………．118

8 7连接管理………………………，．119

8.7.1更新连接……………………119

8.7.2更新信道映射田…………．120

8 7 3交换功能列表……………．121

8.7.4交换版本信息……………121

8.7.5加密连接……………………122

8 7 6重启加密．．…………………123

8.7.7终止连接………………125

第9章逻辑链路控制和适配协议…l28

9l背景……………………………l28

9 2 L2CAP信道…．．………………．130

9.3 L2CAP数据包结构……………130

9.4低功耗信令信道………………．131

9 41奇夸拒绝

9.4.2连接参数更新请求和

响应…………………………．．132

第10章属性…………………………l35

10.1背景……………………………I35

10.1.1精简协议……………“l36

10 1 2无所不在的数据………l36

10 1 3教据与状态………………137

10.1.4几种常见的状态……“…l37

101 5状态机…………………l38

10.1.6服务和规范………………．139

10 2属性……………………………142

10.2.1属性概速………………143

10 2 2属性句柄…………………143

10.2.3属性类型…………………l43

10 2 4属性值…………一…………I44

10.2.5数据库、服务器和客

户端……………………145

10 2 6属性许可“………~……．145

10.2.7接入属性u………………．147

10.2.8原子操作和事务…………148

10.3分组n…………………………l49

10 4服务…………………………l50

10 41扩晨服务………………l51

10 4 2其他服务的重用………153

10 4 3结合服务……”………．153

10 44首要还是次要…………lS4

10 4 5即插即用的客户端应用…l56

10 4 6服务声明……………“l57

10.4.7包含服务……………．158

10 5特性……………………………．159

10.5.1特性声明………………I59

10 5 2特性数值………………l6l

10.5.3描述符……………”．…．161

10 6属性饰议………………………．164

10.6.1协议消息………………I65

10 6 2交换MTU请求…………l65

10 6 3查找信息请求…………I66

10 6 4接类型值查找请求……167

10 6 5按类型读取请求………．168

10 6 6读取请求………………I68

10 6 7大对泉读取请求………168

10.6.8多重读取请求…………l69

10.6.9按组类型读取请求……I69

10 610写入请求……………．169

10 611写入命令……………-169

10 6 12签名写入命令………-169

10 6 13准备写入请求与执行

写入请求………………I70

10 6 14句柄值通知……………l71

10 6 15句柄值指示……………171

10 616错误响应………………17I

10 7通用属性规范…………………l73

10 71发现规程……………．174

10 7 2发现服务………………174

10 7 3特性发现………………175

10 7 4客户端发起规程………-176

10.7.5服务器发起规程………178

10.7.6属性协议数据单元(ATT

PDU)到GATT规程的

映射--……………………．．178

第11章安全…………………………180

Il.I安全概念………………………180

Il.l.I认证……………………l80

11 1 2授权……………………．181

11.1.3完整性…………一………+181

11.1.4机密性…………………182

11.1.5随私…………………………l82

11.1.6加密引擎………………182

11 1 7共享机密………………．182

11.2配对和绑定……………………l85

11.2.1配对……………………l85

11.2.2配对信息交换…………l85

11.2.3认证……………………186

11.2.4密钥分配…………………．187

11.2.5绑定……………………188

11.3数据签名………………………188

第12章通用访问规范……………189

12.1背景……………………………1 89

12 1 1初次发现………………．．190

12.1.2建立初始连接…………．191

121 3服务特性………………19l

121 4长期\*系………”…一…．192

12 1 5重连……………………192

12.1.6私有地址………………．193

12.2 GAP角色……………………………193

12 3模式和规程…………………．194

12 3 1广播模式和观察规程…l95

12 3 2可发现性…………………．195

12 3 3可连接性…………………197

12 3 4绑定……………………l99

12.4安全模式………………- 200

12 5广播数据………………- 201

12.5.1标识………～…- 202

12 5 2服务……………- 202

12 5 3本地名称………- 203

12.5.4发射功率等级…- 203

12.5.5从设备连接间隔范围- 203

12 5 6服务请求………- 203

12 5 7服务数据………- 203

12 5 8制造商指定数据………203

12.6 GAP服务………………- 204

12.6.1设备名特性……- 204

12 6 2外观特性………- 204

12.6.3外围设备隐私标识- 204

12 6 4重连地址………………．205

12 6 5外围设备首选连接

参数……………- 205

第13章中央设备…………- 208

13.1背景…”……………………~208

13 2发现设备………………- 208

13.3连接设备……- 209

13 4这个设备能做什么……- 210

13 5通用客户端……………- 211

13 6与服务交互

13 6 1可读特性………- 212

13.6.2控制点………………- 212

13 h.3状鑫机……………- 213

13.6.4通知和指示……- 214

13 7绑定……………………- 214

13.8变更服务…………………- 215

13 9实现规范……¨，………- 216

13.9.1定义规范…………- 216

13 .9.2查找服务………- 217

13 9 3查找特性………- 217

13 9 4使用特性………- 217

13.9.5规范安全………-217

第14章外围设备…………- 219

14.1背景……………………………．219

14 2仅广播…………………- 219

14.3可发现…………………- 220

14 4可违接………………- 220

14 5公开服务………………- 221

14.6特性……………………- 221

14 7安全事项………………………．222

14.8为低功耗优化………一………一222

14 8 1可发现广播………………224

14 8 2绑定……………- 224

14.8.3可连接广播……- 224

14 8 4定向广播……- 225

14 8 5已连接…………- 225

14 8 6保持连接还是断开……226

14 9优化属性………………………．227

第15章测试和质量鉴定…- 229

15 1启动项目………………………229

15 2选择功能………………- 231

15.3 -致性检查……………- 232

15.4生成测试计划…………- 232

15 5建立舍规文件夹………………2弛

15.6鉴定测试………………- 233

15 7鉴定设计………………- 234

15.8合规声明………………………．234

15 9产品名录……………………- 234

15.10结合组件……………- 235

**第一部分 综 述**

**第1章 介绍低功耗蓝牙技术及其设计目标。**

**第2章 讨论低功耗蓝牙体系结构的设计基础。**

**第3章 介绍低功耗的主要系统架构，包括控制器、主机及其之上的应用层。**

**第4章 描述低功耗技术使能新的使用模型**。

**第1章 什么是低功耗蓝牙技术**

如果我能看得更远的话，那是因为我站在巨人的肩膀上。

——艾萨克·牛顿(isaac Newton)

低功耗蓝牙是一种全新的技术，是当前可以用来设计和使用的功耗最低的无线技术。作为经典蓝牙的扩展，低功耗蓝牙描用了蓝牙商标，并且借鉴了很多父辈的技术，然而，由于针对的设计目标和市场领域均与经典蓝牙有所不同，低功耗蓝牙应被视为一种不同的技术。

经典蓝牙的设计目的在于统一全球各地的计算和通信设备，让手机与笔记本电脑互相连接。不过事实证明，蓝牙最为广泛的应用还是音频传输，比如将音频从手机传到蓝牙耳机。随着技术的成熟．越来越多的蓝牙应用进入人们的视线，包括立体声音频流、汽车从手机下载电子书、无线打印和文件传输。由于每一个新的应用都要求更多带宽，因此，随着时间的推移，越来越快的无线电技术不断地加入蓝牙系统中。1.0版蓝牙为基本码率(Basic Rate．BR)，最大物理层数据速率为1 Mbps（兆比特每秒）；2.0版本为增强码率(Enhanced Data Rate．EDR)，其物理层数据传输率增至3Mbps；3.0版本引入Aitemative MAC，PHY（AMP，交替射频技术）。利用IEEE 802.11实现了高达数百Mbps的物理层数据速率。

低功耗蓝牙选择了完全不同的方向：并非只是增加可达的数据传输速率，而是从尽可能降低功耗方面进行优化。这意味着，也许你无法获得很高的传输速率，但是可以将连接保持数小时或数天的时间。这一选择非常有趣，显然，大部分有线和无线通信技术还在马不停蹄地提升速率，如表1-1所示。



表 1-1

对于那些由纽扣电池供电的设备，经典蓝牙并不能真正达到它们的低功耗要求。理解了这一点，就不难明白选择新方向的原因。然而，在充分考虑低功耗的相关要求时，还有一点必须考虑到，即低功耗蓝牙应被设计成满足极大规模部署的要求，以便用于迄今尚未装备无线技术的装置。要实现极大的规模，就必须要有极低的成本。就好像射频识别（Radio Frequency Identication．RFID）通过一个价格较高的扫描装置获得能量，然而其标签本身的成本极低，从而获得了大量的部署。

因此，从低成本的需求方面审视低功耗蓝牙的系统设计尤为重要。实现低成本的设计有三个关键因素：

**1 ISM频段**

无论从设计的角度还是从使用的角度出发，2.4GHz ISM频段对无线技术而言都是个糟糕的频段。该频段无线电传播特性差，能量极容易被各类物体吸收，尤其是水，而人体主要是由水构成的。尽管有许多显著的不利因素，但不可否认，该无线电频谱的优势是在全世界可以免许可、自由地使用。当然，“免交租金”的标志意味着其他技术一样能够使用该频段，包括绝大部分的Wi-Fi信号。不过．免许可并非等同于毫无约束，使用该频段仍然要遵守相当多的规则，主要是限制设备的输出能量和范围。当然，与许可频谱的高昂费用相比，这些限制就显得微不足道了。因此，选择使用ISM频段能够降低成本。

**2．IP许可**

当Wibree（超低功耗蓝牙）技术发展成熟，考虑将其并人已有柏无线标准工作组时，诺基亚原本有多种选择方案。比如加人Wi-Fi联盟，该联盟也在2.4GHz ISM频段制定标准化技术。然而，鉴于蓝牙组织拥有较高的声誉和优厚的专利许可政策，他们最终选择了蓝牙技术联盟（Bluetooth Special Interest Group，BT SIG）。与其他采取FRAND（Fair Reasonable and Non-Discriminatory）政策的兴趣小组或联盟相比，蓝牙技术联盟的政策使得蓝牙设备的专利许可成本大为降低。而许可成本的降低使得每件设备的成本也显著降低。

**3．低功耗**

设计一款低成本设备的最好方法就是减少制作这个设备所需的原料，比如电池。电池越大，电池盒就越大，这样叉会增加成本。替换一节电池的花费，不仅指消费者需要购买新的电池，而且替换本身也包含了因设备暂时无法使用带来的机会成本。如果设备由第三方维护．比如作为家庭警备管理系统的一部分，换电池还需额外的劳动力成本。因此，设计有关低功耗的技术也是在降低各种成本。这里不妨做个脑力实验，如果只花一毛钱就能买一个兆瓦特级的电池，那事情会变得多么不同？

很多设备能容纳更大的电池，例如键盘或者鼠标内部很容易装下几节AA电池。然而生产商们却倾向于使用AAA电池，并不是因为它们更小，而是因为它们的原料成本更低，降低了设备的总成本。

因此，低功耗的基础设计就是以纽扣电池——这种最小、最便宜并且最容易购买的电池类型作为能量来源。这意味着我们无法令低功耗蓝牙实现很高的数据传输速率，或是将其用于大量数据的传输或者数据流传输。这一点或许是经典蓝牙与低功耗蓝牙的最大区别。下一节将就恢同题进行详细讨论。

**1.1设备类型**

低功耗蓝牙技术可以构建两种类型的设备：双模设备和单模设备。双模设备既支持经典蓝牙又支持低功耗蓝牙。单模设备只支持低功耗蓝牙。当然，还有第三种类型---仅支持经典蓝牙的设备。

由于双模设备支持经典蓝牙，所以能与现有的散以亿计的蓝牙设备通信。双模设备是一类新的设备，要求为主机和控制器分别提供新的软件和硬件（包括固件）。因此，现有的经典蓝牙控制器或主机无法通过升级实现低功耗蓝牙。不过，大部分的双模控制器只是简单地替换了经典蓝牙的控制器的某些部分，这使得手机、电脑和其他设备的设计人员能够较快地用双模控制器替换现有的经典蓝开控制器。

仅支持低功耗蓝牙的单模设备不支持经典蓝牙，无法与现有的蓝牙设备通信，但可以与其他单模设备或者双模设备通信。出于降低功耗的目的，这些新型的单模式设备进行了大量优化，可用于通过纽扣电池供电的元器件。由于单模设备不支持头戴式耳机、立体声音乐或较高的文件传输速率，无法将其用于当今经典蓝牙使用的大部分领域。

表1-2展示了彼此间能相互通信的蓝牙设备类型以及蓝牙设备连接之后使用的无线电技术。单模设备能够与其他的低功耗单模设备通信，也能与支持低功耗的双模设备通信。双模设备能够与其他双模设备或使用BR/EDR的经典蓝牙设备通信。单模设备无法与经典蓝牙设备通信。

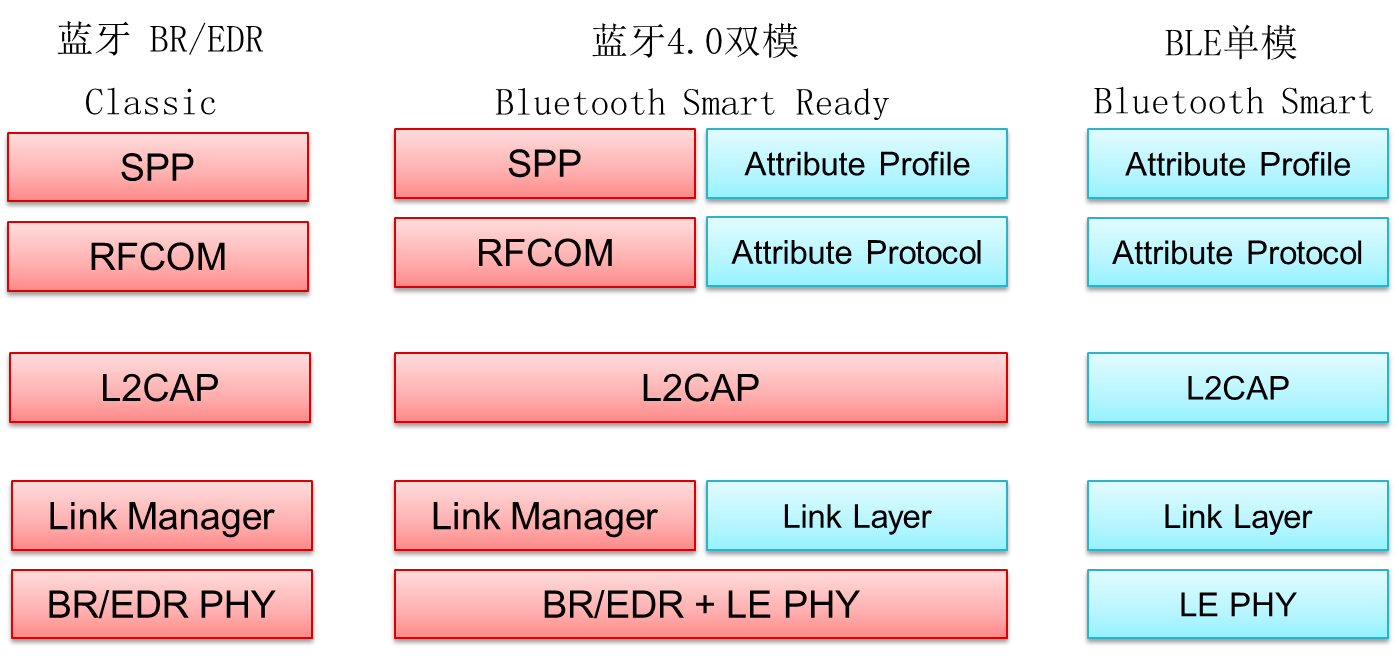


图 1-1

**1.2设计目标**

评论一种技术，一个首先浮现的问题是设计者是如何优化该技术的。大多数技术通常具有一定的适用性，也会有一两项显著的优势。而了解这些优势对于加深对该技术的理解将大有稗益。

对于低功耗蓝牙而言，答案非常简单。它的设计目标就是实现最低的能耗。蓝牙SIG的一个独特之处在于该组织制定并控制从物理层到应用程序的所有层次。SIG在一个合作、开放、由商业驱动的标准化模式下运行，在过去的十多年间不断优化无线规范的制定流程,使得蓝牙规范在发布之后不但可以立即使用，而且具有互通性、鲁棒性以及极高的质量。 在设计之初，低功耗蓝牙的目标在于尽可能创造一种最低功率的、短距离的无线技术。为了实现过一目标，低功耗蓝牙技术对体系结构的每一层都进行了优化，以降低执行任务所需的能耗。例如，与经典蓝牙相比，低功耗蓝牙对于物理层的射频指标要求有所放宽，意味着发送或接收数据时可以使用更低的功率。同样，链路层为了加快重连速度、提升数据广播效率也进行了优化，而并不需要一直保持连接。此外，低功耗蓝牙对主机内各协议也做了优化，主要是为了减少从连接建立到完成数据发送所需的时间。全部这些优化只有当系统中的各部分由同一团队在同一时间完成才有可能实现。

蓝牙技术最初的设计目标井未被人们遗忘，主要包括以下几个方面：

* 全球操作
* 低成本
* 鲁棒性
* 短距离
* 低功耗

全球操作，需要一个在世界范围内都能使用的无线频段。2.45GHz频段是目前唯一一个可实现低成本、大批量制造的频段。该频段之所以可用，是因为它对天文学家、手机运营商或其他商家来说并褴有什么吸引力。不幸的是，正如其他的“免费午餐”那样，当人人都想分一杯羹时，拥挤就在所难免。当然还有其他一些可用的无线频段，比如60GHz ISM频段，但从低成本的角度来看并不适用；800/900MHz区间的频殷也可使用，但由于不同的国家或地区对具体使用将率的定义和使用规定各有不同，同样缺乏可操作性。

低成本的设计目标程有意思，因为它意味着系统应该尽可能做到简洁、高教。举个例子，尽管可以在低功耗蓝牙加人分散网(scatter net)或完整的网状网(mesh)支持，但这要求有更多的内存和处理能量来维护网络，从而增加了成本。所以，低功耗蓝牙系统在进行低成本优化时，并没有涉及在学术界广泛研究的网络拓扑学。

低功耗蓝牙使用的2.45GHz频段已经非常拥挤，仅标准的技术就包括：经典蓝牙、低功耗蓝牙、IEEE 802.11、IEEE 802.l1b、IEEE 802.l1g、IEEE 802.l1n以及IEEE 802.15.4。另外，许多私有的无线电也使用这个频段，包括X10视频中继器、无线警报、键盘和鼠标等。许多其他设备也会在该频段发射噪声，例如街灯和微波炉。

因此，除非沿用经典蓝牙的自适应跳频技术，否则设计一个能在各种干扰下全天候工作的无线设备将是不可能完成的任务。自适应跳频不仅有助于迅速确定干扰源，从而在未来灵活地规避，而且能从无线电渡干扰引发的丢包中迅速恢复。任何无线技术只有具备这样的鲁棒性才能从拥挤的无线频段里脱颍而出。

鲁棒性还包括检测和纠正比特误码的能力，这类错误通常由背景噪声而引起。少数短距离无线标准使用长循环冗余校验(CRC)，但大部分标准做了折中，使用较短的CRC校码。优良的设计应在校检能力和发送校验信息所需的时间之间取得平衡。

短距离实际上存在一些问题。比如，如果想得到一个低功耗的系统，就必须尽可能降低传输功率以减少用于信号传输的能量。类似地，我们必须在接收方保持较高的灵敏度，才能减少从一大片噪声中采集设备的无线信号时所消耗的能量。在这里，“短距离”真正的含义在于低功耗蓝牙网络并不是一个蜂窝基站系统，而是一个个人局域网。

蓝牙的最初设计目标在设计低功耗时并没有太多改变，只是后者将能量的消耗目标降低了一到两个数量级。经典蓝牙的设计目标是能够实现长达几天的待机以及实现若干小时的与头戴耳机通话，而低功耗蓝牙的设计目标是让测量气温或涮量你的步行距离的传感器能够工作几年时间。

**1.3术语**

和许多高技术领域一样，在低功耗蓝牙方丽工作的人们用他们自己的语盲来描述一些技术特征和规范。本节列举了那些具有特殊意义的术语，井给出了解释。

**自适应跳频（Adaptive Frequency Hopping，AFH）**一种使用某个频率子集的技术，使设备可以避免其他非自适应技术使用该频率（比如Wi-Fi接入点）。

**体系结构（archintecture）**低功耗蓝牙的设计方案。

**频段（band）**参看**无线频段（radio band）**。

**跳频（frequency hopping）**两个设备之间使用多个频率通信。某一时刻只用一个频率，各频率按照确定的顺序依次使用。

**层（layer）**系统中实现一个具体功能的部分，例如物理层负责无线电操作。系统中每一层都是根据上层或下层抽象而来的。链路层并不需要知道有关无线电功能的所有细节；逻辑链路控制层和适配层(L2CAP)不需要知道关于链路层如何工作的所有细节。这一抽象概念对管理复杂系统而盲至关重要。

**主设备（master）**微微网中协调与其他设备的操作的一台复杂设备。

**微微网（piconet）**单词pico和network两个单词的缩写。pico星SI的前缀之一，表示。这个词源自意大利语piccolo，意思是微小的。因此，piconet是一个非常微小的网络。一个微微网包括唯一的一个主设备以及一个或多个从设备，主设备负责协调与本微微网中的所有其他从设备的操作。

**无线频段（radio band**）无线电波通过频率或波长进行划分。不同的无线电波具有不同的规则和使用方法。当某一范围内的无线电频率以同一规则组合起来时，该频率的集合就被称为一个无线频段。

**从设备（slave）**与主设备一起工作的简单设备。这些设备通常是用途单一的设备。

**Wi-Fi** 一种为高传输率而设计的补充无线技术，用于计算机等复杂设备的互联网接入。

**第2章 基本概念**

在协议的设计上，只有速到“添一分则多减一分则少”的境界才能算得上完蔓。

——IETF RFC 1925．第12条

要理解低功耗蓝牙，首先应了解如何在短距离无线系统中实现低功耗。通常来讲，一些最基本的设计决策都是围绕典型的使用实例来实现低功耗的。

低功耗蓝牙并非要优化经典蓝牙，相反，它针对的是那些以前没有使用开放无线标准的新兴的细分市场。这些细分市场要求设备间歇性地发送数据，间隔从几秒到几天不等，每次的数据发送包含若干个字节。它们通常是用来执行作业的监测或控制类应用，比如智能家居采暖应用中检测窗户打开或关闭，根据浮动电价开启和关闭电器，切换到另一个电视频道等。

**2.1纽扣电池**

纽扣电弛是低功耗蓝牙的主要设计目标。这类电池（见图2-1）在使用时有非常严格的限制。图中的电池型号为CR2032．标签上的“CR”表示这是一个3V锂二氧化锰电池，“20”表示该电池直径为20mm．“32”表示高为3.2mm。除图中所示的型号外，其他的一些电池尺寸也很常见。



图2\_1纽扣电池

对于这类小型电池而言，各品牌的电池所能存储的最大能量非常接近。一个典型的CR2032的额定容量在3V时为230mAh，如果换一个场景可能更好理解——这些能量仅够一个人维持生命约20s。所以，当你读过这一段的时候，你使用的能量已经超过了一个典型的CR2032提供的所有能量，而这些能量足够低功耗蓝牙设备工作几年时间。

虽然230mAh看来微不足道，但即使这样，设备也无法完全获得它全部的能量。首先，可用能量取次于电池的温度——温度越低，电池可提供的能量也越少。与室温下相比纽扣电池在O℃时仅能提供80%的能量。

其次，如果持续地使用电池，可用的总能量也会显著降低。通常情况下，大多数纽扣电池存在一个峰值电流，一般为15mA．在使用时不应该超过该电流，否则易损坏电池。如果长时间使用该电流供电，总的可用能量将会减少。因此，任何成功的无线电设计都需要考虑这些因素，使电池在大电流或长时间供电后能有一段时期进行恢复。

最后，电池本身具有的内部漏电特性也必须加以考虑。即使设备未从电池取电，电池电量仍会逐渐减少。因此，如果是偶尔使用电池，漏电电流将成为总能量消耗的重要一环。

**2.2时间即能量**

时间即能量是贯穿低功耗蓝牙设计的另一个基本概念。一旦无线电开始工作它便需使用能量，哪怕只是检查是否需要发送或者接收。因此，减少必要操作的执行时间变得十分重要。

许多关键而重复的操作必须通过一定的措施实现优化，包括鲁棒地发现设备、连接设备和发送数据。减少这些活动所需的时间，能耗得以减少，从而延长了电池寿命。

鲁棒的设备发现要求至少有两个设备：其中之一用于寻找其他设备，其他一个或多个设备则处于可发现状态。在低功耗蓝牙中，一个设备如果想被发现就必须每隔几秒发送三次短消息；而如果它想发现打算与之通信的设备，在广播短消息之后该设备应立即进行侦听。寻找其他设备时，设备将打开它的接收器井侦听其他设备的传输。

三次传输分别利用了三个不同的频率进行，以提高鲁棒性。选择数字三其实是为了在鲁棒性和低功耗之间求得平衡：如果频率的数量只有一个，那么像很多其他技术那样只要该频率被阻塞，整个系统将无法工作。如果频率的数量太多，例如为16个，诙设备将花费大量的时间进行传输，将不再是“低功耗”设备。

在选择哪些设备执行发送、哪些设备执行接收时也要经过慎重的考虑。要搜寻一个正在发送的设备可能需要花费很长一段时间，并因此耗费大量的能量，因此这类操作应当交给那些能量储备更多的设备或正好选择低功耗设备。在低功耗蓝牙中，可发现设备执行数据的发送，而寻找其他设备的设备执行接收。

数据报文的长度很短，主要是出于下列三个原因：首先，通过使用高效率编码，短分组可以使用更少的能量更快地发送相同数量的数据。其次，限制设备只可使用短报文，控制器消除了在一欢数据传输中需要不断校准无线电的需求。这是由于无线电模块在传送或接收时不断发热，将改变硅芯片的特性，井因此改变传输频率。如果报文控制在较短的长度，则芯片没有足够的时间发热，因此传输频率的偏移可以忽略不计。此外，对于短报文的要求也略微降低了芯片的峰值功率。最后，短期、间歇性而非长期、连续的取电令设备可以从一个纽扣电池获得更多的能量。可见，较之长报文的解决方案，使用短报文且彼此间保留足够空间让电池得以恢复的方案更为适合。

**2.3昂贵的内存**

众所周知，一台电脑的内存越多，通常而言价格就越贵。然而，计算机内每一个比特内存不仅费钱，还费能量。内存通常需要动态刷新——每隔一小段时间芯片中的内容便要刷新一次。这种动态刷新需要能量，而设备装备的内存越多也就需要更多的能量来为其供电。因此，整个低功耗蓝牙设计在每一层都考虑了降低内存的数量。

例如，在链路层保持较短分组有助于减少无线电发送和接收数据包时对内存的要求。例如，属性协议层不需要处理任何长度大于23字节的数据报文，在状态转换时也不要求存储任何状态信息。所有这些都降低了对内存的需求。

另一个与内存相关的负担在于，当一台设备需要做许多事情时，需要激活大量的协议。例如，想象一副耳机同时支持免提（hands-free）、远程控制(remote control)和电池状态报告（batteIy status reporting）功能。如果以上每个功能都需要一个单独的协议，这些协议对于存储器的需求将被加在一起。而低功耗蓝牙仅有一个协议——属性协议ATT（Attribute Protocol）来支持名称发现、服务发现、读取和写入信息，以实现不同的应用。仅用一个协议显然要比使用多个协议的开销少得多。

**2.4非对称设计**

在理解低功耗蓝牙的体系结构后，你会发现一个设计概念十分明显，即在所有层都采用了非对称设计。为了让能源更少的设备负担更少的事情，这种非对称的设计显得非常重要。

物理层通常有两类无线电：发射器和接收器。一个设备可以既有发射器又有接收器，但也可能只有发射器或只有接收器。假如一个设备只有发射器，而另一个设备只有接收器，则二者构成的网络称为非对称网络。

这种不对称的设计源于一个基本的假设，即所有其他设备必须为资源最有限的设备进行优化。

在链路层Link Layer，设备被分为广播者advertisers、扫描者scanners、从设备slaves和主设备masters。广播者是传输数据包的设备，扫描者是接收广播者的数据包的设备。从设备和主设备相连，但即便在这里，不对称性也是显而易见的。从设备无法发起任何复杂的操作，而主设备则必须负责微微网的定时，自适应跳频集合设置、加密以及一些其他的复杂操作。从设备只需完成要求的操作，不必进行复杂的处理。这样一来，从设备可以非常简单，实现了低成本、低内存和尽可能低的功耗。

在属性协议层Attribute Protocol Layer存在两类设备，称为客户端client和服务器server。服务器保存数据，客户端向其发送请求以获得该数据。服务器好比链路层的从设备，只做被要求做的事情。客户端的工作相对辛苦．比如获得服务器上的数据类型以及理解如何使用该数据。

即便是低功耗的安全架构也是非对称的。安全架构基于这样一种密钥分配方案：由从设备提供密钥，将其交给主设备并由后者保存。主设备因而担负起保存链路绑定信息的重任，从设备则无需记忆任何相关信息。这意味着在从设备上实现安全特性将非常简单．但主设备上实现起来要麻烦一些。

由上述介绍可见，那些资源最为受限的设备都将成为广播者、从设备和服务器。这类设备拥有最少的可用内存和处理负担，得益于非对称的设计实现了超低功耗的目标。

其他类型的设备—一扫描者、主设备和客户端拥有相对较多的可用资源。这些设备通常装备了较大的电池，具有丰富的用户界面，甚至可能由市电供电。把负担从从设备移到主设备，从广播者移到扫描者，从服务器移到客户端的选择是正确的，这一选择降低了资源受限的设备的功耗，将成本转嫁到具有丰富资源的设备上。

**2.5为成功而设计**

许多无线标准出师不利，原因在于一些无线电设计一旦受到欢迎，用户敷量不断增加，往往导致网络拥塞不堪甚至无法工作。如果说有一件事蓝牙做得很好，那便是它可以在非常拥挤的环境中工作。蓝牙技术联盟（SIG）每年组织三次称为UnPlugFest的测试活动，许多来自彼此竞争的公司的工程师们聚到一起，测试他们即将投放市场的产品。这些测试揭示了这样一个事实，即使酒店大厅同时出现了成百上千的无线设备，蓝牙依然能够正常工作。无疑，低功耗蓝牙继承了这一点。

为成功而设计意味着无论是在拥挤的通勤火车或公共汽车上，还是进入热闹的体育场或演唱会的现场，每个人都应该能够操作一些低功耗设备。这意味着在一台设备周围的几米内可能有成千上万的设备，但设备发现和连接仍然能够如期地运作。这也意味着，对一给定设备能够同时连接的设备的数量不应有所限制。如果一台设备想要和另一台设备对话，它就一定能够做到，无需担心类似经典蓝牙“同一时间只允许七个从设备进行连接”的限制。

除了设备的密度，另一个在设计控制器时必须考虑的方面是安全系统。任何流行的无线电系统总是会招致一些人对其进托攻击，当涉及个人账号和财产安全时，这一点就显得更为重要。因此，必须使用最先进的安全和加密引擎。

除安全之外，如果一个人想随身携带多神设备，鉴于它们的有限资源必须间歇地进行广播，那么隐私问题也必须加以解决。隐私是低功耗蓝牙的一个主要设计目标。每个连接使用不同的签名，并与连接双方的身份信息无关，这样一来，攻击者仅凭侦听传输的数据包将无法猜测谁正走在大街上。此外，广播时允许使用一类可解析的私有地址，只有那些有身份识别密钥的设备能够解析该地址，而恶意设备将无法解析或跟踪该地址。

另一个考虑的因素是，由于无线电无处不在，即使是单个比特的错误也可能导致严重的后果。打个比方，如果由你控制一个排污阀，以保护漂亮的城市公园不被污水淹没，你恐怕不会希望已经关闭的阀门由于某个误比特而被错误地打开。为防止这种情况出现，要求数据包使用强循环冗余校验（CRC）以防止1、2、3、4、5位和所有奇数位产生错误。此外，如果想获得更高的鲁棒性还可以执行加密，数据将会附加一个不同的消息认征码，确保其来自正确的发送方，防止攻击者回复消息给排污阀。同时，属性协议层Attribute Protocol Layer也拥有高度的防护方案，它提供了一种称为“准备属性写人”的能力，只允许将通过验证的返回值写入指定的属性（而不允许写入其他属性）。由此可见，仅仅是为了控制“阀门”的一个比特，甩到了14字节的循环冗余校验和认证码数据，足见低功耗蓝牙的鲁棒性。

**2.6凡事皆有状态**

低功耗蓝牙背后有个基本的概念：任何事物都有状态。状态可以是任何东西：当前的温度、设备里电池的状态、设备名称或是对测量温度的地点的描述。它通过属性服务器Attribute Server上的属性协议Attribute Protocol对外公开。

状态并不局限于“可读”状态，还包括“可写”状态。比如有一台可以设置温度阈值的自动调温器，另一台设备设置这个值来控制房间应升温或降温。既然可以公开各种状态，自然也可以公讦状态机的状态。通过使用显式的状态机属性值，设备可以清楚地提供其当前的状态。这为客户端随时断开连接提供了可能——在重新连接时，它们只需简单读取便可快速地确定对方的当前状态。

一些状态是可变的，甚至可以频繁改变。要快速实现服务器到客户端的状态传输，就必须支持状态信息的通知功能notifications。通知notifications直接从服务器发至客户端，无需客户端向服务器执行轮询，这种设计可以支持高效的应用，比如只有当电池出现状况时才向客户端通知电池状态，在此之前，客户端根本不用担心电池的状态。

这个简单的基于状态的模型使其得以构造一个非常高效的客户端一服务器架构。这也让应用能够采用面向对象的方法进行计，重用大量的数据类型和服务行为。同时，这也减少了设备需要包含的代码数量以及用来保存代码的存储器数量，从而降低设备功耗。更少的代码还有另一个显著的好处——更少的错误。总之，越简单的系统越便宜，开发更迅速，包含更少的错误，更加强健，也更容易维护。正如罗伯特布朗宁（Robert Browning）曾说的那样：“少即是多”。

**2．7客户端一服务器架构**

在客户端一服务器架构中还有一个设计元素作为设计低功耗蓝牙的基础。在设计低功耗时考虑了将设备连接到互联网的问题，原本可以在每一个资源受限的设备上集成IP协议，从而将它们公开给互联网。遗憾的是，即使是最简单的IP协议，其消耗的内存和能量也无法满足简单装置的要求。因此，协议设计者作出决定，不允许将IP数据包直接路由到从设备。

作为替代方案，智能网关实现了互联网和低功耗从设备之间的互联互通。采用纯客户端一服务器架构使得这种相互连接成为可能。服务器仅仅作为数据存储库，并不关心客户端是谁。客户端可以直接连接到服务器，或者可以从地球的另一侧通过互联网网关连接。

这为正在度假的用户提供了监视和控制居家环境的能力。由于低功耗蓝牙可用于安全警报、机顶盒或供热系统等各类设备，用户在去海边的途中可以查看所有的窗户是否安全，当躺在沙滩上时可以录制最喜爱的电视节目．当乘机返回时还可打开家里的供暖系统。

与网关连接的能力也让体育和健身设备可以向相关网站即时更新其收集的数据，哪怕是在锻炼者喝水的空隙。它也为老年人提供了监控能力，他们能够安全地待在家中，如有需要就会有人为其提供帮助。

客户端一服务器的网关模式也可以用于支持从客户端到网关的互联网安全，网关可执行访问控制、防火墙和客户端授权等操作，之后再向客户端授予网关外部的访问权限。这些成熟的技术而夸已在家用和商用领域广为使用。

**2.8模块化架构**

一个常被忽视的基本概念是面向未来的架构。大多数无线标准往往是在匆忙间完成的制定工作，只想让技术尽快脱颖而出，并不太关注该技术在未米10年或20年的时间里将如何运作。在“时间就是市场”的压力下产生的很多糟糕的结构设计无疑会对平台的长期可用性带来负面的影响。为了解决该问题，蓝牙技术联盟建立了一个特别架构工作组，专门研究基于通用属性规范GAP（Generic Attribute Profile）的架构，以确保该技术不会很快过时。

该工作组的主要成果是基于通用属性规范GAP（Generic Attribute Profile）的模块化的服务架构，允许设备以标准的方式将原子的、可封装的行为比特装进单个服务并将其公开。（此处，原子指的是只做一件事情，可封装指的是能够从其他功能剥离并自我封装。）服务可以引用其他的服务，比如有温度传感器的电池可以引用温度服务；相同的温度服务可被重复使用，比如家庭温度计、冷藏室的温度传感器、汽车发动机的冷却液温度传感器等。

在此架构中，一个有趣的副作用在于设备上公开的服务不必与某个给定的规范直接相关。规范要求设备提供一组给定的服务，但也仅仅需要一个链接而已。这意味着如果可以利用设备上的不同服务创建新的规范，则利用已有服务将其组合，就可蹦形成多个新的规范。事实就是如此，虽然只有在设备上设计和实现服务之后才会编写规范。

上述高度灵活的模块化架构有利于随着时间的推移逐步建立生态系统。例如，家庭可以首先部署智能电表，提供当前或耆未来的价格信息和使用信息。随后可以部署智能家电，使其支持远程开关，并采用网关模式在离家时实现自由控制。之后，部署的能源管理器可以利用智能电表的信息，“聪明”地调度各种智能家电，为屋于的主人节省用电开销。

**2.9十亿只是个小数目**

任何新兴技术都面临一个严峻的挑战——必须想方设法获得市场的青睬。一种技术若要取得成功就一定要降低成本，要摊薄成本则要增加销量，要提高销量还要获得成功。今天，手机是当之无愧销量最大的消费类电子设备，任何技术一旦进A手机就很容易取得成功，蓝牙便是一个典型的例子。低功耗蓝牙将依附于蓝牙在手机上的高配售率快速拓展市场。

随着最初几年手机厂商们不断地更新平台，低功耗蓝牙技术有望获得该领域超过十亿台设备的支持。有意思的是，它还为手机创造了巨大的配件市场。不但手机会很快带有低功耗蓝牙，事实上，由于纳入到现有的蓝牙系统的成本极为低廉，包括电脑、电视在内的许多支持经典蓝牙的设备都可能获得更新，开始支持低功耗蓝牙。

**2.10无连接模型**

设计经典蓝牙的初衷是代替耳机线、鼠标线、文件传输电缆等有线介质。这意味着建立链路所耗费的成本不那么重要，因为连接将维持几分钟、几小时甚至几天；连接开始时数秒的延迟也显得并不重要。然而，低功耗蓝牙改变了这一切。

低功耗的基本理念就是连接是瞬态的。当你需要做某件事或者检查情况时，可以快速创建一个连接，完成要做的事，而后断开连接。一台设备如果每五分钟通告一次状态信息．一天当中无线电打开的总时间不会超过1s。这说明该无线电在99.999%的时间内处于关闭状态——如果只保留4位有效数字的话，无线电关闭的时间是IOO%。每次建立连接的过程中产生的任何延迟都会显著地导致功耗增加。

低功耗蓝牙可以在大约3ms内建立连接、发送数据并优雅地断开莲接。这是许多设备的福音，这些设备包古状态信息，却一赢无{去负担无线通信的能耗成本，现在可以考虑增加低功耗蓝牙。即使小到按钮一般的简单小玩意也可以使用该技术，借助回收能量(scavenged power)．甚至不需要电池也可工作。

**2.11范式Paradigms**

最成功的技术常围绕不同的范式而设计，低功耗蓝牙也不例外。低功耗蓝牙技术使用两个主要的架构范式：客户端一服务器架构和面向服务的架构。

**2.11.1客户端一服务器架构**

在客户端一服务器架构中，客户端通过网络向服务器发送请求，服务器回复响应。这是互联网背后的主要范式，无疑也是有史以来发布的最成功的网络技术。

举个例子，当你在Web浏览器中键人一个URL地址时，它首先发送地址到DNS服务器。DNS服务器将已分配给该名称的对应的IP地址返回。然后，客户端通过超文本传输协议(HTTP)向服务器发送请求，一旦连接确立，请求服务器发送所要求的资源。接着服务器回复适当的资源，通常是一个包含了标记信息（HTML）的文本文件，告诉浏览器如何显示信息。

该文件还可以包禽一些URL供客户端获取资源，如图片或其他页面。这些额外的链接被枧为HTML贾面彼此链接成一个巨大的Web网络的真正原因，也正是因为这样，才有了Web页面和Web服务器的说法。

关于服务器和客户端的分工问题，二者有明确的区分。服务器通常以结构化形式保存信息，可以说，这些信息正是服务器存在的真正原因。信息可以是任何形式的数据，比如当前夏威夷科纳区的天气，下一班从首尔市区到机场的火车的时刻表，或只是一群朋友间的闲谈。另一方面，客户端没有任何数据，只是将请求发进到服务器。一旦从服务器收到答复，就可以执行指定的任务，例如为用户显示信息或发送通知，告诉他们认识的某人刚在推特上发了些东西。

客户端一服务器架构的主要优点是将客户端和服务器二者划分开来。当系统的不同部分位于不同的设备上时，这种划分必不可少。将其中一部分作为服务器而另一部分作为客户端，系统中二者的关系随之确定下来。

这种架构的主要优点是便干扩展。除了能够访问资源的URL，客户端不需要知道任何东西。客户端的数量也可以十分庞大，一些互联网的网站每天可能收到来自效百万客户的请求。服务器并不真正关心这些客户端是谁或来自何处，它只是对每一个请求做出响应。该服务器架构亦可进行拓展。用一台机器每天响应数百万的请求可能导致过载甚至服务失效。解决的办法是使用许多相同的、有权访问相同信息的服务器。还有个办法是将同一个域名解析成不同的IP地址告诉不同的客户端，使负载均匀地分布在各服务器之上。这就是所谓的负载均衡。

**2.11.2面向服务的架构**

客户端一服务器架构之上的进一步抽象是面向服务的范式。这是一个将服务器中的信息组织成服务的模型。该服务可以被发现、进行交互或用做已知的语义。这意味着该服务具有确定的行为，在给定相同的条件时，总会产生同样的结果。

这种范式是最成功的互联网系统的基础，如SOAP、REST、COBRA、RPC、Wcb服务等。为了更好地说明这一点，一种方法是把它与现实世界的例子相结合。假设有一个包裹需要尽快交付给另一家公司，你可能会做的第一件事是打电话给快速公司安排取件，然后支付服务费。关键在于，你总是知道接下来会发生什么，快速公司始缚遵循一套默认的行为：无论哪天给他们一个包裹，他们所做的事情都完全一样——把它及时地快递到目的地。这项服务具有预先确定的行为和已知的语义，并产生可预测的结果。

上述例子里有一个有趣的地方，你和快递公司的两个不同的人打了交道：接电话并接受业务请求的人以及上门取件的快递司机。此外，也许你并未意识到，其实还有一个处理财务账款的人。三人提供的子服务相互结合，形成了快递公司的主营服务。

这些子服务也是通用的，它们可以用于许多不同类型的公司。比如财务事务处理几乎能以同样的方式应用到各家公司。类似地，利用电话的方式在某个地方装货且在另一个地方卸货的服务也可以被应用于出租车公司。

这一切要运转，每个环节都必须遵循一套规则和惯例，下面将对此进行详细介绍。

**1.正式台约**

一个服务之所以被视为服务，是因为其在公开的功能以及如何工作两个方面提供了正规的描述。例如，快递公司的司机穿着公司制服，驾驶公司车辆，并愉快地迎接客户。他会在不同地点之间快速、安全地驾驶车辆，将包裹完好无损地送达。任何违反这些规则的行为将被视作违反了客户与快递公司之间的合约。因此，大多数快递公司也要求客户首先同意这种正式合约，然后才去取包裹。正式合约的一个好处在于，一个服务的实例很容易被另一个服务的实例所代替。只要两个服务的实例具有相同的功能和行为，这种情况就有可能发生。例如一个财务人员离开了公司，公司应该很容易找到一个知晓相同做账规则的替代者。在低功耗蓝牙中，这些正式合约位于服务规范之中，并为蓝牙SIG所采用。这些规范也有相应的测试规范，以确保实施行为的有效性。

**2．松耦合**

在面向对象的软件中，单独的系统组件是指被设计成元遗界效应的独立对象。那些发生在组件之间的相互作用可以被明确地定义和测试。

将依赖关系减少到最低限度，使修改服务的实现时不会带来意想不到的边界效应，从而降低风险。从遍一点出发可以得到—个台理的推论，即应当将正式合约及其实现这二者分离开来。这样，只要正式合约不被破坏或修改，实现就可以根据需要随意变换。

打个比方，快递公司可以增加更多的司机，把一个司机完成所有工作的局面变成很多司机在城市的一小块区域内取件，运回分拣中心、再发送到目的地，而这些工作可能由不同的司机完成。从客户的角度来看，该服务依然如故，包裹仍然按照预期被收走而后送达，但实现的过程却是完全不同的。注意，这里并未改变财务服务或订单服务。

**3．抽象化**

服务抽象是十分重要的设计原则，如果违背该原则可能造成严重的后果。假如不利用抽象化，而让客户端掌握服务的所有实现细节，那么客户端使用该服务的方式将会严重翩约服务的演化。

知识总是多多益善，这虽然是个常识，但在面向服务的体系结构下，客户端对服务的实现知识了解得越少越好。太多的知识会让客户端与某种具体的实现发生关联，从而妨碍服务的重用或是重新设计。一旦服务的实现发生变化，客户端有可能崩溃。

为了确保遵循诙原则，服务公开的状态应当尽可能少。此外，只应规定服务行为的外在表现。

**4．可重用性**

可重用性的概念多年来一直是面向对象方法所期望的设计目标。但是，真正意义上的可重用性是夸服务适用于多种不同应用的一种能力。如果未经认真思考，设计的服务往往仅能完成某一种工作。而在怠好的设计方案中，服务可以与具体的实现过程相互独立。这意味着该服务能够在其他应用程序中快速、轻松地获得重用。

蓝牙技术联盟为应对这一挑战设立了一个工作组，该工作组的唯一工作就是找寻通用的功能，对其需求进行抽象以实现有效的重用。

**5．无状态**

为了让众多客户端支持服务扩展，服务器不能保存任何客户端的状态数据。服务器或许可以定义一个服务用来记住客户端已经告诉它们的所有信息，使客户端在后续请求时不必重复这些信息。但该方法的问题在于，这些信息占用了大量的内存，且依赖于客户端和服务器二者之间同步的共享状态信息。这将导致服务器完全依赖客户端的正确操作，而这样的假设无疑站不住脚。

因此，无状态的设计目标是删除客户端和服务器之间所有的交互状态。虽然仍然有一些状态信息存储在服务器上，但是它们均为服务器状态．而非客户端状态。这意味着，无论任何客户端在任何时间发送任何请求，服务器都将以完全相同的方式响应相同的请求．而不管请求来自哪个客户端。

**6．可组合性**

上面提及的各种目标意味着，服务应该被设计得小而简单，然而现实世界从来都不是那么简单。现实世界中的服务是复杂的。要解决这个明显的矛盾冲突，面向服务的体系结构鼓励聚合较小的服务，以实现更高级的服务接口。

这一设计目标鼓励服务间相互组合。例如，快递公司的服务是由三个独立的服务组成的，这些单独的服务只要围绕既定目标，就可以合并成一个快递公司。同样，一些服务可以结合在一起组成出租车公司或商务汽车租赁服务。尽管实现不尽相同——送货车、家用车、行政豪华轿车，但本质上是相同的服务。

**7．自治**

为了重用和组合服务，服务本身必须是可靠的。基于系统内其他组件构造的服务在执行请求处理的可靠性上无疑要比对所有事情实施完全控制的服务更差。

自治的服务可以独立执行任务，而不管在其周边发生的情况。这些服务要在其他应用程序中实现重用将会非常容易。非自治服务可能会带来许多额外的支撑服务，并且可能与其他服务相冲突。

例如，快递公司的司机们按照规定自主活动，完成包裹收集和递送。即使办公室那边出了岔子，他们仍能继续独立地工作。

**8．可发现性**

最后，要想使用服务就必须能蟹发现服务。乍看上去似乎显而易见，但其实服务发现能力在移动自组织网络中是必不可少的。没有服务发现，所有服务必颓采用静态编程，复杂、繁琐而且容易出错。

可发现性通常是通过一个单独的、与服务交互的协议来实现的。比方说，要找到快递公司，人们应当使用电话簿或在互联网上搜索，绝不会是随机拨打电话号码并希望接通的是个快递公司。

低功耗蓝牙采用不同的方法；使用同一个协议实现服务发现以及服务交互。该协议称为“属性协议”，服务可发现性在其规范中被称为“通用属性规范”。速二者都将在第10章中进行介绍。

**第3章 低功耗蓝牙的体系结构**

专注简单是我一直以来信奉的价值观。保持简单比丰富功能更难：你曲须努力使思路清晰才能夸事情保特简单。但这终究是值得的，因为一旦做到速点，你将能创造奇迹。

——史蒂夫·乔布斯( Steve Jobs)

低功耗蓝牙的体系结构本质上非常简单。如图3-1所示，它分成三个基本部分：控制器、主机和应用程序。控制器通常是一个物理设备，它能够发送和接收无线电信号，并懂得如何将这些信号翻译成携带信息的数据包。主机通常是一个软件栈，管理两台或多台设备间如何通信以及如何利用无线电同时提供几种不同的服务。应用程序则使用软件栈，进而是控制器来实现用户实例。

在控制器内既有物理层和链路层，又有直接测试模式和主机控制器接口( HCI)层的下半部分。在主机内包含三个协议：逻辑链路控制和适配协议( L2CAP)、属性协议（Attribute Protocal）和安全管理器协议（Sccurity Manager Protocol），此外还包括通用属性规范(GATT)、通用访问规范(GAP)和模式(mode)。

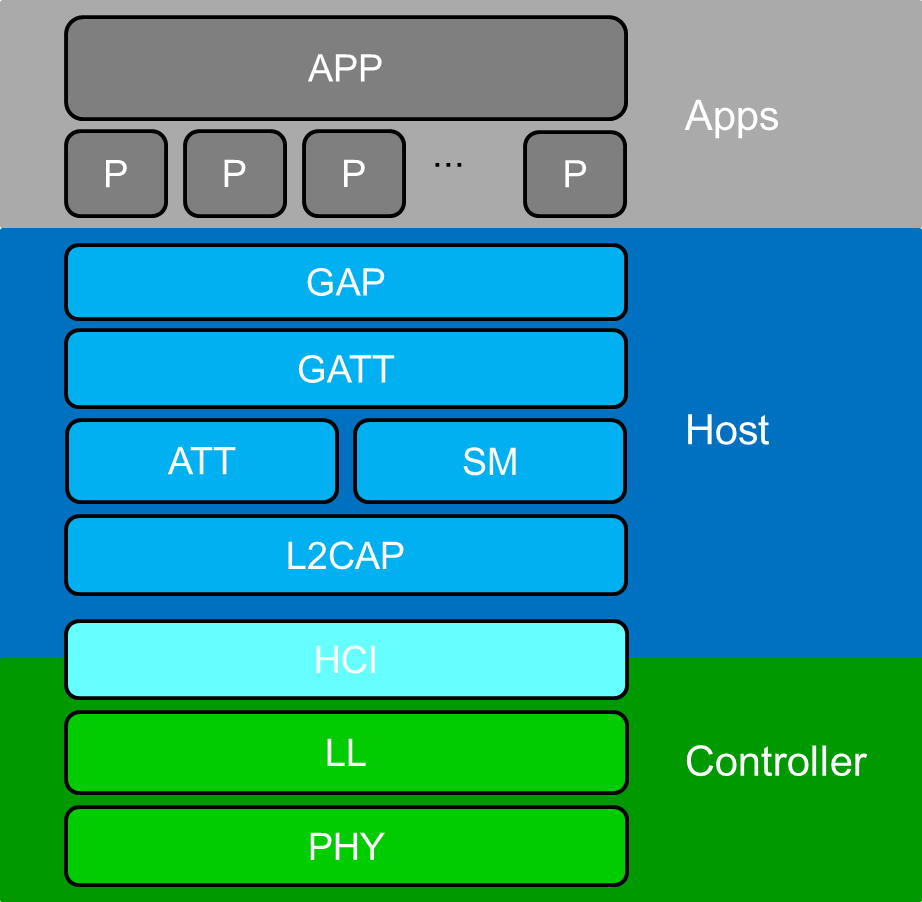


图3-1

**3．1控制器 Controller**

控制器被很多人视为射频器件。然而它并不比单单是射频部分。蓝牙控制器由同时包含了数字和模拟部分射频器件和负责收发数据包的硬件组成。控制器与外界通过天线相连，与主机通过主机控制接口(HCl)相连。

**3.1.1物理层Physical Layer**

物理层是采用2.4GHz无线电、完成艰巨的传输和接收工作的部分。对很多人而言，该层仿佛笼罩着一层神秘色彩。但本质上来讲，物理层其实并没有什么魔法，只不过是简单的传输和接收电磁辐射而巳。无线电波通常可以在给定的某个频段内通过改变幅度、频率或相位携带信息。在低功耗蓝牙中，采用一种称为高斯频移键控（GFSK）的调制方式改变无线电渡的频率，传输0或l的信息。

频移键控部分是指把1和O通过轻微升高或者降低信号频率进行编码。如果频率在改变的一瞬间突然从一端移向另一端，将会在更宽的频段上出现一个能量脉冲。因此用一个滤波器来阻止能量扩散到更高或更低的频率处。在GFSK的情况下，采用的滤波器的波形与高斯曲线一致。用于低功耗蓝牙的滤波器不像用于经典蓝牙的滤波器那样严格，这意味着低功耗无线电信号比经典蓝牙无线电信号要稍微分散一点。经典蓝牙的从2402-2480 MHz划分了79个信道，每个信道的带宽是1MHz。BLE从2402-2480MHz划分了40个信道，每个信道的带宽是2MHz。

适当拓宽无线电信号的好处在于无线电将遵循扩频(spread-spectrum)的约束，而经典蓝牙无线电则受跳频的约束。传输时，扩频无线电要比跳频无线电使用的频率更少。如果没有更宽松的滤波器波形，低功耗蓝牙将不能只在三个信道上广播，而不得不使用更多的信道，从而导致系统的能耗升高（如上一章所述）。

无线电信号的适度拓宽称为调制指数modulation index。调制指数表示围绕信道的中心频率的上下频率之间的宽度。传输无线电信号时，从中心频率出发超过I85kHz的正向偏移positive frequency deviation代表值为1的比特；超过185kHz的负向偏移代表值为O的比特。

为使物理层能够工作，尤其是应对同一区域有大量无线电同时传输的情形，2.4GHz频段被划分为40个**RF信道**，各信道的宽度为2MHz。**发射功率**在0.01mW（-20dBm）到10mW（+10dBm）之间。物理层每微秒传输l比特应用数据。例如要发送80bit的以UTF-S格式编码的字符串“low energy”将花费80us．当然这里没有考虑数据报头的开销。

**3.1.2直接测试模式**

直接测试模式是一种测试物理层的新方法。绝大部分的无线标准并未提供对物理层执行标准测试的统一方法。这就导致不同的公司采用专门的办法来测试物理层。这样一来，整个产业的成本增加，对终端产品制造商来说，从一家芯片供应商快速换到另一家的门槛也很高。

直接测试模式允许测试者让控制器的物理层发送一系列测试数据包或接收一系列数据包。测试者随后可以分析收到的数据包，或根据接收的数据包数量判断物理层是否遵循RF规范。直接测试模式不仅能量化测试，还能用于执行线性测试和校准无线电。比如，通过快速命令物理层在指定的无线电频率发送信号，并测量实际收到的信号频率，可以调节无线电直至正常工作。由于这类校准通常需要为每个单元执行，因而拥有一台能够高效地完成测试的设备将为产品制造商节约大量成本。

**3.1.3链路层**

链路层是低功耗蓝牙体系里最复杂的部分。它负责广播、扫描、建立和维护连接，以及确保数据包按照正确的方式组织、正确地计算校验值队及加密序列等。为了实现上述功能，定义下列三个基本概念：信道、报文和过程。

链路层信道分为两种：广播信道和数据信道。未建立连接的设备使用广播信道发送数据。广播信道共有三个——再次说明，这一数字是在低功耗和鲁棒性之间折中的产物。设备利用该信道进行广播，通告自身为可连接或可发现的，并且执行扫描或发起连接。在连接建立后，设备利用数据信道来传输数据。数据信道共有37个，由一个自适应跳频引擎控制以实现鲁棒性。在数据信道中，允许一端向另一端发送数据、确认，并在需要时重传，此外还能为每个数据包进行加密和认证。

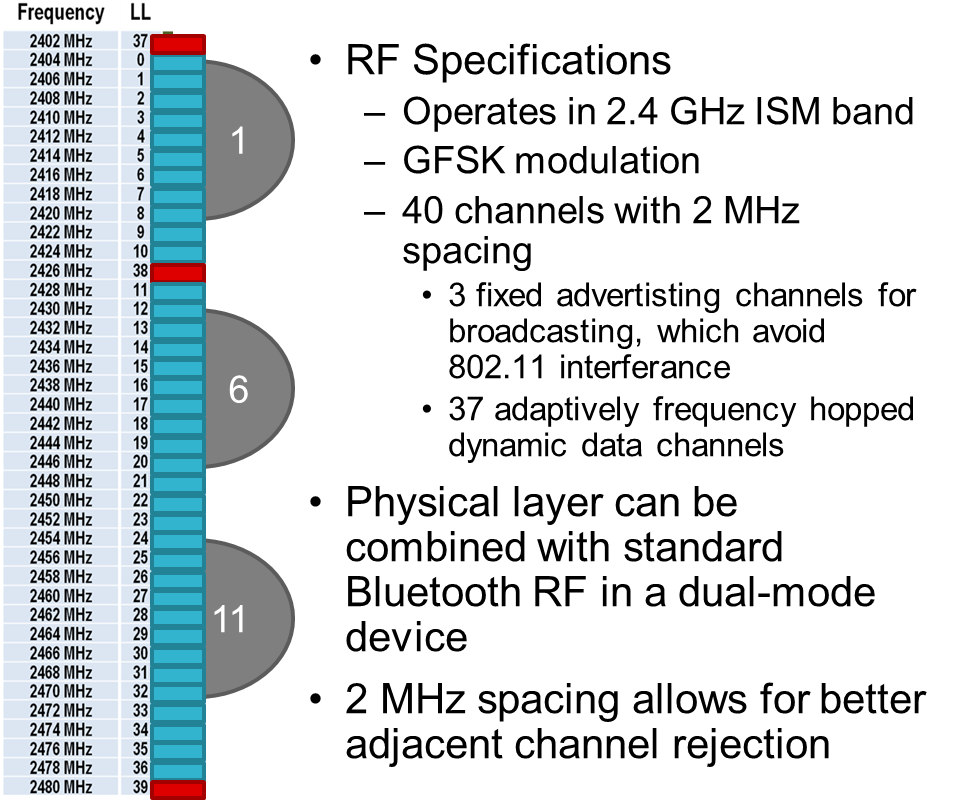


图3-2

在任意信道上发送的数据（包括广播信遣和数据信道）均为小数据包。数据包封装了发送者给接收者的少量数据，以及用来保障数据正确性的校验和。无论在广播信道还是数据信道，基本的数据包格式均相同，每个数据包含有最少80比特的地址、报头和校验信息。

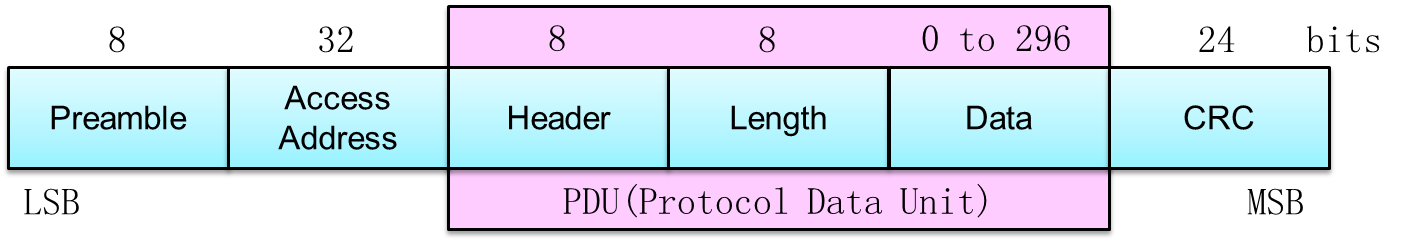


图3-3

8bit前导Preamble优化数据包的鲁棒性，这一长度足够接收者同步比特计时和设置自动增益控制，广播信道的前导码是10101010b，数据信道的前导码取决于接入码的LSB，如果接入码的LSB是1则前导码是01010101b，反之前导码是10101010b；32bit接入码Access Address在广播信道数据包中是固定值（0x8E89BED6），而在数据信道数据包中是完全随机的私有值；8bit报头Header字段描述数据包的内容；另一个8bit长度Length字段描述载荷的长度，由于不允许发送有效载荷长度超过37字节的数据包，不是所有的比特都用来记录长度；紧接着是变长有效载荷data字段，携带来自应用或主机设备的有用信息；最后是24bit的循环冗余校验（CRC）值，确保接收的报文没有错误比特。

可以发送的最短报文是空报文即没有Data的报文，时长为80us；而满载时的最长报文共发送296bit(37byte)时长为376us。大部分广播报文只有128us，而大部分数据报文时长为144us。

**3.1.4主机/控制器接口**

对于许多设备，主机/控制器接口（HCI）的出现为主机提供了一个与控制器通信的标准接口。这种结构上的分割在经典蓝牙里十分盛行，有60%以上的蓝牙控制器都使用HCI接口。它允许主机将命令和数据发送到控制器，并且允许控制器将事件和数据发送到主机。主机/控制器接口实际上由两个独立的部分组成：逻辑接口和物理接口。

逻辑接口定义了命令和事件及其相关的行为。逻辑接口可以交付给任何物理传输(Physical Transport)，或者通过位于控制器上的本地应用程序编程接口(API)交付给控制器，后者可以包含嵌入式主机协议栈。

物理接口定义丁命令、事件和数据如何通过不同的莲接技术来传输。已定义的物理接口包括USB、SDIO和两个UART口的变种。对大部分控制器而言，它们只支持一个或两个接口。考虑到实现一个USB接口需要大量的硬件，而且不属于低功耗的接口，所以它通常不会出现在低功耗蓝牙的单模控制器上。

因为主机控制器接口存在于控制器和主机之内，位于控制器中的部分通常称为主机控制器接口的下层部分；位于主机中的部分通常称为主机控制器接口的上层部分。

**3.2主机**

主机是蓝牙世界的无名英雄。主机包含复用层Multiplexing Layers、协议Protocols和过程Procedures，以此实现许多有用而且有趣的的事情。如图3-1主机构建于HCI接口的上半部分，HCI之上为逻辑链路控制和适配协议（L2CAP）。L2CAP是一个复用层。在L2CAP上面是系统的两个基本构建块：安全管理器SM（用于处理所有认证和安全连接等事务）以及属性协议ATT（用于公开设备上的状态数据）。属性协议之上为通用属性规范GATT，定义属性协议是如何实现可重用的服务的，而这些服务公开了设备的标准特性。最后，通用访问规范GAP定义了设备如何以一种可交互方式找到对方并与之连接。

主机并未对其上层接口做明确规定。每个操作系统或环境都会有不同的方式公开主机的应用程序接口，不管是通过一个功能接口还是一个面向对象的接口。

**3.2.1逻辑链路控制和适配协议**

逻辑链路控制和适配协议(12CAP)是低功耗蓝牙的复用层。该层定义了两个基本概念：L2CAP信道channel和L2CAP信令signaling commands。L2CAP信道是一个双向数据通道，通向对端设备上的某一特定的协议或规范。每个通道都是独立的，可以有自己的流量控制和与其关联的配置信息。经典蓝牙使用了L2CAP的大部分功能，包括动态信道标识符dynamic channel identifiers、协议服务多路复用器protocol service multiplexers、增强的重传enhanced retransmission、流模式streaming modes等。相比而言，低功耗蓝牙只用到了最少的L2CAP功能。

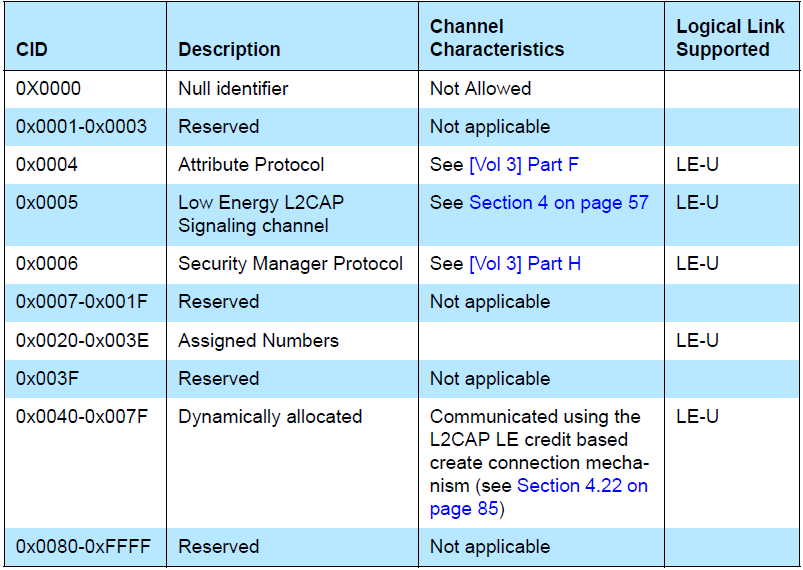


图3-4

低功耗蓝牙中只使用固定信道：一个用于信令信道signaling channel，一个用于安全管理器SM，还有一个用于属性协议ATT，如图3-4。低功耗蓝牙只有一种帧格式，即B格式，包含两个字节的长度Length字段和两个字节的信道识别符Channel ID字段，如图3-5所示。B帧的格式和传统L2CAP在每个通道使用的基本帧格式一致，在协商使用一些更复杂的帧格式之前，传统L2CAP会一直使用该帧格式。关于复杂帧格式的一个例子是经典蓝牙的包括了额外帧序列和校验值的帧。这些帧没有必要用在低功耗蓝牙中，因为链路层已有足够的校验强度，不必使用额外的校验值，而且简单的属性协议不会用多个信道乱序发送报文。通过保持协议的简单性和执行恰到好处的校验，只用一种帧格式也就足够了。

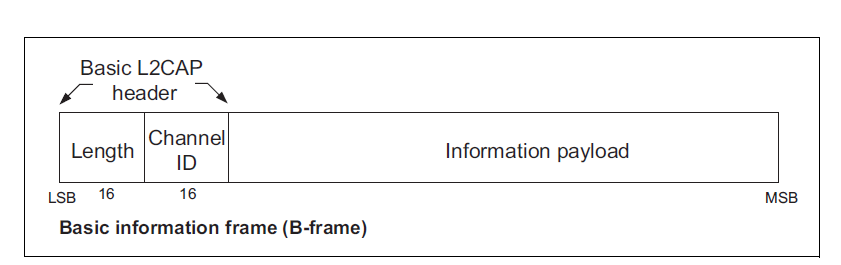


图3-5

**3.2.2安全管理器协议SM**

安全管理器定义了一个简单的配对和密钥分发协议。配对Pairing是一个获取对方设备信任的过程，通常采取认证authenticating的方式实现。配对之后，接着是链路加密和密钥分发过程。在密钥分发过程中从设备把秘密共享给主设备，当这两台设备在未来的某个时候重连时，他们可以使用先前分发的共享秘密进行加密，从而迅速认证彼此的身份。安全管理器还提供了一个安全工具箱，负责生成数据的哈希值、确认值以及配对过程中使用的短期密钥。

**3.2.3属性协议ATT**

属性协议定义了访同一对设备上的数据的一组规则。数据存储在属性服务器的“属性Attributes”里，供属性客户端执行读写操作。一个Attributes由下面的几部分构成。图3-5是一个属性的具体程序实现。

1. **Attribute Type** 属性类型的识别是通过Universally unique identifier（UUID）来实现的。UUID可以是32bit或者128bit。
2. **Attribute Handle**属性句柄是服务器分配给每个属性的的一个16bit的值。在同一个服务器中Attribute Handle必须是唯一的。0x0000为保留值。Attribute Handle可以分组。
3. **Attribute Permissions**属性许可是一个8bit的值，它定义了属性的接入许可，加密许可，认证许可，授权许可。
   * Access permissions接入许可被Server用来决定一个客户端是否可以读写某条属性的值，可以有的值为：

* Readable可读
* Writeable可写
* Readable and writable可读并可写
  + Encryption permissions加密许可的取值可以是下面两种：
* Encryption required要求加密
* No encryption required不要求加密
  + Authentication permissions被Server用来决定当客户端试图接入某个属性的时候是否需要一条认证的物理通路。服务器也可以用这个值来决定当需要发送一个notification或者indication到客户端的时候，是否需要一条认证的物理通路。
* Authentication Required要求认证
* No Authentication Required不要求认证
  + Authorization permissions授权许可用来决定在一个客户端在接入某个属性值之前是否需要授权。
* Authorization Required需要授权
* No Authorization Required不需要授权

1. **Attribute Value**属性值是8bit数组，数组的长度可以是固定值也可以是可变值。由图3-3可知，一个数据包中最多可以发送37个字节。如果属性值的长度超过了一个数据所能发送的最大长度，属性值的值对ATT是不透明的。对属性值数组的编码是通过UUID来实现的。

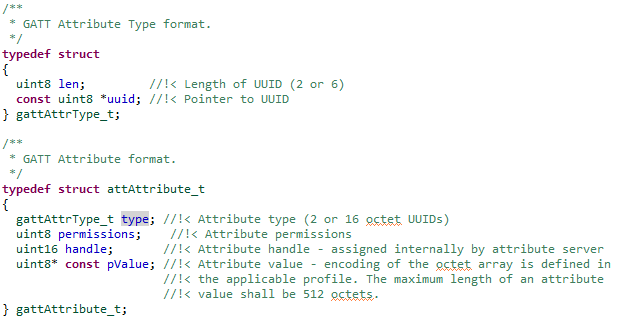


图3-6

客户端将请求发送至服务器，后者回复响应消息。客户端可以使用这些请求在服务器上找到所有的属性并且读写这些属性。属性协议定义了六种类型的信息：

1. **Request**从客户端发送至服务器的请求,。
2. **Response**从服务器发送至客户端的回复请求的响应。
3. **Command**从客户端发送至服务器的无需响应的命令。
4. **Notification**主动从服务器发送至客户端的无需确认的通知。
5. **Indication**主动从服务器发送至客户端的指示，需要Confirmation。
6. **Confirmation**从客户端发送至服务器的回复Indication的确认。

所以，客户端和服务器二者都可以发起通信，发送需要对方回复的消息或者无需回复的消息。

属性是被编址并打上标签的一小块数据。每个属性均包含一个用来标识该属性的唯一的句柄、一个用于标识存储数据的类型以及一个值。例如，一个类型是“温度”、值为20.5 ℃的属性可能放在句柄为OxO1CE的属性里。属性协议没有定义任何属性类型，但规定某些属性可以分组，并且可以通过属性协议发现分组的语义。

属性协议还定义了一些属性的权限：如果客户端验证了自身身份或得到了服务器的授权，客户端将获得读写属性值的权限或是其允许访问属性值的权限。客户端无法显式地获得属性的权限，只能隐式地通过发送请求并且接收错误的响应来尝试获得，该错误响应会说明不能完成请求的原因。

属性协议本身大多是无状态的。每一次事务处理（比如读取请求和读取响应等）并不会让服务器保持其状态，这使得协议本身只需要很少的内存即可工作。不过，仍然有个例外：准备和执行写入请求会将一组数据首先存储在服务器上，然后在一次事务处理中顺序执行所有的操作。

**3.2.4通用属性规范GATT**

通用属性规范GATT位于属性协议ATT之上，定义了属性的类型及其使用方法。通用属性规范引入了一些概念．包括“特性Characteristics”、“服务Services”、服务之间的“包含Include”关系、特性“描述符Descriptors”等。它还定义了一些规程，用来发现服务、特性、服务之间的关系，以及用来读取和写入特性值Characteristic Values。

服务service是设备上若干原子行为的不可变封装。听上去挺复杂，但却不难理解。不可变意味着一旦服务发布就不能再改变。这一点是必要的，因为若要服务能够被反复使用，最好永远不去动它。一但服务的行为发生变化，版本号等许多棘手的设置过程和相应配置将耗费大量的时间，这与低功耗蓝牙背后的“无连接模式”的基本概念完全背道而驰。

封装Encapsulation是指简洁地表达事物的功能。一项服务的所有的相关信息都置于属性服务器的一组属性中，并通过其来表达。一旦知道了某个属性服务器上的服务范围，你就知道了服务将封装哪些信息。原子意味着一个更大的系统中单个的不可逆转的单元或部件。原子服务十分重要，这是因为越是小巧的服务越有可能在其他地方获得重用。如果我们创建的服务非常复杂以致包含许多可能相关的行为，那么它们被重用的机会将大大降低。

行为Behavior是指事物响应特定情况或刺激的方式。就服务而言，行为意味着当你读取或写入某属性时都发生了些什么，或是什么原因导致了向客户端发送属性通知notification。明确定义的行为对互操作性尤为重要。如果服务规定的行为含糊不清，每个客户端在与服务器交互时有可能各行其是，服务器的行为表现将取决于哪个客户端正在连接。更糟糕的是，同样的服务在不同的设备上表现也可能大相径庭。一旦设备间出现这种局面，互操作性将被彻底破坏。因此，明确定义的、可测试的行为哪怕交互起来存在错误，仍能提升互操作性。

服务间的关系是实现设备公开的复杂行为的美键。服务本质上是原子的；复杂的行为不应该仅仅通过某一个服务来公开。举个例子来说，一台测量室温的设备可以公开温度服务。设备可能由电池供电，所以它会公开电池服务。然而，如果电池还有一个温度传感器，我们应该在该设备上公开另一个温度服务的实例。第二个温度服务必须和电池相互联系，以便客户端确定其关系，如图3-7所示。

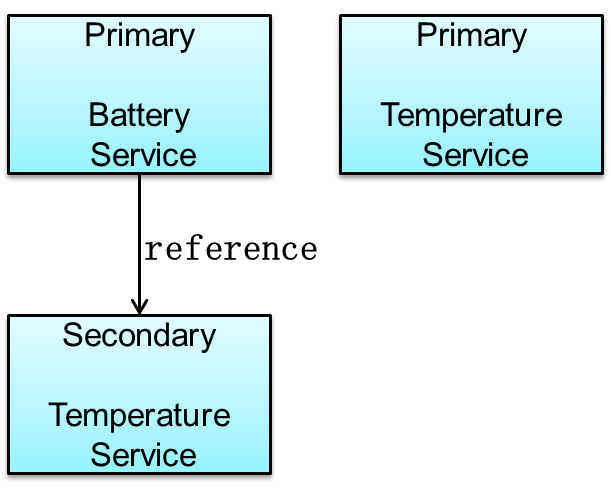


图3-7

为了适应复杂的行为和服务之间的关系，服务分为两种类型：首要服务和次要服务。服务的类型通常不取决于服务本身，而取决于设备如何使用该项服务。首要服务从用户角度公开设备的用途；次要服务被首要服务或另一个次要服务使用，使其能够提供完整的行为。在前面的例子中，第一个温度服务是首要服务，电池服务也是一个首要服务，而第二个温度服务——用来表示电池温度的实例则为次要服务，被电池服务所引用。

**3.2.5通用访问规范GAP**

通用访问规范定义了设备如何发现、连接，以及为用户提供有用的信息。它还定义了设备之间如何建立长久的关系，称为绑定( bonding)。要启用此功能，规范定义了设备如何实现可发现、可连接、可绑定。还介绍了设备如何使用规程procedures以发现其他设备、连接到其他设备、读取它们的设备名并和它们进行绑定。

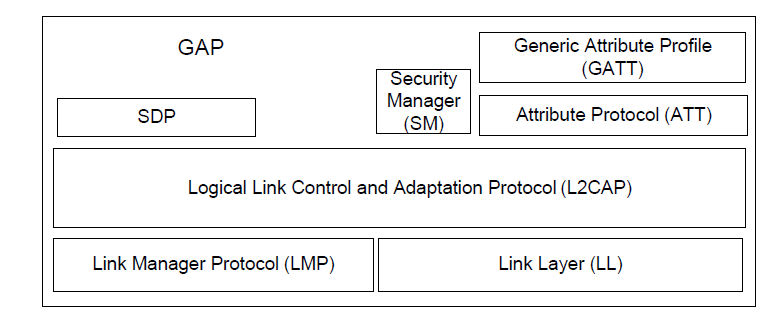


图3-8

通过使用可解析的私有地址，这当中还引入了隐私权的概念。隐私对于那些不断通告其存在以便其他设备能够发现并与之连接的设备而言是非常重要的。然而，希望保留隐私的设备必须采用不断变化的随机地址来广播。这样，其他设备一来不能确定它们侦听的是哪个设备，二来也无法通过跟踪其当前的随机地址判断哪个设备正在周围移动。怛是，让受信任的设备能够判断对端是否就在附近并允许其连接，这就要求私有地址必须是可解析的。因此，通用访问规范不仅定义了如何解析私有地址，而且定义了如何与私有设备进行连接。

**3.3应用层**

控制器和主机之上是应用层。应用层规约（specification）定义了三种类型：特性（characteristic）、服务（service）和规范(profile)。这些规约均构建在通用属性规范上。通用属性规范为特性和服务定义了属性分组，应用程序为使用这些属性组定义了规约。

**3.3.1特性Characteristics**

特性是采用已知格式、以通用唯一识别码（UUID）作为标记的一小块数据。由于特性要求能够重复使用，因而设计时没有涉及行为。只要是掭加了行为的东西，它的重用性就会大打折扣。特性有个很有意思的地方，它们被定义为计算机的可读格式，而非人类的可读文字。这赋予丁计算机相应的能力：当遇到一个索未谋面的特性时，可以下载计算机的相关读取规则，用来向用户显示该特性。

**3.3.2服务Services**

服务是人类可读的一组特征及其相关的行为规范。服务只定义了位于服务器上的相关特性的行为，而不定义客户端的行为。对于许多服务而言，客户端的行为可以隐式地由服务的服务器行为所决定。然而，还有些服务可能需要在客户端上定义复杂的行为，这些行为由规范profile而非服务service定义。

一个服务可以包括其他服务。父服务只能定义自身包含的服务，它不能改变包含的服务的特性或者行为。但是，包含服务时应描述多个被包含服务之间如何彼此互动。

服务有两种类型：首要服务和次要服务，如3.2.4节所述。一个服务本质上是首要服务还是次要服务取决于服务的定义，或者可以由规范文件和规范文件的实现来决定。首要服务表征一个给定的设备主要做些什么。正是通过这些服务，用户才了解了该设备的作用。次要服务是那些协助主要业务或其他次要服务的服务。

服务本身并未规定设备之间如何连接，以及如何发现和使用服务。服务只描述在读写特性characteristic或通知notified和指示indicated特性时究竟做了些什么。服务没有描述通用属性规范采用什么规程来寻找服务和服务的特性，也没有描述客户端如何使用特性。

**3.3.3规范**

规范是用例或应用的最终体现。规范是描述两个或多个设备的说明，每个设备提供一个或多个服务。规范描述了如何发现并连接设备，从而为每台设备确定所需的拓扑结构。规范还描述了客户端行为，用于发现服务和服务特性，以及使用该服务实现用例或应用所要求的功能。

规范和服务之间是一种多对多的映射关系，如图3-5所示。一个服务可以用于许多规范，以便在设备上实现需要的行为。此时，服务的行为与使用该服务的规范是相互独立的。在应用商店里，可以提交某个设备支持的所有服务的列表，从而找到使用这些服务的所有应用。这种灵活性有助于实现即插即用模型，比如通用串行总线便是如此。

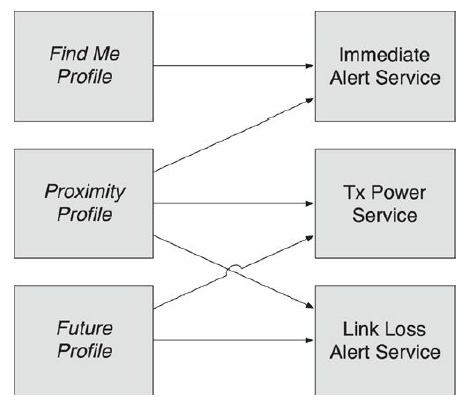


图3-9

**3．4协议栈划分**

构建一个低功耗蓝牙产品可能使用多种不同的协议找划分方案。标准规范定义了一种协议栈划分的方法，即使用主机控制器接口分隔主机和控制器这两部分。即便如此，我们也可以使用许多不同的方案。

**3.4.1单芯片解决方案**

单芯片解决方案可能是低功耗蓝牙里最简单的协议栈划分方案，如图3-6所示。图中其实并未划分协议栈，产品的所有部分都装在一个芯片上。该芯片包括控制器、主机软件和应用程序。这是低功耗产品的极简方案，只需要一个电源、一根天线、一些连接按钮或灯泡的硬件接口，以及一些额外的分立元件。

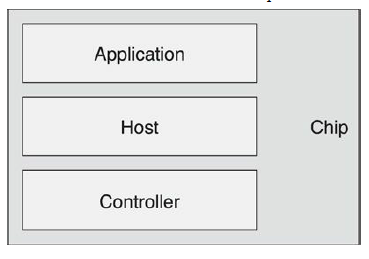


图3-10

不过，使用单芯片解决方案也有一些劣势。首先，因为芯片资源使用受限会导致开发环境使用起来也很困难。其次，为降低成本需要将软件烧录在片内的只读存储器（ROM）中，这就需要为每个产品单独定制芯片。当产品的批量很大时，这样做可以减少物料清单中元件的数量，但在小批量生产时，成本却可能相当昂贵。

对小批量生产或是原型产品来说，可以选用量产的单芯片和一块小的非易失性存储器芯片相结合，前者包含了从控制器到主机的所有一切，后者用于存储应用程序。这会让小批量生产具有非常低的成本。上电时，在非易失性存储器中的内容被读人单芯片中执行。因此，可以同时拥有一个有效的原型平台，以及一个具有成本效益的小规模生产方案。

**3.4.2双芯片解决方案**

经典的双芯片解决方案是将控制器放在一个芯片上，而将主机和应用程序放在另一个单独的芯片上，如图3-11。这种模式通常用于手机和电脑，因为它们已经拥有了非常强大的处理器能够运行完整的主机和应用软件栈。该方案通常使用提供了标准的主机控制器接口的量产控制器芯片。对于拥有强大处理器的设备而言，这种架构上的划分固然是一种理想的方案，不过对其他类型的设备而言，该方案却不能视作理想的选择。

一种替代的双芯片解决方案把控制器和主机放在同一个芯片上，而把应用程序放在另一个单独的芯片上，如图3-11所示。这样做的好处是，由于应用芯片不需要太多的内存或其他资源来运行应用程序，它可以是一个非常小的低功耗微处理器。两个芯片之间的接口通常是一个自定义的接口，比如采用简单的UART。该解决方案的优点是两种标准的量产芯片可以相互组合，同时，还可使用关于应用芯片的标准开发工具。

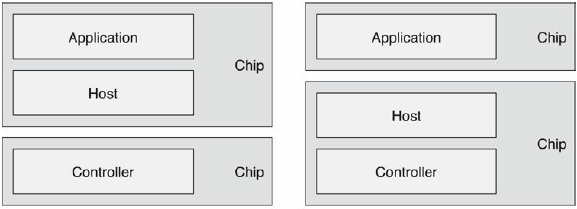


图3-11

**3.4.3三芯片解决方案**

当然，还可以使用多芯片解决方案，例如结合一个标准的控制器芯片、一个主机芯片和一个应用芯片，如图3-8所示。其中，主机芯片将需要两个独立的接口。圉3-8三芯片解决方案

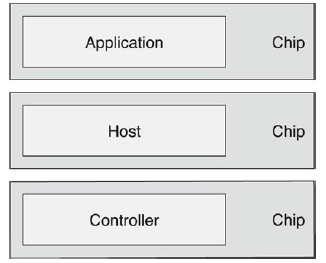


图3-12

这种解决方案往往十分昂贵，通常只限于开发那些使用多个接口以允许每层单独进行测试的系统。小批量生产时，由于减少元器件带来的成本节省较为有限，也能够容忍这种复杂性。对于大批量生产的产品，从成本的角度来看绝对不会采用这种架构。

**第4章 新的使用模型**

世界上所有的书籍包含的信息不超过一个蔓国大城市一年内播救的视频总量。但不是所有比特都具有同等的价值。

——卡尔·萨根（Carl Sagan）

低功耗蓝牙提供了一种使用无线技术的新方式。一些主要的新模型都基于广播模式，包括存在检测、数据广播、无连接模型等，此外还包括从设备接人互联网的网关。

**4.1存在检测**

存在检测是低功耗蓝牙所支持的最为有趣的新无线模型。“存在” presence是指当前发生的、或于某处出现的状态或者事实。使用广播模型，设备可以在后台被动扫描那些正在广播的设备。广播设备可以仅仅通告其地址，也可通告一些基于存在的数据。

广播是在链路层定义的一种新的运作模式。有了它，设备可以周期性地发送其身份信息和少量数据。低功耗蓝牙协议提高了无线电调制指数，允许2.4GHz频段规范发送更宽的无线电信号。由于并未采用无线电跳频方式，为实现设备的连接或发现所需的信道数较少，从而实现了更高的广播效率和扫描效率。

扫描模式可以分成两类：主动扫描和被动扫描。主动扫描时，扫描者从广播者请求更多信息，以获得额外的静态数据；被动扫描时，扫描者仅仅侦听广播报文，并不发送请求。链路层收到广播数据包后会将它们交给主机。

主机可以利用附近存在哪些设备的信息确定自身的位置。例如．假设主机发现了一辆汽车，它可以判断自身位于汽车内部或是在汽车附近，并相应改变其行为，例如与该汽车建立连接。在居家环境、办公室或咖啡厅也有类似的使用实例。正是这种位于后台的基于被动扫描的自动判别能力，使得设备能够根据其所处的位置自动地改变其行为。

正如之前的描述，“存在”说的是判断移动设备的自身位置。事实上，还有一类“存在”与静态设备有关，即判断指定的位置都有哪些设备。它的一个最大的好处在于方便你找人或找东西。譬如在一幢巨大的办公楼里，希望被追踪的人或设备间歇性地执行广播，而在每个房间的设备不断监测那些可被探测的设备，并将这些信息传输到一台中央设备，从而确定其位置。你可以用它来实现自动话务路由功能——把来电转到离你最近的电话机，也可以在建筑物内紧急疏散时用它来追踪员工。

**4.2广播数据**

广播模式还可以广播少量的数据——寥寥可数的数据，仅有几十个字节，但是能将少量信息广播给区域内所有的侦听设备的能力还是非常有价值的。如前所述，低功耗蓝牙设备拥有一项有用的功能，即根据其他设备的广播信息来判断自身位置。然而，该功能必须通过某种映射方式将广播设备映射到某个具体的物理位置。数据广播恰恰为此映射提供了帮助。

我们可以使用广播来传输多种不同类型的有用数据。广播数据有助于提升用户体验的三个主要方面为：发起连接建立、公告、广播信息。

为了便于建立初始连接，设备可以广播其自身的设备类型或是一些相关的信息，比如想要与具有何种服务或规范的设备进行连接。举个例子，当您从包装盒中取出电视机并首次将其打开，电视机会开始搜索遥控器。而当您在避控器里安装丁电池，遥控器会开始广播，表示其正在寻找电视机。

接收到广播数据的电视机随之连接到遥控器并自动配对，允许遥控器安全地对其进行操作。从消费者的角度来看，这就意味着他们第一次打开电视并把遥控器装上电池，随后便可以接下遥控器上的按钮来控制电视。在此过程中，既没有连接按钮，也没有配对菜单。

广播对于许多组织而言是一个有用的工具。有了它，消费者远在百米之外就能发现真实世界的服务。使用免费的无线技术进行广播，在国际机场和火车站等地方都能派上用场。它可以通知航班的登机口或火车的站台号等信息，为那些不想花移动漫游费或Wi-Fi上网费的旅客提供了一种获取信息的新途径。不但如此，在一些简易的公交车站也可以通告下一班班车的到达时刻和运行方向。

另外，设备还可以广播从本地收集的信息。比如一个温度传感器可以向正在侦听温度信息的任意设备广播温度信息。在感知不断变化的信息且需要同时服务大量的设备时，这一点尤为有用。

**4.3无连接模式**

设计并实现无连接模型是从经典蓝牙到低功耗蓝牙的一个最大的变化。在无连接模型中，设备无需为有效信息的快速交互保持连接。由于主要协议不会在设备间建立面向连接的通道，在需要发送数据时，将不再有连接丢失或重连的成本。这就鼓励设备在其需要发送数据时才建立连接，而非维持代价昂贵的数据连接以便让数据可以随时发送。的确，这种连接模式为标准的无线协议带来了一些有趣的设计变化。

在面向连接的信道上，状态信息需要通过协议并耗费一段时间才能建立。因此，在需要状态信息时，它通常还处于不可用阶段，只有通过两台设备间隐式的创建状态并且将其记住后才能使用。该状态信息需要很长的时间才能建立，导致初始连接由于发现和协商状态信息而被延误。基于隐状态的协议通常包含协商和配置过程、功能位和版本号等。如果连接持续很长一段时间将会产生很多的状态信息，一个全状态的系统处理起来可能更有效率。

遗憾的是，很多协议没有定义完全，它们的每个位状态均为隐式定义而非显式定义。这就导致了互操作性问题，每个设备认为连接都有不同的状态，因而对接下来会发生什么或应该发生什么做出不同的假设，而这正是面向连接系统的最大问题之一。该问题可以通过明确定义状态以及定义状态机如何工作得以解决。蓝牙的逻辑链路控制和适配协议（L2CAP）层是一个很好的例子：经典蓝牙定义了一个简单的状态机和配置系统用于建立连接，明确定义了所有的连接状态，并对连接状态机进行了完整的描述。然而，这一堪称典范的设计花费了十余年时间才完善到目前的水平。

无连接模式定义的是设备的状态，而不是连接的状态，从而解决了上述问题。通过无状态的协议（如协议属性）公开状态，允许在任意时刻断开连接，并在重新连接时直接从对端设备获得当前状态。另外，也可以显式地定义状态机，公开其状态和控制点，以便服务推进状态机的运行。另外，当一些信息发生改变、设备已经注册并准备接收当前状态变化信息时，也可以重新建立连接。

这里举一个通知设备的电量的倒子。一个监控装置连接到另一个由电池供电的设备，读取当前的电池电量，并设置其在电量改变时发出通知，然后断开连接。当电池电量发生变化，由电池供电的设备把自身设为可连接状态，而电池监控装置注意到它似乎有情况，向其发起连接。一旦建立连接，电池供电设备立刻将新的电池电平值通知监控装置，并随即终止该连接。这一切大约在3ms之内完成。对于充满电后可以维持一年的设备而言，在一年内总共进行99次电量报告，每次3ms，则无线电的总活动时间还不足300ms。比较起来，用经典蓝牙建立一个面向连接的信道所消耗的时间与之不相上下，而这仅仅是为发送一次报告而已。

**4.4网关**

在过去几年中，有关计算技术的一个最根本的变化在于互联网的传播和普及。从报纸、电视到收音机，好像一切都连接到了互联网，而三者分别是印刷文字、视频、语音三类媒体的门户。当然，互联网还有许多其他的用途，如电子邮件、社交网络、电话会议的音频和视频连接等。然而，下一个巨大的挑战将是在数以百计的设备与现今连接到互联网的设备之间建立连接。这将是一个巨大的变化，但是目前的基础设施却还没有能力应付所有的这些新型数据。

互联网的一个问题在于它的建立围绕的是面向连接的模型。一个TCP连接是在两个设备之间建立的面向会话的通道，而建立过程需要时间。事实上，连接到互联网的每个设备都有一个基于会话的地址。要获得该地址，设备要么预先植入一个地址，要么要求另一台设备为其分配一个地址；后者获得的地址仅可使用一段时间，随后必须要求重新分配。

互联网最大的问题在于其当前的结构是围绕有线的基础设施设计的。线缆在绝大多时候非常可靠；而因为与线缆连接的设备不能移动，它们也可以被连接到其他的供电线路。这意味着，有线协议的能量效率很少会被人们考虑到。事实上，在随机的时间间隔里，路由器不断地检查分配的或非分配的互联网地址到设备之间的映射，通常意味着互联网设备需要一直保持侦听状态。这在设备不断移动并且需要超低功耗的情况下是无法工作的，得采用另外的方法才行。

低功耗蓝牙遵循的模型和许多实现了互联网接人的家庭采用的模型类似，他们大多拥有一台以上的计算机，并且使用支持网络地址转换( NAT)技术的网关。从外部看来，每个家庭通常只有一个唯一的互联网地址，分配给网关或路由器；而该网关为家中与之相连的所有设备分配一个独立的地址空间。关键在于，网关实现了内部地址和外部地址的转换，对外界隐藏了内部网络的拓扑结构。外界只看到一个设备，并不真正关心究竟是哪个设备在发送或接收数据。

互联网协议本身开销巨大。在IPv6网络中，每一个传输的数据包都包括128位源地址和目的地址。这意味着即使不考虑其他的协议开销，一个数据包的最小尺寸也将达到32字节，这一长度已经超过了低功耗蓝牙的最大数据包。因此，在低功耗蓝牙上直接使用互联网协议变得非常困难——虽然在蓝牙协议设计之初有线连接还是主流。然而，网关模型让我们能够对外界隐藏设备的内部寻址。内部寻址的设备可以使用一个独立的IPv6地址空间，也可以使用其他一些对外界透明的解决方案。

通过使用低功耗蓝牙网关，微型无线设备不但可以连接到互联网，而且能使用尽可能少的能量实现。这样一来，低功耗蓝牙把互联网的复杂性推给了有资源应付这种复杂性的网关设备。这些网关设备可以使用任何它们想要的方案实现互联网地址到设备地址的映射，比如可以为每个设备分配一个单独的IPv6地址，或为每个设备使用一个单独的网络地址的端口号。

如果你的冰箱在保修期内出现压缩机故障，需要通知制造商更换，就必须有一种方式来将信息传达给制造商的互联网服务器，这耐阿关的作用就体现了出来。显然，所有制造设备都会需要做类似的事情，比如洗衣机、汽车、真空吸尘器。制造商还可能希望将信息发送给这些设备，比如他们可能要升级洗涤程序。因此，网关为设备提供了一种高教的互联网交互方式，不用负担那些驱动互联网的高能耗的有线协议。

**第二部分 控制器**

**第5章 讲述设备间如何实现无线西信。**

**第6章 强谰直接潮试横式及其在执行{低成本、高效的无线电测试时所起的作用**

**第7章 介绍有关报文格式、广播和连接建立等的底层协议。**

**第8章 描述主机与腔制器进行交互的接口以爰如何用其执行一些有用操作。**

**第5章 物理层**

你看，有线电报是一种很长很长的猫。你在纽约拽它的尾巴，它的头会在洛杉矶喵喵叫。明白了吗？无线电也一样：你在这边发信号，他们在那边就能收到。唯一的区别是没有猫。

——阿尔伯特·爱园斯坦(Aibert Einstein)

**5.1背景**

两个以上的低功耗蓝牙设备使用无线电波收发信息。无线电已经存在了很多年，从最简单的火花隙式发射机开始，经过幅度调制、频率调制，发展到现在相移键控以及其他复杂的调制方案。以下各节将从基础开始，介绍无线电的工作原理以及低功耗蓝牙所用的调制方案。

**5.2模拟调制**

基本的火花隙式无线电可能是你能制作的最简单的无线电。它的发射机简单到只需两个元件：一个正负电极在同一边的9V电池以及一个能导电的硬币。但在做发射机之前，你需要先设置一下接收机：将一个调幅收音机调到一个没有电台的频率。之后，用硬币短暂连接电池的两个电极。当硬币非常靠近两个电极但还未完全接触时，间隙处会产生电火花，你会听到它对收音机造成的干扰。

火花隙式无线电有两个问题。首先，效率不高。长距离传输需要非常高的电压，通常高达数千伏。其次，在一定区域内不能有多个发射机同时发射。这严重制约了在一定区域内传输多个消息的能力，例如，想象一下整个城市的居民只能选择一个电视台。

无线电技术的下一个发展阶段是幅度调制，或者称为调幅无线电。无线电显然可以发送一个单频载波。进而调制这些载波，便可以发送某些信息。幅度调制中，载波的幅度会变化。从根本上说，这是一个巨大的进步，因为可以同时发送多个无线电信号。图5-1是一个具有代表性的模拟调幅信号。



图5-1

各个国家和公司可以建立很多不同的无线电台。只有短波的年代里，政府建立的电台可以选择一个空白的频段，或者故意挑选一些会和其他国家电台冲突的频段，为的是让他们的公民只能收听当地的宣传。最终，各国签署了频率分配的国际协议，以合乎逻辑且无冲突的方式分配频率资源。一直以来，绝大多数的无线电频率分配都以这些协议为基础。

有些情况下，频段的分配也许类似，但使用方式会有所不同。比如，在美国和欧盟，用于调幅广播的中波频段均为530kHz - 1620kHz。不同的是，这一频段在美国以1OkHz为间隔分配给电台，而在欧盟间隔为9kHz。这意味着欧盟可以有更多的电台，但接收机的设计需要能适应两种方式。

当你收听调幅电台时，会感觉到幅度调制有一些明显的问题。当无线电台的音频输入很小的时候，接收机可能完全丢失信号，或者试图接收有用的东西时却输出更多的噪声。噪声（通常称作背景噪声）总是存在的，其来源可能是众多的电子设备，也可能来自闪电以及其他大气效应的影响，包括来自太阳的辐射。

无线电技术的下一个发展阶段显著提高了信号质量。采取的手段是使用频率调制，或称为调频无线电。幅度调制根据输入信号改变载波幅度，因此当输入信号非常微弱时，载波输出也会非常微弱；与之不同的是，频率调制根据输入信号改变载波频率（见图5-2）。输入信号为声音，这意味着当音量非常小时载波几乎没有频偏，而当音量很高时会导致非常大的频偏。调频的最关键之处在于，载波总是以最大功率发射，因此接收机可以镇定信号进而解调出其承载的信息。调额的另一个优势是多个载波可以被放置在彼此十分接近的位置。例如，对于现代调频广播的一个频道，其包含了单声道信号（左声道与右声遭之和）、立体声信号（左声道与右声道之差）、电台的数字信息（无线电数字系统）以及导频音。



图5-2

因此，频率调制解决了火花隙式无线电和幅度调制的主要问韪。调频也比较简单，对于接收机具备锁定任意信号的能力，并且可以发送频率接近的成组信号。还有一些更为先进的调制方案，比如相位调制。与频率调制类似的是，相位调制也会根据输入信号改变载波频率，确切地说是改变信号的相位。相位调制比简单的频率调制要复杂，因此只用于复杂的数字系统，如数字广播。除此以外，还有正交幅度调制，其利用两个相位羞为900的载波进行幅度调制。大多数国家的数字电视广播采用了正交幅度调制。

**5.3数字调制**

在讨论数字调制之前，有必要简单回顾一下码片chips、比特bits和符号symbols之间的区别。输入数据以比特表示，非0即1。一个比特可与其他一些比特组合起来形成一个多比特值。组合起来的多个比特被统称为一个符号symbol。因此，一个符号可表示多个比特。还有另一种在现实生活中用得很少的方式，即一个比特利用多个码片传输。每个码片是一个比特的一部分．或者说—个比特由多个码片组成。最后讨论一个过于复杂的情况，就是多个码片可以组成—个表示多个比特的符号。

比较无线电系统时，会遇到各种指标。当无线电系统无法直接比较时，通常会比较这些指标。其中最有用的是应用层数据传输率。该指标是扣除所有报文额外开销后，应用层能够传输的最大数据传输率。报文额外开销是指除实际应用层数据之外，用于定义和管理传输本身的额外符号。比如时间同步信息、地址信息、描述应用层数据的报头，以及用于保证数据正确性的校验信息等。

最多被引述的是物理层比特率physical bit rate。该指标是指当连续发送时，1s内所能发送比特的最大数量。除了电视与无线电广播之外，能连续发送数据的无线电系统很少。多数无线电系统需要将数据分拆成多个自成体系的报文。另一个程有用的指标是符号速率symbol rate。符号速率指1s内所能传输符号的最大数量，确定该值时需要保证接收机能正常工作。符号速率越高，处理符号获得信息所需的能量也就越多。

与应用层能够发送多少数据相关的另一个重要指标是帧率frame rate。该指标是指一段时间内所能发送报文的个数。电台需要在发送态、接收态之间转换，而转换需要时间。转换时间越长，用于传送应用层数据的时间便越少。

谈到传输数字信息，调制方案更易懂一些。最简单的数字调制是通断键控（OOK）。键控是用来描述如何根据数字信号调整载波的一个专业术语。通断键控是指在特定时间开启（发送）或关闭（不发送）载波。换个思路，可以认为通断键控是输入信号在最大值和最小值间变化的简单幅度调制。因此，如果认为没有载波则为0，有载波则为l，容易看出，通过在适当时刻发送适当幅度的载波，如图5-3所示的波形传输了8比特信息。这种方法十分简单，但其容易受噪声的影响，尤其是信号很微弱的时候。

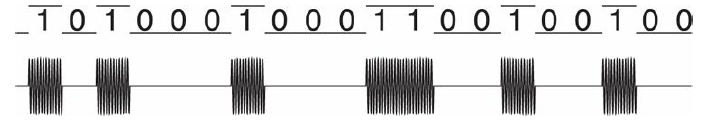


图5 3通断键控

稍复杂的数字调制是幅移键控（ASK），见图5-4。它类似于幅度调制。如果输入信号是二进制的（仅有0%和1OO%），则幅移键控退化为通断键控，如果可以使用4个不同的级别（0%，33%，66%，100%）．则每个级别可以编码2比特信息。8比特的数据仅需4个符号symbols，是通断键控传输速率的两倍。幅移键控和通断键控都有信噪比低的问题，这样的话对一个符号所代表信息的判定可能会有误。

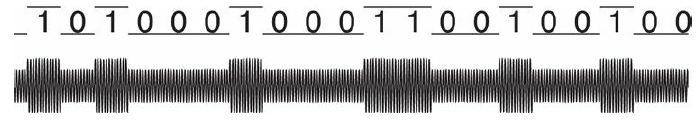


图5-4幅移键控

频移键控（FSK）是更为复杂的数字调制。它与频率调制类似。如图5-5所示，频率的变化用来决定符号的值。最简单的频移键控是二进制频移键控binary frequency-shift keying，即负频偏negative frequency deviation对应比特。正频偏positive frequency deviation对应比特l。频移键控中，可以利用不同的频偏值代表不同的级别levels。频移键控的优点在于总能收到并锁定载渡，因此比幅移键控的传输距离更远。频偏的数学表达式见下式。

（5-1）



图5-5频移键控

偏差的大小称为调制指数modulation index（记做h），式中的是相对于载波中心频率的最大频率偏移，是被调信号的最高频率。如果调制指数大于1，即载波的频率变化大于被调信号的频率变化，这种情况称为宽带传输。如果调制指数小于1，即载波的频率变化小于被调信号的频率变化，这种情况称为窄带传输。调制指数等于0.5是一种称为最小频移键控( MSK)的特殊情况。最小频移键控是一种频谱效率很高的频移键控变种。

**5.4频段**

低功耗蓝牙使用2.4GHz工业、科学及医疗(ISM)频段传输信息。这一频段有两个特殊之处。大多数频谱是需要授权的，即需要购买授权才能使用。比如，当地的广播电台要花很多钱购买授权才能广播音乐和广告，前提是他们的商业收入足以购买音乐、支付工资并获得频谱授权。有些频段需要授权，但不需费用。比如用于航空、军事和民用紧急服务的频段。有些频段是免费且无需授权的。是的，无需授权！

2.4GHz ISM频段的第一个特殊之处在于，这是一个无需授权的频段。只要遵守一定的规则，就可以在此频段上传输。规则也很简单，即该频段仅能用于传输距离和传辕功率受限的个人局域嗣(PAN)和局域网(LAN)。这些规则的细节很复杂，但从根本上说，这个频段用于短距离应用是无需授权的。

2.40Hz ISM频段的第二个特殊之处在于，这是唯一一个在所有国家都无需授权的频段这意味着，无论在哪里购买的这个频段的产品，在任何国家都可使用而无需配置。也有一些别的ISM频段，比如900MHz左右的，其频率和带宽取决于地域。比如美国使用915MHz．而欧盟则使用868MHz。而对于2.4GHz的频段，从2400MHz - 2483.5MHz的约83.5MHz频谱资源在任何地方都可以使用。

**5.5调制**

低功耗蓝牙采用丁高斯频移键控（GFSK）。高斯滤渡器通过增加从一个值到另一个值的频率转换时间，优化了一个符号到下一个符号的转换。如果投有高斯滤波器，频率转换会非常快，从而导致很大的噪声。高斯滤波器的引入，意睐着从比特。到比特1的转换迅速而高效。

低功耗蓝牙的物理层比特率为1Mbit/s（或记为1Mbps），每个符号1比特。调制指数约为0.5，也就是说非常接近最优的最小频移键控。实际的调制指数可以在0.45 - 0.55之间，意味着低功耗蓝牙不是严格意义上的最小频移键控，但它具有最小频移键控的大多数特点，比如减少了边带发射功率，这意味着无需复杂的滤波器就能满足频谱监管的要求。

负频偏代表比特。，正频偏代表比特1。最小的频偏约为180kHz。也就是说，如果中心频率选取了2402MHz．比特。意味着传输2401.820MHz，比特l意味着传输2402.180MHz，如图5-6所示。

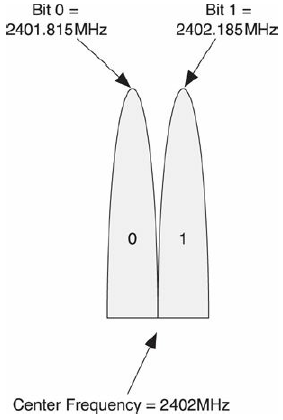


图5-6调制

**5.6射频信道**

实现长距离、高稳定性的无线电系统，通常需要使用多个频率来传输信息。有些系统通过高带宽来实现这一目标，比如802.11使用20MHz或40MHz的信道快速地传输信息。有些系统利用跳频来实现，比如经典蓝牙在79个窄带信道间切换进行信息传输。选择哪种方式通常是随意的，但对于复杂多变的多径环境来说，多个窄带信道比参量宽带信道的方式更为有效。

低功耗蓝牙传输信息所使用的40个无线信道如图5·7所示。每个信道的中心频率可以简单地根据式( 5-2)算出。

（5-2）

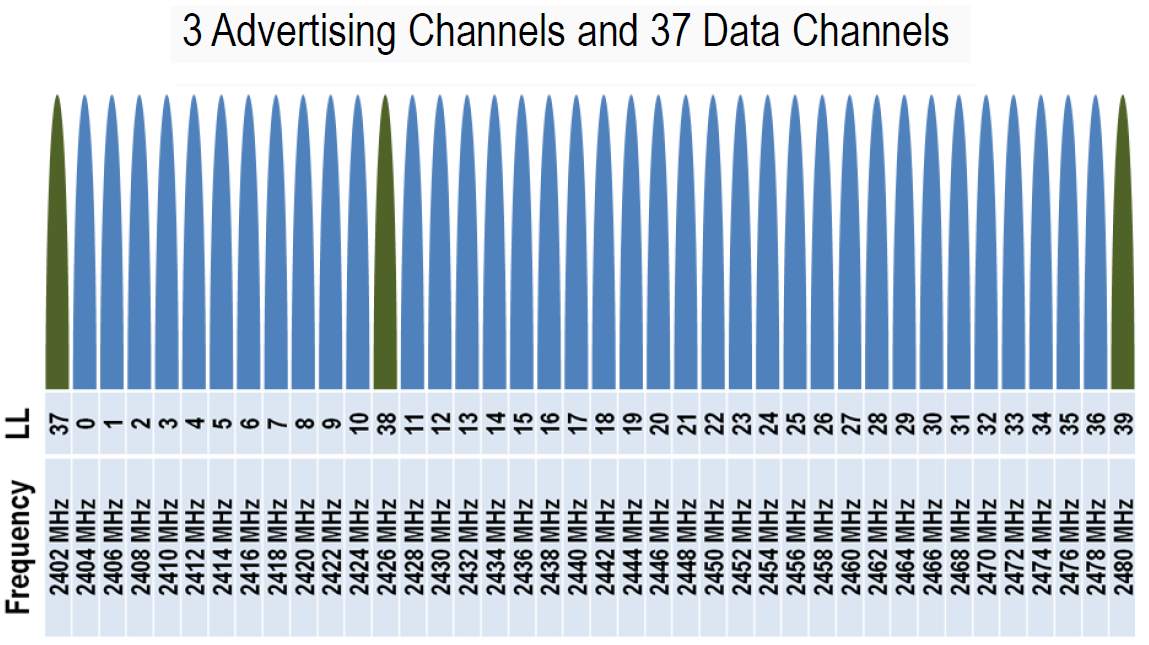


图5-7无线信道

这里的正是无线信道k的中心频率。

这意味着低功耗蓝牙使用的最低频率为2402MHz，最高频率为2480MHz。在较低的频率部分，低功耗蓝牙所使用的频率与2.4Hz ISM频段的最低频率有2MHz的间隙，而在较高的频率都分，二者之间有3.5MHz的间隙。

**5.7发射功率**

2.4GHz ISM频段对无需授权的设备有最大发射功率的限制。对于低功耗蓝牙，规范规定最大发射功率为+1OdBm。同时规范也规定了最小发射功率不应低于-20dBm．否则这些设备将由于“音量”过小而不能被其他设备“听”到。

+1OdBm的发射功率为1OmW，而-20dBm的发射功率仅为IOuW。

**5.8容限**

制造设备时都会给定一个容限。通常来说，容限越精确，设备越昂贵。对于无线电，主要的容限可以通过频率精度确定。即使无线电被设计为在2402MHz发射，实际的工作频率也可能是2401.850MHz或者2402.150MHz，这也是发送一个报文的中心频率容限。对于低功耗蓝牙，整个报文的中心频率容限是±150kHz。中心频率之所以会偏离，原因在于它一般是由一个已知频率的晶振倍频获得的。晶振的频率通常为16MHz，如果想获得2400MHz就要倍频150倍。晶振的任何偏差也会被放大，并在发射频率中体现。举个例子，如果晶振的实际频率为16.OOO1MHz，则中心频率将偏离理想频率150kHz。我们称这个晶振的误差为62ppm，即百万分之62。通常，大量使用的低成本晶振的误差能在约50ppm以下。

另一个很重要的指标是报文发送过程中中心频率的漂移。漂移可能由芯片工作时积聚的热量导致。随着热量积聚，电台的内部频率会发生轻微的漂移。在一个低功耗蓝牙报文的发送过程中，漂移不允许超过50kHz。也就是说，在开始发送一个报文时，如果中心频率是2402.OOOMHz，发送过程中中心频率耍在2401.950MHz - 2402.050MHz之间。与此同时，频率的最大漂移率也不能超过400Hz/us。

**5.9接收机灵敏度**

构建接收机时，一个非常重要的问题是：它有多好。测量接收机的灵敏度可以量化这个问题：接收机检测其他设备发出的无线信号有多灵敏。灵敏度通常以dBm为测量单位，这是一个非常小的单位。低功耗蓝牙规定接收机的灵敏度要高于-70dBm。换句请说，接收机接收到0.000 000 1mW的电磁能量时要能正常工作。但是，噪声总是存在的。如果不能对其正确解码，仅仅检测到信号是没有意义的。因此，实践中，接收灵敏度阈值要在一个可接受的误比特率(BER)条件下定义。低功耗蓝牙选择的误比特率为0.1%。

支持低功耗蓝牙的控制器，其接收机灵敏度可达-90dBm．或者说1pW。这是令人难以置信的、从频段噪声中能检测到的少量能量，而这也导致了客观的通信距离，下一节将会具体解释。

**5.10通信距离**

为了计算低功耗蓝牙电台的通信距离，需要确定系统的链路预算。链路预算由发射机芯片到接收机芯片之间消耗能量的各个部分组成。其中包括天线、匹配电路的增益和损耗。不过．假设天线和匹配电路的差别不大。，链路预算的主要来源是路径损耗。路径损牦是指从发射机天线到接收机天线的能量消耗。式( 5-3)给出了路径损耗与距离的关系。表5-1给出了一些具体教值。图5-8和图5-9是二者关系的图形化表示。应当指出，该等式只是给出了近似的关系，而忽略了发射／接收系统中的所有损耗，并且只适用于全向天线。

path toss= 40+ 25log(d) (5-3)

式( 5-3)中，path loss表示路径损耗；d表示发射机和接收机之间的距离。



表5-1 路径损耗与距离的关系

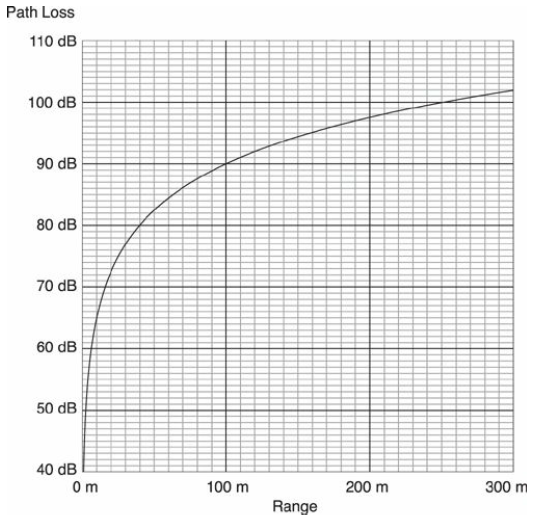


图5-8路径损耗图示

如果发射功率为-20dBm．接收机灵敏度为-70dBm，则允许的路径损耗为50dB，对应于2.5m的通信距离。这也是低功耗蓝牙以最低功率发射，并在接收机灵敏度最低的情况下所达到的通信距离。

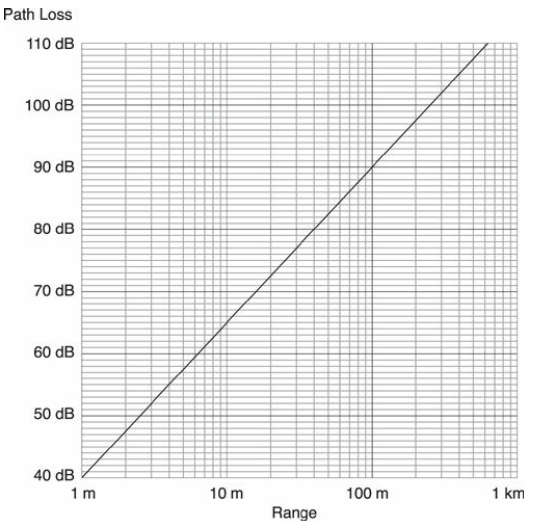


图5-9路径损耗（对数图）

如果发射功率为OdBm，接收机灵敏度为-80dBm，则允许的路径损耗为80dB，对应于40m的通信距离。这是低功耗蓝牙在常见的发射功率和接收机灵敏度情况下所达到的通信距离。

如果发射功率为1OdBm，接收机灵敏度为-90dBm，则允许的路径损耗为1OOdB，对应于250m的通信距离。这是低功耗蓝牙以最高功率发射，井在接收机灵敏度非常好的情况下所达到的通信距离。

**第6章 直接测试模式**

知识必须通过实践来获得；你不能仅凭空想来做测试，而必须真刀真枪做实验。

——索福克勒斯(Sophocles)

**6.1背景**

无线系统的最大问题之一，是如何校准、认证以及进行生产线上的性能测试，这对于希望生产成本尽可能低的无线系统尤为重要。当器件被封装到模块或者产品中之后，就无法像刚被生产出来时那样，抛开主机协议栈而花几秒钟就完成器件测试。直接测试模式通过定义标准测试流程和硬件接口解决了上述问题，即使主机协议栈或者其他部件已经和器件融为一体。

直接测试模式需要三个设备（如图6-1所示）：

* 待测设备(DUT)
* 上位浏试设备(UT)
* 下位测试设备(LT)

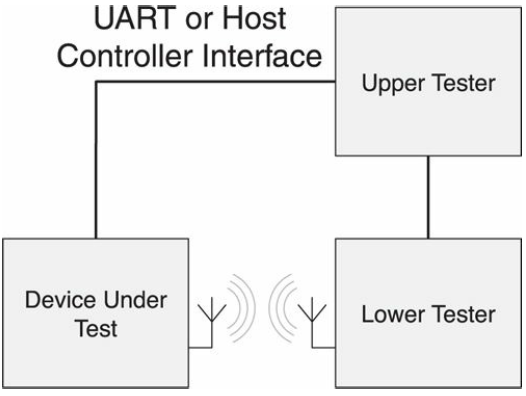


图6-1测试配置

待测设备指那些需要测试的控制器、模块或者最终产品。待测设备需要有一个天线，以及一个通用异步串行通信接口（UART，以下简称串口）或者是连至上位测试设备的主机，控制器接口(HCI)。

上位测试设备通常由测试设备制造商生产。它的软件能够通过串口或者主机／控制器接口驱动待测设备，也能驱动下位测试设备并与之通信。

下位测试设备能够收发报文，通过天线与待测设备有效地通信。

待测设备由上位测试设备告知做什么，并执行报文发、收。同时，上位测试设备通知下位测试设备做相反的操作，即执行报文收、发。也就是说，待测设备会和下位测试设备进行报文交互。测试结束时，上位测试设备能够通过待测设备和下位测试设备的有效信息，比如待测设备的报文计数器以及下位测试设备更丰富的信息，来判断待测设备是否通过测试。

上位测试设备还可以通过要求设备发送已知频率的报文并测量实际的发射频率，来校准生产线上的控制器。通常，用于时钟参考的外部晶振与其设计频率有差别时，需要进行此操作。这种晶振频率教调会在控制器发送报文时完成，可以实现快速的部件校准。

**6.2收发机测试**

为了测试收发机，控制器会被要求发送或接收报文。待测设备控制器什么时候发送报文是由测试者决定的。测试者试图接收这些报文，并从中提取多种物理层特性。例如，下位测试设备可以通过这种方式得到待测设备的频率漂移。待测设备控制器什么时候接收报文也是由测试者决定的。测试者发送一定数量的报文，待测设备对正确接收的报文计数。测试结束时，这一计数信息会上报给上位测试设备。这样一来，测试者便可获知接收机在给定发送报文数量条件下的接收性能了。

**6.2.1测试报文格式**

测试报文的格式类似于7.3节所描述的广播报文，区别仅在于测试报文的接入地址是广播报文接入地址的按位逆序。不过，由于测试报文的接人地址只用在测试过程中，不会在一般产品的连接过程中出现，因此不像主广播报文接入地址那样受到保护。

测试报文与广播报文的报头格式相同，前4位为测试报文类型，其余位为0。因报文类型的不同，测试报文的净荷可能为如下几种情况：

* PRBS9（一种9位伪随机比特序列）
* “11110000”
* “10101010”
* PRBS15（-种15位伪随机比特序列）
* “00001111”
* “01010101”

在低功耗蓝牙认证测试规范中，只使用了前三种报文类型。不过，其他类型在生产线测试中可能有用。对于所有测试报文，均不进行白化处理。这是为了精确测量非随机比特序列的频偏。

**6.2.2发射机测试**

发射机测试是为了获知待测设备发射机的性能。这一测试能得到频偏、频率漂移，以及其他规定的射频参数。

上位测试设备发送特定命令使得待测设备进入发射机测试模式。该命令包括3个参数。第一个参数决定发送的中心频率，用式(5-2)定义的射频信道号表示。第二个参数决定报文净荷长度，有效值为0～ 37字节。第三个参数决定发送报文的数据类型。

可发送的测试报文包括以下三种；

* PRBS9-用于发射功率测试
* 11110000-用于频偏测试
* 10101010-用于载波频率和初始频率测试

PRBS9报文使用多个重复的9位伪随机比特序列。该随机序列由一个线性反馈移位寄存器生成。PRBS9序列是一种常用的快速测试无线性能的测试样式。之所以使用PRBS9序列，是因为它与进行过白化处理的连接报文有类似的随机特征。因此，发射功率测试以及接收机灵敏度测试均可采用该序列。

“11110000”报文使用重复的4个连“1”和4个连“0”。该报文可用于连“0”或连“1”情况下的频偏测试。测试者会搜寻225kHz以上的频偏。这也说明了为什么测试过程中需要禁用白化，否则物理层无法发射连“0”或连“l”。

“1O1O1O1O”报文使用交替的“l”和“0”。该报文可用于“O”、“l”交替情况下的频偏测试，也可用于载波频率测试和发射初始频率测试。

下位测试设备收到足够多用于获得测试结果的报文之后，上位测试设备将命令待测设备停止测试。待测设备应该立即停止发射，并返回一个事件用于确认不再发射。上位测试设备便可以进行下一项测试，比如使用不同的报文类型、报文长度、中心频率或者这三者的任意组合。

**6.2.3接收机测试**

与发射机测试相比，接收机测试简单很多。接收机测试主要用于确定不同发射功率条件下接收机的误比特率。一般情况下，下位测试设备以已知的功率发射，待测设备接收，二者之间用射频线连接，从而避免不确定的路径损耗带来的影响。待测设备对正确接收的报文进行计数，并在测试结束时将这一信息上报给上位测试设备。上位测试设备便可以确定错误报文数量，从而估计特定信号强度情况下接收机的误比特率。

上位测试设备发送特定命令使得待测设备进入接收机测试模式。该命令只有一个参数，即接收的中心频率，和发射机测试一样用射频信道号表示。待测设备收到该命令后，将报文计数器清零，并开始准备接收。

接收机收到一个有效报文后，将报文计数器加1。待测设备重复这一过程，直到上位测试设备命令其停止。一旦收到停止命令，待测设备应立刻停止接收，并返回一个事件确认不再接收。该事件包含了待测设备接收的有效报文数量信息。上位测试设备可以基于发射、接收的报文数量，计算接收机的性能。之后，上位测试设备便可以进行下一项测试，比如使用不同的频率或者不同的发射功率。

**6.3硬件接口**

考虑到控制器可能来自多个不同厂商，为了进行高效、与设备无关的模块测试，低功耗蓝牙标准规定了待测设备与上位测试设备间的硬件接口。该接口是一个简单的双线串口，一根线从待测设备向上位测试设备传输信号，另一根线相反。如果模块或者最终产品的生产厂商能够引出这两个用于连接上位测试设备的引脚或者测试点，就可以在生产线上校准或检验每个产品的发射和接收能力了。

**6.3.1串口**

串口的配置通常很灵括。为了保证兼容性，直接测试模式仅允许修改串口的一个参数：波特率。

待测设备的渡特率可以从如下数值中选取：1200，2400，9600，14400，19200，38400，57600，115200。常用的波特率是38400，该值是在命令、事件的传输效率和实现成本之间的合理折中。

其他串口参数均采用最为常用的设置：8位数据，无奇偶校验，1位停止位。段有软硬件流控。硬件流控很明显无法实现，因为没有流控所用的硬件连线。软件流控也没必要使用，原因在于命令最长只有2字节，并且一次只能发送一个命令。最后的设计参数是共地．即要求两个设备的信号地是连在一起的。

发送命令或事件时，2字节数据之间最大的时间间隔是5ms。一个命令发送后，相应的事件必须在50ms内返回。这是为了保证待测设备能够及时开始或停止测试，进而保证计数的准确性。

**6.3.2命令与事件**

上位测试设备可向待测设备发送四种命令

* 复位
* 发射机测试
* 接收机测试
* 测试结束

待测设备可向上位测试设备上报两种事件

* 测试状态
* 报文报告

顾名思义，复位命令的作用是使得待测设备恢复初始状态。无论控制器处于何种状态，收到该命令都要停止当前操作。如果控制器正在发送或接收测试报文，则必须停止发送或接收。该命令将待测设备置于已知的正常状态之中。待测设备会返回一个事件，确认一切都已恢复到初始状态。复位命令只使用两位，其余位会被忽路。

发射机测试命令用于启动发射机测试。如图6-2所示，该命令包括三个参数：频率、长度和报文类型。待测设备收到该命令之后，会返回一个测试状态事件，并接给定的参散发送报文。如果由于某种原因，待测设备无法发送报文，则需要返回一个带有错误指示的测试状态事件。如果发射机测试命令失败了，上位测试设备需要复位待测设备，使其进入巳知的正常状态。

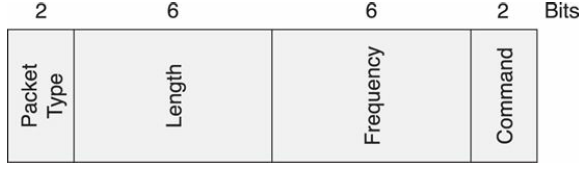


图6-2直接测试模式命令的比特结构

接收机测试命令用于启动接收机测试。该命令只包含一个参数：频率。命令中的其他位会被忽略。待测设备收到该命令之后，会返回一个测试状态事件，并以给定的频率接收报文。同样，如果接收机不能接收任何报文，测试状态事件的状态位会指出这一错误。

在一次成功的发射机或接收机测试之后，上位测试设备会发送测试结束命令终止测试。待测设备收封泼命令之后，应立即停止正在进行的操作并返回报文报告事件。对于接收机测试，该事件包含成功接收的报文数量（见图6-3）。对于发射机测试，由于没有接收报文数量，因此相应的域值为0。上位测试设备收到报文报告事件之后，便可以发送其他测试命令进行其他测试了。

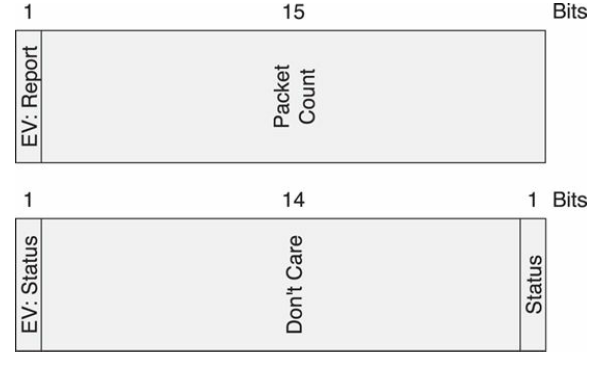


图6-3直接测试模式事件报告的比特结构

测试状态事件仅定义了一个位。如果该位为0，则对应的测试命令执行成功。如果为1．则对应的命令因某种原因失败了。测试命令失败可能有两个原因：

* 控制器处于不正常的状态。这种情况下，上位测试设备应发送一个复位命令。
* 控制器不支持该测试，比如该控制器没有发射机或者接收机。（尽管对于只含发射机的控制器，也可以测试其接收机。）

上位测试设备无法通过命令集确定待测设备是否支持某种测试，而只能尝试每一种测试并通过其状态进行判断。

报文报告事件用于上报测试完成。该事件包含一个15比特的值，用来表示成功接收的报文数量。对于发射机测试，该值总为。。而对于接收机测试，该值可能为。- 32 767。如果测试时间过长发生溢出，将无法分辨接收了1个报文还是32 769个。即便如此，待测设备也无需担心溢出，而应该由上位测试设备保证测试时间不会长到导臻该计数器溢出。

**6.4使用HCI的直接测试模式**

直接测试模式也可以复用已有的HCI传输层（参见8 2节）和逻辑接口访问控制器。这种方式需要更多的部件，尤其是对于主机接口比较复杂的情况。但是，对于单独测试控制器（而不是控制器已经被集成在高度优化的模块或最终产品的情况）．这是一种非常有效的方式。

测试流程与之前的描述相同，区别仅在于完整的HCI命令和事件代替了之前所用的2字节命令和事件。HCI的命令、事件与直接测试模式的命令、事件之间的映射如表6.1所示。



表6-1 直接测试模式命令、事件及对应的HCI命令、事件

当使用HCI时，没有专门的测试状态事件或者报文报告事件，因为命令完成事件已经涵盖了二者的功能。命令完成事件包含触发该事件的命令的操作码，以及与命令相关的参数。

**第7章 链路层**

一切的连接技术只不过让我们越发地明白：万物万事于我们竟是如此陌生。

——希瑟·多纳约( Heather Donahue)

链路层定义了两个设备如何利用无线电传输信息。它包括了报文packet、广播advertising、数据信道data channels的详细定义，也规定了发现其他设备的流程、广播的数据、连接的建立、连接的管理以及连接中的数据传输。2.4GHz ISM频展的干扰、噪声、深度衰落等无线传输系统的挑战加剧了链路层的复杂度。

**7.1链路层状态机**

在讨论报文及其用法之前，有必要了解一下链路层状态机的基本概念以及它们对设计低功耗蓝牙的影响。

如图7-1所示，链路层状态机定义了5种状态：

* 就绪态Standby
* 广播态Advertising
* 扫描态Scanning
* 发起态Initiating
* 连接态Connection

不过，可以认为扫描态有两个子状态：主动扫描和被动扫描。连接态也有两个子状态：主、从。

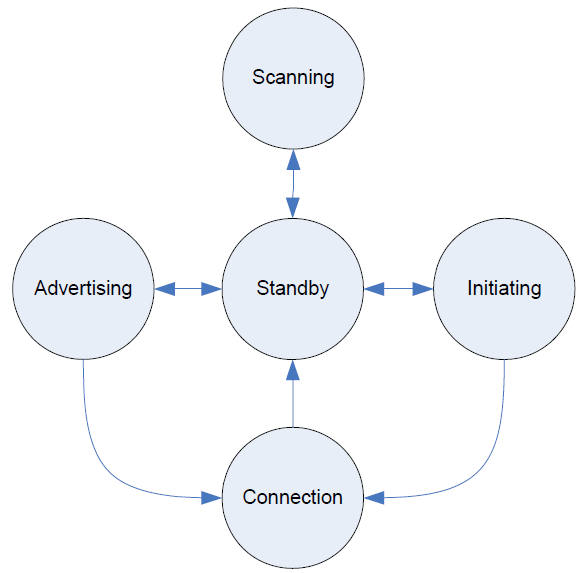


图7-1链路层状态机

链路层状态机不但解释了设备间如何发现和连接，也解释了低功耗蓝牙的另一个基本的设计策略：将广播、发现和连接过程与连接中的数据传输分离开来。该设计的一部分是为了实现广播设备的超低功耗。将广播信道减少到3条，在保证鲁棒性的同时降低了功耗。但这一设计也需要区分广播态和广播报文。链路层状态机的3个状态使用广播报文，而只有1个状态使用数据报文。

**7.1.1就绪态**

上电后，链路层进入并保持就绪态，直到接到主机的命令。从就绪态可以进入广播态、扫描态或者发起态，如图7-2所示。从其他任意状态也可以进入就绪态。就绪态是链路层状态机的中心状态，尽管是非话动状态，但最为重要。

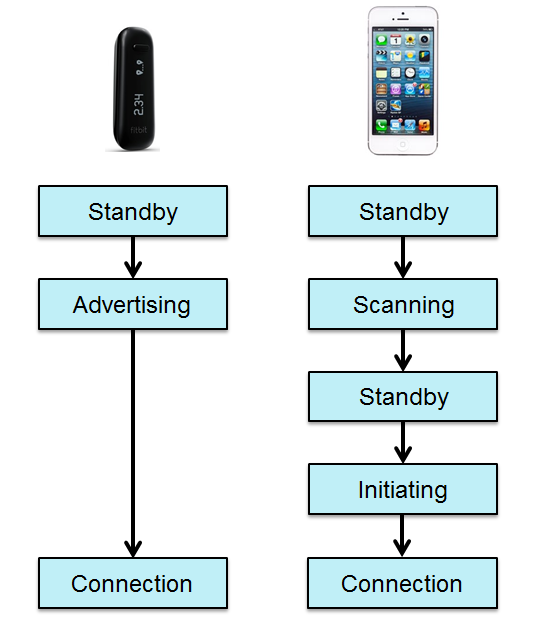


图7-2

**7.1.2广播态**

处于广播态的链路层可以发送广播报文见图（7-2）．也可以发送扫描响应，用以回应主动扫描的设备。可被发现或者可被连接的设备需要处于广播态。想向一定区域内其他设备广播数据的设备也需要处于广播态。

为了成为广播者，设备必须有发射机，但也可以有接收机。只支持广播态的设备可以没有接收机，以降低芯片成本。应当指出，实际中，出货量大的收发机芯片往往比出货量小的单发射机芯片更便宜。

广播态的设备停止广播后可进入就绪态。在收到发起者的连接请求之后，广播态的设备也可以进入连接态。

**7.1.3扫描态**

处于扫描态的设备能够接收广播信道的报文（见图7-2）。扫描态可用于简单地侦听哪些设备正在广播。扫描态有两个子状态：被动扫描和主动扫描。被动扫描仅接收广播报文。主动扫描则发送扫描请求给广播态设备，并获取附加的扫描响应数据。

扫描态的设备只能进入就绪态，转换的条件是停止扫描。

1控动扫描

被动扫描态中，设备其能被动地扫描，不能发送任何报文。因此，被动扫描可以在只有接收机的设备中实现。如果只为了支持被动扫描，则无需发射机，进而可以减少控制器的体积和成本。不过，正如之前提到的，价格取决于由货量的多少，同时带有收发机的设备可能会更便宜。

2．主动扫描

主动扫描态中，链路层一旦发现了新的广播态设备，都会发送扫描请求，并等待该请求的响应。扫描请求和响应报文都在广播信道中传输。为了使主动扫描更为有效地工作，扫描响应应尽量为静态数据，因为就能耗而言，扫描请求、响应这两个额外的报文使得数据更加昂贵。然而，原始广播报文中的数据则可以定期变更，因为这些报文总会被接收到。

**7.1.4发起态**

为了发起连接，链路层需要处于发起态。处于发起态（见图7-2）的发起者，其接收机用于侦听自己试图连接的设备。如果收到了来自该设备的广播报文，链路层会向其发送连接请求并进入连接态，并假设广播者也进入连接态。如果发起者不再试图发起连接，也可以进入就绪态。

**7.1.5连接态**

最后一个提及的链路层状态是连接态（见图7-2）。从广播态或发起态均可进入连接态。两种情况均源于发起者向广播者发送连接请求报文。

正如已经提到的，连接态有两个子状态：主或从。连接态中，两个设备相互传送数据信道报文。这也是唯一一个用到数据信道的状态，其他各个状态均使用广播信道。连接态的设备只能通过断开连接转换为就绪态而不能进人其他状态。

1．主连接态

主连接态只能从发起态进入。为了成为主设备，它必须向对端设备发起连接。主设备必须定期向从设备发送报文。从设备只有通过回复这些报文才能发送自己的数据。

2．从连接态

从连接态只能从广播态进入。为了成为从设备，它必须向对端设备进行广播。对于从设备，只有在正确接收主设备的报文之后才能发送。收到主设备的一个报文后，从设备可以发送一个自己的报文。如果从设备想发送更多的数据，则必须等待主设备发送另外的报文再回复。从设备也可以随时忽略主设备以达到节能的目的。这样一来，从设备可以通过“休眠”来节省大量的能量。

**7.1.6多状态机**

一个链路层的具体实现可以拥有多个独立的状态机。举个例子，通过配置，一个设备可以同时作为从设备、广播者和主动扫描者；或者同时作为主设备、广播者、被动扫描者或者发起者。一个设备也可以同时作为多个主设备连接多个从设备。重要的是需要了解其中的一些限制。

**不能同时成为主从设备。**

链路层是一个“独裁者”，如果成为了主设备，便不能同时成为从设备。类似地，从设备也不能同时成为主设备。这也意味着主设备不能发送可用于连接的广播报文。不过，它还是可以发送不可连接的广播报文或者是可发现的广播报文。

由于上述限制，从设备也不能发起连接，因为这会导致其成为主设备。通过这一限制．设备在任何时间点上应该做什么都没有不确定性。

低功耗蓝牙链路层这一确定性的特性使得设备可以使用高效的调度算法。任何需要在多个时间域内同步的非确定性系统，都需要非常复杂的调度算法。这些算法也因为不确定性而需要大量的计算资源，因而不符合低功耗的设计目标。而确定性设计则可以基于离散逻辑实现高效算法。

**一个设备也不可以同时成为两个主设备的从设备。**

一个已经连接的从设备不能广播可连接的广播报文。事实上，同时作为两个主设备的从设备，是比同时作为主设备和从设备更复杂的情况。对于经典蓝牙，这被称为分散式网络(scattemet)，而低功耗蓝牙不支持这种分散式网络。

**7.2报文**

报文是链路层的基石。报文非常简单，它是带有标签的数据，由一个设备发送，一个或多个设备接收。标签指明了数据由谁发出，以及应该由哪些设备接收。

图7-8给出了报文的基本结构，适用于所有报文，无论其用途是什么。报文的开始是一小段训练序列，称为前导。之后是接人地址，接收机用它将报文和背景噪声区分开来。接入地址之后是报头和长度字节。再之后是报文的净荷，以及用于确保净荷正确性的循环冗余校验码(CRC)。

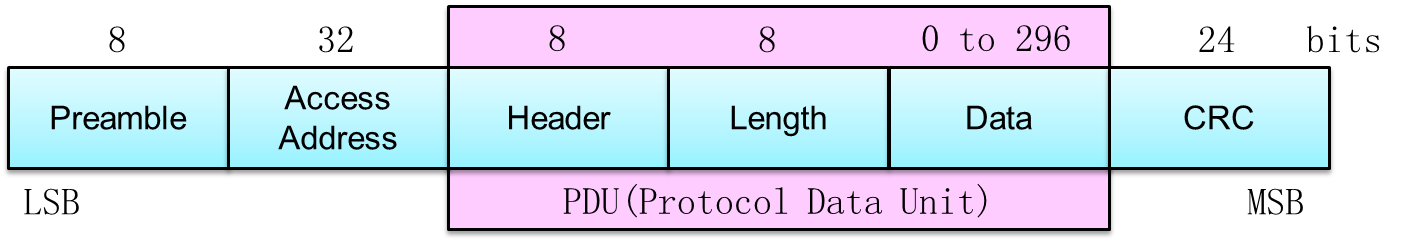


图7-8报文结构

**7.2.1广播与数据报文**

低功耗蓝牙规范中，有两类报文：广播报文和数据报文。两类报文具有两种完全不同的用途。设备利用广播报文发现、连接其他设备。一旦连接建立之后，则开始使用数据报文。而两者的区别在于，数据报文只能被连接中的两个设备（主、从设备）所理解，而广播报文则可以广播给多个侦听设备或者只发送给某个特定设备。

是广播报文还是数据报文由其传输所在的信道决定。低功耗蓝牙规定了3个广播信道和37个数据信道。如果报文在某个广播信道传输，则为广播报文；反之，则为数据报文。

**7.2.2白化**

频移键控( FSK)接收机的一个有趣之处在于，其接收连续帽同比特的能力很差（频移键控接收机的更多详情可参阅第5章）。比如，当发送了一串如“000000000000”的比特序列时，接收机会认为发射机的中心频率向左移动了，进而导致频率失锁。之后的比特“1”会被错过，报文的接收也会因此而失败。为避免这一情况，低功耗蓝牙使用了“白化器”来随机化发进的数据。

白化器是一个很短的输出“0”、“1”序列的随机数发生器，对于给定的报文，“0”、“1”序列的顺序是确定的（见表7-1）。接收机可以使用相同的随机数发生器来恢复原始的比特。为了在输出序列中保留原始信息，需要对原始数据和白化序列逐位进行异或操作得到输出序列。

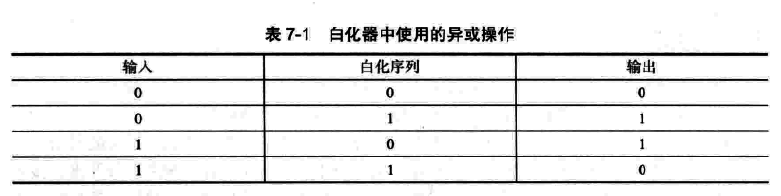


表7-1 白化器中使用的异或操作

通过使用随机白化器，原始信息中的一殷比特会被转化成一殷高度随机化的序列。这降低了接收机频率失链的概率。如果原始信息已经是随机的，进一步的随机化一般也没有什么坏处。

白化随机序列是通过线性反馈移位寄存器生成的，类似于计算CRC。生成多项式如式（7-1）所示。

（7-1）

式中，Whitener表示白化序列；x代表移位寄存器。

移位寄存器的初始值由报文传输所在的链路层信道号决定，最高位置为1。这意味着即使一个报文白化后在某个信道上引起了接收机失锁，当它在其他信道上传输时，由于使用了不同的白化序列，接收机还是能够正确接收的。尽管这是非常罕见的现象，白化器还是能够解决这种问题的。

表7-2给出了白化器如何防止发送连“O”或连“1”。下面是23号信道上一段数据的前三十字节被白化的例子。原始数据为：

输入 = 00000000:00100000:00010000

白化序列为：

白化序列 = 11110101:01000010:11011110

合成之后为：

输出 = 11110101:01100010:11001110

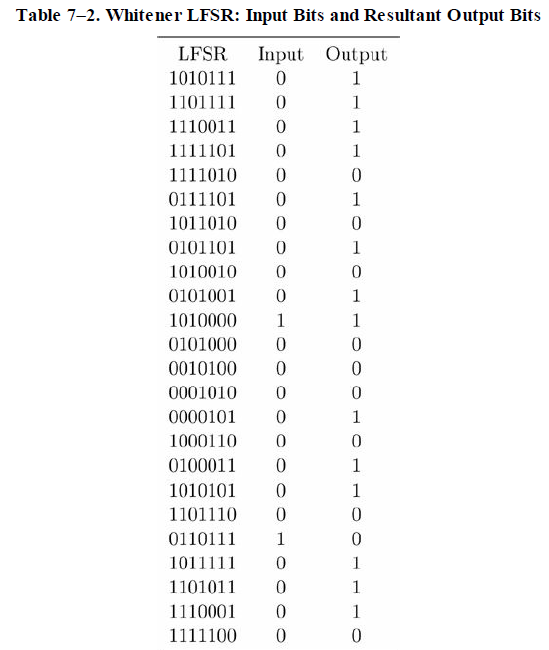


表7-2白化线性移位寄存器示例：输入比特及对应的输出比特

可以看到，原始数据被白化器随机化了。原始输入数据有很长的连续“O”，连续相同的比特数分别为10，l，8，l，4。而输出序列中没有很长的连续相同比特，最长的连续相同比特仅为4个比特，而连续相同的比特数分别为：4，1，1，l，1，1，2，3，1，1，2，2，3，1。

**7.3报文结构**

如图7-9所示，报文结构由一些域构成。各个域将在后续的小节中详细描述。有些域包含多个字节．因此，传输时的字节序以及各个字节中的比特序均有必要说明。

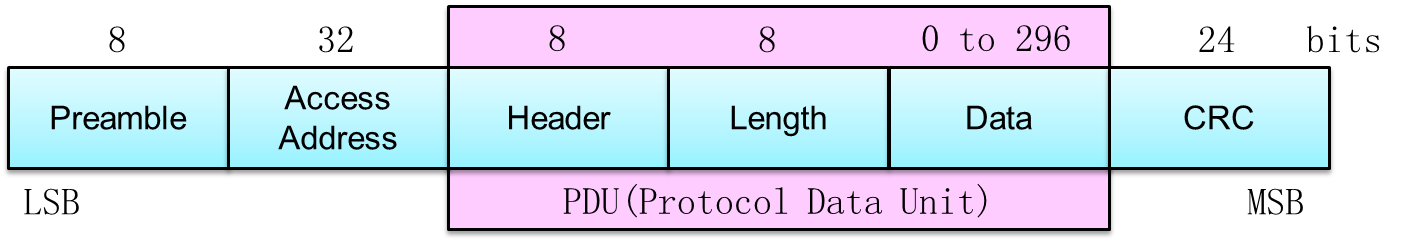


图7-9报文结构

**7.3.1 比特序与字节**

报文是一比特一比特传输的，但他们同时也是由数据的字节组成的。当数据的各个字节传输时，总是从最低位开始。因此，Ox80是按照00000001发送的，而Ox01是按照1OOOOOOO发送的。大多数多字节域是从低字节开始发送。因此．Ox010203会按照如下顺序发送：

1100 0000 0100 0000 1000 0000

**7.3.2前导**

报文最开始的8比特是01010101或者10101010序列。这是很简单的交替序列。接收机可以用它来配置自动增益控制，以及确定“0”、“1”比特所使用的频率。

这段序列之所以非常重要，是因为芯片必须能够应对输入信号强度的可能范围。接收机需要应付-1OdBm ～-90dBm的信号强度，也就是80dB的动态范围。从接收机的角度讲，也就是1pW - O.1mW的能量。自动增益控制器必须检测出输入信号的能量等级，并调整增益，使得信号刚好处于接收机能够轻松工作的范围之中。

**7.3.3接入地址**

接入地址的第一个比特决定了前导是Ol010101还是10101010。如果接^地址的第一个比特为“0”，则使用01010101序列；如果是“l”，则使用10101010序列。这保证了任一报文的前9个比特都是交替的，即要么为101010101．要么为010101010。

报文接下来的32比特是接人地址。有两种类型：

* 广播接入地址
* 数据接人地址

广播接人地址在广播数据，或是广播、扫描、发起连接时使用。数据接人地址在连接建立之后的两个设备间使用。

当控制器试图接收一个报文时，它要事先知道待接收报文的接入地址。接收机开启并调谐到正确的频率，就可以收到数据。即使附近没人发送，接收机还是会收到背景辐射。考虑接收到纯随机噪声的概率，接收一段噪声与前导序列相符的可能性相对还是比较高的。通常，如果低功耗蓝牙接收机一直开着，每隔几分钟就能收到一个假的前导序列。因此，就需要利用接入地址以减少随机噪声造成伪报文接收的概率。

链路层也不知道其他设备什么时候会发送报文，因此只能保留最近40us接收到的比特，并在新的比特移人到寄存器的时候检查序列是否满足前导和接入地址。这一过程称为与接入地址求相关。

对于广播信道，接人地址是固定值Ox8E89BED6。发送时的二进制形式如下（从左到右）：

0110 1011 0111 1101 1001 0001 0111 0001

这意味着广播报文的前导是01010101。之所以选择这个值作为接人地址，是因为其相关特性非常好。该定值使得任何低功耗蓝牙设备能通过与该值求相关获知它正在接收广播报文，即使之前可能从未从该设备处接受到任何报文。

对于数据信道，接人地址是一个随机值，不同的连接有不同的值。这一随机值也需要符合一些规定，主要是为了保证接人地址有足够好的白化特性。

正如7.2.2节关于白化的解释，对无线传输的数据进行白化是有必要的，这保证了接收机可以实现得足够简单。接人地址最基本的规则是不能有超过6个连“O”或连“1”。此外，数据信道接人地址与广播信道接人地址至少要有1个比特的不同。数据信道接人地址也不能有任何重复的模式，各个八位组必须互不相同。数据信道接入地址也不能有超过24次的比特翻转，防止使用逐比特交替的序列。最后一点，最后6比特需要有至少两次比特翻转，这样可以确保报头开始前有一些比特翻转，以防报头有较长的连“。”或连“l”而带来不好的影响。

符合以上规则的有效随机接人地址大概有231个。换句话说，20亿个低功耗蓝牙设备可以在互相可通信的范围内同时工作。这个设计可能有点大材小用，但请记住低功耗蓝牙已经是成功的谩计。数据信道的醢机接^地址还有一个有用的特性，就是攻击者无法根据接^地址来确定某个连接中是哪两个设备。

**7.3.4报头**

报文的下一个部分是报头。报头的内容取决于该报文是广播报文还是数据报文。

对于广播报文（见图7-10），报头包含了广播报文的类型以及一些标记位，这些标记位指出了报文使用的是公共地址还是随机地址。广播报文类型共有7种，每种类型都具有不同的净荷格式及行为：

* ADV\_IND——General advertising indication通用广播指示
* ADV\_DIRECT\_IND——Direct connection indication定向连接指示
* ADV\_NONCONN\_IND——Nonconnectable indication不可连接指示
* ADV\_SCAN\_IND——Scannable indication可扫描指示
* SCAN\_REQ——Active scanning request主动扫描请求
* SCAN\_RSP——Active scanning response主动扫描响应
* CONNECT\_REQ——Connection request连接请求

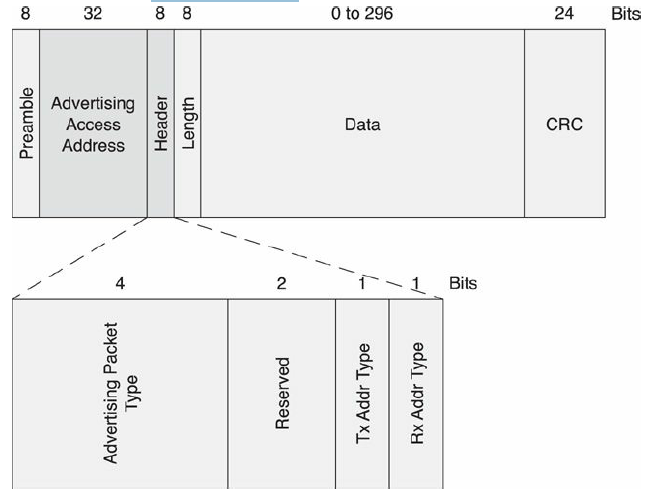


图7-10

图7-11给出了数据报文的报头，包含如下标记位：报文可靠传输使能、低功耗管理、净荷路由（发给控制器或是主机）。图7-11给出了教据报文的报头，包含如下标记位：报文可靠传辅使能、低功耗管理、净荷路由（发给控制器或是主机）。

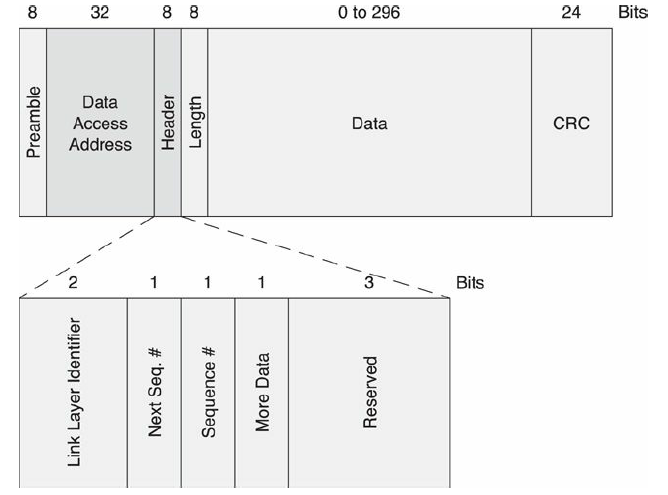


图7-11

**7.3.5长度**

对于广播报文，长度域包含6个比特，有效值的范围是6-37。对于数据报文，长度域包括5个比特，有效值的范围是。0-31。长度域之后是净荷，其长度是长度域指出的字节数。

广播报文和散据报文的长度域有所不同可能显得有些奇怪。这一设计的主要原因是，广播报文除了最多**31字节的数据**之外，还要包含**6字节的广播设备地址**。6个字节加31个字节导致报文长度最多为37个字节（296 bit），因此需要6比特的长度域。

数据报文就简单多了。数据报文长度不是那么关键，多数待传的数据只有几个字节，因此达到最大长度的报文很少用到。这里要注意的是，如果报文被加密，则需要包含**4字节的消息完整性检查域MIC**，实际的净荷数据将被减少到最多**27字节**。为了使得链路层设计得到简化，未加密报文的净荷也不允许超过27个字节的限制，以减低链路层缓存的复杂度。

**7.3.6净荷**

净荷是所传输的“真实”数据，可以是关于设备的广播数据，或者发给一定区域内所有设备的服务数据；可以是主动扫描响应的附加数据，如设备名称，实现的服务；可以是建立或保持连接所需要的信息；可以是从一个设备到另一个设备的应用层数据。

**7.3．7循环冗余校验**

报文的最后是3个字节的CRC（循环冗余校验）。CRC对报头、长度域以及净荷域进行计算。24位的CRC强大到足以检测所有奇数位错误，以及所有2位或4位错误。速意味着低功耗蓝牙可阻检测出所有的1比特．2比特，3比特，4比特，5比特，7比特，9比特等错误。

考虑到多数无线系统采用16位或32位CRC，选择24位CRC显得有点奇怪。不过，考虑到低功耗蓝牙的报文长度，32位CRC也无法检测所有6位错误并且需要浪费额外8us。功耗，因此并不比24位CRC更好。如果报头、长度域以及净荷域的总长度增加，超过39字节，就有必要增加CRC的长度以检测4位错误。对于336比特的总载荷，16位CRC无法检测4位错误，对于低功耗蓝牙而言就不够强壮。因此，选择24位CRC是在鲁棒性和功耗间的权衡。

该24位CRC的生成多项式见式(7-2)。

（7-2）

**7.4信道**

如第5章第6节提到的，低功耗蓝牙使用40个信道。因为调制指数放宽，低功耗蓝牙的信道与经典蓝牙有所不同。这意味着每个信道的功率谱更宽，因此，为避免邻近信道干扰，低功耗蓝牙的信道宽度为2MHz．而不是经典蓝牙的IMHz。

在链路层，这些信道被分为两种：广播信道和数据信道。信道类型对应于之前提到的广播报文和数据报文。如果报文在广播信道传输，则为广播报文；反之，则为数据报文。

如图7-12所示，低功耗蓝牙共有3个广播信道和37个数据信道。3个广播信道分散在ISM频段的不同区域，如果它们集中在某个频段，则可能因为这个频段的深度衰落而造成所有广播无法进行。因此，各个广播信遭之间至少相差了24MHz。

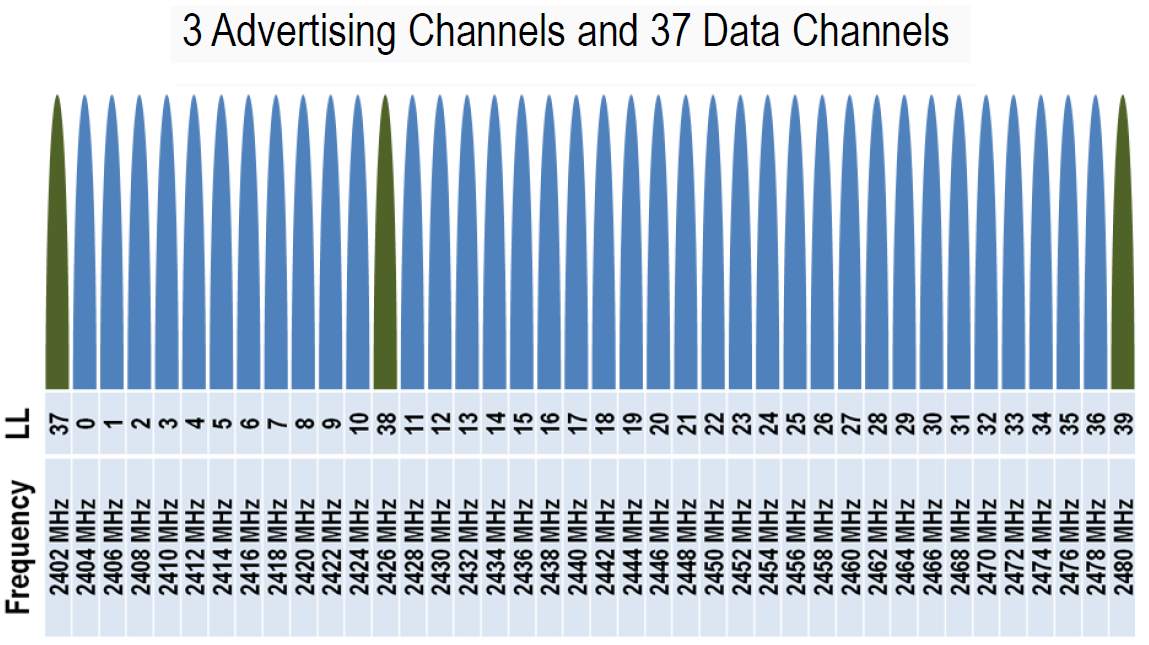


图7-12链路层信道映射

广播信道颇点的选择原则上是为了远离诸如Wi-Fi接入点的严重干扰。这些接入点通常选择802.11信道中的3个：信道l、6和11。这几个信道的中心频率分别为2412MHz、2437MHz和2462MHz，宽度大概为20MHz。这意味着信道1占据2402MHz～2422MHz，信道6占据2427MHz～2447MHz，信道11占据2452MHz～2472MHz。

广播信道位于2402MHz、2426MHz和2480MHz。也就是说，第1个广播信道低于Wi-Fi的信道1，第2个广播信遭位于Wi-Fi的信道I和信道6之间，第3个信道高于Wi-Fi的信道11。如图7-13所示，这3个Wi-Fi信遭覆盖了数据信遵的0～8、11～ 20以及32～34。不过，3个广播信道37、38、39在这些干扰之外。

数据信道在广播信道之间分布，间隔为2MHz。图7-12给出了广播信道和数据信道的完接列表，包括链路层信道编号以及中心频率。

广播信道的编号是37 - 39。数据信道的编号是从O- 36。用这样的编号方式区分广播信道和数据信遭使得跳频算法的实现非常简单。

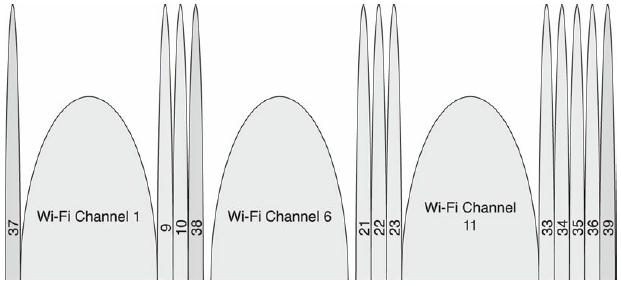


图7-13链路层信道与Wi-Fi信道共存

**7.4.1跳频**

跳频算法用于数据连接中。数据信道共有37个，是一个质数，因此跳频算法非常简单，如式(7-3)所示：

(7-3)

hop是一个5～16范围内的值，每次跳频之后中心频率加hop并模37。这意味着无论hop的值是多少，每个频率均具有相同的优先级。图7-14给出了hop值为13，中心频率随时间变化的例子。此外，算法实现时，也可以先加hop值，如果大于36，则减去37。这样一来，就不需要除法、乘法或者其他复杂的数学运算了。

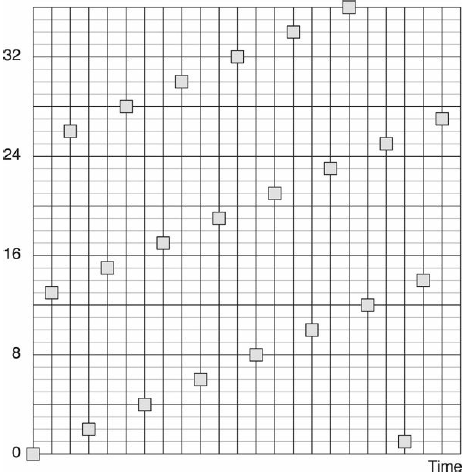


图7-14数据信道随时间的跳频规律

**7.4.2自适应跳频**

自适应跳频能够将一个已知的坏信遭映射到一个已知的好信道，从而减少其他设备对数据报文传进的干扰。为实现这一点，连接中的两个设备都要记录好、坏信道的映射关系。如果如式（7-3）所示的信道是一个好信道，则使用该信道；如果是一个坏信道，则会重新映射到好信道的集合中，如图7-15所示。主设备需要至少将两个数据信道标记为好信道。

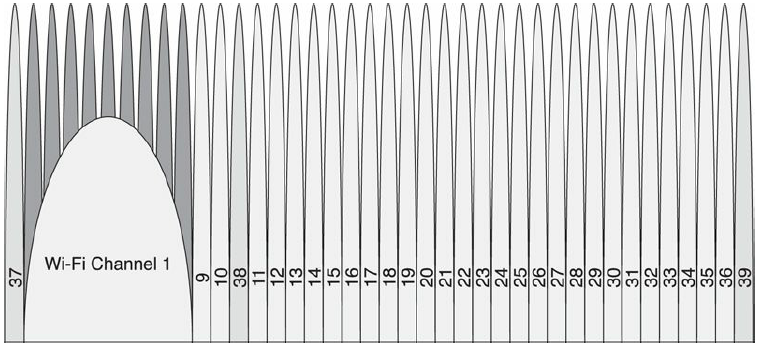


图7-15链路层自适应跳频，以应对Wi-Fi的1号信道导致的坏信道

举个例子，假设一个低功耗蓝牙设各处于使用Wi-Fi 的l信道连续通信的Wi-Fi接入点和设备的干扰范围之内。低功耗蓝牙设备会将编号为。0～8的链路层数据信道标记为坏信道。这意味着当两个设备通信时，它们会循环切换这些坏信道，并将其映射到好信道之上，如表7-4所示。

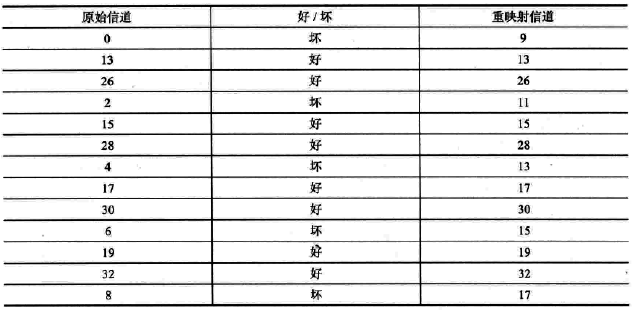


表7-4自适应调频信道重映射示倒

信道的重映射保证了低功耗蓝牙面对严重干扰时依然能够传输数据。同时，这一机制也能让设备迅速应对新的干扰。对于经典蓝牙，大多数控制器能够在几秒钟之内应对一个新的干扰，之后两者就可以很容易地共存，没有任何问题。

为了辅助重映射，主机可以告知控制器当前信道的情况。这一信息可以直接来自该设备的其他干扰无线电收发器，或者来自其他更加非本地的地方。大多数蓝牙控制器能够进行被动式频段扫描，在无需主机任何帮助的情况下，确定干扰的范围和程度。

**7.5设备发现**

低功耗蓝牙设备通过广播信道发现其他设备。一个设备进行广播，而另一个设备进行扫描。共有4种不同类型的广播：general通用的、directed定向的、onconnectable不可连接的以及discoverable可发现的。

如图7-17设备每次广播时，会在3个广播信道上发送相同的报文。这些报文被称为一个advertising event广播事件。除了定向报文以外，其他广播事件均可以选择20ms～10.28s不等的间隔，称为advertising interval广播间隔。主机可以控制该间隔也就是图7-17中advInterval的值。

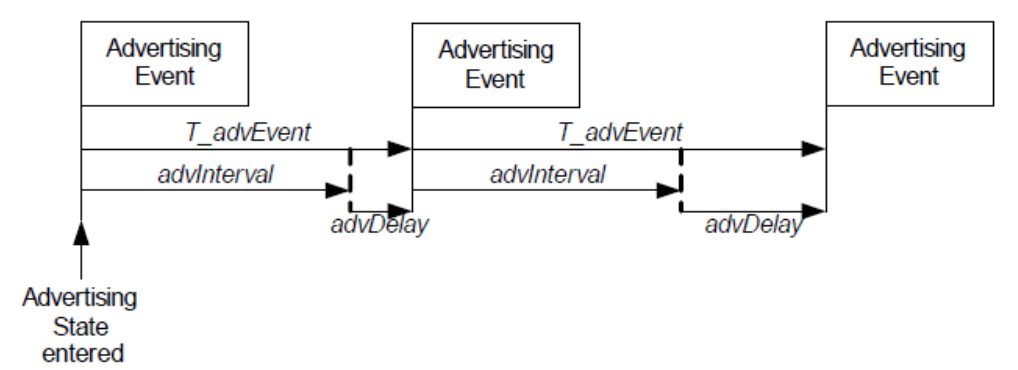


图7-17

但是，设备周期性的发送广播会有一个问题：由于设备间的时钟会不同程度的漂移，两个设备可能在很长一段时间同时广播而造成千扰。为防止选一情况的发生，除定向广播之外的其他广播事件，发送时间均会被扰动。实现该扰动的方式为，在上一次广播事件发生后加入0～1Oms的随机延时如图7-17中的advDelay。这意味着，即使两个设备广播间隔相同，并在相同的信道及时间点上发送造成了冲突，它们发送下一个广播事件时也很可能不再冲突。

扫描是低功耗蓝牙广播的重要组成部分。扫描用于接收广播事件。扫描时间取决于有多少时间用于扫描以及带要多快发现其他设备。例如，如果用户点击屏幕搜索设备，扫描可以持续几秒钟，就可以将一定区域内正在广播的设备一网打尽。

但是，如果用户是在走来走去，扫描设备也许只能每秒钟扫描几个毫秒；或者根据用户是否到家，是否坐在咖啡馆里，或者是否走进某个会议室，每分钟扫描几百毫秒，用来搜索有用的信息。这种背景扫描可以根据地点改变设备的行为。比如：在咖啡馆则将手机自动静音；在家时则将手机的来电转接到家庭电话系统中；在会议室时则将来电转接到语音信箱，并给呼人者发送一条短信，告诉他你正在会议中，不能被打扰。

**7.5.1通用广播**

通用广播是用途最广的广播方式。进行通用广播的设备能够被扫描设备扫描到，或者在接收到连接请求时作为从设备进入一个连接。通用广播可以在段有连接的情况下发出，换句话说，没有主从设备之分。

**7.5.2定向广播**

有时候，设备间需要快速建立连接。如果从设备想这么做，就需要进行广播。定向广播事件就是为了尽可能快的建立连接。这种报文包含两个地址：广播者的地址和发起者的地址。发起设备收到发绐自己的定向广播报文后，可以立即发送连接请求作为回应。

定向广播事件也有特殊的时序要求。完整的广播事件必须每3.75ms重复一次。这一要求使得扫描设备只需扫描3.75ms便可以收到定向广播设备的消息。

如此快的发送会让报文充斥着广播信道，进而导致该区域内的其他设备无法进行广播。因此，定向广播不可以持续1.28s以上的时间。如果主机没有主动要求停止，或者连接没有建立，控制器都会自动停止广播。一旦到了1.28s，主机便只能使用间隔长得多的通用广播，让其他设备来连接。

当使用定向广播时，设备不能被主动扫描。此外，定向广播报文的净荷中也不能带有其他附加数据。该净荷只能包含两个必需的地址，别无其他。

**7.5.3不可连接广播**

不想被连接的设备使用不可连接广播事件。这种广播的典型应用包括设备只想广播数据，而不想被扫描或者连接。速也是唯一可用于只有发射机而没有接收机设备的广播类型。

不可连接广播设备不会进入连接态，因此，它只能根据主机的要求在广播态和就绪态之间切换。

**7.5.4可发现广播**

最后一种广播事件是可发现广播。这种广播不能用于发起连接，但允许其他设备扫描该广播设备。这意味着该设备可以被发现，既可以广播数据，又可以响应扫描，但不能建立连接。这是一种适用于广播数据的广播形式，动态数据可以包含于广播数据之中，而静态数据可以包含于扫描响应数据之中。

可发现广播不会进入连接态，而只能在停止后回到就绪态。

**7.6广播**

如上一节所述，设备可以进行广播。但是，一个广播设备必须在广播中包含一些有用的数据。这意味着可以通过4种广播事件中的3种进行广播：通用广播、不可连接广播以及可发现广播。

进行广播时，需要在广播报文中给数据打上标签。之所以要这么做，是因为并非所有设备都能理解所有可能的广播数据。因此，需要给广播数据打上标签并指出其长度。每个数据片段均起始于一个长度域，用以指示后面的类型及数据域的长度。接下来是类型域，接收机可根据其内容判断自己是否能够理解后面的数据（详见第12章12.5节）。通过使用“长度：类型：数据”这样的格式，无法理解特定数据类型的设备可以根据该数据片段的长度跳过该片段，从而不影响对后续数据片段的解析。

广播数据能够被附近的任何被动或主动扫描设备接收到。广播数据的接收无法被确认。广播设备无法知道是否有设备接收到它的数据，或者是否有设备试图侦听它的数据。也就是说，必须认为广播是一种不可靠的操作。

**7.7建立连接**

应付比广播更为复杂的数据传输，或者要在设备之间实现可靠的数据交付，这些都要依赖于连接。连接使用数据信道在两个设备之间可靠地发送信息。它采取了自适应跳频增强鲁棒性，同时使用了非常低的占空比，尽可能地降低功率消耗。

设备创建连接的过程如图7-18所示。简言之，设备首先广播可连接广播事件，其他设备收到之后即可发起连接。在此过程中，广播者发送的事件类型要么是通用广播事件，要么是直接广播事件。发起者收到正确的广播报文时，将向广播者发送一个连接请求，其中包括了连接开始时需要的所有信息，包括下列各项：

* 连接中使用的接入地址。Access Address to be used in the connection
* CRC初始值。CRC initialization value
* 发送窗口大小。Transmit window size
* 发进窗口偏移。Transmit window offset
* 连接间隔。Connection interval
* 从设备延迟。Slave latency
* 监控超时。Supervision timeout
* 自适应跳频信道图。Adaptive frequency-hopping channel map
* 跳频算法增量。Frequency-hop algorithm increment
* 休眠时钟精度。Sleep clock accuracy

一旦收到或发出连接请求报文，设备即建立了连接，数据交换随之开始。

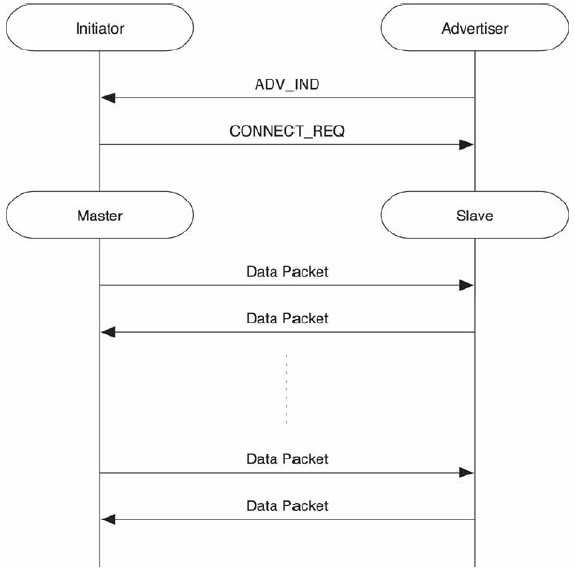


图7-18

**7.7.1接入地址**

连接使用的接人地址总是由主设备来提供。地址通过随机生成，但是需要遵循一些规则，详见第7.3.3节。如果主设备有多个从设备，它会为每个从设备选择不同的随机接入地址。地址的随机性确保了在不同的主从设备之间发生的碰撞概率会很低。随机性也增强了隐私，扫描者无法得知是哪两个设备正在通信。

**7.7.2 CRC初始化**

CRC初始值是另一个由主设备提供的随机数。随机的意义在于，如果在同一区域有两个主设备正在和不同从设备通信，那么使用相同的接人地址的概率将会很低。如果确实发生了这种情况，从设备会从错误的主设备收到干扰数据包。因此，为每个从设备设置随机的CRC初始值，主从设备既有相同的接人地址又有相同的CRC初始值的概率就会变得微乎其微。

**7.7.3发送窗口**

从设备可以自行决定广播的时间，它们是最需要保存电量的设备，这样的设计并无不妥。但是，如果主设备已经在做其他事情，也许是一些更重要的事情，那么它必须让低功耗蓝牙错开当前的活动。在连接建立期间，有两个参数用来传递上述信息：窗口大小和窗口间隔。

**当连接请求数据包发送完毕，存在一个1.25ms的强制时延**，紧接着是发送窗口偏移和发送窗口。发射窗口偏移可以是0到连线间隔connection interval之间的任意值，但必须为1.25ms的整数倍。从发送窗口开始，从设备打开其接收器，并等待来自主设备的数据包。如果到发送窗口结束时仍未收到数据包，从设备终止侦听，并会在一个连接的间隔后再次尝试。

关于连接过程，最有趣的一个地方是，一旦发送了连接请求，主设备便认为自身有了连接；该连接已经创建(created)，但还不能算是完全确立(established)。当从设备收到连接请求时，它也认为自己已经处在连接之中；连接已经创建，但不能证明完全确立。

出于效率的考虑，连接一经创建，主机就会立即收到通知。连接可能不会成功，从设备也许收不到连接请求，或者两个设备的距离很远。这些情况如果发生，连接失败的概率将变得很高。然而，由于主机收到了连接已创建的通知，它可以开始将数据发给控制器从而为首个无线数据包的传输做好准备，并且节约时间和能量。因为第一个数据包的发送至少会在1.25ms延迟之后，主机堆栈应该有足够的时间向控制器提供数据，以便在第一时间发送出去。这种强制性的延迟为设备提供了喘息的机会：在繁重的广播规程与连接正式确立之间，电池获得了宝贵的恢复时间。

只有收到了数据包确认，连接才能视为正式确立。确立（established）不改变连接的工作方式，但它改变了链路监控超时的时间——从之前的**6个连接间隔**，变为了在连接请求中设定的值。这样一来，如果连接不能迅速确立的话将会立即终止。

**7.7.4连接事件**

在一个连接当中，主设备会在每个连接事件里向从设备发送数据包。一个连接事件是指主设备和从设备之间相互发送数据包的过程。连接事件的进行始终位于一个频率，每个数据包会在上个数据包发完之后等待**150ps**再发送。

连接间隔决定了主设备与从设备的交互间隔；它是指两个连续的连接事件开始处的时间距离，可以是7.5ms ～4s内的任意值，但必须为1.25ms的整数倍。要确定从设备与主设备的实际交互间隔，需要用到从设备延迟这一参数，它实际上是一个连接间隔的倍数，代表从设备在必须侦听之前可以忽略多少个连接事件。应当指出，从设备延迟必须短于监控超时时间。

如图7-19所示，连接事件被一个个的连接间隔分开。从主设备发送数据包开始，每个连接事件可以持续进行，直至主设备或从设备停止晌应。在连接事件之外，主从设备之间不发送任何数据包。

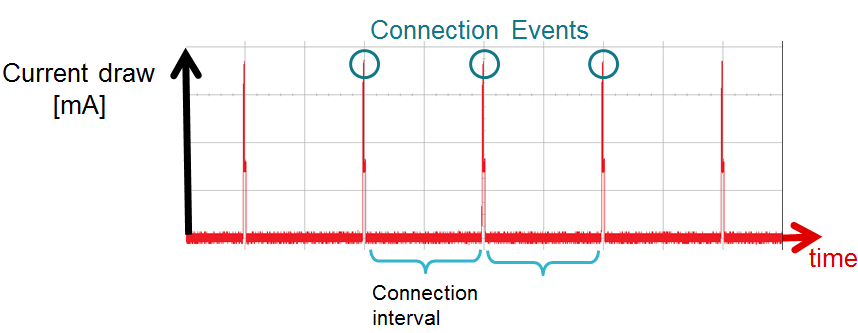


图7-19连接事件

如图7-20举个例子，如果连接间隔为1OOms，从设备延迟是9．那么从设备可以忽略9个连接事件，但不得不侦听第10个连接事件。换言之，从设备必须每秒侦听一次，而此时监控超时的最小值应为1010ms。反过来，另一个极端的例子是，如果监控超时使用了32s的最大值，对于间隔为1OOms的链路，从设备延时必须小于等于319。

虽然如此，如果将从设备延迟设为可行的最大值，在监控超时发生前从设备只能获得唯一一次侦听主设备的机会，这可不是一个好主意。因此，建议至少给从设备留出6次侦听的机会。在前面的例子中，如果连接间隔为iOOms，从设备延迟为9．那么监控超时应该至少为6s，这样一来，链路在最终断开前从设备至少会有6次侦听的机会。

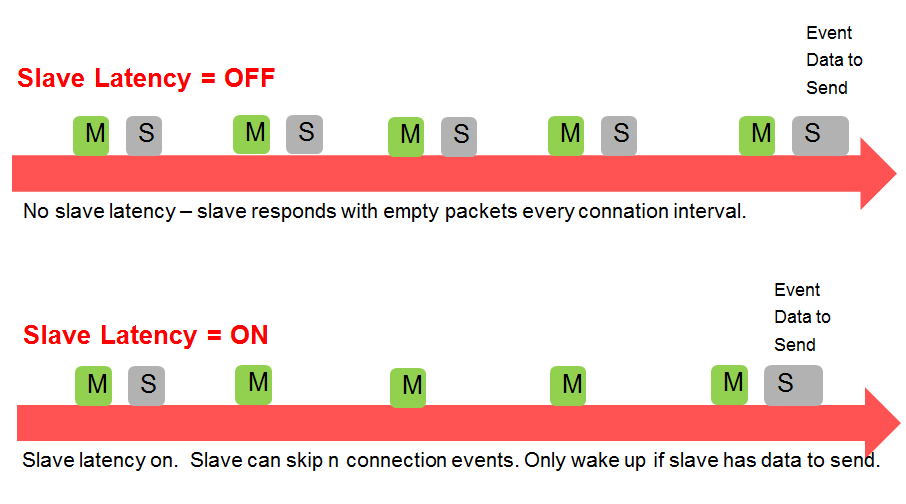


图7-20从设备延迟

**7.5.5信道图**

自适应跳频信道图是数据信道的位掩码，用来标记信道的好坏。由于共有37个数据信道，信道图的长度于是设为37位。如果某一位被设置为l，表明该信道良好，可用于数据通信；如果该位为0，表明该信道很糟，不可用于数据通信。

跳频算法的跳数值是在5 ～16之间的一个随机数。跳数值被用在自适应重映射之前的跳频算法中，详见第7.4.1节中关于跳频的介绍。显然，跳数值不能为0，否则频率永远不会改变。

非常低的跳数也是不可取的，因为大多数的干扰一般会占据好几兆带宽，使用非常小的跳数值将不能快速地将传输带离干扰源，可能造成持续干扰。同样的逻辑也适用于17或更高的值。试想，如果增量为17，由于跳频算法中要对37个信道取模(参见公式(7-3))，每两个频率之间的差距将仅仅只有3个信道。

**7.7.6睡眠时钟精度**

最后，主设备发往从设备的信息中还包括睡眠时钟精度值。该值定义了时钟能够保证的精度范围。如果时钟由晶振提供，晶振有一个根据温度变化的精度范围，比如在室温下为20ppm，在0℃或85℃时为50ppm。如此一来，设备可以声明其时钟精度达到50ppm。

时钟精度可以帮助从设备消除连接事件的不确定性。如果从设备与主设备的不同步时间达到1s．而这两个设备的定时精度均为500ppm，那么，将合并的不确定性——1OOOppm乘以上述时间，就得到一个1ms的不确定窗口。也就是说，从设备必须提旱1ms醒来，并在这额外的1ms内保持侦听，以防主设备和从设备的时钟在不同的方向发生了最大漂移。

更精确的时钟可以降低功耗。还是以使用晶振的设备为例，如果两个设备的时钟精度分别为50ppm和150ppm，二者合并后的精度只有200ppm。一秒钟之后，从设备只需要提前200us醒来并保持侦听。如果设备被唤醒的频率不高，它们的可用工作时间将5倍于两个使用500ppm晶振的设备。因此，如果对降低功耗有特别的要求，建议在设备中使用高精度的晶振。

**7.8发送数据**

在一个连接里，设备通过在连接事件中发送数据包来相互通信。数据报文和广播报文不同，前者是单独进行的通信，后者则是对所有的侦听设备发送广播。广播报文和数据报文之间的最大差别在于报头的格式和净荷的长度。

数据报文的净荷的可以为0～31字节不等。净荷长度为零的数据包是一个空包，它不含应用程序的数据，但仍然可以在报文的头部包含一些信息。净荷的最大长度(**31字节**) 小于广播报文的最大长度。通常只有加密报文才会使用31字节的长度。无论链路层加密与否，传给控制器的未加密数据包最多只能携带**27字节**的数据。

**7.8.1数据报头**

数据报文的报头如图7-20所示，包含下面四个字段：

* + 逻辑链路标识符（LLID），2 bit
  + 序列号（SN）,1 bit
  + 下一个预期序列号（NESN）,1 bit
  + 更多数据（MD）,1 bit

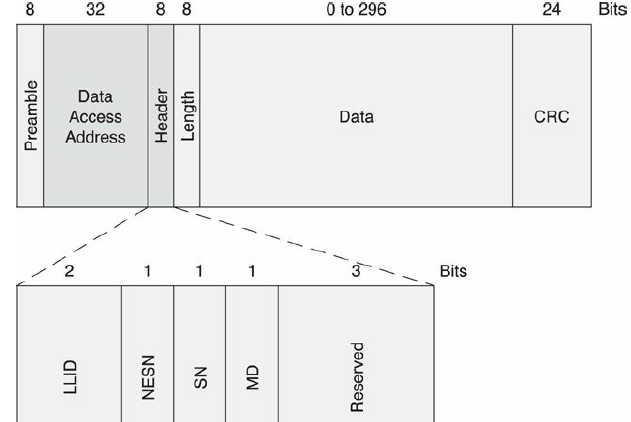


图7-20数据报文的报头

我们看不到某个位用来表示“数据包已加密”，因为它属于连接的模式属性，就好像自适应跳频或连接事件间隔。

**7.8.2逻辑链路标识符**

逻辑链路标识符( LLID)用于判断数据报文属于下列哪种类型：

* 链路层控制报文(11)——用于管理连接
* 高层报文开始(10) ——也可用于一个完整报文
* 高层报文延续(01)

如果数据包为链路层控制报文，那么其逻辑链路标识符将设为11，并被直接交给链路层控制实体，由后者来判断该数据包内的数据的具体古义，如第7.10节所述。

所有其他的数据包都要经过主机——要么来自主机，要么发给主机。主机能够发送大于27字节的数据，但由于无法放入单个链路层的数据包，因此必须支持分段。具体做法是把数据包贴上“高层报文开始”或“高层报文继续”的标签。图7-21给出了一个例子，一个很长的高层数据包被拆分为三个链路层的数据包：第一个数据包标有“高层报文开始”的LLID．而其他两个标有“高层报文延续”的LLID。

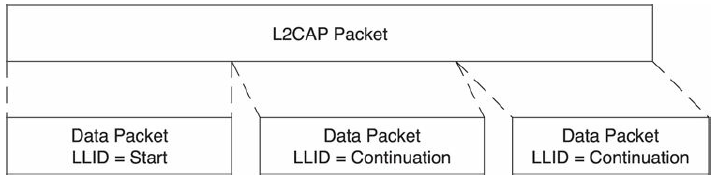


图7.21数据报文的报头

这种做法带来了两个有趣的结果。首先，在收到开始报文时，链路层并不需要知道整个报文的最终大小。在发送新的开始报文之前，通常是开始包、延续包……延续包的发送方式。可见在消息开始时，连续消息的数量是不固定的。

第二个有趣的结果在于：总是可以发送长度为零的延续包，它们对高层的数据没有任何影响。这意昧着可以随时使用长度为零的延续包对消息进行确认。我们将这些长度为零的延续包称为空包。

**7.8.3序列号**

为了使数掘传输变得可靠，所有的数据包均带有序列号。连接建立后，第一个数据包的序号为0；每次发送新的数据包时，其序列号与上个数据包的序列号不同。这使得接收装置能够判断接收的数据包的性质：如果序列号与之前的一样，则为重传报文，如果序列号和之前的不同，则为新报文。

在数据包中，用一个比特来表示序列号；在发送的第一个数据包中将该位设为0，接下来每当设备发送新的数据包时，该值在1和0之间交替。

**7.8.4确认**

数据包的确认需要用到另一个比特，即所谓的下一个预期序列号( NESN)。NESN的发送方用其通知对方自己期望接收的**对方的数据包的序列号**。

如果设备成功接收序列号为0的报文，在其确认报文中，应将下一个预期序列号设为l，否则序列号为O的数据包将被重传。因此，可以将NESN作为一个标志来判断数据包是被正确接收还是需要重传。图7.22说明了这一点。

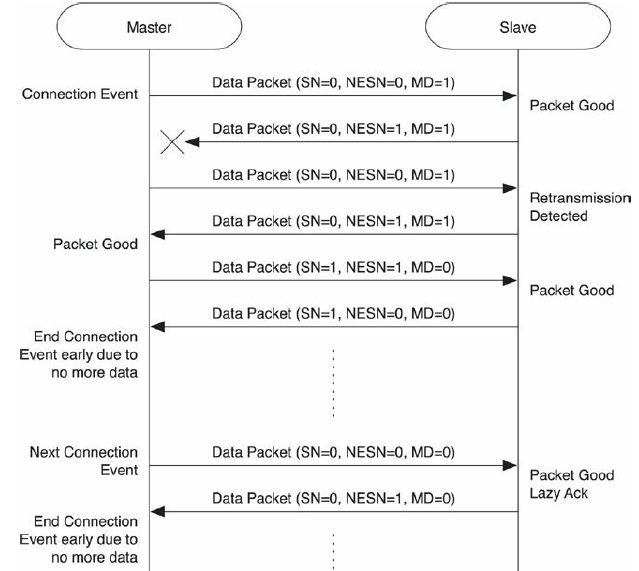


图7-22序列号

**7.8.5更多数据**

数据信道报文的报头里还有最后一位——更多数据位，用来通知对端设备自己还有其他的数据准备发送。如果收到设置了更多数据位的数据包，应该在当前连接事件中继续与对端设备通信。这样一来，只要还有数据要发送，连接事件会自动扩展；一旦不再有散据发送，连接事件会迅速关闭。如果把更多数据位设为。，设备可以快速、优雅地结束连接事件，从而节省能量。因此，更多数据位可以提供一种非常有效的方式，用尽可能少的报文来实现大量数据的可靠传输。

**7.8.6使用序列号和更多数据的例子**

1. 图7-22讲述了一个关于序列号、下—个预期序列号和更多数据位的例子，说明如下：主设备发送第一个数据包，默认的序列号为0，下一个预期序列号也为0;同时由于有两个数据包要发送，更多数据位设为1（，）。从设备正确地接收了该数据，更新下一个预期序列号（）。
2. 更新了下一个预期序列号（）之后，从设备发送它的第一个数据包（）。因为从设备还有一些数据要发送，该数据包也对更多数据位进行了设置（）。然而，主设备没有收到这个包，所以主设备的下一个预期的序列号没有改变。从设备还将继续侦听主设备，因为二者都设置了更多数据位。
3. 主设备再次发送第一个数据包（，），这是因为主设备没有收到从设备的数据包，因此必须重传上一个包。从设备接收该数据包后发现序列号与上个相同，判断其为重传报文，于是不更新下一个预期序列号。由于看到主设备还要发送更多数据，从设备在当前连接事件里继续发送其他报文。
4. 从设备重新发送它的第一个数据包（，）。主设备成功接收丁该报文，于是更新它的下一个预期的序列号（）。主设备发现从设备还有更多的数据等待发送，于是发送其他报文继续该连接事件。
5. 主设备第三次传输的是一个新的数据包，使用了一个新的序列号（，）。由于该数据包含有所有的剩余数据，主设备把更多数据位设置为0。从设备在成功接收该数据包后更新它的下一个期望序列号（）。此时，由于从设备仍然有数据耍发送，因此将继续连接事件。
6. 从设备第三次传输的是一个新的数据包，使用了一个新的序列号（，）。从设备将更多数据位设为0，表明没有其他的数据需要发送。主设备在正确接收该数据包后更新它的下一个期望序列号（）。因为主从设备的最后一个数据包都表示没有更多的数据，连接事件随即关闭。
7. 一段时间后，主设备在下一个连接事件醒来，向从设备发送新的数据包，包含默认的新序列号和最后更新的下一个预期的序列号（，）。这个包同时确认了从设备的上个数据包。从设备成功接收了该数据包，并更新它的下一个期望的序列号（）。虽然没有要发送的数据，从设备仍然发送一个空包以做出响应。
8. 从设备第四次传输的是一个空的数据包，使用一个新的序列号（，）。由于没有更多的数据要发送，从设备将其更多数据位设为O。主机在成功地接收该数据后更新它的下一个预期序列号（）。因为主设备的最后一个数据包都表示没有更多的数据，连接事件随即关闭。

正如前面的例子中所述，序列号和下一个期望序列号总是处于互锁的状态，确保了可靠的数据送选。CRC校验值用于检验数据包，如果内容校验失败，数据包不算正确接收。

另外，通过下一个期望序列号还可以进行流量控制。设备一旦没有足够的缓冲区空间来处理消息，可以不更新下一个期望序列号。这将迫使对端设备重新发送当前消息，从而把对缓存的要求从接收设备转嫁到发进设备。

对于连接事件而言还有一些需要注意的事项。如果由于一些位的错误导致了CRC校验失败，数据包接收出错，这种情况在同一连接事件里一旦发生两次，设备将立刻停止当前事件，并在下次连接事件中重新同步和尝试传输。这样一来，如果某信道因干扰产生了拥塞，两个设备很快能发现干扰，并停止使用该信道。下一个连接事件到来时将更换新的信道，干扰随即减轻，数据又可以快速地传给对方。

**7.9加密**

在连接时，可以对净荷中的数据进行加密，确保数据的机密性，从而抵御攻击者。机密性是指第三方“攻击者”由于没有加密链路的共享秘密，因此无法拦截、破译或读取消息的原始内容。

加密数据包括一个消息完整性校验值，表明该数据包已经过认证。为了验明发送方的有效身份，认证使用共享密钥为已加密的数据计算签名，可防止第三方篡改数据包中的任何内容。通过认证，消息的接收方能够确信收到的数据包来自一个可信设备。一个关于认证的典型例子是银行卡持有人采用个人识别码（PIN）进行认证；PIN码在这里能够验证银行卡的使用者是否即为授权人。

加密数据包还包含一个数据包计数器，用来防止重放攻击。重放攻击是这样一种攻击方式：攻击者截取一个既有的消息，随后再次发送该消息以期望收到响应。试想，假如没有重放攻击保护．攻击者有可能扫描到某个设备的大量数据包，然后再次发送这些数据包，看看有什么“有趣”的结果。如果接收设备是一个城市公园附近的排污阀，其结果可能不堪设想。显然，防止重放攻击是一件非常重要的事情。

**7.9.1 AES**

低功耗蓝牙中的所有加密和认证都基于同一个加密引擎，称为高级加密系统( AES)。AES最初源自美国的一项政府计划，试图寻找未来可用的加密引擎。一直以来．AES被用于许多有线和无线标准，迄今为止安全研究人员还没有找到其算法的弱点。

AES可以有多种形式，取央于在给定的时间内能够处理的数据块以及密钥的大小。低功耗蓝牙使用128位的密钥和128位的数据块。也就是说，所有密钥的长度均为128位，每次加密生成的密文长度为16个字节。

AES加密块非常简单，它包含两个输入和一个输出。两个输入分别为128位的密钥值和128位的纯文本数据块，输出则为128位的加密数据块。密钥和纯文本在使用上有一些不同：纯文本可以直接为加密块使用，但密钥必须经过处理后才能使用。可见，更有效率的方式是只设立一个密钥，用于不同的纯文本块以进行快速加密，而非使用不同的密钥为每个块加密。

于是，执行算法（E）并利用密钥(key)对明文(plaintext)加密以生产密文的过程可以用函数（7-4）表示。ciphertext表示密文，plaintext表示明文。

(7-4)

在低功耗蓝牙里．AES加密引擎被用于下列四个基本功能：

* 加密净荷数据
* 计算消息完整性校验值
* 数据签名
* 生成私有地址

数据签名在安全管理器中定义，生成私有地址在通用访问规范中定义。

**7.9.2加密净荷数据**

要加密净荷数据，首先应将净分割成16字节的块，对每个块生成一个密文比特流，再与纯文本执行异或运算。上述过程遵循IETF RFC 3610标准，其中定义丁一种加密和认证的方法——计数器密码区块链信息认证码模式（CCM）。这种标准的加密方法适用于任意长度的密钥或任意长度的明文消息。

低功耗蓝牙采用了加密块——一种具有已知格式、经过了初始化的明文块。加密块中包含一个随机数nonce，由包计数器、方向位和一个初始化向量(IV)构成。注意，在下面的公式里符号||表示串联。

nonce表示随机数；Packet Counter表示包计数器；Direction表示方向位；IV表示初始化向量。

包计数器是一个39位的数，在启用加密时，包计数器总是从零开始，每发送一个新的非空数据包就增加1。空包是不加密的，因此不必为其增加包计数器。初始化向量是一个长为64伍的随机数，加密链路两端的设备分别提供了其中的32位（参见第7.10.3节）。总之，包含39位计数器、l位方向和64位初始化向量的nonce总长度为13个字节。

的其他字节由CCM规范决定。第一个字节是一个标志字段，始终为0x0l，表示这是一个且，块。最后两个字节是块计数器，在加密净荷的前16个字节（）时，块计数器应设为0x0001，在加密净荷的后11个字节（）时，块计数器应设为Ox0002。CCM规范也可用于对消息的完整性校验值（）进行加密，此时，块计数器应设为0x0000。

随后，加密块与消息的各个部分进行异或操作，生成加密的净荷。

式中encrypted表示加密的净荷；Blockl表示加密块1；Block2表示加密块2；MIC表示消息完整性校验值。

加密的净荷接下来被发往对端设备。因为对端设备知道共享秘密，即密钥值，它能够使用相同的包计数器、方向位和Ⅳ值对消息进行解密。当发送加密的净荷时，将针对密文净荷而非明文净荷计算CRC。数据包的报头和长度字段不会被加密。

**7.9.3消息完整性校验**

消息完整性校验(MIC)用于验证数据包的发送者。如图7-23所示．MIC位于数据和CRC之间，确保加密的数据包来自于对端设备而非第三方攻击者。计算MIC时会再次用到AES加密引擎。此时，一个块的输出将用于下一个块的输入，各个块串联起来，确保原始消息中的每一位都被用于MIC的计算。

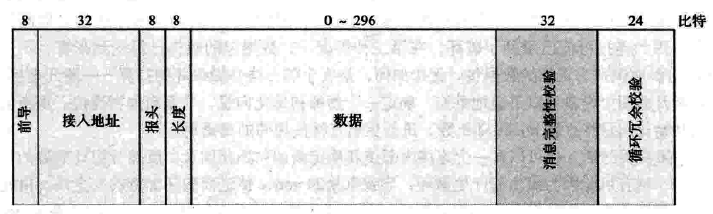


图7-23加密的报文格式

计算MIC总共需要三到四个数据块。第一数据块包含nonce和验证数据的原始长度，这里的nonce与加密净荷时用到的值完全一样。长度字段的大小为16位，尽管低功耗蓝牙中用于验证的净荷最长不超过27字节。

接下来的数据块包含了一些额外的数据．它们应该与净荷一样进行验证，但又不包含在净荷内。在低功耗蓝牙中，这些额外数据指的是一些报头中的位，确切地说是最需要进行身份验证的逻辑链路标识符。计算时，报头里的其他位用0屏蔽，这样做不仅简化了计算，而且在尚未获得SN或NESN的情况下还可提前进行计算。从安全的角度来看，这些被屏蔽的位实际上不是太重要。

接下来的一到两个块包含验证的净荷数据。包含0～15字节的净荷数据，包含16～26字节的净荷数据。

最终计算MIC时，使用曾经为净荷加密的密钥对这些块逐层进行连接和加密。注意，这里MIC只会用到最终加密数据的32位最高有救位(MSB)。

接收加密的数据包时，在接收设备上执行相同的MIC计算过程，检查计算值是否与报文携带的MIC值相同。如果二者不符，接收设备将立即断开连接，停止进一步通信，面对端设备最终会自动进人监控超时。这种方法乍看上去似乎有些激进，不过由于CRC的存在，上述情况的发生概率往往比较低；如果CRC校验失败就不会进行we检查，接收方将拒收该数据包，并且等待对端的重传。

导致MIC校验失败的可能性只有两种：一是有攻击者正在试图攻击连接，二是出现了一些位错误，但是CRC校验没有检测出来。在第一种情况下，最安全的办法是立即断开链路，因为连接可能已遭到了破坏。在第二种情况下，数据包的数据已经受到损害，不过CRC错误地识别为正确的数据包。无论如何，最安全的方法是做最坏的打算——断开连接。连接断开后两个设备可以迅速地重连，确定一个新的初始化向量，并重新加密连接。重新确定的初始化向量将更新nonce随机数，进而更新连接使用的加密密钥。

如果需要的话，也可以在一个连接内部重新确定新的初始化向量。既然分组计数器的长度一定，就有可能因为溢出而产生循环，导致重复的nonce值进而破坏加密的安全性。由此看来，定期更新初始化向量还是有必要的，只不过更新并不会频繁进行，这是因为在nonce重复之前，共有个包可以发送，即使不断发送数据包也需要至少12年才能绕完一圈。话说回来，低功耗蓝牙的成功之处正是在于该技术能让设备稳定地连接很多年。出于这种考虑，它规定了一个nonce的更新过程，配合加密暂停和继续规程重新确立新的初始化向量，相关细节详见第7.10.4节。

**7.10管理连接**

两个设备一旦连接，便可以收发数据和管理连接。连接管理涉及在链路层发送控制消息，包含下列七个链路屡控制规程：

* 更新连接参数。Updating the connection parameters
* 改变自适应跳频信道图。Changing the adaptive frequency-hopping channel map
* 加密链路。Encrypting the link
* 重加密链路。Reencrypting the link
* 交换功能位。Exchanging feature bits
* 交换版本信息。Exchanging version information
* 终止链路。Terminating the link

**7.10.1连接参数更新**

连接建立时，主设备通过连接请求数据包发送连接参数，详见第7.7节。当连接活跃了一段时间，连接参数也许不再适用于当前使用的服务。出于提高效率的目的，连接参数需要进行更新。较之首先断开连接、接着更换新参散重新连接，还有一种在链路中更新参数的更为简单的途径，如图7-24所示。

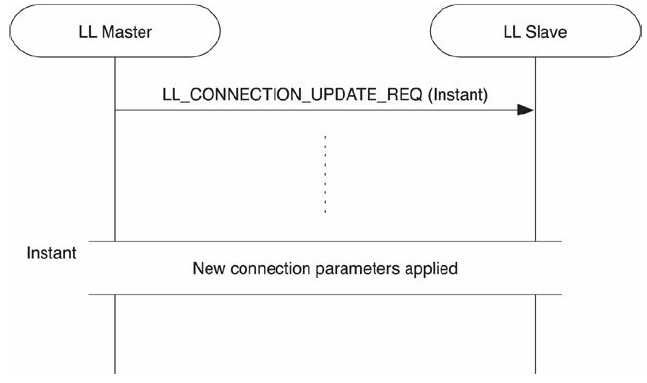


图7-24执行连接参数更新规程

为此，主设备向从设备发送连接更新请求，即LL CONNECTION\_UPDATE\_REQ，当中携带了新的参数。这些参数不必进行协商，从设备或者接受和使用它们，或者断开链路。连接更新请求中包含了早先创建连接时用过的一部分参数，还有一个称为瞬时(inatant)的新参数：

* 传输窗口大小。Transmit window size
* 传输窗口偏移量。Transmit window offset
* 连接间隔。Connection interval
* 从设备延迟。Slave latency
* 监控超时。Supervision timeout
* 瞬时。Instant

瞬时参数决定了连接更新的开始时刻。发送消息时，主设备为连接更新选定一个未来的时间点，并且放在消息中。接到消息后，从设备会记住这个未来的时刻，届时再切换至新的连接参数。这有助于解决无线系统里的一个最大问题——报丈重传。只要数据包的重传次数足够，并最终在瞬时之前传输成功，上述过程执行起来就不会有问题。但是，如果该数据包届时没能宪成传输，链路就有可能丢失。

由于低功耗蓝牙没有时钟，要决定瞬时时刻只有依靠计算连接事件的个数。因此，每一个连接事件都会被计数，链路上的第一个连接事件，也就是在连接请求之后的位于首个传输窗口里的连接事件记为0。因此，瞬时实际上是一个连接事件的计数器，相应的连接事件到来时就使用新的参数。为了让从设备收到数据包，主设备必须为其提供足够的机会。不过从设备延迟是多少，都应该至少保证六次数据发送机会。也就是说，如果从设备延迟为500ms，那么瞬时通常被设定在3s之后的某个未来时刻。

瞬时到来时，从设备开始侦听发送窗口，就好像连接建立的过程那样。主设备能够调整从设备的计时，总体而言不超过1.25ms。不过，由于主设备可能还是一个经典蓝牙设备，上连调整使其得以协调低功耗蓝牙从设备，从而更好地完成调度。一旦该过程结束，新的连接间隔、监控超时、从设备延迟值将投入使用。

**7.10.2自适应跳频**

在一个开放的无线频段中，自适应跳频对于任何无线电技术的成功生存是非常重要的。不幸的是，某些技术没有采用自适应跳频，因此容易受到干扰。自适应跳顿，尤其是在一些典型设备中，存在一个很大的挑战：任何时刻，好的信道集合和差的信道集合都在不断地变化。也就是说，设备必须能通过信令更改当前的信道图，相关规程如图7-25所示。

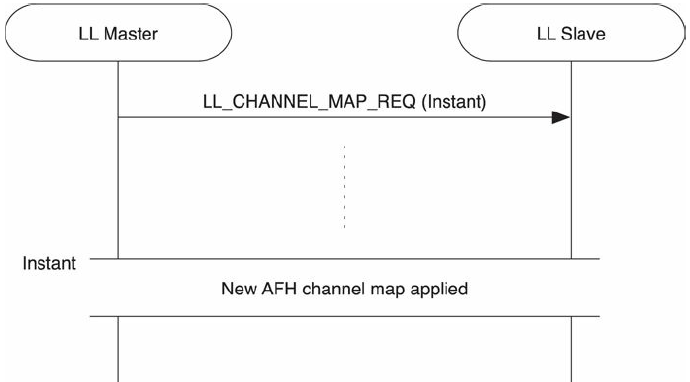


图7-25信道图更新规程

主设备通过信道图请求报文LL\_CHANNEL\_MAP\_REQ向从设备发送自适应跳频更新信息，其中含有下面两个参数：

* 新的信道图。New channel map
* 瞬时。Instant

这里的瞬时和连接更新中提到的瞬时概念完全一样。它确定了一个时间点，在该时间点之后的连接事件将使用新的信道图。更新后的信道图可以一直使用下去，直到再一次更新信道图。

信道图是一个37位的字段，每一位代表一个数据信道。如果字段中的某个信道位被置为1．则认为信道良好，允许使用；反之．如果字段中的某十信道位被置为0，则认为信道很差，无法使用。

要重发信道更新请求，必须等待瞬时经过以后。这限制了信道图的更新频率。通常情况下．只有主机发现之前的坏信道有所好转，或者当前信道集导致连接变差时才会执行信道图的更新。链路层控制规程不允许从设备改变信道图，甚至不能向主设备告知自身的信道条件。

**7.10.3启动加密**

只有尚未加密的链路才能启动加密。链路在加密时，需要创建一个随机数(nonce)和一个会话密钥session key（SK）。随机数需要每个设备分别提供4个字节的信息，会话密钥要求每个设备分别提供8个字节的信息。此外，还需要一个长期密钥long-term key（ITK）作为配对时的共享秘密（更多信息请参阅第11章11.2节）。

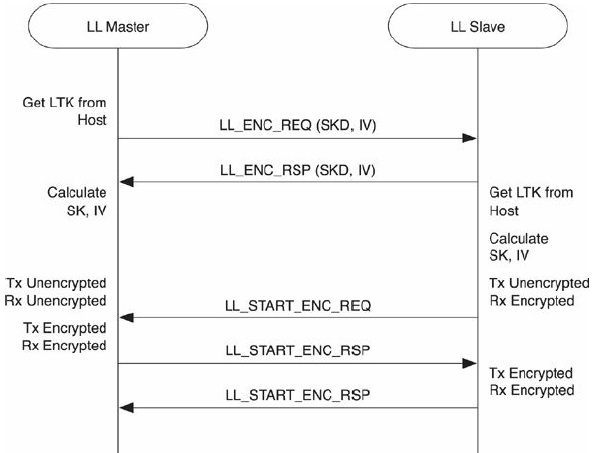


图7-26启动加密规程

启动加密的操作如图7-26所示。主设备首先向从设备发送加密请求消息（LL\_ENC\_REQ）, 从设备回复加密响应消息 （LL\_ENC\_RSP）。主设备的LL\_ENC\_REQ报文包括其贡献的4字节的初始化向量（initializatiou vector），8个字节的会话密钥分散器（session key diversifier），以及一些在首次配对时由从设备提供的信息。对主设备来说，这些信息是静态的，从设备可以用此信息来确定正在与哪个主设备通信，并可能根据这些信息获得该主设备的LTK。这样一来，从设备完全不必保存任何与之绑定的设备的信息。在从设备的加密响应报文中，包括了其贡献的4字节初始化向量以及8字节会话密钥分散器。

如果从设备侧没有LTK，它将会立刻向主设备发送拒绝加密指示并附上拒绝的原因。如果有LTK．从设备将启动三次握手进行加密。此时从设备既要向主设备发送未加密的数据，又要准备接下来接收主设备的加密数据。因此，三次握手是必需的，只有这样两个设备才能以相同的步调最终实现链路加密。

加密开始时，使用一个会话密钥来进行链路加密。会话密钥的生成将由LTK和来自两个设备的会话密钥分散器来决定。

会话密钥分散器的存在使得LTK能够被多次使用。其原理在于，每个连接使用了一个不同的、由会话密钥分散器生成的密钥。主设备和从设备各自贡献分散器的一半内容，即使其中—个设备是攻击者，另一个设备也可以采用不同的分散器，从而得到不同的加密密钥。这背后的主要原因是为了保护AES的单侧的脆弱性，每个密钥决不能使用第二遍。因此，即使已经拥有了共同的密钥LTK，我们却不能用它来加密应用数据；相反，我们必须将LTK分散，变成一个个的会话密钥SK。

会话密钥分散器( SKD)包括两部分，由主设备提供，由从设备提供。二者串联在一起，作为明文输入AES加密引擎(参见公式(7.4))；同时，LTK作为密钥输入AES加密引擎。引擎的输出便是会话密钥，用来进行链路加密。

初始化向量(IV)同样来自两个设备，由主设备提供，由从设备提供，二者串联在一起。

在加密启动时所有的数据传输将会暂停。对主设备来说，暂停发生在发送加密请求报文之前；对从设备来说，暂停发生在发送加密响应报文之前。这确保了启动加密之后不会再有未加密的数据发送，也有助于三向握手得以正确地执行。

从设备一旦算得SK和IV．将以未加密的方式发送开始加密请求（LL\_START\_ENC\_REQ），同时准备用SK和IV接收加密的数据包。假如主设备没有收到该请求，则响应一个空报文再次请求该数据包。这是因为其他的所有数据包都被暂停发送，而空报文永远不能被加密（没有可加密的净荷）．因此从设备只能接收空报文或是主设备接下来的加密数据包。

收到来自从设各的请求，主设备用通过计算得到的相同的SK和IV值加密数据包LL\_START\_RSP，此数据包为已加密的数据包。并将其发给从设备，随后将自己设置为接收加密数据包。由于从设备之前已配置了加密包的接收，它能够接收该密文，之后启动自身的加密传输。

收到主设备的加密数据包后，从设备发送加过密的LL\_START\_RSP作为回应。因为主设备已经配置了加密包的接收，它能够接收该密文。

主设备一旦收到了该数据包，它会打开应用程序数据流，让所有的数据都执行加密。从设备如果从链路层得到了数据包的确认，也会打开应用程序数据流，让所有的数据都执行加密。

经典蓝牙的认证必须在加密之前进行，这种方式不但耗时，而且要发送额外消息。低功耗蓝牙为每个传输的数据包携带了身份验证信息，因此无需执行这一过程。换言之，低功耗蓝牙对链路的认证是通过每一个数据包来进行的，而不仅仅在加密过程的开始处执行一次。

**7.10.4重启加密**

重新启动加密有助于刷新会话密钥，一般发生在包计数器即将过期，或者主机选择了新的链路密钥，并且想要用它来获得新的会话密钥的情况下。重启加密的情况比较少见，事实上，一些设备的连接很少会持续一段较长的时间，所以没有必要重新启动加密。

重启加密的过程和启动加密类似，但是首先要暂停加密。暂停意味着停止应用数据的发送以及关闭加密功能，从而避免发送未经加密的应用数据。重启加密的过程实际上分为两步：主设备必须先暂停加密，然后利用启动加密规程重新启用。

暂停加密是另—个三次握手的过程，只不过跟启动加密时三次握手的顺序恰好相反。

如图7-27所示，在暂停应用程序数据后，主设备向从设备发送一个暂停加密请求报文（LL\_PAUSE\_ENC\_REQ）。

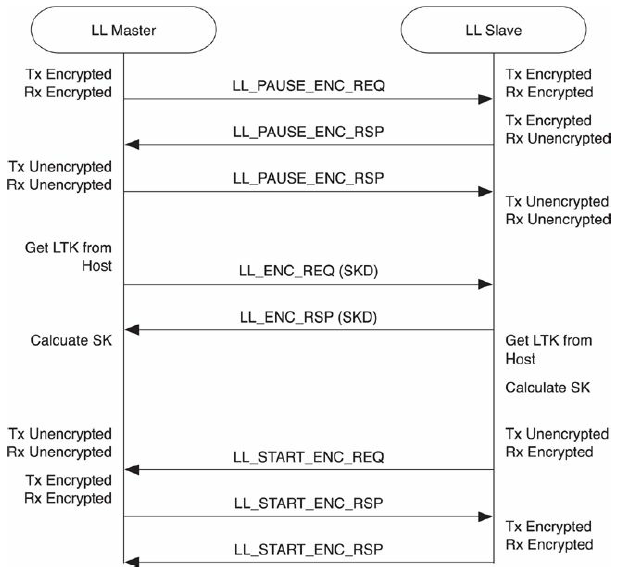


图7-27

收到主设备的暂停加密请求后，从设备将暂停其应用数据，然后发送一个暂停加密响应报文( LL\_PAUSE ENC\_RSP)并停止接收加密的数据包。这个报文仍然是密文，但它已经准备好接收明文了；主设备对任何来自从设备的数据包只能回复空包进行确认。

收到从设备的暂停加密响应后，主设备关闭加密数据包的发送和接收，以明文的形式回复另一个暂停加密响应报文（LL\_PAUSE\_ENC\_RSP）。从设备因为已经禁用了加密数据的接收，因此可以接收该报文。收到之后，从设备也将禁用加密数据包的传输。

至此，加密已被完全禁用，此时应用数据已经暂停发送，双方都不允许发送任何数据，从而防止了敏感的应用数据未经加密便被发送出去。

一旦加密被禁用，主设备将立即向从设备发送加密请求（LL\_ENC\_REQ）发起启动加密规程，如上一节所述。

**7.10.5版本交换**

有时候出于调试的目的，有必要查明设备地址之外的一些信息，比如设备主机和应用层的一些有用信息。版本信息仅用于调试目的，设备的链路层能够自动获取，也可以通过主机请求获得该信息，如图7-28所示。版本信息不能用于改变设备的行为，因此救有必要在每一次建立连接时交换它们。然而，大多数设备会每隔10个（或更多）连接请求一次版本信息，这样嗅探器可以获得该信息，并帮助调试连接。

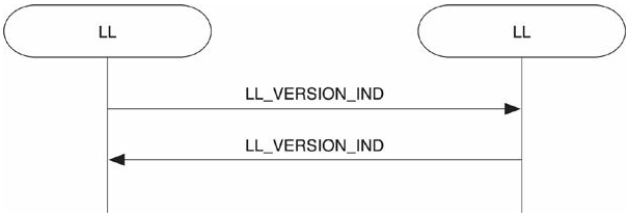


图7-28

链路层这样的一种自主行为多少会让一些人觉得不舒服。为什么非要每10个连接接收一次版本信息呢？简单来说，当你在调试两个设备时碰到了问题，你极有可能无法再通过设备的主机执行版本交换。因此，为了描述这些设备，偶尔进行一下版本信息的交换至少要比什么都没有强一些。

进里介绍一个经典的例子，某设备只与另一个设备进行了配对（因而没有其他设备与之连接，没有谁知道它的版本信息），随后的几个月里该设备一直正常工作，不过最后还是出现了一些毛病。链路层坚持每隔几个连接自动交换一次版本信息，这在过去的几个月里或许意义不大，但此刻将为问题的解央提供极大的帮助。

版本信息包括：

* 版本号
* 公司标识符
* 子版本号

版本号是一个为设备兼容的蓝牙规范版本所分配的编号。

公司标识是蓝牙技术联盟（SIG）为生产这款控制器的公司所指定的号码。

子版本号是由控制器的生产商直接指定的一串数字，每一次实现或修订控制器，子版本号都应相应地改变。如何指定数字并没有具体的规定，如果设备的调试非常重要，那么应直接联络设备制造商。设备制造商的联系方式通常在UnPlugFest事件里，工程师们在大规模出货前会测试和验证这些控制器。相关的联系信息也可以在每件货运产品的合格证数据库中找到。

要交换版本信息，任何一个设备都可以发出版本信息指示报文，只要当前连接还未进行过该操作。如果设备从对端设备收到版本信息指示报文，并且未曾发送过自己的消息，那么它将回应版本信息指示数据报文。上述过程一但完成，不允许再次执行该规程。由于版本信息是静态的，在一次连接中将不会改变，所以再次询问相同的静态信息其实毫无意义。连接处于活动状态的情况下，控制器应当把该信息保存在高速缓存里，以免主机再次询问。

**7.10.6功能交换**

对端设备利用功能信息来判断本端设备到底能做些什么。功能信息公开了设备支持的一些可选功能。如图7-29所示，主设备通过功能请求报文（LL\_FEATURE\_REQ）询问该信息，这种操作在每个连接里最多只能执行一次。收到主设备的请求，从设备回复功能响应报文（LL\_FEATURE\_RSP）。要使用可选的功能，不必进一步交换信息，因为报文中的所有功能都必须是支持的功能。同时，对于不支持的可选功能，从设备自有方法能够拒绝该请求。例如，如果不支持加密，可以返回一个拒绝指示报文，安全地拒绝加密请求。

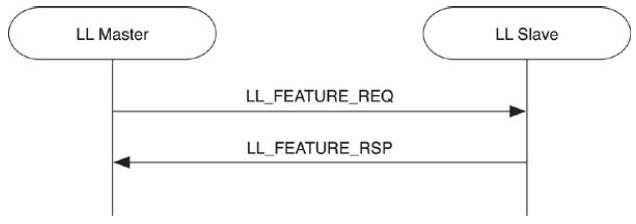


图7-29功能交换规程

低功耗蓝牙的第一版只定义了一个功能位——支持加密，用来表示本设备支持加密。

**7.10.7终止连接**

最后一个链路层控制规程为终止规程。毋庸置疑，它是指断开链路并把主从设备从连接态转为就绪态。任何一方可以基于任何原因在任何时间终止链路，如图7-30所示。



图7-30终止规程

要终止链路，设备首先发送一个终止指示报文( LL\_ TERMINATE IND)，等待链路层对该报文进行确认，然后断开连接。如果一个设备发送了终止指示报文，但没有收到此报文的任何确认，发送设备将视为规程超时，仍然会断开莲接。超时的时间与监控超时值相同。

设备一收到终止指示就立即发送一个**空包**来响应该报文，随即断开连接。

连接也可能由下列原因终止：

* 监控超时
* MIC失效

遇到上述原因的时候，不会发送终止指示报文；链路自动断开，双方的主机随即得到通知。

**7.11鲁棒性**

为实现鲁棒性，链路层采用了两个强有力的算法以确保数据不受干扰地传输，以及从本端主机发送到对端主机时不会发生变化。

**7.11.1自适应跳频**

要在2.4GHz ISM频段上进行有效的数据通信，自适应跳频技术是必不可少的。由于有如此多的不同设备使用上述频段，抗干扰技术就显得十分重要。2003年底，自适应跳频首次被引入经典的蓝牙规范，与2003年以前的设备相比，该技术在性能方面带来了质的飞跃。

简单地说，自适应跳频是一种频率跳变的无线电技术，能够屏蔽差的信道并将其重新映射到好的信道。要做到这一点，两个设备都有一张信道映射图，长为37位，每一位分别表示链路层的一个数据信道。信道映射图中的第0位代表链路层的0号信道，第36位代表链路层的36号信道。如果信道位被设为l，表示该信道为一个使用的信道(used channel)；如果信道位被设为0，表示该信道为一个未使用的信道(unused channel)。在信道图中，最少应设置2个使用的信道。

我们来看一个信道映射图的例子。假如所有三个主要的Wi-Fi信道都在使用，它们的中心频率分别为2412MHz、2437MHz和2462MHz。这意味着低功耗蓝牙只有9个链路层信道可以枧为“好”的信道．它们是9、10、21、22、23、33、34、35和36。定义信道映射图需要用到三个值：ChannelMap是在设备间传输的信道图，Used是一个集合，用来标明信道图中好的信道，而numUsed用来表示这些好信道的数量。因此，对于前面的例子，上述各值将分别如下。ChanneIMap表示信道图；Used表示使用的信道；nwnUsed表示使用的信道数。

如果使用该信道图建立连接，主设备会把信道图放在连接请求报文中发往从设备，并将跳数间隔（hop interval）设为7。正如7.4.1节所述，二者将使用式（7-5）所示的跳频公式计算连接的自适应跳频信道：

(7-5)

频率的跳变只有在计算了首个频率之后才会发生。

链路上的第一个连接事件本来要用7号信道。由于它并不属于好的信道集，所以会被重新映射到好信道集中的某个信道。要实现这一点，Used集合会根据信道号的大小进行排序，而未映射的信道则根据好信道的数量，即numUsed被调整到好的信道上。在上述情况下，我们有9个好信道。其中的第七个，即链路层35号信道将被用于首个连接事件。

在下一个连接事件，用上个未映射信道来计算信道：

14通道也是一个未使用的信道，所以还要再次重映射到使用的通道。

在下一个连接事件，仍然是用上个来映射信道来计算信道：

21信道是一个使用的信道，所以这次可以直接使用它。

正如你在表7-5中所看到的，用上述方法可以继续为未来的连接事件服务。



表7-5使用自适应跳频进行信道重映射

即使在该区域有大量的干扰，也总是可以通过使用自适应跳频将来使用的信道重新映射到那些使用的好信道上，保证数据传输正常进行。

**7.11.2强CRC**

在低功耗蓝牙中，CRC值的长度要比经典蓝牙或者其他大部分短距离无线技术50%。经典蓝牙已经证明，16位的CRC对于2.4GHz ISM频段来说强度不够，它在逻辑链路控制和适配协议层（L2CAP）另外加上了16位CRC，这样一来，带有误比特的数据即使通过了控制器的CRC校验．在经过L2CAP层到达主机时，还可能被第二层CRC检测出来。很遗憾，这种做法导致L2CAP数据包变长了许多，对于那些资源有限的设备，复杂的软件计算让CRC的实现也异常麻烦。

从经典蓝牙获得的宝贵经验，告诫低功耗蓝牙在控制器上应采用更强的CRC，最后，低功耗蓝牙选择了24位的CRC值来保护数据包。该CRC能够检测数据包中所有的单比特、2比特、3比特、4比特和5比特错误，以及其他所有的奇数位错误，这远远强于一个16位的CRC校验码。相比大多数的有线协议，比如互联网协议(IP)使用的数据包，它提供了相同的强度。这是因为低功耗蓝牙的报文要短得多，因此，虽然CRC的长度要短一些，但是仍然提供了大致等效的保护性能。

经典蓝牙的另一个问题在于，数据报文的头部用了一个较弱的报头错误校验值来进行保护，即一个8位的CRC。显然，如果该棱验值没有检测出数据错误，主CRC也有可能校验失败。

这方面的经验已经再次说明，用一个单独的报头错误校验和一个主CRC没有太大的帮助。如果该数据包正确无误，那么控制器仍然要运行两次单独的LFSR．一次针对报头，另一次针对净荷。如果数据包已经损坏，在报头中出现错误的概率是比较低的，整个数据包仍然可能会被接收。无论如何，通过使用一个单独的报头错误检查不能达到期望的节能效果。

低功耗蓝牙选择了另一种设计。它仅用了一个强CRC来保护数据包的报头、长度字段、净荷字段，以及加密的消息完整性校验值。这意味着强CRC的所有力量作用在了整个数据包上，只有前导码和接入地址除外——作为接收数据包的前提，二者必须逐个比特进行接收。由此可见，低功耗蓝牙不仅更加强健，而且同样具有效率。

和经典蓝牙使用三个CRC相比，仅有一个CRC的系统更易于实现。这说明了为什么同时考虑所有层的设计是创建一个从上而下、高品质、能够满足市场需求的规范的最佳方法。

**7.12为低功耗优化**

下列章节探讨低功耗蓝牙是如何束实现极低功耗的，主要的方法如下：

* 使用短报文。Keeping the packets short。
* 物理层使用高比特率。Using a high physical bit rate。
* 提供低开销。Providing low overhead。
* 优化响应机制。Optimized acknowledgement scheme。
* 单信道连接事件。Single-channel connection events。
* 亚速率连接事件。Subrating connection events。
* 使用离线加密。Using offline encryption。

虽然功率测量有多种方式，但我们这里只考虑设备内部的两个功率指标，它们对于低功耗而言至关重要。首先，较低的峰值功率(peak-power)是那些使用纽扣电池的低成本设备的关键。高的峰值功率无疑会让电池快速耗尽，从而减少设备的使用时间。其次，较低的每应用比特功率（power-per-application-bit）也是支持设备长时间使用的重要因素，它决定了设备能够发送应用数据的数量。

**7.12.1短报文**

无线技术中有一个十分复杂的部件，即实际使用的无线电收发装置。这些装置采用大规模CMOS技术制造，往往将设计人员推人两难的窘境：一方面要尽一切可能减少成本，另一方面为了无线电的稳定性不得不增加成本，比如增加电路以减少频率漂移。低功耗蓝牙解决了这个问题，数据包的长度被设计得足够小，收发数据导致的热效应因而被降至最低。须知，使芯片发热并不需要很长的报文，在经典蓝牙里，3ms的数据包已经足以引起发热问题。

通过这样一种简单的设计，低功耗蓝牙的设计者对于细节的拿捏可见一斑。他们为了优化链路层规范甚至考虑到了硅制造工艺的物理性质。

如果报文长度不超过几百微秒，就没有校准无线频率或是增加频率稳定电路的必要。用户不必担心在此期间产生的频率漂移量超过规范要求的最大漂移量。在蓝牙低能量中，允许的最长数据包为376us，在如此短的时间里，硅的发热量对于数据传输频率的影响十分有限，不会超出允许的限度。连接中的最大报文长度更小一些，仅为328us，如图7-31所示。

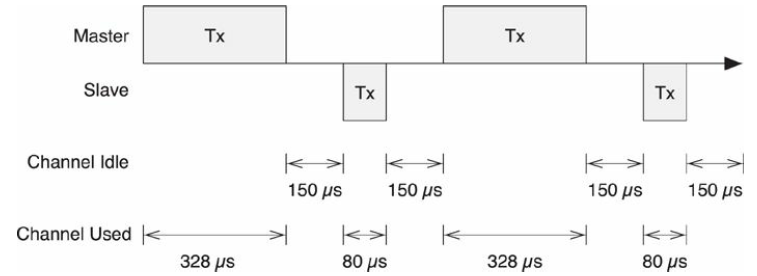


图7-31短报文

因此，使用短报文无需经常校准无线电。该设计减少了用于数据包收发的电路数量，降低了设计的复杂性，使得峰值功率得以降低。

还应当指出的是，发送一个很长的数据包后必须有一个150ps的空隙。该空隙让硅芯片在数据包之间得以降温。因此，无论是先发送后接收，还是先接收后发送，都不必为了数据包的操作而校准频率，从而进一步降低了功耗。上述分析说明，在一个加密链路上，某个传输方向上的最大的数据传输占空比仅为：

58%的占空比在无线技术当中是一个很低的值。经典蓝牙的占空比为72%，而超高速无线技术的占空比通常高达90%。低功耗蓝牙的优化目标是少量、间歇性的数据传输，而非为了尽可能高的数据吞吐量。

**7.12.2高比特率**

当发送数据时，无线电需要大量的电流。大部分电流用于运行一个2.4GHz的振荡器，以便调制无线信号。CMOS运行在极高的频率时需要相当大的电流。由于数字系统中的绝大多数门电路不会随时间改变状态，CMOS为这类门电路进行了优化。因此，对于任何一个运行在2.4GHz、并且使用大规模CMOS工艺制造的无线电而言，其内部消耗的电流产生2.40Hz信号所需的电流相当。

既然振荡器的存在使得能耗难以降低，调制信号的效率变得重要起来。对于给定的数据量，传输的速率越快，无线电的效率就越高。低功耗蓝牙的数据传辖速率为每秒1M比特。

作为对比，如果有一个设备使用了另一种技术，其传输速率只有低功耗蓝牙的四分之一，在传输同样的数据时该设备将会需要四倍的时间，消耗的能量也会是低功耗蓝牙系统的四倍。可见，高比特率是一件好事。

应当意识到，或许我们在数据传输率方面强调的太多了。事实上，使用复杂的调制方案虽然能获得10倍于低功耗蓝牙的速率，但通常也需要更多的能量用于调制和解调；尽管数据可以用更少的时间来完成传输，用于调制的额外能量意味着每比特功率其实相差不大。不过由于时钟漂移的原因，没有数据接收时的电流会显著提高。因此，一个简单的、峰值功耗低、数据传输率高的调制方案是最有效的数据发送方式。这也是低功耗蓝牙的设计师们的最佳选择。

**7.12.3低开销**

鉴于每个比特都有价值，无线电技术中对于开销的考虑非常重视。低功耗蓝牙的开销是指一切不含应用程序数据的部分，包括前导、接入地址、报头、长度、CRC字段以及可进的MIC值。

对干未加密的数据包，衡量其效率时采用应用程序数据的大小与传送所需数据包的总大小之比，范围从29% - 73%，如表7-6所示。数据包越大，无线传输的开销越小。对于加密的数据包，其效率要比未加密时更低，主要是因为每个数据包都要增加额外**4个字节的MIC**。比同类无线电技术相比，这种效率其实已经不错了。ZigBee在任何的加密报文之前都有一个数据包的开销，长度从15字节至31字节不等。考虑到Zigbee的物理比特率比低功耗蓝牙慢四倍，传输一个四字节的应用数据时．Zigbee需要的能量要比低功耗蓝牙高出十倍。



表7-6应用数据的开销

**7.12.4确认机制**

链路层的确认机制有一个有趣的性质，它不要求立即执行数据包的确认。这是经典蓝牙和低功耗蓝牙的一个根本区别。

在经典蓝牙里，接收机在下个传输机会到来之前必须对数据包进行确认。如果没有及时收到确认，接收机必须在发送的下个数据包中携带一个否定的确认。如果在同步链路时从设备试图发送数据，上述确认机制将导致很多同题。

低功耗蓝牙发送的每一个数据包都能确认上一个数据包，哪怕是一段时间之前传输的数据包。这意味着设备再也不必立即发送应答，相反，它可以选择等待，直到有效据需要发送（或因为超时等其他的原因执行传输）。这种快速而有效的执行确认的方法尤其适用于大数据量的传输。

**7.12.5单信道连接事件**

主设备和从设备之间的所有通信都发生在连接事件当中。连接事件包括一个主设备首先发送的数据包，以及随后从设备和主设备交替发送的一系列数据包。在蓝牙V1.1的标准里，每一个数据包——无论来自主设备还是从设备——都会在不同的信道上传输。比如主设备在某个信道上向从设备发送一个数据包，响应报文将通过另一个信道发送。这样傲的问题在于，即使从设备成功地完成接收，发送回复的下一个信道却可能存在干抗，导致发往主设备的确认数据包无法送达。

蓝牙v1.2增加了自适应跳频，从设备的每次传输使用与主设备相同的信道。在前面的例子中，从设备将在主设备传输数据的同一个信道进行传输，数据包成功送达的可能性因而增大许多。上述结论基于这样一种假设，如果主设备发往从设备的数据包完好无损，那么反向传输的数据也将完好无损。这是一个正确的假设，已经在一些干扰严重的场景下提高了实际的数据传输率。

低功耗蓝牙沿用了上述台乎逻辑的结论。既然找到一个好信道，为什么不用它呢。好用的通道当然应该尽可能长时间地利用，基于这个常识，我们有可能最终得到“禁用跳频”的结论。毕竟，如果数据能够送达，就应该一直停留在该频率，直到其不能正常工作再切换到另一个信道。

可惜的是，在需要改变频率那一刻，设备却不能发送任何信令来协调这种改变。因为一旦信道失效又要耗费许多的能量重新进行同步。鉴于干扰的瞬态特性，尤其是Wi-Fi网络中的互联网流量具有突发性，这种模式很容易出现问题。

此外，由于处在同一区域的网络都会自发地选择干净的频率，单信道模型减少了处于相同信道的网络数量。即便干净信道的数量有限，挤在相同信道的网络也不会太多。跳频算法把网络流量在时域和额域空间做了分配，允许大量的网络在同一区域同时工作。而“待在一个频率直至失效”的替代算法却行不通。

低功耗蓝牙采用的方法是，在一个连接事件之内保持一个频率发送数据和确认，然后在下次连接事件到来时换至另一个频率。在任意的时间点上，低功耗蓝牙使用的频率都是完全确定的。如果有哪个信道不工作，两个设备将立即停用该信道，并且在下次连接事件时使用新的信道重新进行同步并继续发送数据（参见图7-32）。这意味着即使在一些充满了未知干扰的地方，比如在许多家庭和企业里，数据流的延迟影响也被减小至最低。

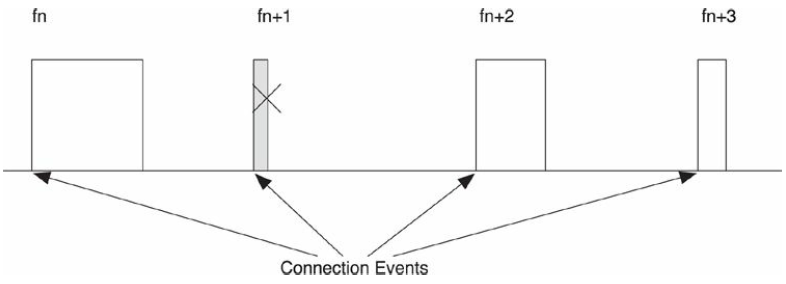


图7-32单信道连接事件

**7.12.6亚速率连接事件**

许多情况下，主设备比从设备拥有更多的资源，就好比笔记本电脑的电池要比无线键盘的电池大得多。电脑可以定期充电，键盘的电池却要用好几年。而且许多设备对延迟有非常苛刻的要求，比如用户的按键操作，其结果必须尽可能快地传输到对端设备。低延时是用户交互的要求，但它和低功耗是一组矛盾，必须设法予以解决。

低功耗要求从设备尽可能少地侦听主设备，而低延迟却要求从设备尽可能多地传输。在一个连接事件里，由于从设备只有在主设备发送数据之后才能获得发送机会，主设备必须不断询问从设备，看它是否有数据要发送，以满足低延迟的要求。然而，如果从设备不得不侦听每一个连接事件，将不可能实现低功耗。

解决的办法是，让从设备忽略大部分的连接事件，如图7-33所示。如果有数据需要发送，从设备可以完成同步并尽快进行数据传输，如此产生的数据延迟变成了主从设备之间的连接事件的间隔。此外，从设备还能够忽略一定数目的连接事件，称为从设备延迟( alave latency)。可忽略的数量决定了从设备能够节省多少能量：错过的连接事件越多，从设备的功耗就越低。

从设备延迟有一定的限度，它既不能大于连接的监控超时时间**supervision timeout**，**又不能让从设备与主设备重新同步的机会少于6个**。举个例子，如果监管超时时间为5s，连接事件的时间间隔为50ms，那么从设备延迟可以为4650ms．以便在延迟结束之后让从设备有6次机会重新和主设备取得同步。选样一来，从设备每4.5s能够执行一次通信，占空比仅为0.0069%；但在通信时，从设备发送数据到主设备的平均时延仅为25ms。

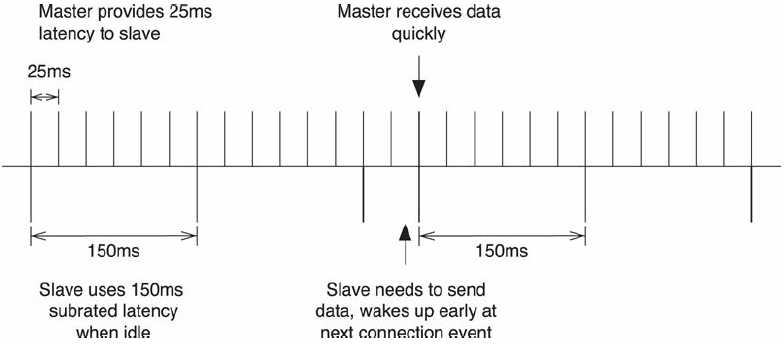


图7-33亚速率连接事件

**7.12.7离线加密**

数据加密通常被认为是一个非常耗能的过程。一个设备不但要计算消息认证码，还要为净荷和消息认证码计算密码块。以最大长度的数据包为倒，数据加密需要对AES-128加密块执行7次迭代。如果上述操作需要实时进行，数据包收发的峰值电流将显著提高。

低功耗蓝牙把数据加密和验证码的计算放在了后台。一个数据包在发进之前就已经完成了加密，而此时无线电仍然是关闭状态。数据加密不依赖于序号、下一个期望序号或者更多数据位。因此，数据加密可以在数据到达链路层之后、被发送之前的任何时间进行。此外，数据可以重发任意多次，即使下一个期望序号或数据位发生了变化，每次重发时加密和认证代码都不会改变。这种做法降低了峰值功耗，消除了重新发送加密报文的成本。

收到加密数据时，设备实时计算其CRC值并且判断数据的正确性。加密的数据可以保存在链路层，直到无线电停止活动并且有剩余的能量时才将其解密。实际上，数据包在被传送到主机之前随时可以完成解密操作，因而降低了接收机的峰值功率。另外，对于重传报文而言，接收时一旦发现重复的序列号，就不必再对报文重新解密。这种做法降低了峰值功耗，消除了重新接收加密报文的成本。

**第8章 主机/控制器接口**

预测未来的最好办法，就是把它创造出来。

——阿伦·凯(Alan Kay)

**8.1介绍**

如图3-1所示，主机控制器接口(HCI)是主机与控制器之间的接口，主要完成两个任务：一个是发送命令给控制器和接收来自控制器的事件，另一个是发送和接收来自对端设备(peer device)的数据。

一般来说，主机接口既是两个设备之间的物理上的接口，也是逻辑上的接口。逻辑接口定义了命令、事件和数据的封包格式。而物理接口则规定了主机和控制器之间如何传输这些数据包。

**8.2物理接口**

在蓝牙规范中，定义了4种物理接口，每一种对应着不同的用途：

* 通用异步收发器(五线UART)提供了最简单的传输方式。
* 3线UART则提供一种优化的可靠的UART传输方式。
* USB为电脑和类似的设备提供高速的、通用的传输方式。
* 安全数字输入输出(SDIO)为消费电子设备提供中等速度的传输方式。

**8.2.1 五线UART**

UART是所有传输协议中最简单的一种。在零调制调解器配置下，为主机和控制器之间定义了一个简单的通用异步收发器（UART）传输规范。零调制调解器是指发送线和接收线在设备两端交叉对接，即控制器的发送端（**TXD**）可以与主机的接收端（**RXD**）相连，反之亦然。3线UART仅有TXD、RXD和**地线**，5线UART则多定义了两条流控线。控制器的请求发送端（**RTS**）与主机的清除发送端（**CTS**）相连，反之亦然。

在低功耗蓝牙中，UART接口一般使用不带奇偶校验位但带有一个停止位的8比特字符。该停止位是在每一个字节末端的额外一位。在蓝牙控制器和主机之间的链路较短的时候，通常很少发生错误，所以不需要用来进行错误检查的奇偶校验位。另外．UART使用硬件控制流的方式来驱动CTS和RTS。每一个HCl数据包都带有下列三种报文类型编码中的一种：

* 命令(command)=Ox01
* 数据(data)=0x02
* 事件(event)=Ox04

不幸的是，UART接口不能在低电压的情况下进行信号收发，因此，该接口不适台低电压的设备。一些设备为了使UART工作在非常低电压的工作模式下，其会使用额外的硬件信号线来驱动UART．不过它们通常属于专有扩展，不同设备的实现方式也有所不同。

在经典蓝牙里面，Ox03类型的HCI数据包也可以被定义为同步数据包，不过低功耗蓝牙则不能使用这种类型的数据包。因此，仅仅拥有低功耗模式的蓝牙控制器是不能发送和接收此类数据包的。

**8.2.2 三线UART**

3线UART比之前描述的UART复杂一些，它在设计上舍弃了硬件流控线，但考虑了一些误比特情况。如果你的主机和控制器之间的链路超过几毫米，并且它们处在一个有噪声的电气环境中，那么使用3线UART就再适合不过了。

对于3线UART工作模式来说，它使用了信道来标识不同的数据包，这种方式和之前描述的UART划分数据包类型有些相似。信道号码0x01标识命令，Ox02标识数据，Ox04标识事件。另外3线UART为链路建立和确认定义了另外两条信道：Ox0F用于链路建立，而Ox00则用于确认。需要注意的是，在3线UART中，信道号码长度仅为4个比特。

3线UART主要有3种模式：

* 链路建立。Link establishment
* 活动状态。Active state
* 低电压状态。Low-power state

链路建立信道是用来确认对端设备运行模式、配置参数和自动探测波特率。实现自动的波特率探测，采用的办法是依次选择不同的波特率发送链路建立信息，然后判断哪个波特率得到了对端设备的应答。两个设备链路建立使用三次握手协议，在链路建立过程中，还设置了可靠滑动窗口的大小、是否使用循环冗余校验(CRC)以及是否使用帧错位软件流控。

在活动状态下的数据传输是可靠的。所有的数据包都封装成帧，每帧带有一个序列号(sequence number)和确认号(acknowledgement number)。这些号码长度均为3个比特，它们实现了多个数据包的同时发送。这一特性对于快速UART来说相当有用。

数据包被划分为帧时遵循了SLIP（串行线路口协议）协议，它在RFC 1055中定义。SLIP规定每一个帧以0xC0（一个字节）作为开始标志和结束标志。如果一个数据包里面出现了0xC0。则使用OxDB OxDC（两字节）替代。由于OxDB是一个转义序列，因此，它也需要转换成OxDB OxDD（两字节）。如果使用了帧错位软件流控，那么XON和XOFF字节也需要分别转换为OxDB OxDE和OrDB OxDF。

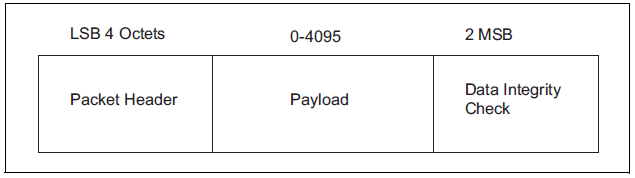


图8-1

如图8-1数据包由Packet Header，Payload，Data Integrity Check三部分组成。每帧的头部如图8-2包含数据包的长度和头部校验和。头部校验和用来确认头部信息正确与否。如果不正确，那么就拒绝接收整个数据包；根据重传机制，对端设备会自动重发这个数据包。

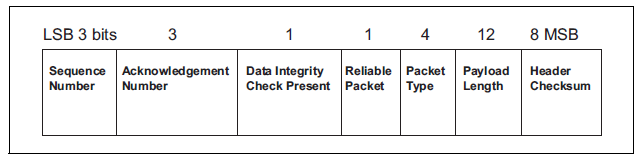


图8-2

净荷的最大字节散为4095，并且使用CRC（循环冗余校验）进行校验。3线UART使用16位的CRC-CCITT（生成多项式）算法，与经典蓝牙的基带数据包所采用的16位CRC-CCITT完全一致。另外，如果一个数据包的CRC校验失败，那么就忽略这个数据包，对端设备会将整个数据包重新发送。

通过发送睡眠（sleep）信息可以将连接状态切换为低功率模式。在该模式下，UART通常处于关闭状态，传输的数据包不能保证完整地接收。假如有数据包要发送，需耍花费一些时间等待对端设备唤醒UART硬件。为了实现高效的唤醒机制，在向对端设备发送HCI信息之前，首先向其发送一段很短的唤醒信息。对方一旦回复了已唤醒（woken）信息就会切换至活动状态。这个时候，设备就可以发送任何的数据包了。

如果一个设备只有UART接口，并且这个UART接口不能提供抗误码的鲁棒性，那么使用3线UART是一种正确的选择。

**8.2.3 USB**

USB接口主要为那些拥有USB Host的设备而设计。它定义了主机和设备之间是如何传送命令、事件和数据的。蓝牙的USB接口规范也定义了一个标准类代码（standard class code）来标识蓝牙设备。这使得各个制造商能够销售即插即用的蓝牙软件狗（Dongle）。

主机通过控制端点Control endpoint（0x00）给设备发送命令，使用“主机向设备请求，设备为目标”作为请求类型。主机使用中断端点Interrupt endpoint（Ox81）轮询来自设备的事件。端点能够每1ms轮询一次，这使得每个事件可以在非常短的延迟内输出。

数据在两个端点中传输：一个是Ox02，主机向设备输出块数据Bulk data out。另一个是Ox82，设备向主机输入块数据Bulk data in。此外，每隔1ms就有一次轮询。然而，当需要非常高的数据吞吐量时，可以在单个帧上批量发送USB块数据。

USB接口的最大缺点就是对功耗不太敏感。实现USB接口需要大量的高速硬件和软件控制，而它们常常抬高了成本。另一个缺点是主机必须每隔1ms向设备查询数据。通常情况下，当设备正在运行的时候，查询动作会让主机无法进入低功耗的工作模式。不过，可以使用USB的链路电源管理特性来解决这些问题，也非常建议在蓝牙设备上使用这种技术。

**8.2.4 SDIO**

SDIO协议定义了一种高速的传输接口。主机能够通过SDIO卡Type-A接口来和控制通信。SDIO是一种基于分组的总线，可以工作在很低的功耗下，使用4-8线实现双向数据传输。

SDIO接口用来传输命令、事件和数据的信道与UART以及3线UART使用的信道完一样。

另外．SDIO接口误码率非常低，这对那些拥有SDIO接口的设备来说十分有用。

不过，要获得完整的SDIO传输规范，你必须是SD协会的成员之一。否则，你只能到一份用于评估SDIO技术的简版规范，它缺少了一些用于设备生产的细节，不过提供了够的信息来帮你理解系统是如何运作的。

**8.3逻辑接口**

在8.2节描述的HCl物理传输接口就是一种逻辑接口。所谓的逻辑接口是指在单芯片备上，你不需要在组件之间真正实现这样的信息传输接口。然而，如果系统中的控制器和机分别位于两个芯片上，HCI逻辑接口就体现为在二者之间传输数据的物理接口。

关于逻辑接口．我们需要理解三个相关的概念；

* 信道（channel）
* 数据包格式（packet format）
* 流控（flow control）

**8.3.1 HCI信道**

一旦控制器与其他BLE设备建立了连接，控制器的底层HCI接口就创建一个HCI信道，们使用一个连接句柄(connection handle)来标识这个HCI信道。连接句柄用来识别从主机发送到控制器，并准备发往对端的数据。连接句柄也用来识别控制器从对端收到的准备交给主机的数据。

无论何时，只要主机尝试刨建一个连接，控制器就为主机提供一个连接句柄。直到连结束连接句柄才会失效。连接结束的原因可能是本地的链路层终止规程Link Layer termination procedure，也可能是链路监控超时link supervision timeout导致。

**8.3.2命令包**

主机通过向控制器发送命令数据包来执行命令。这些命令通常用于配置控制器的状2或者请求控制器完成操作。

如图8-3所示．HCI命令数据包包括：用来确认发送命令的操作码、参数的长度字段命令参数。每个命令都有自己的一套独特的参数。

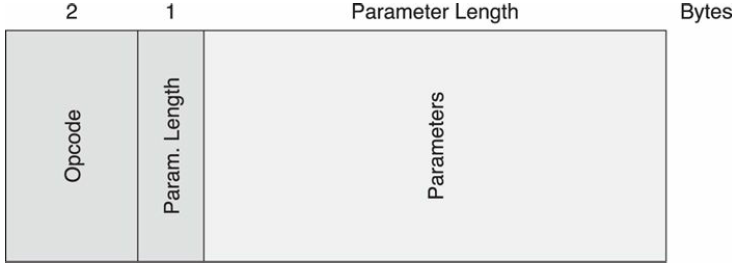


图8-3 HCI命令数据包格式

在低功耗蓝牙中一共有三种基本命令类型，分别实现如下功能

* 配置控制器的状态。Configure the controller state
* 请求执行特定的操作。Request a specific action
* 管理连接。Control a connection

**1．配置控制器状态**

我们可以将控制器视为一个大状态机，有一系列的参数可以进行配置。例如，如果将广播看做成一个状态，那么我们可以通过如下的命令进行配置：LE Set Advertising Parameters、LE Set Advcitising Data、LE Set Scan Response Data和LE Set Advertise Enable。

在低功耗蓝牙中，如果状态机里的某个状态正在被使用，通常是不能对其进行调整的。就好像如果启动了广播就不能再改变广播参数。这个时候，我们需要首先停止广播，修改广播参数，然后重新进行广播。

**2．请求指定操作**

一些命令可以请求控制器执行指定操作，但不会改变设备状态或者连接状态。例如．LE Encrypt命令向控制器输入密钥和文本并要求其生成加密数据。

**3．管理连接**

设备建立连接之后，可以通过发送命令来管理连接，比如用LE Read Channel Map命令来获取当前连接的自适应跳频信道图。注意，这类命令均包舍含连接句柄。

**8.3.3事件包**

控制器发往主机的事件数据包主要用于发送信息和反馈之前输入的命令。

如图8-4所示，HCI事件数据包由事件类型编码、参数的长度字段和事件参数组成。每个事件都有一套独特的参散。

低功耗蓝牙主要有下列三种基本事件类型：

* 通用命令完成事件。Generic command complete events
* 通用命令状态事件。Generic command status events
* 特定命令完成事件。Command-specific completion events

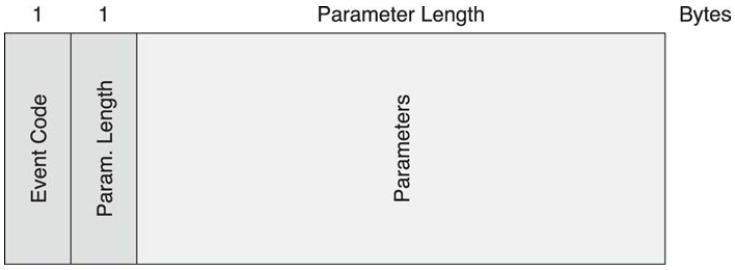


图8-4 HCI事件数据包格式

**1.通用命令完成事件**

当主机发送给控制器的命令执行完毕时，控制器立即返回一个通用命令完成事件。事件的参数包含了之前发送的命令操作码和执行命令的返回参数。返回参数中的第一个参数是状态码，状态码用来表示命令执行成功与否。

例如，LE Rand用来命令控制器返回一个随机数。它有两个返回参数，一个是状态码，另一个是返回的随机数。

注意，每当控制器执行与无线传输无关的任务时，都将使用通用命令完成事件。例如．LE EncIypt命令不涉及链路层数据包传输，因此将返回一个通用命令完成事件。而LE Create Connection命令需要在连接建立之前至少发送一个链路层连接请求数据包（CONNECT\_REQ）。因此不会返回通用命令完成事件。

**2．通用命令状态事件**

上节提及的LE Create Connection等命令类型需执行无线传输操作，通常返回的是通用命令状态事件；一段时间后才会返回特定命令完成事件。

**3．特定命令完成事件**

有些命令需要一段时间才能完成执行，它们都有对应的、并且是唯一的特定命令完成事件。例如．LE Create Connection命令首先返回一个命令状态事件，随后等待连接建立完成或者失败，再返回LE Connection Complete事件。需要注意的是，命令是否执行完毕并不以收到了特定命令完成事件作为标志。

**8.3.4数据包**

数据包是指主机和控制器之间传输的应用数据。控制器接收来自主机的数据包，将其传给对端设备。对端设备收到数据之后，将其从控制器发往主机。

如图8-5所示，数据包总是以连接句柄作为第一个字段，在LE Connection Complete事件中该字段占用了12位。如果主机一直没有收到LE Connection Complete事件，则不能发送任何数据给对端设备；只有收到了该事件，主机才可以启动数据发送或者接收对端设备发送过来的数据。数据长度

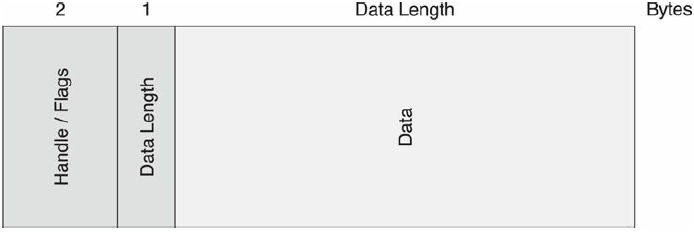


图8-5 HCI数据包格式

HCI数据包有两种标记（flags）：数据包边界标记（**Packet Boundary Flag**）和广播标记（**Broadcast Flag**）。由于其中一些标记继承自经典蓝牙，它们在低功耗蓝牙里没有什么意义。数据包边界标记用来确认上层协议（逻辑链路控制和适配协议，L2CAP）的信息是一个开始数据包还是一个延续包。这有点像链路层的数据信道PDU中的LLID位（详见第7章7.8.2节）。

需要注意的是，从主机发送给控制器的数据包，标记00表示开始包，标记O1表示延续包，然而从控制器发送给主机的数据包有些不同，标记1O表示开始包，标记01表示延续包。这是因为在经典蓝牙中，主机发送给控制器的标记10表示可以在必要时将该数据包清除。而在低功耗蓝牙中不存在清除数据包的概念；相反，假如数据包无法送达，链路将被断开而不会用到上述值。基于上述原因，低功耗蓝牙在主机到控制器的报文中使用标记00表示开始包（取代链路层LLID中的1O），而在经典蓝牙中，标记00用来表示该数据不可清除。

**8.3.5命令流控**

HCI接口有两种流控形式：**命令流控**和**数据流控**。控制器使用命令流控同时处理多个HCI命令。究其原理，简单来说是因为控制器内部拥有足够的缓存，能够存储一定数量的命令；主机可以通过控制器来获知缓冲区的长度，从而得知可以同时发送的命令的最大数量。

HCI接口不支持事件流控。因为事件的数量受限于可处理命令的数量，另外主机比控制器拥有更多的资源，所以能够顺序地缓冲和处理这些事件。

要启动命令流控，所有命令完成（Command Complete）事件和命令状态(Command Status)事件都必须包含一个Num HCI Command Packets参数。该参数用来表示控制器能够缓存多少条命令。控制器每次都从缓冲区中获取命令来执行。控制器每次向主机发送命令完成事件或命令状态事件，都会在其中包含缓冲区的剩余空间。控制器可以随时发送操作码为No Operation的命令完成事件，其中携带新的Num HCI Command Packets参数。这在启动控制器的时候非常有用，控制器可以提供更多的缓冲槽加速完成初始化配置。

**8.3.6数据流控**

数据流控和命令流控的方式差不多。一共有两种数据流：**主机到控制器**以及**控制器到主机**。主机到控制器的数据流控是必须的，而控制器到主机的流控可以忽略。因为大部分的主机都能处理从控制器到主机的大量数据，因此该没有必要进行流控。对于主机到控制器的数据流控，控翻器拥有一定数量的缓冲区，每个缓冲区存放一个数据包。每次控制器都从一个缓冲区提取数据包来发送给对端设备，一旦数据包成功发送，控制器释放该缓冲区，以便装填主机发送给控制器的新数据包。

双模控制器的流控相对复杂。它有两个不同的缓冲区，一个用于基本速率数据（basic rate data），另一个用于低功耗数据。要获得两种缓冲区的剩余空间，应分别使用如下的HCI命令：Read Buffer Size命令和LE Read Buffer Size命令。对于单模设备来说，只有LE Read Buffer Size命令才会返回大于零的值。

主机通过命令获知缓冲区的剩余数量，每发送一个数据包给控制器，就消耗控制器中的一个缓冲区。控制器发送Numbcr of Completed Packets事件给主机，当中包含连接句柄的列表和发送数据包的数量，之后释放相应缓冲区。因此，主机不仅知道控制器释放了多少个缓冲区，也知道哪些数据发送到了对端设备。

**8.4控制器的配置**

利用控制器与对端设备通信之前，主机可以先对控制器进行下列相关操作：

* 重置控制器的状态
* 读取设备地址
* 设置事件掩码
* 读取流控缓存
* 读取控制器支持的功能列表
* 生成随机数
* 加密数据
* 设置随机地址
* 配置白名单

**8.4.1重置控制器为已知状态**

在执行其他操作之前，有必要重置一下控制器的状态，如图8-6所示。由于控制器可能正在执行其他操作，或者主机传输才刚刚建立。在这种情况下，我们可以把控制器重置为就绪态，这样会把所有可配置参数恢复为默认值。

主机通过发送Reset命令来重置控制器。控制器一旦重置，将向主机返回Command Complete事件。这里要注意两点：一是重置命令不会重置物理链路，如果要重置链路的话，需要执行另外的链路重置操作；二是虽然主机可以发送多条命令给控制器（通过命令流控实现）．如果控制器正在执行重置，也不能接受其他任何命令。此外还应注意，当控制器正在执行某条命令时，假如主机发送了重置命令，那么控制器会停止执行当前命令，转向执行重置命令。

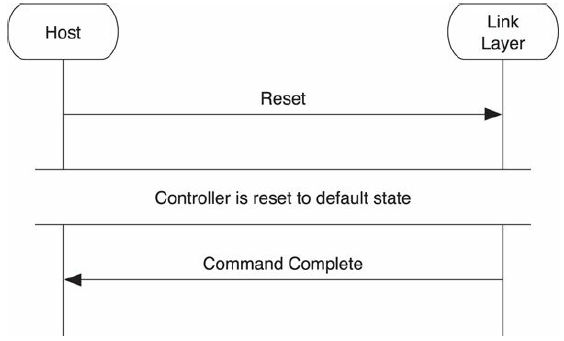


图8-6 HCI重置

**8.4.2读取设备地址**

很多低功耗蓝牙设备有预先设置的设备地址，可供主机读取。

如图8-7所示，主机通过向控制器发送Read BD\_ADDR命令来读取设备地址，后者返回一个含有固定设备地址的Command Complete事件。如果控制器没有固定地址，则返回全零地址00:00:00:00:00:00。此时，主机需要为控制器生成一个随机地址，否则无法进行数据传输。

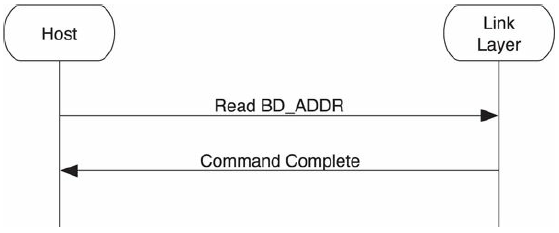


图8-7读取设备地址

**8.4.3设置事件掩码**

控制器定义了很多事件类型。考虑到来来的蓝牙规范还会添加新的事件和功能，控制器必须知道主机能够接受和处理哪些事件。如果主机不能理解控制器的事件，将会导致互操作性问题。解决问题的一个办法是设置事件掩码，让主机告诉控制器哪些事件能接收，哪些不能接收。选样控制器只会发那些能被接收的事件。

如图8-8所示，设置事件掩码需要用到两条命令。一条是Set Event Mask命令，另一条是LE Set Event Mask命令。Set Event Mask命令曾用于经典蓝牙设置各类事件．由于其中的“meta-event”事件仍然适用于低功耗蓝牙，主机可以使用该命令来启动或屏蔽meta-event。另外的LE Set Event Mask命令则是用来启动或屏蔽相关的低功耗蓝牙事件。

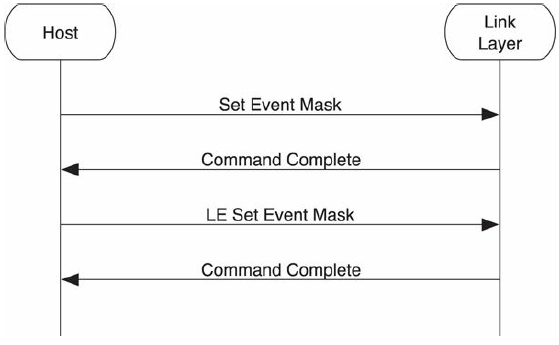


图8-8 HCI设置事件掩码

**8.4.4读取缓冲区大小**

在数据流控那一节，我们提到控制器最多可以有两种类型的缓冲区来满足数据传输。

如图8-9所示，要获得这两种类型的缓冲区的大小，主机可以分别发送LE Read Buffer Size命令和Read Buffer Size命令。一旦收到相应的Command Complete事件，主机就能知道缓冲区的可用空间，从而确定还能向对端设备发送多少个数据包。

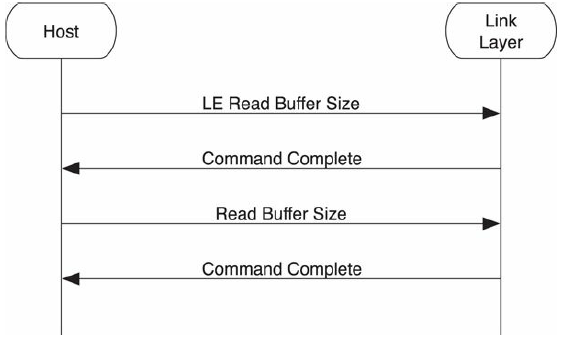


图8-9读取缓冲区大小

**8.4.5读取控制器支持的功能**

确保主机和控制器兼容的另一个办法是，主机向控制器发送命令之前首先确认控制器支持的功能。

如图8-10所示，主机通过发送LE Read Supported Features命令来获得控制器所支持的功能列表。在控制器返回的Command Complete事件中包含了此功能列表。一般来说，主机在发送与功能有关的命令之前，应该首先发送LE Read Supported Features命令。

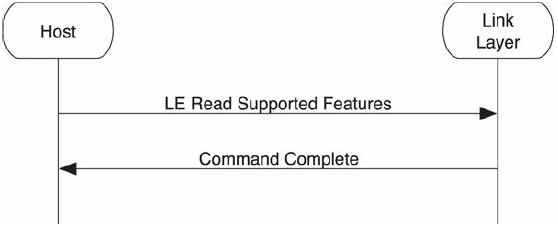


图8-10获取控制器支持的特性列表

**8.4.6读取控制器支持的状态**

控制器的设计可以非常简单，也可以非常复杂。因此主机必须知道控制器支持哪些状态及其组合，以免因设置了无效的状态而导致错误。

如图8-11所示，主机发送LE Read Supported States命令后，控制器返回带有支持的状态列表的Command Complete事件。

* 控制器可以支持如下状态：
* 不可连接广播（nonconnectable advertising）
* 可扫描广播（scannable advenising）
* 可连接广播(connectable advertising)
* 定向广播（directed advertising）
* 被动扫描（passive scanning）
* 主动扫描（active scanning）
* 发起连接成为主设备（initiating/a connection to the master role）
* 连接成为从设备(a connection to the slave role)

另外，还可以读取一些标志位来了懈控制器支持哪些组合状态，例如：

* 同时处于不可连接广播(nonconnectable advertising)状态和被动扫描(passive scanning)状态
* 同时处于不可连接广播(noncomectable advatising)状态和连接成为从设备(aconnectIon to the slave role)状态

利用这些状态支持信息，主机可以知道控制器能否成功执行广播、扫描或发起连接等

命令。

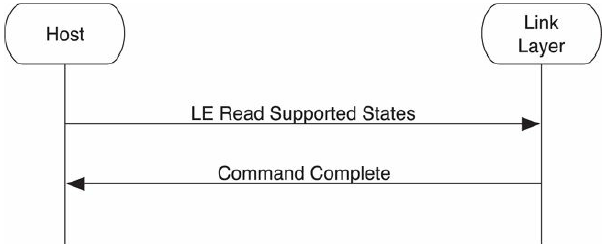


图8-11获取控制器支持的状态列表

**8.4.7随机数**

控制器可以方便地生成随机数。这些随机数通常源于设备自身的物理特性，例如自激振荡器和外部晶体时钟的相位噪声。鉴于大部分的随机数都源自这些固有的物理特性，可生成的随机数往往较为有限。然而，主机可以将这些随机数怍为种子，将其用于传统的伪随机数生成器，或者间歇性地为其注入一些随机性。

如图8-12所示，主机发送LE Rand命令来让控制器为其生成随机散。随机散包含在返回的Command Complete事件中。



图8-12生成随机数

**8.4.8加密数据**

主机可以使用低功耗蓝牙的AES-128加密引擎来加密数据。由于控制器已经实现了加密功能，给主机提供相关的接口也就顺理成章了。

主机发送LE Encrypt命令来进行数据加密。在此命令中包含了需要加密的数据和加密密钥，如图8-13所示。控制器使用AES-128加密算法进行加密，并在返回的Cammand Complete事件中包含已加密数据。

需要注意的是，在低功耗蓝牙中是没有解密命令的。主机可能要为使用私有地址的设备加密数据，但是无需实现数据解密。换言之，主机只能检查相同的明文在变成密文后是否一致，而不能使用密钥将密文恢复为明文。

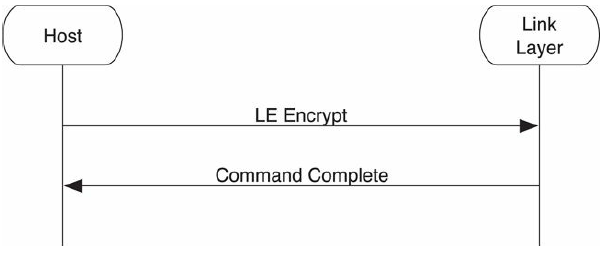


图8-13 HCI加密数据

**8.4.9设置随机地址**

如果控制器没有固定地址，或者主机希望使用私有地址来替代固定地址，那么主机必须为控制器设置一个随机地址，以便用于广播、主动扫描和发起连接等操作。

通常来讲，随机地址由随机数生成或者由随机数和IRK（Identity Resolving Key）的组合生成。如图8-14所示，主机首先使用LE Rand命令来生成一个随机数。接下来，此随机数和IRK分别作为明文和加密密钥执行LE Encrypt命令。主机获得了返回值后将其作为随机地址，使用LE Set Radom Address命令对控制器进行设置。只有收到了地址设置的Command Complete事件，该随机地址才可用于其他命令。

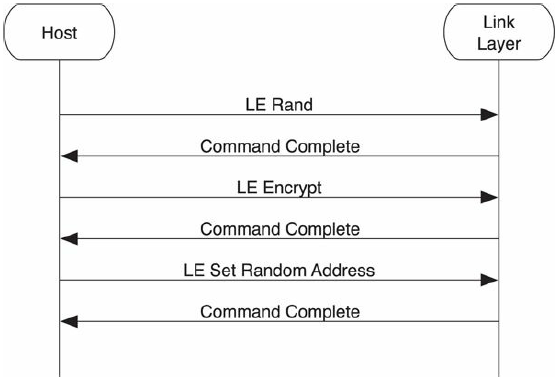


图8-14设置随机地址

**8.4.10白名单**

控制器里面存储着一个设备地址列表，称为白名单。白名单在充满广播报文的密集环境中可用来搜索已知设备。不过由于白名单的容量有限，在使用之前首先要确定其容量的大小。主机可以通过命令对白名单进行添加、删除以及重置等操作，而控制器可以根据白名单来过滤广播数据包。

如图8-15所示，主机使用LE Read White List Size命令来读取白名单的容量大小，控制器在返回的Command Complete事件中包含了最多能容纳的条目数量。在管理白名单方面，可以用LE Oear White List命令来清空列表，要从列表中添加和删除条目则应使用LE Add Dcvice To White List命令和LE Remove Device From White List命令。

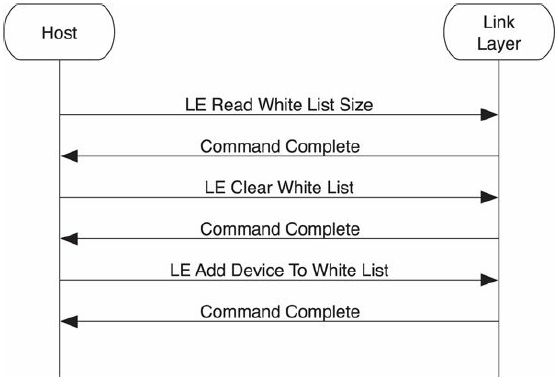


图8-15—管理白名单

当控制器正在使用白名单的时候不能改变其列表内容。比如一个广播设各正在使用白名单来过滤某些设备的扫描请求或者连接请求，如果要修改白名单的内容，必须首先停止广播，待白名单修改之后再重新启动广播。

**8.5广播和观察**

两个低功耗蓝牙设备之间的最基本的通信模型就是广播(broadcasting)和观察(observing)模型。二者采用**广播advertising**和**扫描scanning**来传输数据。

**8.5.1广播**

控制器有两类数据可以通过广播发送：**广播数据**和**扫描响应数据**。另外，控制器还有一系列的参数，用来设置如何发送以及何时发送广播报文。

如图8-16所示，主机使用LE Set Advertising Parunerers命令来设置广播参数。配置参数包括广播的最小间隔时间和最大间隔时间，范围从20ms～10.24s。另外，广播类型一共有四种，它们分别是：

* **可连接的非定向广播Connectable undirected advertising**。这是一种常用的广播类型，包括广播数据和扫描响应数据，它表示当前设备可以接受其他任何设备的连接请求。
* **可连接的定向广播Connectable directed advertising**。表示设备仅仅能接受某一特定设备的连接请求，不过，它不包括广播数据。
* **可扫描的非定向广播Scannable undirected advertising**。发送广播数据和扫描响应数据，用来激活扫描者。
* **不可连接的非定向广播Nonconnectable undirected advertising**。仅仅发送广播数据。

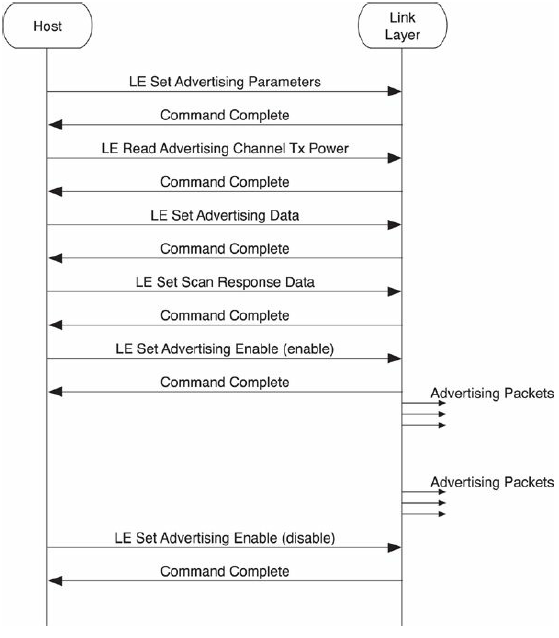


图8-16 HCI广播

LE Set Advertising Parameters命令还可以设置地址类型，或者为固定设备地址，或者为如上所述的随机地址。如果当前的广播类型为定向广播，那么在广播数据包中应含有对端设备的地址。另外，还有两个参数可以配置，它们分别是**广播信道映射**和**广播过滤策略**。广播信道映射用来决定使用哪三个广播信道，而过滤策略用来过滤不符合规则的广播数据包，过滤策略可以设置为如下规则之一：

* 接受任何设备的扫描请求或连接请求。
* 仅仅接受白名单中的特定设备的扫描请求，但是接受任何设备的连接请求。
* 接受任何设备的扫描请求，但仅仅接受白名单中的特定设备的连接请求。
* 仅仅接受白名单中的特定设备的扫描请求和连接请求。

如果想获得广播时的发射功率，那么可以使用LE Read Advertising Channel TX Power命令。另外，我们还将发射功率的值放在广播数据包中或者扫描响应数据包中，从而实现靠近配对，或者供用户界面(user intetface)根据路径损耗对设备进行排序。

要设置广播数据包和扫描响应数据包的内容，主机可以分别使用LE Set Advertising Data命令和LE Set Scan Response Data命令实现。

当一切配置就绪，可以使用LE Set Advenising Enabled来启动或关闭广播。广播一旦启动，控制器将使用配置好的参数进行广播。

**8.5.2被动扫描**

要接收对端设备的广播数据包，可以使用被动扫描。

如图8-17所示，使用LE Set Scan Parameters俞令来进行设置控制器的扫描参数。可配置的参数如下：

* 扫描类型Scan type——可设置为被动扫描或主动扫描。
* 扫描间隔Scan interval——控制器间隔多长时间扫描一次。
* 扫描窗口Scan window——每一次扫描的持续时间。
* 扫描策略Scanning filter policy——接受任何的广播数据包或者仅仅接受白名单设备的广播数据包。

**扫描间隔**和**扫描窗口**是两个非常重要的参数，因为它们决定了控制器的扫描频率和扫描时间的长短。扫描间隔和扫描窗口共同决定控制器的扫描占空比，即扫描窗口与扫描间隔二者的比值。

例如，如果扫描间隔为1OOms，扫描窗口为1Oms．那么控制器的扫描占空比为10%。理论上，可捕获到定向广播数据包的最低的占空比为04%，即每一秒中扫描3.75ms。

当然，我们可以把间隔和窗口都设置为相同的数值。在这种情况下，扫描连续进行，每个间隔改变扫描频率。例如，把间隔和窗口都设置为10ms，那么控制器就在10ms内一直处于扫描状态，并且每1Oms切换一次扫描信道。

扫描过滤策略决定了是否使用白名单。控制器可以接受任何的广播数据包，也可以仅仅接受白名单中符合过滤策略的广播数据包。不过，得注意一点，如果定向广播数据包中的目的地址并非是自己，那么，该数据包必须要抛弃，即使广播数据包的炭送者在自己的白名单中也不例外。

一旦扫描参数设置完毕，主机就可以使用LE Set Scan Enabled命令启动扫描。扫描过程中，如果控制器接收到的符合过滤策略和其他规则的广播数据包，则发送一个LE Advenising Repon事件给主机。除了广播者的设备地址外，报告事件还包含广播数据包中的数据，以及接收广播数据包时的信号接收强度。我们可以利用该信号强度以及位于广播数据包中的发送功率，共同确定信号的路径损失，从而给出大致的范围。

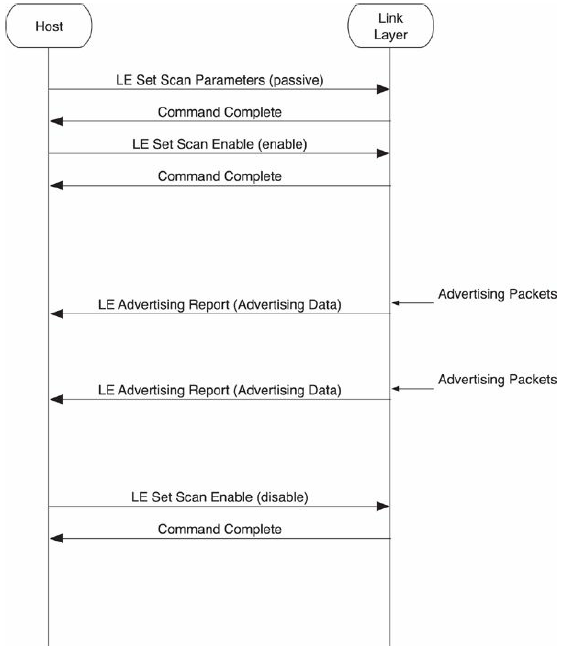


图8-17 HCI被动扫描

主机如果想停止扫描，还是使用LE Set Scan Enable命令，只是参数应设为“停止扫描”。

**8.5.3主动扫描**

我们来看更为复杂的主动扫描。它不单可以捕获到对端设备的广播数据包，还可以捕获可能的扫描响应包。

在参数配置和启动扫描方面，主动扫描和被动扫描使用的命令完全一样。不过，因为控制器要发送SCAN\_REQ数据包给对端设备，以便获取扫描响应数据包，而这些数据包需要包含设备地址，因此，在使用LE Set Scan Parameters命令时需要配置一个额外参数，决定链路层的数据包使用固定地址抑或是随机地址，如图8-18所示。

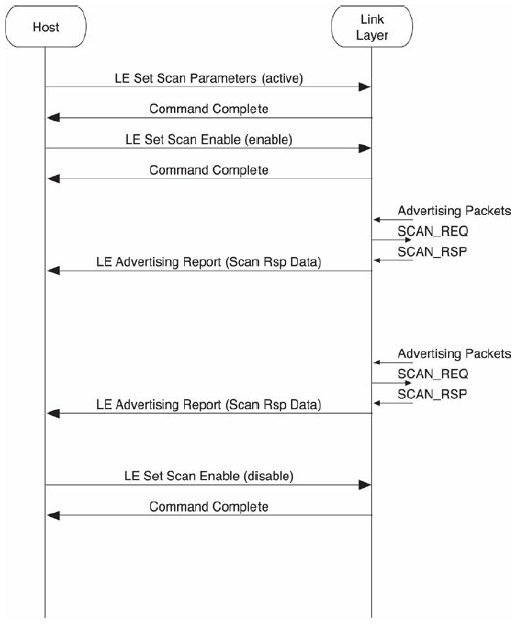


图8-18 HCI主动扫描

控制器收到SCAN\_RSP数据包后将向主机发送一个LE Advertising Report事件。该事件同样包括了链路层数据包的广播类型。因此，主机能够判断对端设备是否可以连接或扫描．并且区分出广播数据包和扫描响应数据包。

**8.6发起连接**

广播和扫描仅仅是蓝牙功能的一部分，要实现更多的应用功能必须依靠两个设备之间的连接。要建立连接，其中一个蓝牙设备应处于可连接的广播状态，由另一个设备发起连接。当然，连接的对象既可以是一个白名单，也可以是指定的单一设备。建立连接需要耗费一定的时间，如果用户或者应用程序不再需要连接时，可以在连接未建立之前取消。

**8.6.1与白名单设备发起连接**

最常用的连接方式是主机先将对端设备添加到白名单中，然后再与白名单中的设备进行连接。通过这种方式，控制器可以同一时间和多个设备发起连接。实际上，这能让主机请求控制器同时与A、B、C、D、E、F等设备发起连接。

要与一个或多个设备发起连接，白名单必须包含这些设备。如图8-17所示，主机使用LE Add Device To White List以及其他白名单管理命令实现该功能。一旦主机打算与白名单中的设备连接，则向控制器发送LE Create Connection命令。

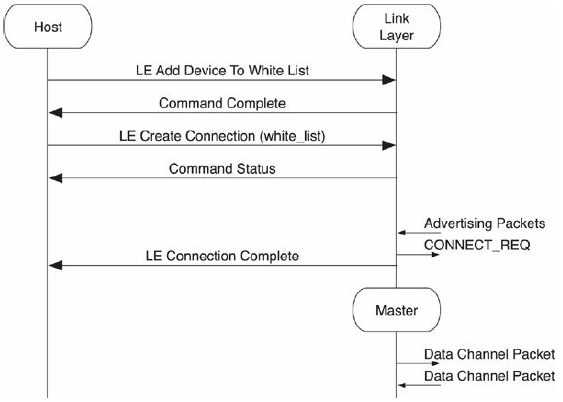


图8-19 HCI与白名单中的设备发起连接

LE Create Conneclion命令可配置如下参数：

* 扫描间隔和扫描窗口——与主动扫描参数共同决定控制器侦听广播的频率。
* 发起者过滤策略—一可设置为“使用白名单”，表示仅与白名单中的设备发起连接。
* 发起者地址类型——该参数设置CONNECT\_REQ数据包的地址类型是固定地址抑或是随机地址。
* 发起连接参数——该参数用于设置主设备传输数据给从设备的频率、从设备可以忽视主设备的等待时间(latency)、监控超时(supervision timeout)以及每个连接中发往从设备或来自从设备的预期的数据传输质量。

注意，发起连接参数对白名单的所有设备是完全相同的。在与大量相似的设备——比如自动传感器进行连接时，该命令尤为有用，但是当连接的设备类型各不相同时，作用较为有限。如果要与不同类型的设备进行连接，那么主机应该为LE Create Connection命令设置最坏情况下的连接参数。

如果控制器收到白名单中某一设备的可连接广播数据包，则向对端设备发送CONNECT\_REQ数据包，其中含有所有连接所需的信息。另外，控制器还向主机发送LE Connection Complete事件。如果对端设备接收到CONNECT\_REQ数据包，它也会向其主机发送LE Connection Complete事件。

该LE Conncction Complete事件包含连接句柄，用于标识主机和控制器之间传输的数据包。该事件还包括当前控制器的角色信息（主设备或从设备），以及从设备的地址、间隔时间、等待时间、监控超时和主设备的时钟精度等。从设备需要利用时钟精度来决定其窗口扩展（window widening），而该参数提供给主机当做信息参考之用。

注意，**当连接已经建立并且发送了LE Connection Complete事件，所有的广播或者其他的发起连接请求将会自动停止。**如果主机想继续广播或与其他设备进行连接，则必须再次执行相应的命令。

**8.6.2与单一设备发起连接**

与单一设备进行连接，主机同样使用LE Create Connection命令（如图8-20所示）。区别在于，发起者的过滤策略将设置为“忽略白名单”，另外还需要设置对端设备的地址等相关参数。除了设置参数的细微不同，与单一设备的连接过程和与白名单的连接的过程完全类似。一旦连接建立，双方设备都将生成LE Connection Complete事件。

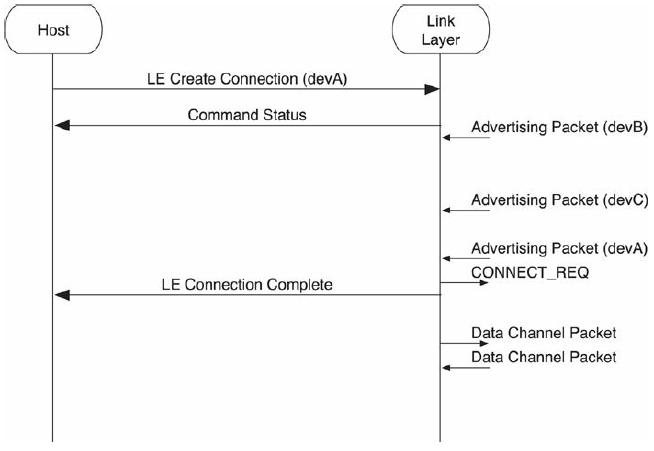


图8-20与单一设备进行连接

**8.6.3取消连接请求**

有时候由于连接的设备不在附近，连接请求的接收方长时间没有响应，此时，主机可以取消此连接请求，转而执行其他任务。

如图8-21所示．主机使用LE Create Conncction命令向对端设备发出连接请求，如果主机想取消此次的连接，发送LE Create Connection Cancel命令即可，该命令会返回Command Complete事件。另外，不管连接成功与否．LE Create Connection命令的都会返回LE Connection Complete事件。

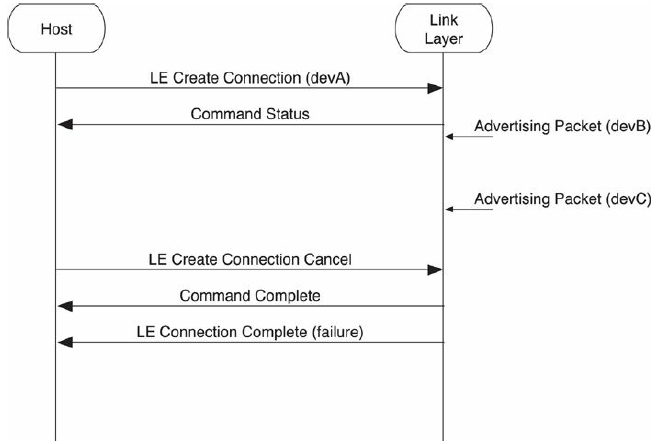


图8-21取消连接请求

注意，这里有可能出现竞态条件。打个比方，主机向控制器发送LE Create Connection Cancel命令的同时，可能控制器已经向对端设备发送了CONNECT\_REQ．但尚未返回LE Connection Complete事件。在这种情况下，LE Create Connection Cancel命令返回Command Complete事件，但还可能收到一个连接已经建立的LE Connection Complete事件。所以，**即便试图取消某个连接，该连接也完全有可能成功地建立。**

**8.7连接管理**

建立连接之后，设备就可以对连接进行管理，包括切换至低功耗模式、增加或减少间隔时间、设置加密或者终止连接等。

**8.7.1更新连接**

如果当前的连接参数不再适用时，主设备可以进行修改。比如为了加快设备韧始化，当初发起连接时使用了短间隔，假如设蔷已经正常运行服务了，那么使用长连接间隔会更加节能。

如图8-22所示，主设备使用LE Connection Update命夸修改连接参数。连接参数包括新的连接间隔时间、等待时间、监控超时以及连接事件长度。

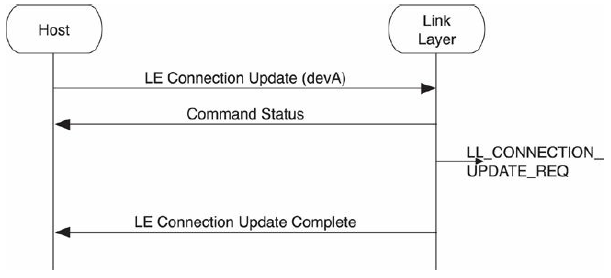


图8-22修改连接参数

控制器收到命令后先返回Command Status事件，随后才发送链路连接更新请求数据包给对端设备（详见第7章7.10.1节）。该数据包包括了瞬时时刻。一旦计数器倒数结束，就必须更新连接参数，随后控制器返回LE Connection Update Complete事件表示连接参数已经更新。

**8.7.2更新信道映射图**

主机可能存有关于本地信道使用情况的信息，并且希望将该信息告知控制器。例如，本地有 Wi-Fi设备正在与访问点进行通信，而低功耗蓝牙的工作信道如果能与该Wi-Fi设备的工作信道错开，无疑能减少互相干扰。

低功耗蓝牙并不能直接向对端设备发送请求以设置链路层信道图（详见第7章7.10.2节），但是主机可以使用LE Set Host Channel Classifcation命令来达到相同的目的，如图8-23所示。该命令中含有一个位字段，表示每个链路层的数据信道是坏信道或是未知信道。显然，主机无法知道某个信道是不是“好”的信道，但控制器可以通过监测信道中数据包的错误率来判断哪些未知信道的确很糟糕。

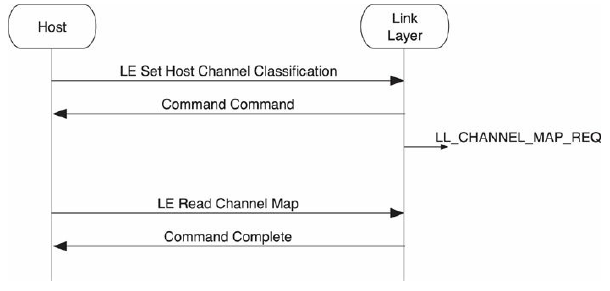


图8-23更新信道映射图

LE Set Host Channel Clasaification命令将返回Command Complete事件。控制器可以在任何时候启动链路层控制规程修改信道映射图（详见第7章7.10.2节）。另外，主机也可以使用LE Read Channel Map命令获得当前的信道映射图，该命令的返回事件包含了每个链路层数据信道的使用情况。

**8.7.3交换功能列表**

主机可以获知当前连接所支持的功能。这一点很有必要，因为尽管本地控制器支持加密，但这一功能只有在对端设备的控制器同样支持时才能使用。

如图8-24所示，主设备的主机使用LE Read Remote Used Features命令请求获取对端设备的支持功能列表。该命令除了返回Command Status事件外，还会让控制器发送链路层功能请求（LL\_FEATURE\_REQ）以及获得对端设备的链路层功能响应包（LL\_FEATURE\_RSP）（详见第7章7.10.6节）。随后，控制器在返回的LE Read Remote Used Features Complete事件中包含了对端设备的功能信息。

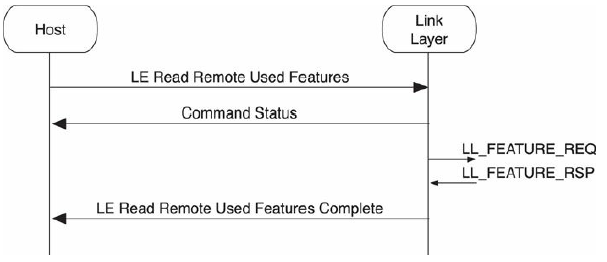


图8-24公开连接的功能

**8.7.4交换版本信息**

在调试设备的时候，特别是那些非直接控制的设备，获得链路层的版本信息有时候十分有用。我们可以将该版本信息提供给设备生产商，从而方便问题的解决。另外，当两端设备在调试链路的时候，主设备或从设备都可能需要获知对端设备的版本信息。

如图8-25所示，主机使用LE Read Remote Version Informatian命令交互版本信息。控制器除了Command Status事件外，还将交换链路层版本信息（详见第7章7.10.5节）。

变换版本信息后，控制器将该信息包含在LE Read Remote Version Information Compicte事件中，并通知主机。

注意，当再次执行LE Read Remote Veraion Information命令时，虽然仍然返回相同的事件，但是控制器并不会执行相关的链路层规程。这是因为版本信息会被当做一种静态信息缓存在控制器中。如果设备之间已经交换了版本信息，当再次执行版本交换命令时，控制器会在状态事件之后立即返回命令完成事件。

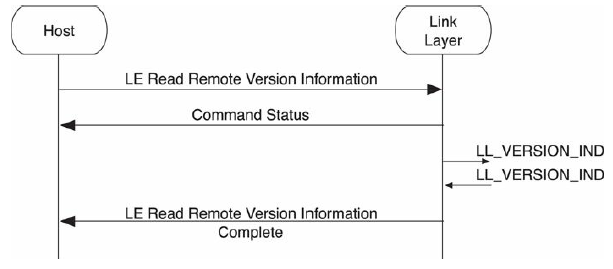


图8-25获取版本信息

**8.7.5加密连接**

只要连接双方都有一共享密钥，主机就可以加密传输数据。共享密钥由安全管理器（Security Managcr）设置，无论是初始化配对过程还是绑定过程中的密钥分配都是由它管理。

连接加密涉及两组命令和事件：一组与主设备相关，另一组与从设备相关。

主设备的主机使用LE Start Enption命令要求链路层启动加密，该命令包含了连接的加密密钥（见图8-26）。当加密规程开始后，控制器将返回Command Status事件。随后链路层开始加密（详见第7章7.10.3节）。加密规程启动之后，控制器返回Emryption Change事件，告知主机加密已经成功完成或者加密过程出现了问题。

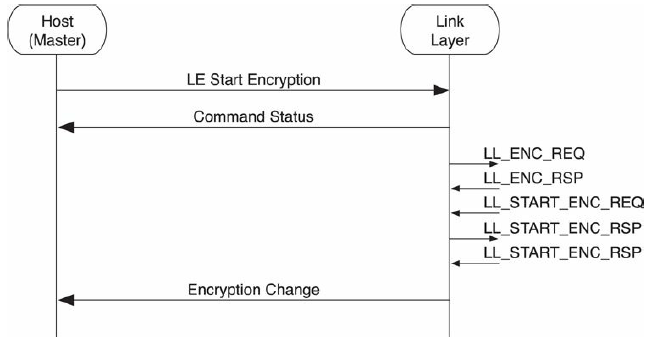


图8-26 HCI主设备启动加密

站在从设备的视角，相关的事件和命令稍微有些不同。如图8-27所示，从设备的主机首先收到LE Long Term Key RcquesL事件，通知从设备启动加密。主机接着发送LE Long Term Key Request Reply命令作为响应，其中包括了加密连接的密钥。由于这是一条命令，所以会返回相应的Command Complete事件。连接成功加密后，从设备的主机将收到Encryphon Ch4nge事件，通知其更新加密状态。

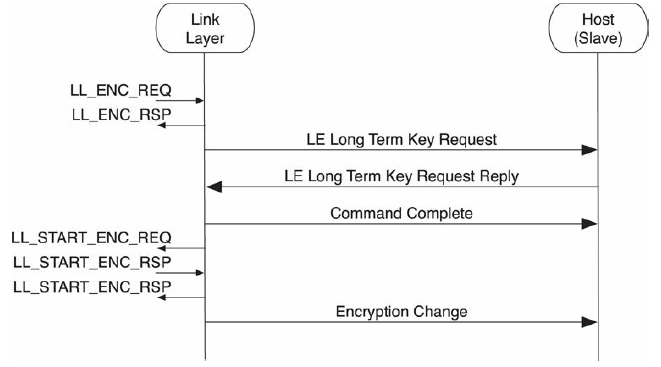


图8-27 HCI从设备启动加密

**8.7.6重启加密**

有时候主机需要重新启动加密，或许是为了使用新的加密密钥，或许是为了刷新用来生成密钥的初始化向量。

如图8-28所示，主设备用到的加密事件和命令与启动加密时的完全一样，不过链路层需要发送更多的数据包：首先要停止加密，然后再重启加密。

从设备用到的加密事件和命令也和初始加密时完全一样，如图8-29所示。

注意，在低功耗蓝牙中，不允许在停止加密后发送应用数据或主机数据。因此一旦暂停加密后，不需要发送Encryption Key Refresh Complete事件。

这是因为在早期的经典蓝牙版本中曾经发现过一个问题。主机希望尽可能安全地发送数据，为此定期更新密钥，令其难以破解。然而，更新密钥需要停止加密，但是即使加密已经停止，数据依然能够发送。因为控制器的缓冲区仍可能有数据等待发送，主机不能简单地停止向控制器发送的数据流。这意味着在早期的经典蓝牙版本中，主机本想提高安全性，结果却反而降低了安全性，最终导致控制器发送了未经加密的数据。不过，后来的经典蓝牙版本已经修正了该问题。

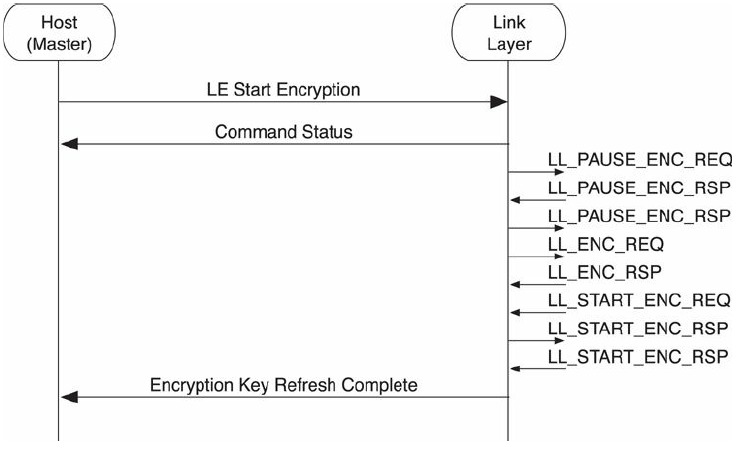


图8-28 HCI主设备重新加密过程

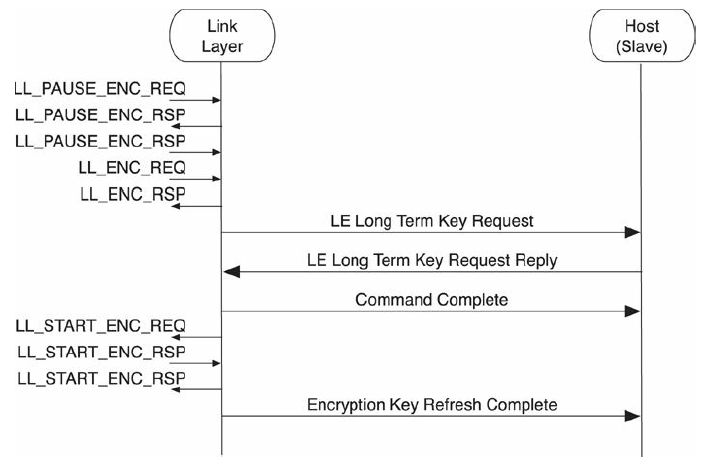


图8-29 HCI从设备重新加密

主机一旦请求加密连接，通常没有理由禁用它。因此，当加密暂停，数据也将停止发送，早期版本的蓝牙安全漏洞也不再存在了。

**8.7.7终止连接**

如果不再需要发送数据，或者维持连接比断开后再重新连接需要更多的能量时，主机可以终止连接。

如图8-28所示，主机向链路层发送Disconnect命令终止连接，随后链路层返回Command Statua事件并尝试终止连接（详见第7章7.10.7节）。如果连接成功终止，控制器将返回Disconnect Complete事件。

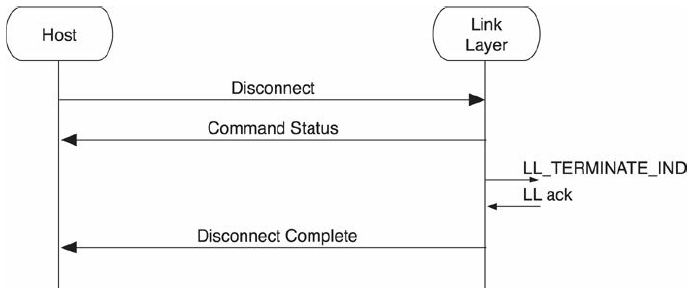


图8-30终止连接

当发生监控超时（supervision timeout）或者加密消息完整性检查失败导致了连接中断，主机也会收到Disconnect Complete事件。监控超时（supervision timeout）一般发生在两个设备距离较远时，被此收不到对端的数据包。加密消息完整性检查失败通常不会发生，除非有攻击者试图接管连接，或者发生了某种特殊的情况，比如出现了较为罕见的位错误，导致报文通过了循环冗余校验而来通过消息完整性检查。

**第三部分 主 机**

**第9章 描述逻辑链路控制和适配协议。**

**第10章 解释无状态协议是如何有效地维护保持设备状态的。**

**第1l章 描述安全服务的主要协议。**

**第12章 把各种要部分合成一个高级抽象层，供程序开发者使用。**

**第9章 描述逻辑链路控制和适配协议。**

那些认为协议不重要的人一定从未遇到过真正的麻烦。

——罗伯特·海因莱因(RobertA．Heinlein)

逻辑链路控制和适配协议（Logical Link Control and Adaptation Protocol，L2CAP）是个复用层，可以让低功耗蓝牙复用三条不同的信道。它也支持数据的分割和重组功能，使得较大的报文可以在底层无线电中传输。在经典蓝牙无线电中．L2CAP层提供的功能比这更多，也更为复杂。

**9.1背景**

和经典蓝牙完全不同，低功耗蓝牙的一个基本概念在于无连接模式connectionless model。用户只在需要发送数据的时候才建立连接，其他时候设备可以长期处于断开连接状态。为了实现该功能，无连接模式必须扩展到L2CAP层，并且只能使用固定信道。这是由于固定信道无需配置任何的参数，一旦底层链路建立连接，固定信道便可立即投入使用，免去了建立信道耗费的额外时间。

在低功耗蓝牙设计之初并未选择L2CAP，而是使用了另一个称为协议适配层（Protocol Adaptation Layer，PAL）的设计。由于PAL被过度地优化，结果严重影响了协议间的复用。PAL就像属性协议外加了一个信令层，其主要缺点在于缺乏灵活性和扩展性。PAL仅支持两种数据包：高层协议信令数据包以及自身的信令数据包。PAL不支持分割或重组，也没有区分不同协议的功能。协议设计的一个基本原则是每一层协议都是独立的。这意味着你可以用系统中的其他协议来设计一个安全管理器（Security Manager）；而在实现时，相互独立的各协议可以单独进行测试。显然，PAL打破了这种简单的规则，糟糕的还不是这种设计，缺乏灵活性才是其宣判死刑的主要原因。

大多数复用层都能实现分割与重组。也就是说，来自高层协议的大数据包可以被分割成多个带有标记的小数据包，从而在一个报文长度受限的系统中传输。ATM网络就是一个很好的侧子，在ATM网络中，每个数据包只限于几十个字节大小，却可以在不同的流之间快速切换。这有利于在同一时间传输低延迟音频数据和大块数据。

HCI支持分割和重组，使用“start”和“continuation”字段标记每个数据包。然而，PAL并不支持分割重组这一基本特性。这意味着在PAL层的应用程序数据长度被限制在24个字节，而这种严格的限制最终导致了PAL的消亡。

当L2CAP被提出来替换PAL的时候，低功耗蓝牙的设计工作组分成了两派：已经实现蓝牙的公司和未实现蓝牙的公司。如果换作其他标准化组织，这种情况可能会通过数月的激烈投票来消除分歧，当中还会牵扯到一些党派斗争，比如把投票成员召集到某个房间，设法动摇他们的想法。蓝牙标准化组织通常不会采用这种做法，相反，他们撰写了一篇论文研究各种方法的成本消耗，并且着重研究了L2CAP所带来的额外成本。确定的结论是，对于每秒发送一次数据的设备来说，其电池寿命将从3 3年降至3 2年。虽然L2CAP减少了设备的寿命，但是相比于它在每个数据包前添加的7字节数据以及之后添加3字节的循环冗余校验，减少的这点寿命其实不算什么。这是另一个说明了低功耗蓝牙的设计师关注细节、综合考虑系统设计的好例子。

L2CAP本身即具有直接添加低功耗蓝牙的扩展能力。这使得低功耗蓝牙从经典蓝牙上直接获得了完整的分割和重组功能，相应地，数据包的最大长度也将扩大至**65535字节**——虽然没有应用会在低功耗蓝牙里发送这么大的包。此外．L2CAP也保留了经典蓝牙所使用的信道模式。

在经典蓝牙中，信道分为两种类型：**固定信道**和**面向连接信道**。只要两个设备建立连接，固定信道就已经存在。固定信道主要作为信令信道，或许是基本L2CAP信令命令，或许是2.0版本或更高版本的MAC/PHY信令信道。面向连接信道可以随时建立，只需给对端发送一些L2CAP信令即可。

在经典蓝牙中，面向连接信道可以把某对单独的应用数据与其他信道的数据相分离。举个例子，虽然所有的面向连接信道都能添加额外的数据完整性检测，但它们可能有不同的流规范，或者隶属于流信道而非尽力交付信道。面向连接信道非常适合那些复杂的、多种数据同时传输的系统。例如，手机和汽车之间同时运行多种不同的协议：手机向汽车的立体声发送高质量的语音流、免提操作数据流、电话薄数据流或着互联网连接数据流。

打开面向连接信道是一个非常复杂的过程。每个L2CAP信道都有大量的配置参数，在最新的规范中就有七种。这意味着，除了交换请求以便建立连接的两条信息外，在发送任何数据之前，双方必须对所有的配置参数达成一致。上述协商过程快起来会很快，只用交换另外的四条信息，不过慢起来也可能很慢，包括一系列的提议和反提议。面向连接信道的复杂性还表现在，当所有的信道参数都已经配置完毕且数据传输开始，设备还可以协商不同的参数。所有的这些虽然带来了更多的灵括性，但是增加了数据连接的延迟。好在对于大部分的经典蓝牙协议和规范这些成本都是可接收的，因为它们的连接往往会维持很长的时间。

**9.2 L2CAP信道**

L2CAP有个很简单的概念——信道。毕竟L2CAP是一种**复用层**，有多个信道也不足为怪。信道是指一个数据包序列，莲接两个设备上的一对服务。在两个设备间允许同时启用多条信道。

低功耗蓝牙**只支持固定信道**。固定信道指的是两个设备一建立连接就已经存在的、没有任何配置参数的信道。得益于未来扩展的灵话性，还可以日后根据需要添加面向连接信道。

表9-1列出了L2CAP的信道标识符。在蓝牙中，每个信道标识符为一个16位的数字。信道标识符0x0000为保留标识符，不能被使用；信道标识符0x000l为经典蓝牙信令的固定信道。

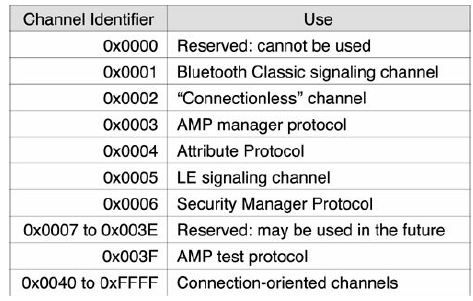


表9-l L2CAP信道标识符

信道标识符Ox0002为用于传输“无连接数据”的固定信道。不过，目前并没有规范使用该信道。信道标识符Ox0003用于当需要高速率传输数据时所使用的Alternate MAC/PHY协议。信道标识符Ox003F则用于测试Aitemate MAC/PHY控制器。

低功耗蓝牙一共使用了**3条信道**：信道标识符**Ox0004用于属性协议**（详见第10章）；信道**0x0005用于低功耗蓝牙信令信道**；**信道Ox0006用于安全管理**（详见第11章）。信道标识符Ox0007～Ox003E为保留信道，此外，信道标识符0x0040～OxFFFF则用于面向连接信道。

**9.3 L2CAP数据包结构**

每个L2CAP数据包的净荷前端都包含一个32位比特报头。假设使用分割和重组，那么数据包的长度信息必须包含在报头中，以便判断数据包的结束。使用分割和重组机制需要为每个通过HCI接口的数据包打上标记（详见第8章），分为开始数据包或延续数据包。不过，这里没有定义怎样标记当前数据包的最后一个片段。这意味着，判断当前数据包是否结束的唯一方法是发送一个新的数据包（假定这个数据包正准备发送），或者将数据包的长度信息放在开始数据包中。

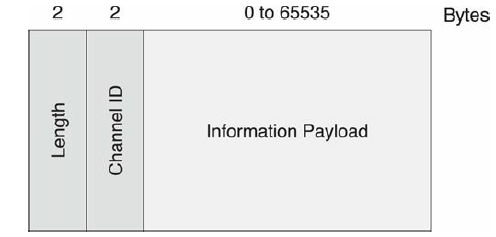


图9-1 L2CAP数据包结构

如图9-1所示，报头包括**2字节的长度字段**和**2字节的信道标识符**。长度字段表示报头后的信息载荷字节数。在经典蓝牙中，信息载荷还可以包含额外的报头和信息，但在低功耗蓝牙的L2CAP层中并没有其他有意义的信息结构。



图9-2 SDU和PDU

SDU (Service Data Unit) 是来自上层的数据包，如ATT层。PDU (Protocol Data Unit) 是包含了L2CAP层信息的数据包。MTU规定最大可接受的SDU包的大小。在低功耗蓝牙中MTU最小是23字节。这意味着，所有低功耗蓝牙设备必须最小支持在空间传输27字节数据包4字节为L2CAP报头，23字节为信息载荷。

**9.4低功耗信令信道**

低功耗信令信道用于主机层级的信令。如图9-3所示，每个低功耗信令信道的数据包均含有一操作码，随后为各种参数。低功耗信令信道支持的命令操作码如下：

* 命令拒绝（Command Reject）
* 连接参数更新请求（Connection Parameter Update Request）
* 连接参数更新响应（Connection Parameter Update Response）

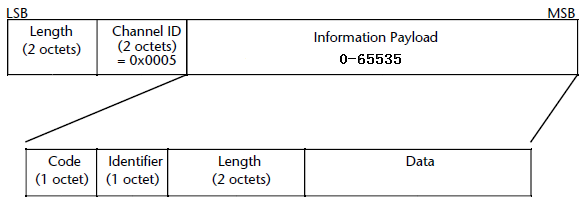


图9-2 L2CAP命令数据包

无论什么时候发送信令命令，其信息载荷总是包含**1字节标识符**。该标识符用于匹配请求和响应。举例来说，如果请求的标识符为Ox35．则响应该请求的数据包也必须使用Ox35作为标识符。如此一来，只要每个请求有不同的标识符，多个请求便能同时发送。由于在用完所有标识符之前不允许重复使用，实现时一般会采用增量操作以确保符合该规定，但有一个例外；不能使用标识符Ox00。使用标识符的一个副作用在于重复的命令可能被默默丢弃。如果命令信道不可靠，上述做法能起一定的作用，不过考虑到命令通常都在可靠连接中传输，所以此规则极少使用。

在低功耗蓝牙中，由于只定义**一种请求命令类型**，并且该请求命令仅用于没有其他请求发送的情况，标识符的逻辑相对简单。

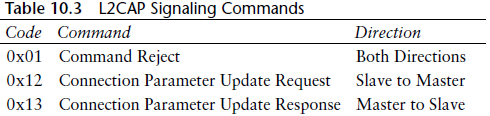


图9-3信令类型

**9.4.1命令拒绝**

“命令拒绝”用于拒绝设备收到的不支持的信息包。该命令与经典蓝牙中的命令拒绝完全一样。它包含一个原因代码以及相关信息。其中的原因代码可以是“命令不理解”或“信令MTU溢出”。

在设备收到了不支持的命令时可以使用“命令不理解”原因代码；当收到未定义的命令时也可使用，这使得设备可以前向兼容未来的规范版本。

当接收到的命令大于23字节时可以使用“信令MTU溢出”原因代码。默认信令信道MTU仅23字节，因此如果接收到的命令大于或等于24字节，则发进拒绝命令作为响应。

经典蓝牙还定义了另一个有关请求的原因代码Invalid CID，不过由于在低功耗蓝牙中没有定义相应的信道标识符，所以没有使用。

**9.4.2连接参数更新请求和响应**

“连接参数更新请求”命令可以让从设备更新链路层的连接参数，如图9-3所示。这些参数包括连接事件间隔（从设备希望主设备允许从设备发送数据包的频率）、从设备延迟（从设备能够忽略主设备的连接事件的最大值）以及监控超时。

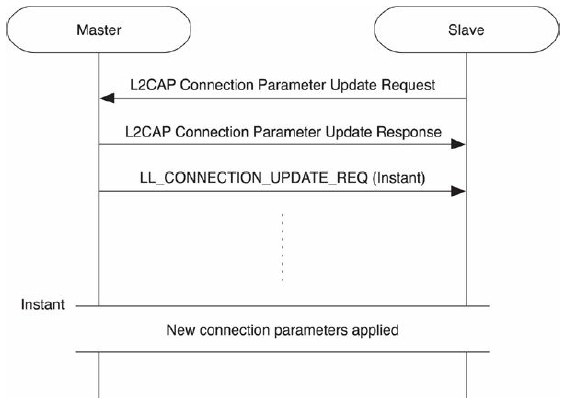


图9-4 L2CAP连接参散更新请求命令

在连接中，如果从设备希望修改当前的连接参数则可以使用该命令。比方说，如果连接事件的间隔有可能太快了，导致过多的电量浪费。这在从设备时延很大时段有同题，但如果不是这样，从设备将会频繁地侦听链路。这在一些情况下是必要的，例如设备间首次绑定、互发多个数据包、探索服务和设备特性等。但在很多其他情况下，尽可能地减少从设备必须侦听连接事件的数量对提高电池寿命至关重要。

**连接参数更新请求命令仅用于从设备向主设备发送**，这是由于主设备随时都能启动链路层连接参数更新控制（Connection Parameter Update Control）规程（详见第7章7.10.1节）。如果该命令由主设备发送，从设备会将其视为一个错误，并返回带有“命令不理解”原因代码的“命令拒绝”命令。

从设备可以在任何时候发送该命令；收到该信息的主设备如果可以修改连接参数，则将返回“连接参数更新响应”（Connection Parameter Update Response）．其中的结果代码设为“接受（accepted）”。随后，主设备将会启动链路层连接参数更新控制规程。

当然，如果主设备不同意从设备的请求参数，它可以发送结果代码为“拒绝（rejected）的连接参数更新响应命令以拒绝请求。此时从设备有两个选择：要么接受主设备希望的正使用的连接参数，要么终止连接。终止连接的做法乍看起来可能让人觉得很激进，但是，假如使用当前的参数从设备将会在一周内耗尽电量，而使用请求的参数则可以持续数年，很明显，合理的选择只有一个。

修改连接参数时，如果要减少主设备拒绝从设备请求的可能性，可以在请求里设置一个可接受的参数范围。经过精心设计的从设备会乐意接受很宽的参数范围。由于主设备可能正忙于实时会话音频连接或者高质量语音连接等任务，它可以接受一定范围内的连接间隔参数。设备可接受的间隔参数会根据当前任务的不同而不同，可能有别于上一次设备连接时的参数。

要提高主设备接受连接参数的机率，还有个方法是从设备提供一个合理的从设备延迟。主设备可以选择最适合的连接事件间隔，从设备则使用最佳功耗的从设备延迟参数。举个例子，如果从设备想每600ms同步一次，它可以请求范围1OOms - 750ms的连接间隔参数，并带上从设备延迟5。如果主设备选择1OOms，则从设备每6个连接事件同步一次；如果主设备选择200ms，则从设备每3个连接事件同步一次，实现其所期望的600ms间隔；如果主设备选择300ms，则从设备忽略每隔一个连接事件同步一次；如果主设备选择400ms，则从设备每400ms同步一次。

**第10章 属性**

数据是宝贵的东西，它的持续时间超过系统本身。

——蒂姆·伯纳新·李(Tim Bemers-Lee)

文明的进步归功于扩展一些重要的、不假思索便能执行的操作数量。

——阿弗烈·诺夫·怀海德(Aifred North Whitehead)

本章节涉及协议栈的两个层次：属性层Attribute Protocol Layer和通用属性规范层Generic Attribute Profile Layer。二者密切相关，因而一般放在一起介绍。在蓝牙技术联盟（SIG）构建低功耗蓝牙时，属性协议ATT的概念最初是在非核心的工作组里被提出，之后才被引入到核心规范中。而在属性协议集成到核心规范时，SIG做出了一个架构性的决定：将其分割为一个抽象协议和一个通用配置文件。从协议的角度出发，抽象化本身是有帮助的，不过它对于理解属性的工作原理毫无益处。把通用属性规范GATT从属性协议里抽象出来，理论上允许在属性协议上架构其他的通用规范，但就目前来看，这一点尚未被业界纳入考虑范畴。

**10.1背景**

在低功耗蓝牙的设计之初，使用什么样的协议就成为了一个难题。协议应当十分简单，因为任何的复杂性都会增加额外的成本和所需的存储空间；同时，协议的数量要越少越好。由此，有人认为使用一种单一的、普适的协议在该技术的起步阶段是最佳的选择。不过这个想法并没有完仝实现。低功耗蓝牙最后使用了三种协议：逻辑链路控制和适配协议(L2CAP)，安全管理协议(SM)和属性协议(ATT)。

随后的目标便是尽可能减少协议的数量，让通用属性规范(GATT)之上的所有服务都使用属性协议（ATT），包括用来提供名字和存在发现的通用访问规范(GAP)。这样，在GATT上构建其他服务的成本被降至最低。

**10.1.1精简协议**

也许你会质疑：为何会有如此多的糟糕的协议。然而，不可否认的是，所有的计算乃至世上的大部分事物都围绕着协议运转。大多数的行为有其自身的协议，例如：载入网页用到了超文本传输协议(HTTP)；传输文件用到了文件传输协议(FTP)；安全的登陆另一台电脑，用到了安全外壳协议(SSH)。每一种协议都专攻于它自己的应用场合，试想要是用HTTP协议传输大量的文件，或是用FTP协议登录电脑，那显然是欠缺效率的。

低功耗蓝牙技术和成堆的因特网协议的最大区别在于，低功耗蓝牙技术并不试图传输多样化的数据类型。考虑到不会用来传输大批量的数据或是流媒体音乐，为其开发一种能够处理有限的几种数据类型的协议就可以。这种协议被称为属性协议ATT，是整个蓝牙技术的基石和构造组件。只有理解了属性协议才能理解低功耗蓝牙技术。

**10.1.2无所不在的数据**

在人们最初探讨低功耗蓝牙技术时，显然当时所有的通信系统都和数据有关。许多事物都包含着数据，低功耗蓝牙技术被其他设备用作接纳、使用这些数据的一种手段。数据可以是任何信息：手机的信号强度、玩具里的电池剩余电量、今天打开了几次冰箱，今早骑自行车走了多远、当时的时间、用耳机聊了多久、最新的新闻、谁给你发了条短信、椅子现在有人坐吗、谁在会议室里、这个月通话时间多长…这一切无一不是数据！

如图1O-1所示，某个蓝牙设备可以公开其无线传输功率，用作连接中断时通知用户的告警值。其中可以包含设备名，便于用户识别具体设备。最后，由于设备通过电池供电，它还可以公开当前的电池余量。

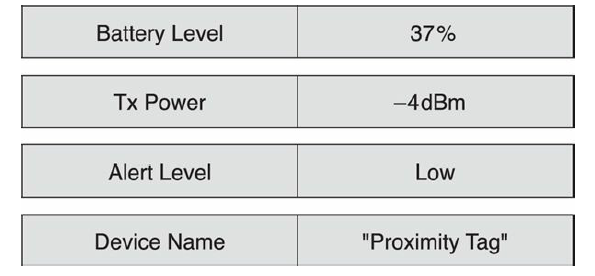


图1O-1低功耗蓝牙设备可能具备的数据类型范例

关于数据，这里需要理解的一个重要概念：一些设备具有数据，另一些设备需要这些数据。这两种设备在角色上的区别对于低功耗蓝牙来说尤为重要，因为它决定了谁是服务者servers，谁是客户clients。提供数据的设备称为服务器，使用其他设备的数据的设备称为客户端。图10-2阐明了这种关系。

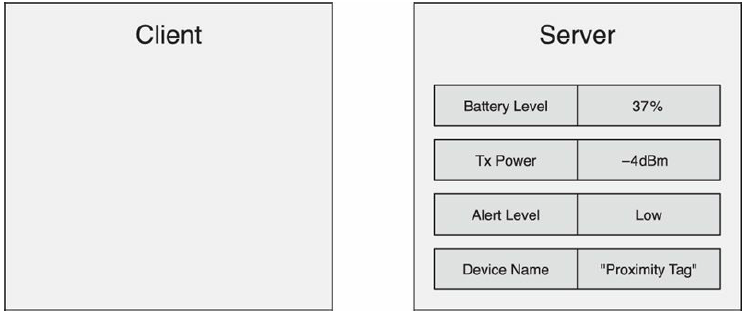


图10-2服务器提供数据；客户端使用这些数据

**10.1.3数据与状态**

还有一个需要理解的概念是：数据( data)与状态（state）这两者有着显著的差别。数据是一个值，它反映了某种客观性质，比如某种测量的结果、读数。数据可以是温度计涮出的房间温度，也可以是供暖系统测出的室内温度；它们都可以被不同的设备测量出来。而状态则反映了某个设备的当前状况或处境：它在做什么、是怎么运作的。设备的状态只有它自己知道，并靠自身维持。温控器测量室温，温度的度数则反映了房间温度的状态。

这本书里，“状态”措的是保存在服务器上的信息；“数据”指的是服务器传给客户端的信息，或是客户端当前持有的信息。因此，服务器可以认为是一个持有许多状态信息的设备。客户端是读写该状态信息或者在本地缓存为数据的设备。由于客户端在获得服务器的数据时后者的状态可能已经发生了改变，因此，客户端的数据不具有权威性。在阅读后续章节的时候，请牢记设备具有状态，并且这些状态保留在服务器上。

**10.1.4几种常见的状态**

低功耗蓝牙用到了三种不同种类的状态类型：外部状态、内部状态与抽象状态。

物理测量值反映了物理传感器或者类似接口的当前状态+让我们设想一个浴室里的体重

计，如图10-3所示，该设备的测量值包括了当前室温、当前剩余电量以厦上一个用户的体

重。这些都可以视为“外部”状态，由于采用了外部传感器测量，每一次读取时它们都有可

能呈现不同的读数。

[黼扭■曩i—两j■]

圈10-3铡量物理量

38◆第三部舟土 机

第二种状态类型是内部状态（见图1 0-4）。一些设备具备状态机，用来反映它们的内

部状态。有别于反映传感器的外部状态，内部状态是设备当前的工作状态，可以是电话的

通话状态，当前时间是否正在通过GSP接收机同步，或者灯光是否正在根据先前的指令

调节明暗。

困10-4内部状态

最后一种状态叫做抽象状态（见图10-5）。这是一种只和某一瞬时有关的状态信息，它

不反映设备当前的外部或者内部状态。抽象状态的例子有：命令灯光切换开关状态、要求设

备立刻报警、启动设备同步、取消进行中的同步等。在属性协议里，这些抽象状态称作控制

点（control point）。一般来说它们不可读取，但可以被写人或被通知。

图ia-s抽象状态

I开始同步

l取消同步

10.1.5状态机

有趣的是，有限状态机可以明确地使用属性协议和可公开的状态来表示。状态机反映了

设备的内部状态，并且有一个或多个外部的辅人接口。这些外部辅人为瞬时命令，根据其他

状态信息或行为来改变状态机的状态。这是一个抽象状态，或称为控制点。

结合内部状态和控制点，便可完整地反映设备上有限状态机的工作流程与行为。可以从

两个角度来看这种有趣的结合：第一，通过公开有限状恋机、它们的输入以厦当前状态，可

以明确的公开某设备的行为；通过开放的输入接口，其他设备可以与该设备姬。第二，可

以定义有限状态机的所有可能的行为，包括无效的行为；任何设备都可以通过任一控制点向

状态机进行输入，而状态机将执行相应的预设行为。

来看一个简单的电灯状态机。可以将电灯看做一个包含两种状态的有限状态机：亮和灭

（如图10-6与图10-7所示）。我们既可以读取当前的状态，叉可以通过擦写该状态来改变灯

的开关。我们也考虑到这个状态机会有三种可能的输入：打开，关闭和开关切换。状态机大

部分的输入都能被映射到一个有效、合理的下个状态。例如，发送“打开”指令会点亮原先关闭的灯，发送‘戋闭”指令会让点亮的灯熄灭。但是，向已经亮着的灯发送‘j于P’指夸，

尽管是一个无效的动作，电灯仍将保持点亮状态。类似地，“熄灭”指令将让原先关闭的灯

保持熄灭。此外，“切换”命令会把亮的灯熄灭，把原先熄灭的灯点亮。

图10-6灯的状态转变

打开

关闭

圈10-7灯的状态机

打开

有趣的是，在用无线指令改变电灯状态的示例中，使用“切换”输入能显著地减少所需

要的通信成本。在不公开抽象控制点的情况下，之前的做法需要电灯开关读取当前的开关状

态，在内部切换数据，再把新的值写到灯的状态。该过程最少需要发送三条不同的信息：请

求当前的状态，返回当前的状态，以及把灯设置为新的状态的命令。相比而言，通过在电灯

的状态机中加入切换指令，井命其抽象控制点接纳切换指令，大部分的通信信息都能省路。

这样一来，开关仅需向电灯发送一条切换指令，电灯服务器就会执行状态的切换，而开关无

需知道切换前后的两种状态。

公开状态机使得无线传输信息变得更有效率，同时提高了互操作性，因为不可能命令状

态机进人那些投有定义过的状态。因此，通过定义所有的可能状态以及这些状态在任何的输

入下将执行的行动，便能实现一种最优的交互性协议。

10.1.6腮务和规范

从经典蓝牙到低功耗蓝牙，最有趣的转变是服务和规范的体系结构。经典蓝牙里的规范

和协议大多定义的是行为与交互指南，它们极其复杂，糅合了许许多多不同的概念。其中，一个最大的问题是这些规范倪定义了两种设备类型，位于链路的两端，再对各自的行为分别

定义。粗略地看来，这种做法似乎很有必要：好比体有一部手机和车载（免提）套件，我们必

须基于具体应用为其定义各自的工作方式以及交互方式。不幸的是，这也造成了一些麻烦。

首当其冲的问题在于，现有的规范对网络中的设备自身的行为定义不够明确。换言之，

即便看起来两个设备都定义了各自的行为，但有时候设备自身到底应该怎么做其实并不清楚。

速会导致很多误会，两个设备都误以为对方应该执行操作，结果操作迟迟未能执行。譬

如，免提规范( HFP)中提到“免提组件(HF)和音频网关(AG)在任何必要的情况下，都

可以发起音频连接。”问题来了，究竟谁应该去初始化连接？是免提组件还是音频网关呢？

如果它们同时都初始化会如何々这真是交互操作的噩梦。

显而易见的解决途径是，设备单独定义各自的行为，使之明确自身应做的事情。

第二个问题是，现有的规范几乎无法用于当初没有预想过的新场景。由于规范定义了两

个设备的交互方式，设备稍稍不同，规范便难以使用。即使对规范自身来说，这也是一个难

题。例如，免提规范定义了一部手机和对应的车载套件，但现实中最多的应用组合是手机和

茸机。手机并不知道它在和车载套件还是和耳机通话，由于对方可能是车载套件，因此手机

会持续向对方设备发送用户界面的状态更新，以便其更新界面显示。这浪费了电量，因为假

如对方是耳机，根本不会关心信号强度从四格掉到三格。显然，解决方法是定义每一台设备

的行为，而不用知道它的功能。

低功耗蓝牙使用了一种截然不同的方法来解决上述问题。首先，它采用了纯粹的“客户

端一服务器”的结构，针对不同的用例对服务器和客户端的行为单独进行描述。服务器的行

为在服务器规格书中定义，而用户的行为在另一规范说明中定义。正如图10-8所示，通过

一个属性数据库，服务器规格书定义了需要公开的状态以及通过属性可以实现的行为。

图IO-S规范与服务结构

服务包含一些可读属性，用于返回历史数据或当前数据，还有一些可写属性用于接收命

令。此外，一些属性显示了有限状态机的状态，只要与控制点相互结合，所有的行为将一览第10章属 性+ 14

无余。规范说明定义了如何使用一个或多个服务来满足某种特定的应用需求（例如，如何通

过服务器上的属性数据库配置某个服务的属性，使得该服务能执行客户端所期望的操作）。

将服务器和客户端区分开的最大的好处是，服务器的行为将是预先定义并可知的。它只

会做服务说明中定义的“该做”的事情，不会关心客户端将怎样去使用它。这意味着服务可

以独立执行单元测试，而与客户端无关；任何客户端可以在必要的时候使用这些服务。举个

例子，假设有一种提供时间的服务，某个客户端可用其获取当前时间；另一个客户端周期性

地读取当前时间来判断自身的时钟漂移；其他客户端还可以请求其使用GPS接收机以便获

得最精确的授时。你瞧，这个时间服务并不关心客户端在做什么，它仅仅提供该服务罢了。

再举个例子，如图10-9所示，一盏电灯能够通过两块数据来提供照明服务：当前电灯

的物理状态和让客户端控制状态的抽象控制点。电灯开关通过执行“照明规范”来获知如何

查找照明服务，如何读取当前灯的状态，以及如何控制灯的开关状态。电灯开关不会误解这

些公开的行为，因为所有的照明服务都具有相同的行为。

图10-9照明规范与服务

客户端也从这个系统中获益。服务器昀行为由服务定义，而不由规范定义，因此规范变

得相当的简洁。客户端规范从本质上是一组有关发现、连接、配置和使用各种服务的规则。

同样，服务也包括了如何执行客户端要求的一系列操作的标准步骤。客户端及其规范可以

使用任意的服务组合来达到目的，比如既可以使用时间服务，叉可以使用温度服务，从而

在时间维度上监控温度的变化。在这里，客户端省去了自身的实时时钟，也不用实时地去

收集数据。

举一个具体的例子，一个房屋安全系统得知当前没有人在家，但房屋的主人想让屋子随

机地开关照明灯来造成家中有人的假象。这可以通过定义一个新的规范来实现，定义的过程

仅需下面给出的几条简单指令：

loop forevar :

wait <random period f.om 10 secoods t0 3 hours>

.onnect to a light :

send 'to{Jgle" to .ont.ol point 类似地，如果休是一个办公室负责人，想在职员下班后确保关fjl所有照明灯，这也能用

另一个规范来实现，指令如下：

ioop forever :

wait <until start ot next Horxing daY>

.onnect to lightS :

send 'on' to control point

diaconnaet

urait <until end of working day>

.o…t to lights：

send "off“t..ontrol point

diS connect

客户端对多种服务的组合正是低功耗蓝牙技术里最强的概念。每种单独的服务十分简

单；而服务之间的不同组合提供了系统的复杂度和多样性。为了这个目标，服务必须是“原

子”(atomjc)的。“原子”在本书中表示一种服务只执行特定的操作（该特定的操作在系统

中是和原于一样的不可分割的最小的单位）。把服务封装为独立的“原子”，各个客户端都能

重复使用这些服务来做不同的事情。

这便解决了传统蓝牙规范的第一个问题：传统规范中难以定义一个明确的状态。用单独

的规格书定义服务，这确保了它们各自的原子状态，于是每种服务的行为数变得很少。更少

的行为意味着能更明确地定义该服务所要做的事情，因拙经典的单元测试方法也变得更加适

用。同时，由于原子服务有着明确的状态，客户端可以依齄这种状态。

这也同样解决了传统规范的第二个问题：传统规范无法适用于没有被定义过的场合。通

过将应用场景分解为具备已知行为的原子服务，可以很容易地定义使用这些服务的客户端。

客户端既能使用其中的一种服务，也能复用多种服务。因为每种服务都有自己明确的行为，

多种服务复用的复杂性降到了最低。也正由于服务是原子的，在不同的服务间跳转时，行为

不会“伤筋动骨”。这使得服务可以用任意的顺序来组合使用。

这也意味着，客户端能够在一样设备上用全新的方式使用各种服务。它们跳出了规范的

范畴，服务定义了设备如何交互，而规范不过是某些应用场合下执行交互的标准做法。因

此，一台设备不但可以使用温度和时间服务，还可以使用太阳能电池板的发电服务来判断当

前的天气状况。或许这不是一个SIG定义的规范或用例，但它可以被某个厂商所定义。将

服务器的行为分解为服务，使得客户端有可能组合出一些妙趣横生、新颖实用的功能。

10.2属性

要理姆属性协议，首先要理解什么是属性。宽泛地讲，属性是一条带有标签的、可以被

寻址的数据。在后续的章节里，我们将深入介绍属性的含义以及茌实际操作中的使用。10.2.1属性概述

如图IO-IO所示，属性由三种数值组成：属性句柄、属性类型和屑性值。

2字节 2或6字节 。- 512字节

图lO-iO属性的结构

10.2.2属性句柄

一台设备可以有许多的属性，例如温度传感器可能包含温度属性、设备名称属性和电池

电量属性。表面看来，通过属性类型似乎足以判别某种属性：比如使用温度属性来获取温

度，通过设备名称属性来获取设备名等。但是，如果设备包含了两种温度属性，比如一个室

内温度传感器加上室外温度传感器，情况会变得怎样。这时你便无法直接读取温度传感器，

而必须读取第一个或第二个温度属性。考虑到可能有任意多个温度传感器，问题将变得更加

复杂e。

为了解决这个同题，我们使用了一个16位的地址，也就是属性句柄。有效的句柄范围

从Ox0001 - OxFFFF。Ox0000为无效句柄，不能用于寻址属性。你可以根据自己在软硬件

或嵌入式方面的背景，把句柄相应地想象为内存地址、端口号、属性值对应的硬件寄存器

地址。

10 2 3属性类型

可以被公开的数据有许许多多的类型：温度、压强、体积、距离、功率、时间、充电状

态、开关状态、状态机的状态等。所公开的数据的种类称作属性类型。为了区分如此多的数

据类型，一申128位的数字被用来标识属性的类型。这个唯一的标识码就叫做通用唯一识别

码（删Ⅲ）o。

128位的UnD相当长，设备间为了识别数据的类型需要发送长达16个字节的数据。

为了提高传输效率，蓝牙技术联盟( SIG)定义了一十称为“蓝牙UUID基数”的128位通

用唯一识别码，结合一个较短的16位数使用。二者仍然遵循通用唯一识别码的分配规则，

只不过在设备阃传输常用的UUID时，只发送较短的16位版本，接收方收到后补上蓝牙

UUID基数即可。

蓝牙UUID基数如下：

e用于写f车书的电脑就拥有害达23十不同的温度传感器。

。通用唯一识别码在RFC 412/中定义，在功能上和IS。『IEC 9834-8等同。例如要发送的16位识别码为Ox2AOI

00002AOI - OOOO

．完整的128位的UUD便是：

iOOO - 8000 - 00805F9834FB

谈到16位的UUID，通常不直接使用其数值，而是冠以一个名称并加上书名号。例如

用《include》来表示数值为Ox2802的UUID。类似这样的名称定义还有很多。UUID并没

有定义自身的用法，为了增加人工调试时的可读性，低功耗蓝牙使用的那部分uUID被分为

下列几组：

口Ox1800 - Ox26FF用作服务类通用唯一识别码

口Ox2700 - Ox27FF用于标识计量单位

口Ox2800 - Ox28FF用于区分属性类型

口Ox2900 - Ox29FF用作特性描述

口Ox2AOO - Ox7FFF用于区分特性类型

10.2.4属性值

属性值用于表示设备公开的状态信息。属性值的长度可以从。字节到最长512字节，但

某些类型的属性值的长度则是固定的。属性值对属性协议来说并不重要，但它对于上层，包

括通用属性规范和在其之上的服务与规范来说有着相当重要的意义。

1服务通用唯一识别码

每一种服务都能用一个UUID米辨认。它既可以是16位的UUID，也可以是完整的128

位的LnnD。在I6位的UUID申已经分配了3840种服务，而128位的ULnD可以容纳近乎

无穷e的专有服务。

2．单位

许多公开的数值反映了某种传感器测量的物理量，因此有必要为所有可能的数值定义

相关单位的UUD。这些单位由国际计量局制定，亦称国际单位制(SI，源自法语Symme

Intemational d'UnWs)。这使得低功耗蓝牙传感器采集的数值可供其他同样采用国际单位制的系

统所使用。要注意国际单位制用的是公制单位，人们还定义了英制单位的UuD。所吼速度既

可以用米每秒来表示，也能用千米每小时(km/h)或英里每小时(mph)来表示。

3．■性类型

用于识别属性类型的UUID中包含了一些最基本的属性类型，一般用于通用属性规范，

而非具体的服务。这些属性类型包括：

口首要服务

口次要服务

口包含

口特性

Ⅱ以容纳近10'”s种服务，牡地球上的设备使用好些年。\*lO幸属 性+ 145

4特性描述符

一些服务公开的数据包含了额外的信息。这种额外的信息便被冠以特性描述符。例如一

个描述相关数值的格式的信息（单位与表示方式）便是所谓的描述符。

5特性类型

特性类型是16位UUID中使用最多的一组。服务为公开的每一类数值都分配了一个

“特性类型”UUID。客户端从而能够发现服务器提供的所有不同类型的数据。每一个特性类

型都有既定的格式和表达方式。现有的分组里可以定义多达22 015种特性类型，无需使用

那近乎无穷的128位UUID。

10.2.5数据库、服务器和客户端

一组属性的集合称作数据库。数据库可以很小很简单，小至仅包含六种属性e．也可以

很大很复杂。属性数据库的复杂度不在于其层次，而在于服务和规范如何使用这些属性。

属性数据库总是位于属性服务器当中；通过属性协议、属性客户端和属性服务器进行通

信。不管是低功耗蓝牙还是经典蓝牙设备，每台设备最多只有一个属性服务器和一个属性数

据库。对低功耗蓝牙设备而言，由于强制要求属性数据库支持通用访问规范( GAP)服务，

每一部低功耗蓝牙设备都包古有一个属性服务器和一个属性数据库（见图IO-II）。

这样一米，要公开类似设备电池电量这样的简单倍息，需要的开销是相当少的。这是因

为每一台设备已经包含了一个属性数据库，要加人电量指示的服务，只需向当中追加三种或

更多属性。假定每一台设备最初有六种属性，为电量服务再加入三种属性是相当简单的。

10.2.6属性许可

一些属性服务器上的属性含有可读或可写信息。为了提供这类访问限制，数据库中的每

一个属性都含有一个许可。许可自身可以分为三种类型：使用许可、认证许可和授权许可。

使用许可央定了某一属性所能执行的请求的类型。回到之前的例子，电灯的状态可以为既可

读又可写，电话的通话状态只能是可读的，而电灯的控制点只能是可写的。相应地，任何人

都能读取灯的状态，但只有被信任的设备才能写入该状态，获取通话状态需要授权，而想要

写入灯的控制点状态则需要认证。

要注意的是，属性许可只适用于属性值，并不适用于属性句柄和属性类型。每一台设备

都有发现其他设备公开的所有属性的许可，包括属性句柄和类型。这是为了使设备在得到认

证、获取授权之前事先获知对方设备是否有自己可以使用的功能。倒如先判断某设备是否支

持灯的控制点属性，然后再决定是否认证。这不但使初始设备和服务的发现变得容易，而且

保护了设备在服务中提供的私有信息或保密信息。

e曩小的属性教据库必须包古以下的六种属性：‘OAP般务）的‘首要服务)、储备名)的‘特性）、C鞋备名）

的值、< pl jW )的‘特性）、‘外观）的值、（o vrr E务）的t首要服务’。但这样的数据库社能公开足簪

的状态，故没有实用价值。146．第三部务主 机

┏━━━━━━━┳━━━━━━━━━━┳━━━━━━━━━━━━━┓

┃ 属性句柄 ┃ 属性类型 ┃ 属性值 ┃

┃ 。oooi ┃ 首要服务 ┃GAP服务 ┃

┃ OX0002 ┃ 特性 ┃设备名 ┃

┃ OA0003 ┃ 设备名 ┃“接近标鉴” ┃

┃ Ox0004 ┃ 特性 ┃外观 ┃

┃ oxoo05 ┃ 外观 ┃标签 ┃

┃ OX0006 ┃ 首要服务 ┃OATT服务 ┃

┃ OX0007 ┃ 首要服务 ┃发射功率服务 ┃

┃ oXa008 ┃ 特性 ┃盅射功率 ┃

┃ OX0009 ┃ 发射功率 ┃\_4dBm ┃

┃ OrOOOA ┃ 首要服务 ┃立即报警服务 ┃

┃ oXaooB ┃ 特性 ┃报警级别 ┃

┃ omoc ┃ 报警缎别 ┃ ┃

┃ O~OOD ┃ 首要服务 ┃连接丢失服务 ┃

┃ omOOE ┃ 特性 ┃报警级别 ┃

┃ fffooaF ┃ 报警级别 ┃ ┃

┃ oxoolo ┃ 首要服务 ┃电量服务 ┃

┃ oxooll ┃ 特性 ┃电诎余量 ┃

┃ W0012 ┃ 电池泉量 ┃7i% ┃

┃ Cr0013 ┃ 特性表达格式 ┃8位无符号整形．。，百分比 ┃

┃ 陆0014 ┃ 特性 ┃电池余量状态 ┃

┃ Qx0015 ┃ 电池余量状态 ┃75%放电中 ┃

┃ O加OJ6 ┃ 客户端特性配置 ┃oxoool ┃

┗━━━━━━━┻━━━━━━━━━━┻━━━━━━━━━━━━━┛

图】O-Il属性数据库

定义的使用许可如下：

口可读

口可写

。可读并可写

当客户端读取一个属性时，首先检查它的使用许可，判断属性值是否可读。如果不可

读，客户端会收到一条属性值不可读的错误信息。类似地，在客户端试图写入某属性时，必

须具备对应的写人许可，否则也会收到不可写的错误信息。

定义的认证许可如下：

口需要认证

口不需要认证

每当使用一种属性时，不论读写都要验征其认证许可。如果需要认证，客户端发起认证

请求；如果不需要，属性值必须允许访问，并受到其他许可的约束。只有之前通过认证的客

户端可以访问需要认证的属性；如果未认证的客户端试图这样做，它会得到一条未认证的错革lO章属 性．147

误信息。

若客户端收到来认证的错误时，它会有两种选择：跳过当前请求，把错误码返回给应用

程序；或者使用安全管理器重新发送请求来试图通过认证。要注意的是，错误码并不表征所

需要的认证级别。因此客户端需要请求认证或是提高自身的认证级别来获取想要的属性值。

有趣的是，客户端能够完全掌控发起认证的时机和方式，服务器也不用保存接收到的请

求的状态。在经典蓝牙中，认证的行为是在L2CAP信道建立时发起的。当响应者在需要认

证的信道上收到请求．它会试图接人到需认证的通道，它会保存这条请求，发送一条“挂起

（pendingr回复，启动安全程序，最后再返回去处理最初的请求。该过程既复杂又消耗内

存。在低功耗蓝牙中，服务器只需尽可能回应每一条请求；复杂的认证工作由客户端执行，

并在必要时重发最初的请求。

定义的授权许可如下：

0无授权

口授权

授权和认证略有区别，尽管发起的操作与之类似，但是当来授权的客户端试图使用某属

性时，会收到授权不足的错误码，而它对此错误无能为力。

授权是服务器的一种行为。服务器可以选择是否向某一客户端授权，使其能够使用一系

列的属性。授权与否完全由服务器来决定，更重要的是，客户端无法用任何倍号来请求服务

器设备的使用者给自己授权。所以，只要客户端试图使用一个需要授权的属性时，服务器要

么向客户端请示是否对该客户端授权，要么立刻拒绝其请求。在重新发起请求之前，客户端

只能等待。通常，客户端设备的用户会在服务器设备上把自身加入授权列表，然后再重新发

起请求。

10.2.7接入属性

客户端可以通过使用下列任意一种消息类型来访问属性数据库的各个属性：

口寻找请求

口读取请求

口写入请求

口写人命令

口通知

口指示

客户端先使用“寻找请求”来寻找属性数据库中的属性，随后才能使用效率更高的基于

句柄的请求。

发送读取请求用以读取某个属性值。要确定读取的具体属性，应指定一个或多个属性甸

柄或者句柄范围，以及属性的类型。

发送写入请求用以写入某个属性值。该请求常常会用到一个属性句柄和要写人的数值。

也可以事先准备多个需要写^的数值，然后在一个原子操作中一并写人。 这里的任意一种请求都会使属性服务器发送一条响应。如果还需请求更多的数据，客户

靖还须发送另一条请求。例如要读取的属性值太长，无法用一条响应来发送，客户端就要发

送“宏读取请求”来获取诙值的后续部分。

为了降低服务器设备的复杂度，客户端一次只能发起一条请求。只有收到了上一条请求

的响应以后，才能发起下一条请求。

还可以用写人命令来写入属性值。该命令不要求服务器回复，正因为这样，写人命令可

以在任意时刻发送。这说明可以把命令写入公开的状态机的控制点。

除了这几种信息，还有通知和指示两种，它们都是由服务器自发地向客户端发送属性

值。通知可以在任意时刻发送，它包含了要通知的属性的句柄和当前的属性值。类似地，指

示也包含了属性句柄和属性值，不同的是它需要客户端回传一条属性确认信息。客户端通过

回复确认信息，既表明了指示已经进达，还告诉服务器设备可以发起下一条指示；相比之

下，通知则可以在任意时刻发送。

10 2 8原子操作和事务

在客户端和服务器之间发送的每一条属性协议的信息都隶属于某个事务（transaction）

的一部分。事务可以是一条请求及其回响应，也可以是一条指示及其确认信息。事务的重要

性在于它限定了连续的事务之间霈要保存的信息的数量。这意味着当某设备接收到一条请求

后，它无须为了处理下一条请求而保存当前请求的任何信息。

关于事务模型，另一个重要的性质是在一个事务结束之前不能发起新的事务。例如，设

备发送了一条读取某属性的请求，在收到相应的响应之前，不允许其发送另一条请求。事务

仅针对单个设备。发起事务的设备虽然不能初始化另一个事务，但可以处理其他配对设备的

请求。

1命令和通知

属性协议里还有两种信息，分别是命令和通知。借助这两者，设备得以将信息发送给另

一台设备，并且在发起下一条命令或通知之前无须等待相应的响应。当你正处于某个事务，

却需要发送一条特定的命令或通知，这种性质便很有用。举个例子，假设你刚向某特定设备

发送了一条读取请求并正在等待响应，这时需要向同一设备写入一个值，这时你就可以用到

写人命令。

命令和通知无需响应或确认，这也意味着发送设备无法知道信息是否被对方收到并处

理。对一些请求需要响应、指示需要确认的应用场景，这种性质并不适合。然而对另一些应

用场景，它们却能满足要求。无需响应与确认的一个副作用在于设备可以借此发送任意多的

信息。实际上，它们完全可能用命令和通知向对端设备发起泛洪攻击。对于这样的情况，如

粜某设备没有足够的缓冲区来储存、处理命令或通知，它可以选择丢弃这些信息。总之，必

须记住逮两类信息并不可靠。萆io幸属 性‘149

2准备写入请求与执行写入请求

上文提及的事务规则还有一个例外，那便是准备写人请求与执行写人请求消息。用这些

信息，设备可以准备一系列的写入操作并在—个事件中执行。从事件的角度来看，每一条准备

写A请求及其响应属于—个单独事务。另外，在准备和执行操作的中途，允许插人其他请求。

这种命令有两种有趣的用法：长写入和棱验写人。这两种准备写入请求不仅包含了要写

入的属性的句柄和值，还包含了需要写入的数值在目标属性值中的倡移量。这意味着你可以

在一次操作中用一系列的准备写入请求来写入某个很长的属性值的不同部分。

另一个有趣的用法是准备写人请求的响应也包含了原请求中的属性句柄，偏移量和待写

入部分的数值。表面上看逮浪费了带宽，但由于响应中包含的信息必须和原请求相同，这能

够避免错误的出现。

低功耗蓝牙有一定的数据保护能力，所有载荷中的二进制位都用了2位循环冗余校验

码(CRC)，最多可检测5位错误。如果一个收到的包中有6位错误，CRC错误地接收这个

包的可能性极小。下一层保护是每一个加密包中的32位消息完整性检查(MIC)值。虽不能

确保检出所有的无效包，但它应该能够丢弃掉绝大部分通过了CRC校验的错误散据。因此，

一个错误的包能够通过所有校验的可能性极小。

有时哪怕很小的可能性也会导致严重的后果。例如你用低功耗蓝牙控制一座城市的下水

道外流阀门，你本想发送“关闭”指令，但阀门收到的却是“完全打开”，然后公园和孩子

们的操场被污水所淹没，这肯定不是你想看到的。

正是出于这样的安全考虑，准备响应中才需要重现原请求中包含的所有数据。由于数据

包在不同的方向上，使用不同的无线信道和不同的加密包计数器发送，响应和请求里的相同

数据产生相同的错误的可能性几乎为零。当然，如果响应的数据有所出入，你可以使用执行

写入请求中的“取消”代码来取消整个写^操作序列，随后重新准备写人。

10 3分组

通用属性规范协议仅仅定义了一个属性的平面结构。每一种属性有一个地址——句柄。

但现代数据组织的方法论需要的是一个更加复杂的结构而不是这样简单的平面结构。这便是

属性规范的意义所在：它定义了多组属性，而不只是一组属性。

为了理解这样做的必要性，让我们分析一下它的实现步骤。我们可以把属性用页面来划

分，每一页定义了一些数值。每一页面为特定的用途而定义，例如用一个页面描述设备，设

备使用电池的话则要用另一个页面，再用一页面来公开温度。对复杂的设备．同题会变得有

趣起来：试想如果有两组电弛会怎么样？如果有两个温度传感器会怎么样？

过去的几十年里，软件工程上最重大的飞跃是逐步引进了面向对象范例。本质上，它通

过“方法”来描述实体的数据，并将其聚台到一起以控制数据的行为。使用面向对象体系结

构的主要优点是每一个对象都是自包含的。

接下来用一点时间定义一些术语。谈到面向对象的编程，你也许会想到接口、类和对象：接口是对外部行为的描述，类是对该接口的实现，对象是该类的实例。例如，一辆车是

汽车类的一个实例，这辆车实现了驾驶接口。但并不是所有车的对象外观都一样；汽车多种

多样，但严格说来，它们都具备相同的基本驾驶接口，比如方向盘、油门和刹车。驾驶接口

彼此相同，但实现该接口的类可以互不相同，一个类也能有许多实例，看看堵车时的车龙便

会明白。

在低功耗蓝牙中，服务和特性都用到了分组。使用服务声明来对服务进行分组，用特性

声明来对特性进行分组。

服务是一种或多种特性的组合；特性则由一种或多种属性组成。

10 4服务

在软件工程中，当体定义并实现某个粪的行为时，一旦该类的接口确定下来，系统的其

他部分便能重新使用基于该类的对象。这也意味着若该类中出现错误，你只需一次性的修复

它，系统中其他使用该类的部分便能立刻从该修复中获益。

为了保证类的重复使用，你必须定义一个不可变的抽象接口。“不变性”是一个程度较

强的字眼，意味着“随时间保持不变”。只有“不变性“才能蟮确保接口长期地存在。如果

接口是不定的，该对象的客户端就要用更多的时间去弄懂该对象有什么接口，而不是去弄懂

它耍做什么。

当需要对接口做出变动时，面向对象的系统一般使用继承舶方式；新的类有着自己的接

口，它继承了旧类的行为以后，根据要求在该行为中做相应的扩展或变更。确保接口的不变

性可以使其重复使用许多年。

在低功耗蓝牙中，通用属性规范定义了两种基本的分组模式：服务分组模式与特性分组

模式。服务等同于一个具有不可变接口的对象，一般包含一种或多种特性，并能引用其他服

务。特性是数据的单位或公开行为的单位。这些特性是自解释的，这样一般的客户端便能读

取、显示这些特性。

因此，一种服务只是一组特性和通过它们所公开的行为的集合(如图l0-12)o正是这些

特性和它们对应的行为组成了该服务的不可变的接口。

┏━━━━━━━━┓

┃ 膳劳A 、 ┃

┣━━━━━━━━┫

┃I 特性 } ┃

┣━━━━━━━━┫

┃j 行为 } ┃

┗━━━━━━━━┛

阿10-12一个不可变的服务接口由特性和行为构戚第10章属 性．151

如图lO-13所示，服务可以引用其他服务。这样的一个简单的概念赋予了该体系结构巨

大的括力。引用是指一种服务可以指向另一种服务。引用可咀有多种说法：这种服务扩展了

原服务的行为；这种服务使用其他服务；这种服务和其他服务结合在一起实现更多的服务。

让我们依次剖析这几个概念。

┏━━━━━━━━┓

┃ 服务^ ┃

┣━━━━━━━━┫

┃1 特性 l ┃

┣━━━━━━━━┫

┃ 1 行为 I ┃

┣━━━━━━━━┫

┃ j包含 ┃

┣━━━━━━━━┫

┃ 孽L务B ┃

┣━━━━━━━━┫

┃ l 雠 l ┃

┣━━━━━━━━┫

┃ I 行为 I ┃

┗━━━━━━━━┛

图10-13服务A引用了服务B

ia.4.i扩展服务

服务A已经使用了多年，现需要扩展( cxtcnd)其功能。由于服务A是不可变的，我们

无法简单地在原服务中添加新的行为。因此必须保证不触及原有服务的同时做必要的扩展。

为此，我们定义了新的服务AB，包含所需的扩展行为（如图10-14所示k不过为了保证成

千上万的仅支持服务A的现有设备具备后向兼容性，我们也必须在每一部提供服务AB的

设备上包含服务A的实例。

现在假设我们的一台设备上有服务AB的两个实例，AB:I与AB:2。我们在该设备上也

需要有两个服务A的实例．A:l与A:2。但哪一个服务A从属于服务AB呢？为了解决这个

疑问，我们需要在每一个服务AB中包含服务A的特定实例，以便使扩展变得明确(见图

10-15)。152．第三鄙舟主 机

图10-14服务AB扩展了服务A

图10-15服务AB的两个实例扩展了服务A

一台只识别服务A的旧设备仍旧可以发现之前的服务A的两个实例，并像以往那样使

用它们；该旧设备会忽略其他服务AB的实例，仅能够使用那些未扩展的行为。

一台能同时识别服务AB和服务A的新设备将发现服务AB的实例，并遵循它们对服务

A实例的引用。这样该新设备便能够使用服务AB定义的新的行为。第10幸再 性+ 153

一部在和旧设备通话的新设备会先试图发现服务AB，失败后再试图发现服务A；这样

该新设备便能退而求其次，使用两者通用的服务A。

虽然看起来有些复杂，但这其实比下列方案妥简单得多：先将服务A扩展为新的版本，

并利用特征位来表明特定服务支持的特征——这样做可能非常的复杂，因为要对每一种可能

的特征组合进行测试。相比而言，之前的扩展方法意味着每一种服务都是自包含、不可变

的，服务之间的关系变得显而易见。使用这些服务的设备都能确定其行为。原先服务器的不

变性也保证了对旧设备的兼容。

1 0.4.2其他服务的重用

另一个重用其他服务的方法是引用(reference)。引用再简单不过了：服务A想使用服

务B的行为和状态信息，那么服务A其需引用服务B即可。这不是经典的面向对象意义上

的重用，它更像一个指向另一个类的实倒的通用指针。

这在原服务和被引服务都有许多实例时极为有用。比如存在服务A:l、服务A:2、服务

B：1和服务B:2（如图10-16）。没有了引用，我们便无法判断A:l重用的是B:l还是B:2．

亦不知A:2重用的是B:I还是B:2。通过包含对其他服务的引用，我们能辨认出被重用的服

务实例。

┏━━━━━━━━┓

┃ 雁务^：2 ┃

┣━━━━━━━━┫

┃ l — I ┃

┣━━━━━━━━┫

┃I I 行为 l ┃

┣━━━━━━━━┫

┃ 』包畲 ┃

┣━━━━━━━━┫

┃ 魔务B：l ┃

┣━━━━━━━━┫

┃ l 特性 I ┃

┣━━━━━━━━┫

┃ l 行为 l ┃

┗━━━━━━━━┛

10 4 3结合服务

┏━━━━━━━━━┓

┃I 胁“ I ┃

┣━━━━━━━━━┫

┃l l 特性 ll ┃

┣━━━━━━━━━┫

┃㈠ 行为 l 1 ┃

┣━━━━━━━━━┫

┃ l包古 ┃

┣━━━━━━━━━┫

┃ 服务B1k ┃

┣━━━━━━━━━┫

┃ l — I ┃

┣━━━━━━━━━┫

┃ I 行为 l ┃

┗━━━━━━━━━┛

囝IO．16服务A重用了服务B的行为和特性

最后一种引用方式较前两者复杂些，因为它把接口从实现中分离了出来。有时，我们需

要将与某个服务相关的两个独立的服务实例结合起来(combine)．并使其具备额外的行为。54．摹三部分主 机

为了用服务来满足这种需求，我们必须定义第三种服务，该服务分别引用了原来两种服务。

例如有某服务的两个实例：服务A:1与服务A:2．需要把它们融合在一起并具备额外的“组

合”行为。为此，如图10-17所示，你可以实例化第三个服务——服务C．它引用了Al与

A2两种服务。

图10-17服务C结合丁服务A的两个实倒的行为

在处理服务A:l与A:2对，服务C可以公开所需要的行为；它封装了结合后的服务的

行为。例如灯光服务和日光传感器服务可以结合在一起来提供这样一种服务：让灯只能在投

有日光的情况下打开e。服务结台后的状态机会包含一些基础服务没有的状态。独立的A:1

与A:2服务仍然具有各自不可变的行为。这意味着服务C必须能够清楚地区分组台服务的

行为和独立服务的行为。

1 0.4.4首要还是次要

关于服务，最后一个概念是它可以有两种不同的“风格”：它既可以是首要服务，也可

以是次要服务。通过前面对服务的描述以及服务设计的介绍，想必你已经明白，有时候必须

通过服务来公开设备的外部行为，有时候又必须通过服务来公开许多功能，可以采用多种方

式重用多次的功能。

公开设备功能的服务通常属于首要服务。例如你有一台支持胍务B的设备，则服务B

将被实例化为首要服务。如果你需要设备提供在服务D中才有的额外的信息，但该信息又

和设备自身功能无关，那么可以将服务D实例化为次要服务（见图10-18）。于是，次要服

务可以理解为一组无须用户理解的行为与特性的封装。

e一个更戏剧性的捌于是让烤箱在其中的童物若火的时蛱自动关断。当然我家没发生过这种事┏━━━━━━━━┓

┃ 苗饕服务B ┃

┣━━━━━━━━┫

┃ 1 一 l ┃

┣━━━━━━━━┫

┃ I 行为 l ┃

┣━━━━━━━━┫

┃ 』 ┃

┣━━━━━━━━┫

┃ 次要\_I努D ┃

┣━━━━━━━━┫

┃ I — I ┃

┣━━━━━━━━┫

┃ I 行为 l ┃

┗━━━━━━━━┛

围10-18首要服务和次要服务的关系

\*10幸属 性0 155

使用属性协议可以快捷有效地寻找到首要服务。它们既可以有“双亲”服务，也可以是

独立的服务。次要服务只能通过引用来发现，必须通过另一个服务来指向它们。这也意味着

可以建立“服务树”，首要服务在树的顶端，每一个分支指向首要服务或次要服务，后续的

分支同样指向更多的首要或次要服务。

首要服务可以指向另一个首要服务，如图10-19所示。例如，服务的扩展允许公开某服

务的新“版本”，并实现这些服务间的后向兼容性。

首要服务可以指向次要服务，从而使用次要服务公开的行为。次要服务可以指向另一个

次要服务或首要服务。不过，从次要服务指向另一个次要服务的情况很少见，因为次要服

务通常是服务树上的叶节点。从次要服务指向另一个首要服务的情况就更少见了，但也是

存在的。

首要服务还有一十优势：当客户端寻找某特定服务时，往往能迅建地查找到首要服务。

如果设备的首要服务只有一个实蜘的时候，这种优势会更加明显：例如在一台设备上，某服

务只能有它的一个实例，客户端设备只要对该服务进行一次快速的搜索，就能明确地判断该

服务是否存在。

这种忧化有着显著的优点：当简单客户端只想查找某一首要服务的实例时，它所需的开

销降到了最低。该简单客户端既不用读取设备上的整个服务列表，也不用判断服务之间的关

系就能使用它想要的服务。若没有上述优化，那么每一个简单客户端都需要遍历整个服务树

才能决定如何最有效地使用设备上的服务。这显然浪费了大量的宝贵资源：不仅浪费了通信

消耗的能量，还浪费了用来储存中间结果与做计算消耗的内存。┏━━━━━━━━┓

┃ 首要服务^ ┃

┣━━━━━━━━┫

┃l 桃 l ┃

┣━━━━━━━━┫

┃ l 行为 l ┃

┣━━━━━━━━┫

┃ I ┃

┣━━━━━━━━┫

┃ 首X:ESC ┃

┣━━━━━━━━┫

┃ l 她 I ┃

┣━━━━━━━━┫

┃ l 行为 I ┃

┗━━━━━━━━┛

图10-19首要服务

1 0.4.5即插即用的客户端应用

服务模基还有个有趣之扯，即可以使用某设备上的服务树的集合，并搜寻能使用这些服

务树的应用。为此，通用客户端会先进行一次完整的服务枝举：先是首要服务，随后是它们

与所引用的服务之间的关系。一旦建立了服务树，就能将服务的“森林”递交给应用库，由

此获得能够与该服务森林的整体或局部协作的应用。

有些应用也许仅支持一个首要服务。有些应用也许支持一个扩展了另一个首要服务的首

要服务，它可以是原服务的扩展，也可以是该应用的第二个版本。而另一些应用也许支持多

于一个服务树。这些应用也许会在某个有趣的多合一应用中展示这些服务的信息，也可能在

客户端上用某种创新的方式将这些信息结合在一起使用。

举个例子，假设某服务器支持如图10-18与10-19所示的服务。该服务器支持首要服务

A，它包含另一个首要服务C；一个首要服务B．它包含了次要服务D:最后还有一个独立

的首要服务C。表示为：A(C)，B(D)．C。随后客户端可以使用这些信息来判断哪些应用将

支持这些服务。它将对照每一个应用的服务列表来判断哪些应用支持该设备。有的应月也许

仅支持一种服务（应用l仅支持服务C），有的应用可能支持包含了C的扩展服务A．有的

应用，例如应用6同时支持形如A(C)与B的服务树，还有的应用，例如应用7支持包含了

次要服务D的服务B（如图10-20）。

另一种途径是使用通用应用，例如支持所有的服务的应用8。这类应用通常不像专门开

发的应用那样具有良好的交互性，但是对于那些尚未开发专门应用的设备却很适用。服务器 客户端

图10-20将一个服务器上的服务映射到某客户端的应用

第10幸属 性+ 157

通用客户端的行为是通过服务的组合以及纯客户端服务器模型来实现的。不过，这里的

关键还在于不可变的服务。试想，假如服务会发生变化，没有了巳知的确定行为，那么使用

该行为的通用客户端便不知道该如何写入。

通过制约系统中每一个独立部件，降低各自的灵活性，整个系统便具备丁最大的灵活

性。正是这些不可变的独立部件的结合使用，提供了市场产品所需要的多样性和灵活性。

10.4.6服务声明

我们使用服务声明对服务进行归类。图10-21显示了首要服务和次要服务的属性以及属

性类型。所有列于该服务声明之后、下—个服务声明之前的属性都归类于且隶属于当前服务。

属性类型 属性值

广—i][二\*一l

图10-21首要与次要服务声明

正如之前所定义的，首要服务封装了设备的功能。次要服务协助首要服务实现它的行

为。所有次要服务都被一个首要服务所引用。原因很简单：这保证了客户端的简单性。

简单客户端是不具备用户界面的设备，但它们仍可以使用配对设备上的服务。简单设备可以只搜索首要服务并找到它需要的服务。它不须遍历该设备公开的整个服务树。实际上为

了顾及简单客户端，属性协议做了相应的优化：允许它们搜索某一特定的首要眼务。

有些服务具备帮助服务，后者能够协助前者公开其行为或状态。例如大多数医疗设备会

包含设备信息；在自动化和电池的使用场景中，也需要有类似的设备信息。此时，可以定义

一个设备信息服务，且只需定义一次，之后便能反复使用多次。这也使得简单客户端能够不

去关注它们的设备信息；它们只要在搜索首要服务的时候忽视这些次要服务即可。

服务声明的值是服务的UUID。它既可以是16位的蓝牙UUID也可以是128位的

UUID。设备可以安全地忽略掉任何不理解的服务。例如某设备包含了一个次要服务，设备

不理解该服务的16位或128位UUID．那么该服务声明中的所有属性都可以被忽略。为了实

现这一功能，属性协议允许利用属性句柄的范围来发现服务，但只进一步处理已知的服务。

1 0.4.7包含服务

次要服务必须逐个发现。为此，每一个服务可以有零或多个包含（include）属性。包含

的声明总是紧随着服务声明，并列于该服务的其他属性之前。包台的声明也包括了引用服务

的甸柄范围，以及引用服务的胍务UUID（如图10-22）。这使得设备能够快速的发现引用服

务、包含的属性和服务类型。由于没有参照物，这里并不说明该引用服务是首要服务或是次

要服务。

属性类型 属性值

图10-22包古声明的结构

由于包含的值使用了四个字节的句柄．128位的服务UnD将无法放人用来查找“包含”

服务的标准响应数据包。因此，如果包含的服务具有128位UUID，服务ULnD将不会作为

声明值的一部分。这也说明需要读取额外的属性协议来获知所包含的服务种类。

引用服务的种类可以是首要服务也可以是次要服务，一个首要服务可眦引用另一个首要

或次要服务，一个攻要服务同样也能引用另一个次要或首要服务。

通过使用图10-23中所示的数据库，可以建立上述包古四种服务的例子“A(C)，B(D)”。

一个服务最初发布为首要服务，假如随后扩展了其他首要服务，则后者需要引用最初的

服务。最初的首要服务不能转化为次要服务，因为那样做会使得旧的客户端无法发现原先的

旧服务。\* 10幸属 性’159

┏━━━━┳━━━━━┳━━━━━━━━━━━┳━━━━┓

┃ 向桶 ┃ 煳 ┃ ┃ ┃

┃OrWOJ ┃首要艇务 ┃融努B ┃ ┃

┃ 00002 ┃包古 ┃k0200，诹02凹，服务D ┃弋 ┃

┃Qoioo ┃首要服务 ┃服务^ ┃ ┃

┃Qr0102 ┃包音 ┃啪，0．Q柏317，服务C ┃ ┃

┣━━━━╋━━━━━╋━━━━━━━━━━━╋━━━━┫

┃ ┃次要最务 ┃服务D ┃J ┃

┃ ┃ ┃ ┃．一 ┃

┃ ┃首要服务 ┃服务c ┃．．一 ┃

┗━━━━┻━━━━━┻━━━━━━━━━━━┻━━━━┛

图10-23服务A{C)，B(D)的属性敷据库范例

10.5特性

将—个服务的属性归类到一起．可以更好地说明这些属性的组合如何为行为提供一蠢的

接口。低功耗蓝牙的体系结构也使得通过属性分组公开服务的状态和行为成为了可能。

特性只是一个数值。它可以是当前的温度、某人骑行的距离或是时间同步有限状态机的

状态。当然，特性的意义远不止这些。特性还需要指明数值的数据类型，数值是否可读或可

写，如何配置数值的指示、通知、广播，数值的含义等。

由此可见，特性包含三个基本要素：

口声明

口数值

口描述符

特性首先从声明开始，包括了特性的所有其他属性；数值属性包含了该特性的实际值；

而描述符包含了该特性的附加信息或配置。

一个常见的问题是：为何数值是包吉在特性里的一个属性，而不是一个独立韵属性？答

案确实根复杂。特性不仅仅是一个数值，它还包含权限、附加的配置和可作为特性组成部分

的描述性数据。原本我们可以在属性协议中加^额外的语意来获墩这些信息，但没有必要为

了这些少有的特殊情况而把协议变得复杂。

因此，为了保持协议所公开的属性的平面结构，人们在通用属性规范中定义：把设备结

构与它的特性分离。这让获取某些特性信息变得更加复杂，但对大多数特性而言，信息获取

变得简单了许多。

筒而言之，特性由特性声明、特性数值和零个或多个描述符组成。

10 51特性声明

要建立一个特性，首先耍用到特性属性，它包含三个字段：特性性质、数值属性句柄和160．第三郜舟主 机

特性类型。如图10-24所示。

图10-24特性声明

特性性质是一个八位宇段，确定了特性数值属性对一系列操作的支持情况，包括：读

取、写入、通知、指示、广播、命令、签名认证。如果设置了该字段的某一位，就能用相应

的规程来访问该特性数值。此外，如果设置了特性的广播位，就必须有服务器的特性配置描

述符。类似地，如果设置了通知位或指示位，必须要有客户端的特性配置描述符。

特性性质宇段还包含一个扩展性质位。如果含有额外的性质，并且相应的描述符数值超

出了最大长度，则应使用扩展特征位。这些附加性质包含在特性扩展性质描述符中，共有两

类：可靠写入支持与可写辅助。可写辅助是个很有意思的特征位，它决定了特性用户描述符

是否可写。

特性数值句柄字段是包含特性数值的属性句柄。有了该字段，客户端只需返回特性声明

给服务器，后者使用该声明便可以立刻获取属性的数僮。若段有该字段，客户端便需要对属

性进行一次额外的检索，并快速地判断声明后面的哪一个属性才是想要的数值。目前，数值

属性紧随着特性声明，但将来的改动可能会是在声明中包含该数值属性的句柄。

最后的字段是特性UUID，它用来识别特性数值的类型。该UUID的类型必顼和包含该

特性数值的属性一致。实际上，通过向服务器发送额外的请求也能获取特性UUID相关的信

息，特性UUID是该信息的副本，但发送这些无线协议消息浪费了能量。相比之下，直接在

声明中加A特征数值类型的信息显得更有效率。

也有人曾经质疑：为何特性数值属性类型不像数值那样设为静态的UUID，这样还能减

少之前描述过的麻烦。原因是还有别的更理想的优化途径（图10-25）。例如某个简单客户

端只想获取某设备的电池状态，它可以直接询问该电池状态，也可以在某个字段中搜索包含

电池状态UUID的特性，相比之下前者更有效率。这些简单的优化构成了声明的结构，如下

所示：

s e rvi ce\_ra n ge -. dl S . ove r\_p r ima r Y\_ S e rvi .e\_b y\_UU I D { "B a t te . y\_S e rvi .eb

eha.s - di acove. al l cha.a.te.is t i.s o f a serviee { sanrUe .anqe )

foreach cha. in char. :

if Char. UUid-- <Batte.y Leval)+ :

bat te.y\_level - read cha.ac ter t stic value ( ctiar . nanae )

il摘单的方法苹10幸属 性‘161

圈10-25特性的例子

10.5.2特性数值

特性数值是一个属性，它的类型必须符台特性声明的特性UCrID字段。除此以外，它和

普通属性无异。最大的区别在于，特性性质字段公开了特性数值属性可以执行的操作类型，

该类型也可以包含于特性扩展性质描述符中。

对每一个特性，对应的说明文档描述了特性舶格式。特性自身不包含行为，如果要确定

某特性实例公开的行为，必须查阅描述该特性的服务规格书。

10.5.3描述符

—个特性可以包含任意多的描述符。大多数描述符是可选的，正如上面解释的，我们可

以根据特性声明或服务规格书来选用指定的描述符。

特性可以包含如下的描述符：

口特性扩展性质

0特性用户描述

口客户端特性配置

口服务器特性配置

口特性表示格式

口特性聚合格式

1特性扩展性质描述符

该描述符用来获得附加的扩展性质。目前定义了两类扩展性质：可靠地写人数值的能

力，以及写人“特性用户描述”描述符的能力。

2特性用户描述描进符

设备使用该描述符将一字符串与某一特性相关联。它适用于那些用户可以自己配置的设

备。例如，用户可以配置恒温装置，使其用文字描述正在楼房的哪个房间测量温度。一些设

备可能有许多温度传感器，出于可配置性考虑．有必要在特性的层面加人这样的配置。162．第三卑舟主 机

3客户端特性配置描述符

支持通知或指示的特性必须使用客户端特性配置描述符。该描述符是一个2位的数值，

分别用于设置通知与指示，但不允许同时设置。关于如何通知或指示的细节并不在核心规格

书中定义，它们由服务规格书定义。

4服务器特性配置描述符

该描述符与客户端特性配置描述符类似，但它还有一个广播位，设置该位将使设备广播

该特性所属服务的相关数据。另外，广播的时间由服务所决定。

有趣的是我们不能广播某个单独的特性。相反，设置了广播位后，由包含该特性的服务

来定义所要广播的数据。一些服务有可能将多种特性用于广播；服务也同时定义了接收方如

何辨别广播中的各种特性。

你可能会感到疑惑：为何在特性中会有一个位用来启动该特性的广播，而不是在实际中

直接广播该特性。这是因为特性自身不包吉任何行为；也就是说，若脱离了服务，广播的特

性数据是投有意义的。这就好比收到“温度：20.50C”这样的数据是毫无意义的，而“室温：

20.50C”或“汽车引擎温度：65eC”这样的数据才阐明了温度服务的对象。

5特性表示格式描述符

使用通用属性规范的目标之一是支持通用客户端。遥用客户端定义为这样的一类设备：

它能读取某特性的数僮，并在无须理解它们的含义的情况下向用户显示这些值。通用客户端

可以连接一台冰箱，并显示其内部温度，它不必知道内部温度大于looc对冰箱来说情况不

妙。与此不同的是，规范定义了客户端可以与冰箱上的温度服务变互，并且定义了当温度超

出有效范围时，客户端要怎么傲。

为了让通用客户端工作，它们必须能够找到可以显示给用户的特性，随后在充分理解这

些特性数值的基础上显示给用户。发现所有可用特性的方式之一在于寻找特性声明中已知的

属性类型；可读的特性通常也是可用的。不过，判断通用客户端能否能使用某种特性，最重

要的方法还是利用特性表示格式（characteristic pr∞nat6ion format）描述符。有了它，通用

客户端才能安全地读取并显示特性值。

表示格式是个多字段的数值，包含了以下的字段：

口格式

口指数

口单位

口命名空间

口描述

格式是标准数据类型的枚举，它决定了数值的构成方式。常见的格式如下：

口布尔类型

口无符号2位数与4位数

08位到128位的无符号或有符号整数值

口标准IEEE-754浮点数，有两种长度，大多数高端电脑使用

口基于整型的定点数，有两种长度，医疗设备使用

口UTF-8与UTF-16编码的两种字符串格式

如果特性的格式不符合上述类型，我们可以采用不适明数据结构或使用随后介绍的组合

格式。

跟在格式后面的是指数。该字段只对整数有效；它定义了一个固定的指数，可以加在整

数上传给用户。该指数的底数是10．这样便能在最后的输出环节中控制十进制小数点的位

置，而不必使用复杂的数学表示。特性的值可以用如下的公式来表示：

显示值一特性值×iOjart

例如，特性值是OxFD94，表示格式为有符号的16位整型加上指数-2，那么显示值便是：

-620×lo-z= -620×O.OI= -6.20

下一个宇段是单位宇段。单位是由指定的数字说明文档定义的UUID，它包括了许多的

单位。在前面的例子中，如果单位是摄氏度，那么显示值便是-6.2℃。显然，我们假定通用

客户端知道每一个单位UUD所代表的单位名。

最后的两个字段应视作一个字段。命名空间与描述字段携带了数值的附加信息。命名空

间是一个字节，它指明了控制描述字段的机构。描述字段则是一个16位的无符号数。

描述字段仅仅是一个用来修饰特住的“形容词”，据此用户能够判断某个数值对应于设

备上哪一种特性。举个例子，假设有个温度计，它有内部与外部两个探针。该温度计会有两

种温度特性，唯一的区别是在特性表示格式描述符上将是“内部wr或‘'#F部的”描述字段。

描述与单位字段还必须根据用户所使用语言来定制，以帮助他们理解特性。例如某特

性的单位是“weight (kg)”，描述是“hanging”，那么根据用户使用的不同语言所定制的

字符串就会是“hanging weiliht”、“bengende vekt”、“vje§蛐je teiiina”、“riippuva paino”、

“penjant de pes“中的一种。

6特性聚合格式描述符

有些特性值远比单一的数值复杂得多。譬如用标准的符号来表示地球上的某一个位置，

通常应包含经度值与维度值，二者组合在一起便构成一个“坐标值”。为了构造类似的复杂

的特性值，特性聚合格式描述符允许引用多个表示格式描述符来解释组合值的每一个字段。

使用上面的例子，该坐标特性具有两个特性表示格式描述符（分别描述经度与纬度）和

特性聚合格式描述符，后者用正确的J嚼序引用了前两者。于是通用客户端便能正确地解读该

特性值的格式并将数值显汞给用户。

要注意的是，特性聚合格式描述符所引用的多个特性表示格式描述符不必从属于同一特

性。它们甚至可以来自不同的服务或设备。引用它们只是为了表达特定的格式；与它们自身

所属的特性和聚合格式无关。64．第三部分主 机

10.6属性协议

属性协议是非常简单的协议，客户端通过它可以发现并获取属性服务器上的属性。它由

六种基本操作构成：

口请求

口响应

口命令

口指示

口确认

0通知

客户蜡通过向服务器发送请求，以要求服务器执行相关操作井进行响应（如图10-26所

示）。客户端一次只能发送一条请求。这减少了服务器的复杂度，降低了对内存的要求，使

得建立一个属性服务器只需少量的代码。请求有两种可能的响应：与请求直接相关的响应，

或是说明请求为何失败的错误响应。

图10-26属性协议

客户端也向服务器发送命令，但不一定会收到响应（如图10-27）。客户端可以发送命

令让服务器执行某种操作，比如让其换一个电视频道，这种情况下不必立即收到响应。如果

响应延迟发出，也可以使用命令，此时命令将以指示或通知的形式发送。

∈己…

图IO．27属性协议命令

服务器通过向客户端发送指示，让客户端获悉某属性的数值（如图10-28）。指示和请

求有些类似：客户端也需要返回一个确认响应。另外，服务器一次只能发送一条指示，这也意味着只有在接收到上一条指示的确认以后它才能发起下一条指示。

图10-28属性坍议指示

服务器以发送通知的形式来告知客户端某属性的数值（如图10-29）。通知不要求响应

在这点上它与命令类似。

图10-29属性协议通知

由于命令与通知都不要求响应或确认，因此它们的发送次数没有限制。当发送了过多的

命令与通知，使服务器或客户端无法一一处理时，接收端可以将它们忽视。由此可见，命令

与通知是不可靠的；而请求和指示则可看作是可靠的，因为它们都要求响应以确认接收端已

经处理了该请求或指示。

10.6.1协议消息

图10-30列出了属性协议中的所有属性协议数据单元(PDU)。大多数消息既有请求

PDU也有响应PDU。例如凄取消息具有读取请求PDU与读取响应PDU。每一种消息都有

一系列的参数，概括如下：

下面的章节参考了图10-11中的关于属性数据库的例子。

10.6.2交换MTU请求

在低功耗蓝牙连接中，属性坍议默认的MTU的长度为23字节。如果设备想发送更大

的数据包，那么它就要协商一个更长的MTU。只有客户端可以发起这种请求。但鉴于许多

设备兼具客户端和服务器的功能，这不是个问题。客户端的请求包含客户端接收的MTU长

度；服舟器请求则包含了服务器接收的MTU长度。两者的长度必须相同。对客户端与服务

器而言，二者提供的接收MTU长度中较小的那个即是连接将会使用的MTU长度。图10-30属性PDU

该值并非通过协商得到，实际上，交换MTU请求与交换MTU响应中发送的数值是固

定的，这让实现尽可能地简单。对于既是客户端也是服务器的设备，其客户端接收MTU与

服务器接收MTU必须使用相同的数值，也就是说，不管是谁发起了MTU交换，得到的

MTU都一样。这也意味着如果双方设备同时发起MTU交换，结果会和异步发起的情况完

全一致。该限制也说明了在随后的连接中发起另一次MTU交换是没有意义的，因为结果总

是相同的。

对仅支持默认MTU的简单设备而言，服务器总是回复不变的23字节作为接收MTU。

显然仅支持默认MTU的客户端不会发趋MTU交换。换言之，所有的设备都必须保证至少

支持默认MTU。

10.6.3查找信息请求

查找信息请求和回复用来查找一系列属性的句柄和类型信息（见图10-31）。这是唯一一

个能让客户端发现任意属性类型的消息。幕IO幸属 性’167

图10 31查找信息请求

查找信息请求包含两个句柄：起始句柄与结束句柄。它们定义了该请求用到的属性

句柄的范围。为了找到所有数值的属性，该请求的起始句柄将是O@OOI，结束句柄设为

OxFFFF。通常，响应只能包含该范围内的部分句柄；因此，为了查找所有的属性，必须使

用一连串的查找信息请求，相邻的请求中的起始、结束句柄前后相接。

查找信息响应包含句柄一类型对。它有两种可能的格式，对应于低功耗蓝牙使用的两

种长度的UUD：对于16位UUID的格式，允许在一个查找信息响应中包含至多5个属性

句柄一类型对；对于128位UUID的格式只能在响应中包含一个句柄一类型对。显然，当

MTU比默认值大时，包含的句柄一类型对的数量也将随之增加。

对于响应，另一个有趣的地方是同一个响应报文不能同时包含16位与128位的UUID。

否则，每一个句柄一类型对就要消耗一个额外的字节，使得默认MTU响应中所能包含的16

位UUID的数量减为4个。此外，标准属性服务器中的绝大多数LnfiD都是16位的，也极

少会在一个响应中同时包含16位与128位UUID．因此没有必要提供这种灵活性。

10.6.4按类型值查找请求

使用按粪型值查找请求和回复可以根据给定的类型与数值查找相应的属性。该请求包含

两个句柄：起始句柄与结束句柄。它们定义了该请求用到的属性句柄范围。对任一属性，当

它的甸柄处于起始与结束句柄范围内（包吉结束句柄）．并且具备丁请求中所指定的类型与

数值，那么该属性将在响应中被返回。

响应可以包古一个或多个句柄，各自对应某个属性。假如请求的类型是分组属性，首要68．第三郜争主 机

服务、次要服务或特性，则响应也将包含该分组的最后一个属性的句柄。不过，由于特性声

明的格式不含静态的数值，因此无法使用该请求来查找特性。

按类型值查找请求的主要用途是查找特定的首要服务。发送该请求时，客户端将其中的

类型设为首要服务，并将数值设为该服务的UUID。随后的响应将包含查找到的各个首要服

务的实例的句柄范围。某些特殊的服务只会在服务器中实现一次，针对它们的响应只会包含

一个句柄范围。

用该请求也能够查找次要服务，不过当前的低功耗蓝牙不采用这样的方法。次要服务总

是包含在其他服务中，使用按类型读取请求将更为适合。

10 6 5按类型读取请求

按类型读取请求髓在句柄范围中读取某属性值。当客户端仅知道属性的类型而非句柄时

可使用该请求。请求包古了起始、结束句柄和需要读取的属性的类型。响应将培出符合的句

柄和数值。

符合请求的类型且落在句柄范围内的每一个属性都将被返回。响应中包含一系列的句柄

和对应的数值。对具有相同长度数值的属性，响应做了相应的优化：比方说要读取某服务的

所有引用，假如被引属性的数值长度相同，这时仅会返回一条响应；假如被引属性的数值长

度不同，在第一条响应中只包含那些具有相同长度昀属性值；如果有好几个相同数值，但一

条响应只能容纳其中的一部分，这部分属性将被首先发送，随后，客户端需要更新请求中的

起始句柄并重新发送谓求，以便获取剩余的其他属性。

按类型读取请求用于搜索被包含的服务，并通过特性类型来发现服务中所有的特性。它

也被用来读取已知类型的特性值。举个倒子，如果你只想快速地读取设备的电池电量，一种

便捷的方法是使用按类型读取请求，将该请求中的类型设为“电池电量”即可。随后．响应

中将包含电池状态的特性值的句柄以及它的数值。可见，当客户端想要快捷地读取一些数

据、获取属性类型的数值时，只需发送一条快速的请求即可，不必去发现其他的特性。

10.6.6读取请求

读取请求也许是属性协议中最简单的请求。该请求包含一个句柄，响应将返回该句柄对

应的属性值。只有在客户端已知属性句柄的情况下，才能使用该请求读取属性值。

10.6.7大对象读取请求

有时属性值太长，无法装入一个读取响应，须使用大对象读取请求来获取剩余字

节。“大对象”这个词来源于数据库术语，完整的含义是二进制大数据对象( Binary Large

Object．BLOB)。大对象读取请求不光包含属性句柄，还包含属性值在整个数据中的偏移量。

萁响应将从属性偏移量开始，包含尽可能多的属性值。

在获得了属性值的前22十字节后，假如客户端还想获取后续的属性值，则应使用大对

象读取请求。下一条响应将返回第23个字节到第44个字节i如此继续，直到客户端读取到完整的属性值。

该请求用于读取长特性值与长特性描述字。

弟10幸属 性+ 169

10.6.8多重读取请求

多重读取请求用来在一个操作中读取多个属性值。该请求包古一个或多个属性句柄，响

应则按照请求的顺序返回相应的属性值。以体重计为例，假设它可以测量人的体重值与体重

索引值，使用多重读取请求便能同时读取这两个数值。

因为响应中的数值之间没有界限，因此，在请求中除了最后一十属性允许可变的长度，

其他属性必须为定长的属性值。也就是说，如果客户端用一个多重读取请求来读取三个属

性，前两个属性的长度必须固定，最后一个属性的长度则可以变化。

如果客户端请求读取的属性值长度超过了响应数据包所能承载的最大长度，那么无法故

人响应数据包的数值将被丢弃。

1 0.6.9按组类型读取请求

最后一种读取请求是按组类型读取。它和按类型读取请求类似，也包括一个句柄的范

围，读取时将其视为一个属性的类型来处理，只不过属性的类型必须为分组属性。其响应包

含所读取的属性句柄、属性分组中最后一个属性以及属性的数值。

这意味着，如果分组类型是首要服务，它将返回所有首要服务声明的属性句柄、该首要

服务中最后一个属性以及首要服务声明的数值。因此，可以仅凭单个请求米发现设备上的所

有首要服务、与之关联的属性甸柄的范围以及这些服务的类型。

如同其他返回多个句柄一数值对的响应，如果返回的值长度可变，那么只有长度相同的

属性值将在第一个响应中被返回。因此，客户端必须再次发送请求，更新起始句柄来发现想

要获取的下一个属性。

10.6.10写入请求

写入请求类似于读取请求。该请求包吉了一个句柄和一个打算写^该属性的数值。响应

中将确认该数值已经被写入。

10.6.11写入命令

写人命令和写入请求类似，区别在于写入命令没有响应。写人命令包含要写入的属性的

句柄和要写入的数值。

当无祷响应时，可以使用写入命令。此外，因为该命令可以在任何时刻发送，即使刚发

送了一条请求还未收到相应的响应，对命令的发送时延有较高要求时，该请求也适用。

10.6.12签名写入命令

签名写入命令和写人命令类似，区别是前者包含认证签名。通过这样的机制，发送端可以在向服务器发送写入命令时认证自己，而无须加密通信连接。签名写入命令适用于以下两

种场合：1）发起加密将显著增加数据连接的延迟；2）发起加密将显著增加简短且无需加密

的数据的送达成本。

认试签名由签名计数器和消息认证码构成。签名计数器对设备间发送的每一条消息赋予

不同的值，不论捎息发送的间隙连接中断与否。消息认证码源自NIST（美国国家标准技术

研究所）的特别版本800-38B中的CMAC函数，该函数使用了通常在设备绑定时可分发得

到的连接签名解析密钥( CSRK)。

签名计数器是个32位长度的值，也就是说一个CSRK可以用于执行四十亿次的签名写

入。使用签名计数器可以抵御消息重放攻击，因此，假如签名写入命令的签名计数器值和先

前的值相同，该命令会被忽视。消息认证码长度为64位，该码位于句柄、数值和签名计数

器之后。注意，服务器需要为每一个客户端保存其最后使用的签名计数器。

10.6.13准备写入请求与执行写入请求

准备写入请求与执行写^请求实现了两种功能。第一，它们提供了长属性值的写入功

能。第二，它们允许在一个单独执行的原子操作中写人多个值。

属性服务器包含一个准备写人队列，其中保存有准备写入请求。队列的大小面独立配

置，但通常它足够储存所有需要准备写入的服务。只有在收到执行写入请求时，准备写入的

值才会写入属性，也就是说执行写入请求给出了执行这些准备写人操作的开始信号。

准备写入请求包含句柄、偏移量和部分属性值，这和大对象读取请求类似。这说明客户

端既可以在队列中准备多个属性的值，又可以准备一个长属性值的各个部分。这样，在真正

执行准备队列之前，客户端可以确定某属性的所有部分都能写入服务器。

准备写入响应也包含请求中所有的句柄、偏移量和部分属性值。之所以近乎偏执地重复

这些数据是为了保证数据传递的可靠性。客户端可以对比响应和请求的字段值，保证准备的

数据被正确地接收。对一些应用来说，这个验证环节是必要的。回忆先前的下水遭阀门的例

子，对这种重要的操作最好发送两次来保证可靠性，避免由于一个位的错误而导致未处理的

污水淹没游乐园里的孩子们。

想知道客户端有多么的谨慎，不妨让我们向服务器发送一个字节，看看这个字节的数据

受到了哪些保护：当客户端通过加密连接向服务器发送准备写入请求时，将有一个24位的

CRC码来避免位错误，还有一个32位的MIC值来保证该请求发自正确的客户端；准备写

入响应也用同样的24位CRC码和32位昀MIC值来保护。

这意味着，为了保护属性值仅为一个字节的准备写入请求，总共用到了112位的错误检

查值：保护字节与数据字节的比例是14：1。此外，如果客户端对准备写入响应不满意，它

可以快速地发送一条标识位为“取消”的执行写人请求，终止准备写人队列，并重新开始值

的写入。

一旦接收完所有的准备写入请求，服务器将拥有一个随时可以执行的准备写人队列。客

户端发送标识位为“立刻写入”的执行写入请求，随后服务器将在一次原子操作中写人所有值。属性将按照其准备的顺序写人。如果客户端多次准备了同一个属性的值，那么服务器将

按照顺序向该属性写入这些值。这意味着如果使用准备队列配置硬件状态，例如硬件需要先

后执行禁用、配置、重新启用的操作，那么可以用一个准备队列来实现：在队列中先将对应

的属性写入“禁用”，接着写入“配置”，之后在准备队列的末尾写人“启用”，随后执行该

队列。这样便能在一个原子操作中实现该硬件的重新配置。

10.6.14句柄值通知

当服务器想要向客户端发送快速的属性状态更新时，它可以发送一条句柄值通知。这是

服务器能够发给客户端的两种消息中的一种，并且是不要求响应的那种。服务器可以在任何

时刻发送该通知，同时该通知也是不可靠的。

句柄值通知包古属性句柄和数值。因此该通知是服务器发往客户端的一条消息，用来告

知某属性的当前值。它是属性协议中最重要的消息之一。它不仅让客户端能够有效地从服务

器获取当前属性数据库的更新，而且也被用来通知客户端有限状态机的变化。

若没有通知，客户端就要频繁地询问服务器属性值的变化情况。一旦配置了通知功能，

客户端只需等待服务器通知它属性值的更新情况。由此可知，比起周期性询问的方法，客户

端用通知获取新的值显得更如迅速有效。

一般来说，服务器会为所有的绑定设备配置通知。这样做的好处在于当某客户端重新与

设备建立连接时，该设备上的服务器能够立即将自己当前的状态通知该客户端。例如，可以

将某设备的电池电量配置为通知；一旦客户端重新连接到设备，并且电池电量已经改变．那

么客户端能立即收到电量的通知。

因为通知不具备任何确认机制，它们可以在任何时刻发送，而不管当时正在进行什么样

的操作。即使客户端与服务器正在执行请求或指示那样的复杂交互，也能够发送通知。因

此，通知适用于必须立即向客户端发送信息的情况。

10.6.15句柄值指示

句柄值指示类似于句柄值通知。它有着相同的属性句柄字段和数值，不同的是客户端收

到指示以后应回复。服务器一次只能发送一条指示，并且只有收到确认响应后才发起下一条

指示。

句柄值确认不含任何数据，主要用于流控。因为具备了确认机制，指示被视为可靠传

输。一旦服务器收到了确认信息，它便能确定客户端收到了该指示。

10.6.16错误响应

当某设备无法完成请求所要求的操作时，它可以发送错误响应。倒如某设备通过发送写

入请求试图写入某属性，但该属性是只读的，那么服务器应发送错误响应来给出请求失败的

原因，而不是发送写入响应卢明一切正常。

错误响应包台导致错误的、和请求相关的所有信息，例如请求失败的属性、导致错误的首要原因等。一旦客户端收到错误响应，它会认为该响应与其发送的最后一条请求相对应。

因此，错误响应是另一种利用响应消息来终止请求操作的方法。这也说明对每一条可以发送

的请求都会有两种可能的响应：失败错误响应与成功响应。比方说读取请求有可能收到错误

响应或读取响应。

下面给出了导致错误的不同原因：

口无效句柄请求中的属性句柄是无效的，它在服务器中不存在。该错误是由于服务器

不使用或不具备请求读取或写入的属性句柄。当属性句柄被设为Ox0000时也可以发

送该错误。

口不允许读取该属性不允许读取其属性值。例如当客户端试图读取某控制点的只写属

性时，可以发送该错误。

口不允许写入该属性不允许写入其属性值。例如试图写^只读的属性值将导致该

错误。

OPDU无效服务器不理解收到的请求。通常当客户端发送了错误格式的请求时，将

导致该错误。例如，读取请求必须有两字节的甸柄作为其唯一参数。当读取请求不具

备两字节的参数时将导致该错误。

口认证不足当两设备缺少相互的认证时，便无法执行某属性值的读取或写人操作。为

了进行认证，可以将连接加密，或是在设备还未配对时使用安全管理配对规程对设备

进行配对与连接。

口请求不支持服务器能理解发送的请求但驾前无法执行。当服务器末实现某个已知的

请求，或是不理解某个未来的请求时，可以使用该错误。该错误可以作为设备无法执

行请求时的默认错误。

口偏移量无效请求包含偏移量，但偏移量是无效的。当读取长属性时，设备借助偏移

量来读取属性值中的一部分字节。如果偏移量大于属性值的长度，那么该偏移量便是

无效的。偏移量和属性值的长度相同不会导致该错误，但会得到一个长度为零的值。

口授权不足当服务器未给予客户端使用某属性值的权限时，试图读取或写人该属性值

的请求便无法被执行。授权是服务器本地的特性。服务器需要决定如何让用户为该客

户端授权。例如，在电话或电脑上实现授权，可以用提示信息来询问用户是否允许客

户端使用所请求的效据。

口准备队列已满由于用于存储等待写入队列的空间已满，准备写入请求i{壹被接收。

准备队列的长度是有限的，一般足以处理该设备所支持的所有服务。不过客户端可能

尝试在一个准备队列中放人过多的数据，这将导致队列已满的错误。

口属性不存在属性经查找不存在。该错误使用于在一个属性范围内查找一种或几种特

定的属性类型时。只有查找信息请求、按类型值查找请求、按类型读取请求与按组类

型读取请求会导致该错误。对查找信息请求而言，诙错误意味着授有找到给定的句柄

范围内的属性。对按类型值查找请求、按类型读取请求和按组类型读取请求而言，该

错误说明在某句柄范围内没有找到给定类型的属性。第10章属 性．173

口豆性非大对象当大对象读取请求所要读取的属性不是大对象类型时，该请求将被拒

绝，客户端必须改用读取请求。该错误仅和定长的属性值有关，并且该长度小于大对

象读取请求当前用到的MTU的长度。由此，简单的做法是先使用读取请求读取属性

值的前22个字节．随后使用大对象读取请求读取后续的字节。

口密锅长度不足该错误是由于在连接已经加密、认证与授权充足的情况下，配对期间

协商的密钥长度不满足服务的要求。有些属性值需要使用高强度的密钥来充分保证数

据的机密性。

口■性值长度无效写入请求中的属性值长度错误，或是执行准备队列期间队列中的属

性值长度错误。假如属性值的长度固定（比如两个字节）．但写人请求试图改变该属

性的长度（比如一个或三十字节）．此时便会导致该错误。

口未知错误这或许是最棒的错误码了。它的基本含义是所发生的情况无法解释，不适

用于任一种错误码。该错误的意义在于如果无法解释错误，可以为该错误剖建一个错

误码。如果导致该错误的情形事先从来被考虑过，这便是“未知错误”。

口加密不足可以读取或写入属性值，但连接未加密。一些属性值需要加密连接来保证

数据的机密性。

口组类型不支持服务器不认为请求中包含的属牲类型是组类型。按组类型读取请求只

能使用服务器已知的组类型。

口资源不足服务器的资源不足以接收或执行某特定的请求。例如，当某些服务配置某

设备广播数据时，若当前广播的数据加上需要增加的广播的数据后，数据总长度过

长，则台导致该错误。

口应用错误某请求对服务中的属性执行了不允许的操作，服务分配了错误码来报告错

误的所在。这是一个128 - 255范围内的错误码，实际的古义由包含该属性的服务规

格书来定义。一个典型的应用错误是写入某特性的值无效。

要注意的是，一象错误响应蒋终止当前请求。若客户端修复了该错误，譬如执行了授权

操作或执行了认证连接操作，客户端必须重新发起该请求。不存在任何意为“待定”的响

应。特定响应假定了服务器在抛出该错误时能保持请求的状态。在低功耗蓝牙中，客户端总

是那些更加复杂的设备，因此必须由它们来解决问题并重新维持状态。这也说明客户端可

以向服务器发送任意请求，因为至少能够收到错误响应；如果无法修复该错误才转到下一

个属性。

10.7通用属性规范

属性的最后一个未解之谜是通用属性规范( GATT)规程。属性协议定义了客户端与服

务器如何相互发送符合标准的消息，而GATT规程则定义了如何发现与使用服务、持性与

描述符的标准方法。GATT规程可以分为三种基本的类型：

口发现规程

74．\*三部分主 机

口客户端发起规程

口服务器发起规程

还有一种规程不属于上述分组，即交换MTU规程，它使用属性协议中的MTU交换请

求来确定随后的消息所使用的MTU长度。该规程不是必须的，默认的MTU长度是23个

字节。

10 71发现规程

有四种基本对象需要发现。首先，客户端需要发现首要服务。一旦发现了首要服务，服

务器上所有其他的隶属于该首要服务的信息都可以被发现。随后，客户端为每个首要服务设

置句柄范围，以发现被引用的次要服务，以及该服务实例所公开的特性及其描述符。只有进

行上述的操作以后，客户端才能“使用”这些服务，服务器发起规程才能使用该服务进行读

写特性值或描述符的操作。

10.7.2发现服务

有三种发现服务的途径：

口发现所有首要服务

口按服务UUID发现首要服务

口查找包含服务

1．发现所有首要服务

当客户端连接设备后，为了发现设备上公开的所有首要服务来判断设备的功能，客户端

可以使用按组类型读取请求，请求的句柄范围设为oxoool：OxFFFF并把属性类型设为首要

服务。随后服务器把发现的一个或多个首要服务回复给客户端，当中既包含服务声明的句

柄，也包含该服务中属性的最后一个句柄。响应也包含该服务的声明内容．这样客户端便能

理解每一种服务。

如果最后一个服务的最后一个句柄不是OxFFFF或者收到了错误响应，客户端将发送另

一条按组类型读取请求。该请求的起始句柄紧随着上一条响应中最后一个服务的最后的句

柄。这样客户端便能够枚举设备上的所有服务。

要注意的是，按组类型读取响应在一条响应中不髓同时包含I6位UUID与128位

umD的服务。为此，服务器必须在发送128位UUID的属性之前，先用一条响应发送所有

的16位UUID．之后在下一条响应中发送128位的服务，再在后续的响应中发送剩下的16

位UUID。这种做法为标准服务中普遍使用的16位UUD做丁优化。因此建议基于16位

UUID的服务使用数值较小的句柄。

2．按服务UUID发现首要服务

有时客户端只想使用某个服务丽不想列出其他服务。例如灯泡开关只需发现照明服务，

并不关心该设备公开的其他服务。为了方便这一类简单客户端，可以使用一种特殊的规程来\* IO幸属 性．175

帮助它们发现类型已知的首要服务。

客户端向服务器发送按类型值查找请求，将句柄范围设为oxo001：axFFFF，把类型设

为首要服务，并把数值设为想获取的已知服务类型，譬如照明服务。随后服务器将响应所找

到的每一个照明服务的句柄范围。

有些服务被定义为单件服务（singleton service），在任何服务器上它们只能被实例化一

次。对这些服务，响应中只会包含一个句柄范围。其他的服务允许实例化多敬，响应将包含

多个句柄藏围，每个范围对应一个服务。

还有一种可能性：某个服务的实例个数超出了八个句柄范围，在一条响应中无法容纳。

这种情况比较少见，此时应使用前面提到的更新起始句柄的方法来查找后续的服务宴倒。

3查找包含腮务

一旦发现了首要服务，便能发现次要服务与其他的引用服务。这个过程要使用按类型读

取请求来查找一个包含声明。此时，应将起始甸柄与结束句柄设置为之前发现的服务的句柄

范围。通常来说，一个响应只能返回两个引用服务，因此需要发送多个请求才能获得所有服

务；每个请求中，起始句柄应紧接着上一个返回的句柄值。

引用服务被发现以后，可以使用同样的规程来发现这些服务的引用。

10.7.3特性发现

在服努被发现以后，便可以发现每一个服务的特性。要获取完整的特性，需要发现特性

和特性描述符。

1发现服务的所有特性

为了发现特性，使用接类型读取请求，将句柄范围设为该服务的句柄范围，并把类型设

为特性。这样便能发现并读取所有的特性声明。

在一个服务中，强制特性应该排在首位，其他可选的特性紧随其后。这样，有需要的客

户端便能尽快地查找到服务中的强制特性。

响应中将包古服务中每一个特性的声明和句柄。特性声明包含它的特征、承载特性值的

属性句柄吼及该特性的类型。这意味着一旦你发现丁特性声明，你便能获悉该特性表述的内

容、该特性的功用以及后续读写规程所用到的句柄。

2发现所有特性描述符

发现了特性声明以后，便可以发现每一个特性的描述符。为此，使用查找信息请求，将

句柄范围设为归类于该特性的每一个特性声明的句柄范围。

我们无法直接通过特性声明来获得所有分类后的属性的句柄，但是可以先确定某服务的

所有特性声明的句柄，以此来确定每个特性包含的句柄。例如，已知某服务的服务声明位于

句柄oxoloo．特性声明位于句柄Ox0102与Ox0108．服务的结束句柄是OxOIOF，那么可知

归类于该服务第一个特性的属性句柄范围将是Ox0103到Ox0107．第二个特性的属性句柄范

围是0@109到OxOIOF。 查找信息响应包含该特性的所有描述符的句柄和类型。客户端可以安全地忽略掉它们无

法理解的特性描述符。任何能够被客户端理解的特性描述符既可以用来槔人地理解特性，也

可以用来配置特性的行为。例如，客户端可以使用特性表示格式来理解如何在客户端上显示

这些数值，也可以使用客户端特性配置描述符来配置这些特性的通知或指示。

10.7.4客户端发起规程

对于特性，客户端可以执行四种相关操作

口读取特性值

口写入特性值

口读取特性描述符

口写人特性描述符

1读取l长）特性值

在发现了服务的特性之后，便可以读取特性值。特性声明中的某个句柄指向了保存该特

性值的属性。该属性的类型也和特性声明中的特性UUD -致。也就是说，一旦使用前面描

述的特性发现规程发现了特性描述符，特性值便能够通过读取请求或大对象读取请求读取。

对于读取请求与大对象读取请求，两者的使用有着微妙的差别，容易让人感到疑惑。当

属性的长度固定时，如果该长度小于属性协议的MTU长度，那么就可以使用读取请求来读

取特性值。如果属性的固定长度大于属性协议的xrru长度，首先将使用一条读取请求来获取

特性值开头的22个字节，随后使用一个或多个大对象读取请求来获取该特性值的剩余部分。

加果特性值的长度不固定，譬如字符串，那么必须假定该值较长，通过使用读取请求和

后续的大对象读取请求来读取完整的值。如果特性值的长度比预期的要短，可以提前终止该

读取规程。例如，如果特性值的长度是21个字节，那么使用一条读取请求便足够了；如果

特性值长度是22个字节，就有必要使用一条读取请求和一条大对象读取请求，这时大对象

读取响应中不包古该值的额外字节。因此无法获知读取响应或任何一条大对象读取响应是否

包台了属性值的最后一个字节，也无法知道值的长度是否恰好填满了响应的长度。因此，客

户端将反复使用大对象读取请求，直到它收到的响应数据少于22个字节。

2．使用特性UUID读取

某些情况下，你只想读取某个特性的值，不愿先找到服务上所有的特性声明再读取该

值。倒如读取电池电量，若能直接请求电池电量的特性值将大大提高效率。为此，可以使用

按类型读取请求，井将类型设为所要求的特性UUID。

3读取多重特性值

我们也能鳢同时读取多个特性值，前提是已知每—个特性值的句柄。该规程的关键问题

在于必须知道每一个值的长度，这是因为每—个值之间没有特定的界限。该规程用到了多重

读取请求。

不过，对于数值间没有特定界限的这一限制，也有一些商趣的可能性：请求的最后一个特性值的长度可以不固定，因为可以通过属性协议数据包的大小米判断该值的长度。

4写入l长）特性值

写人请求被用来写人特性值。我们只能用它来写入短特性值；默认的特性长度小于或等

于20个字节。在使用该请求之前，必须先发现该特性值的属性句柄。

如果要写^的属性值长度大于20十字节，就要用到另一种规程。该规程包含了准备写

人请求用来准备要写入的长值，以及执行写入请求用来真正执行写入操作。从协议的角度来

看，为了可靠地写入长特性值而使用这样的规程显得大材小用，不过有着充足的理由这样

做。属性协议在对一条请求的操作上是原子的。如果长属性值的写^持续了多个请求，在两

条请求的间隔里，值的一部分可能会发生变化，这可能导致任何试图读取该值的其他设备读

取到无效的值。使用准备写入队列来预存储要写入的长值，随后在一个执行写^请求中执行

写入，这样写人操作的原子特性才能得到保证。

5特性值可靠写入

当希望尽可能可靠地写人值的时候，可以使用特性值可靠写入规程。该规程也能够用来

在一个原子操作中一次性写人多个特性值。例如，试图将某机械工具移到新的二维坐标时，

如果在先后两条请求中分别写Ax与y的值，那么它的运动轨迹将是两条直线段，一条水

平．另一条垂直。这时，如果预先准备了T与y的值，并随后在一个原于操作里写入到设备

中，那么它的运动轨迹将是一条直线。

该规程使用了与写^长特性值规程中相同的准备写入请求与执行写入请求，稍有不同的

是多了一个检验步骤。通过逐一比较准备写入响应与请求中的每一个值，保证了响应中的句

柄和值与所请求的一致。如同10.6.13节中解释的那样，这样做保证了不会有错误出现。如

果检查中发现甸柄或值与请求有出人，那么可以使用执行写入请求，将其标识参数设为取

消，从而取消准备的队列。如此一来，所有准备过的值必须再重新准备一遍。

6（签名I无需响应写入

有时，我们需要快速地将值写人，并且不需要协议层面的回复。该规程使用写^命令将

需要写入的值发送给目标特性值属性。该命令没有响应，因此假定如果需要回复，那么它将

经由另一个特性以通知的方式发送，或是通过带外方式发送。

例如在开灯的场景中，没必要让对端设备通过属性协议发送响应，因为用户能够看到灯

被打开。另一个使用该规程的情形是响应无法立刻被获取。例如一个时间同步服务．它可能

公开—个可写的特性来启动同步操作，并且无需响应。在同步结束后，可以通过发送刚刚更

新的时间值的通知来结束整个操作。晟后一个使用该规程的原因是，响应内容可能过于复杂

而无法用一条消息发送。例如，消息服务器可以公开一条消息的状态，它具备一个可写的无

需响应的特性来改变需要公开的值。在这里没有必要回复，因为在这些值被读取的时候，相

应特性的值已经被更新了。

对一些设备而言，向其写入数据时需要对该信息的发起者进行认证。为此用到了签名写

人命令。它同样不需要响应，因此适用于和写人命令一样的情形，只不过认证提供了额外的安全性。

例如，电视机收到一条“关机”消息，它想确定发送该消息的是不是和自己绑定过的遥

控器。通常，为了确保认证，需要链路执行完全加密。但加密过程需要时间，还需耗费设备

上的许多资源才能创建密文流来加密每一条消息。对某些应用而言，加密连接所带来的额外

的时间消耗超出了它们的延时容限。无需响应的签名写入规程解决了这个问题。此时，可以

在连接建立之前对消息进行认证，随后作为连接中的第一条消息通过非加密的连接发送。虽

然选并不能防止窃听者对当前通信的窃听，但是这大大减少了向另一台设备发送消息的延

时。无论如何，若消息仅仅想传达“打开”或“关闭”设备这样的指令，这里便没有什么所

谓的秘密可言。

7读，写（长）特性描述符

特性描述符不同于特性值，但它的获取规程与读写特性值的规程十分相似。

读取请求与大对象读取请求用来读取描述符，写入描述符则用到了写入请求和准备写入

请求以及执行写入请求。

10.7.5服务器发起规程

并不是所有规程都由客户端发起，有一些由服务器发起，例如通知与指示。通常由客户

端配置服务器来发送这些消息，或者通过向服务器发送命令来生成消息并引发传输。

有两种GATI规程是由服务器发起的：

口通知

口指示

1．通知

通卸是由服务器发起的消息，可以在任何时刻由服务器发给客户端。这种消息没有流控

机制，所以当客户端没有足够的缓存空间来保存所有收到的消息时，允许它们丢弃这些消

息。要注意的是，通知属于不可靠的消息。通知使用了句柄值通知消息。

2指示

指示也是由服务器发起的消息，它可以在任何时刻由服务器发给客户端。它包含流控机

制，在服务器确认上一条指示被客户端收到之前，它不能发送新的指示。因此指示属于可

靠的消息。指示用到了句柄值指示消息和句柄值确认消息，后者由客户端发送以告知指示

的送达。

10.7.6属性协议数据单元(ATT PDU)到GATT规程的映射

属性协议散据单元 lGATT规程

史捶MTU礴求 交换MTU规程

查找信息请求 发现所有特性描述符规程

按类型值查找请求 接服务UUJD发现首要吸务规程

续

┏━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓

┃ ┃ 查找包含服务．发现所有服务特性、# UUID发现特性、接特性 ┃

┃ 按类型读取请束 ┃ ┃

┃ ┃UUID读取规程 ┃

┃ 读取请求 ┃ 读取特性值、读取特性描述符规程 ┃

┃ 大对象读取请求 ┃ 读J取忙挣陛值、读取长特性描述{牛规程 ┃

┃ 多重读取请求 ┃ 读取多重特性值规程 ┃

┃ 按组类型洼取请求 ┃ 发现所有首要服务规程 ┃

┃ 写^请求 ┃ 写^特性值、写^特性插述符规程 ┃

┃ 写^奇夸 ┃ 无需响应写入规程 ┃

┃ 签名写A命夸 ┃ 无需响应的签名写AiX程 ┃

┃ 准备写^请求，执行写人请求 ┃ 写人长特性值、特性值可靠写人、写人长特性描述符规程 ┃

┃ 甸捕值通知 ┃ 通知规程 ┃

┃ 句柄值指示 ┃ 指示规程 ┃

┗━━━━━━━━━━━━━━┻━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┛

第11章

安 全

有两类加密：一娄用来提防你的姐妹偷看你的日记，另一类用来提防你的政府。

——布鲁斯·施奈s尔(Bruce Schneier

11.1安全概念

安全是个相当复杂的课题，对许多人来说好比一个黑匣子。知道某一种技术是“安全”

的，这对大部分人来说就可以了。然而，倘若要真正理解低功耗蓝牙的安全性，你会发现要

了解的东西还真不少。安全性包括了下列内容：

口认证

口授权

口完整性

口机密性

口隐私

11.1.1认证

认证是一种证明身份的方式，用来证实所连接的设备真正是其声称的设备，而非第三方

攻击者。认证采用了下列两种基本方法：

0初始认证和秘密共享

口使用预先共享的秘密重新认证

举个饲子，当某个用户去银行开户时，她必须出示本人的证件以证明其身份。出于对证

件的发行方的信任，银行可以认证用户的身份。在这里，护照、身份证、其他政府颁发的

文件（如驾驶证）都能使用。然后，银行交给客户一张塑料制的银行卡和一个个人识别号码

( PIN)．从此，用户可以凭借它们向银行重新验证自己的身份。当顾客从银行账户取款时，

她必须使用银行卡和共享的秘密PIN码在取款机上进行认证。只要拥有了该银行卡和PIN綦ll章安 鲁’18

码，任何人都能够向银行证明他就是账户持有者，哪怕是一个现实里根本不存在的人。

在低功耗蓝牙中，执行认证有三种不同的方式：

口在初始配对时，设备使用一个认证算法来验证连接的对端设备。该过程可能涉及向一

个或两个设备键人密钥。输入的密钥用来支持链路加密，继而可以分配那些日后使用

的共享机密。只要设备保存了共享机密，即称之为“已绑定”。

口当重新连接到以前绑定过的设备时，其中的任一设备可以向对端设备发送一个签名的

命令来进行身份验证，表明它知道先前分配的共享机密。签名使用了上次绑定时彼此

交换的共享机密来创建，因此不能被第三方伪造。该签名的命令含有一个计数器部

分，每个消息发送时递增，以防止重放攻击。

口当重新连接到以前绑定过的设备，二者之中任一设备都可以启动加密。此后传输的每

个数据包将包含消息完整性检查( MIC)值，接收设备使用先前分配的共享机密验证

消息的发送方。

11.1.2授权

授权是指分配权限做某事，通常包括两种方式：

口文档提供授权

口直接进行授权

音乐会一般没有认证要求，但授权进入音乐台通常需要采取验票的方式。这些票由事件

的组织者提供给观众，井在音乐会现场出示，作为人场的凭证。不会有对持栗人是否与购买

人相同实施认证；换句话说，没有必要使用照片鉴定身份。

另一个授权的例子是即时批准某人做某事。例如，如果把电脑借给某个朋友并且告诉他

只能使用某个程序，那么你正在授权他使用该程序，而不包括其他程序。

这一授权模型也可用于无线通信。当某设备与你的设备建立连接时，你可以授权它来访

问你的设备的某些部分，但不是所有部分。

11 1 3完整性

完整性的定义是指数据的内部一致性和无讹误性。无论使用有线还是无线通信协议，数

据从一个设备传送到另一个设备时都容易产生各种错误。因此，错误检测和防范非常重要。

从安垒的角度来看，一些错误可能是由第三方攻击者试图将—个合法消息修改为一个恶

意消息所引起的。例如，攻击者可以重放捕获的消息，但是更改其中的某些位，把内容从

“锁门”变成“开门”，从而使房屋的安全性受到严重损害。应当指出，循环冗余校验(CRC)

可以用来防止位元的变化，但其功能较弱，不能视作一种安全措旌。这是因为既然很容易篡

改消息的位，当然也很容易修改CRC中的一些位来匹配篡改的内容。为了确保完整性，需

要一个更强大的消息认证机制，能够同时检查原始消息的完整性。”．1.4机密性

机密性是指将事物保持机密的意图。机密性最常见的代表是在电影中，你会看到标有

“机密”字样的公司文件或政府报告。不幸的是，把这个词印在文件上其实并不妥当，因为

任何人只要看到就可以阅读它们。在低功耗蓝牙中，机密性意味着即使一个第三方窃听者接

收到一个消息，她也无法解码。消息的编码过程称为加密。第二次世界大战期间开发的恩尼

格玛机是一个可以加密或解密消息的设备的经典例子。

11.1.5隐私

另一个安全的关注点在于任何沟通都应是私人的。完全的匿名性很难做到，例如有一个

名人登机，有许多人会通过他的样子认出他。因此，要在每一个位置授予完全匿名性几乎是

不可能的。

无线通信应该为设备跟踪设置一定的壁垒。如果其他人能够利用你携带的设备在一个区

域内不断追踪你的运动，既会带来一些有趣的应用，也可能造成可怕的后果。比如，根据你

在其他地方购买的物品，当前的商店为你提供一些特别的优惠。这看起来不怎么恐怖，对

吗？但如果一个跟踪狂将一些设备布置于某个区域，当你经过该区域时便自动跟踪，这就没

有获得一些打折信息来的诱人了。

因此，隐私是能够防止他人根据你的设备认出你，而且无法在一个空间跟踪你的运动的

能力。

”．1.6加峦引擎

低功耗蓝牙有一个密码块，实际上是一个单向函数，用于产生密钥、加密和提供完整性

检查。这种加密引擎被称为先进加密系统( AES)，在NIST的出版物FIPS-197中有具体的

定义e。低功耗蓝牙使用AES的128位版本，称为AES-128。

了解AES的一般方法是考察单向函数E，它将一个密钥和一些简单的明文数据作为输

入，最终生成密文数据块。因此AES属于分组加密。密钥长度和明文数据的长度均为l28

位，所得到的密文的长度也是128位。进可以用式( 11-1)表示；

密文= Ek,（明文） (II-I)

需要注意的是，如果不是频繁地变更密钥，该算法是非常有效的。每次使用新的密钥都

需要进行大量计算以设置加密引擎的内部状态。设置完成之后，每一个新输入的明文就能迅

建转化为密文。低功耗蓝牙采用的安全算法正是利用了AES加密引擎的这一性质。

11.1.7共享机密

几乎所有的安全性都基于共享机密：你的银行卡密码是一个机密，由你和你的银行之间

共享；你的电脑密码是一个机密，由你和你的计算机或公司信息系统部门的计算机之间共

e参考NISTFlPS，l97 h却／恼…j“gov『pubI“享；还有，你家的钥匙是握在你手上的金属与门钝内的金属之间的共享机密。

在低功耗蓝牙里也有许多共享机密，称之为密钥。一个密钥是一个“共享机密”的速记

方式。可以有很多密钥，就如同你有车钥匙、门钥匙（一到两个或者三个）、自行车钝的钥

匙或存取工作文件的钥匙等。

低功耗蓝牙主要有五个密钥：

口临时密钥(IK)

口短期密钥(STK)

口长期密钥(LIX)

口身份解析密钥(iRK)

口连接签名解析密钥(CSRK)

1临时密钥

配对过程需要用到临时密钥(TK)。临时密钥值通过配对算法生成，用于计算短期密钥。

立即工作（just work）是一种设计模式，当有限的用户界面制约了用户键人或验证密钥

时，可以用来连接慨功耗蓝牙设备。在应用立即工作模式时，TK的值应设为零。这意味着

并没有执行认证，因此该连接以及任何在其之上分配的密钥很容易受到中间人的攻击。

万能钥匙进入（passkey entry）是另一种设计模式，当设备的界面支持数值显示或输人

时，可吼使用该模式。应用万能钥匙进人模式时．1X值设为两个设备上的数字输人值，取

值范围从O - 999 999。这意昧着，中间人攻击只有很小的概率碰上当前连接使用的值——

只有百万分之一的猜中机会。就讥证而言这已经是一个合理的概率。因此，这种算法所生成

的密钥可以视为一种能够有效荫止中间人攻击的可认证密钥。

还可利用一些带外(out of band)算法设置TK值。如果两台设备能蟮通过蓝牙之外的

另一种技术获得信息，例如采用近场通信( NFC)在两个设备之间传输一个值，该值可以作

为认证的1X值。使用带外数据生成的密钥一般视为已认证的密钥，可以防止中间人攻击，

这是因为我们假定了带外技术也不易受到这些类型的攻击。

2短期密钥

两个首次配对的设备可以使用短期密钥( STK)加密其连接。STK源自三部分信息：临

时密钥以及两个随机数‰和M州。其中，前者作为加密引擎密钥，后两者则由从设备和

主设备在发起配对请求时提供。S\_d和M州用符号“l”连接，如式（11-2）所示，然后使

用临时密钥加密。

STK - Erx (S rmdl M~ (11-2)

从设备和主设备提供的随机数提高了整个系统的安全性，因为任何攻击者仅能提供128

位随机数当中的64位。作为“中阔人”的攻击者如果想要猜测对端设备使用的随机数，必

须在2科个可能性中做出选择，难度可想而知。 3长期密钥

一旦初始配对过程完成了连接加密，设备将会分配长期密钥( LIX)。LIX可以是存储

在安全数据库中的—个随机数。在趴设备上也能生成LTK。

在设计上，从设备的资源通常较为有限，所以要让其维护一个安全数据库可能只是一种

奢望。为了解决这个问题，从设备向主设备分配两个值：EDIV和Rando这两个值存储在主

设备上，当与从设备重新连接时再发回给对方。从设备随后可以计算要使用的LIK，或者

更准确地说，计算它先前给过主设备的LIX值。

在重新连接到之前配对井绑定过的设备时，用LTK加密连接。这意味着不需要为每次

的设备连接执行完整的配对。

4身份解析密钥

只要知道对端设备的身份解析密钥(IRK)，本方设备就能解析出对方的身份，这就是身

份解析密钥的功能。保护隐私的一种做法是使用完全随机的地址，并让设备随时改变自己的

随机地址，但这样的话，那些可信任舶设备也将无法追踪它们或者与之建立连接。关键的问

题就在于：如何做到既能用随机地址确保隐私，又能被可信设备所识别？

为了解决这个问题，我们使用IRK来产生随机地址。地址分成两个部分：一个随机部

分和一个使用IRK对该随机部分进行哈希预算获得的值，如式(11-3)所示；

izash= EmK(ram0 (11-3)

利用随机数和哈希值构造地址字段，这样的话，已经知道IRK的对端设备就可以检查

二者是否匹配。

打个比方，考虑一个人每次与另一个人说话时使用不同的别名。简单起见，假设没有

2'4那么多的可选名称，我们只有三十。第一次他可以称为“美罗拉·本杰明”，下一次是

“鲍勃·霍普”，再下一次是“查利·狄默科”。如果你看到的名字荚罗拉和本杰明在一起，

或鲍勃和霍普在一起，那你知道这就是那个人。如果名字是“美罗拉·霍普”或“鲍勃·狄

默科”，这肯定不是那个人，因为“美罗拉”总是与“本杰明”在一块。

因此，设备可能有一串名单记录了每一个已绑定的设备的IRK，在收到私有地址时将其

与所有IRK执行穷举匹配。一旦匹配上就意味若发现了正确的设备。

应当指出的是，要为每一个设备找一个不同的随机数和哈希值是不现实的。大约有

700 000亿个固定的设备地址，而可以使用的随机数仅有400万个。因此，即使拥有相同的

随机数和不同的ⅡLK，两个设备也可能获得相同的哈希值。这就是为什么隐私通常还会与认

证相结合，以确保找到正确的设备，而非拥有相同的地址组合的其他设备。

5．建接签名解析密铜

连接签名解析密钥( CSRK)绐予了接收装置解析签名的能力，从而能够验证消息的发

送方。CSRK从该报文的源端分配给消息的目的设备，经过加密的链路发送，随后可以断开

该链路。在重联时，如果消息只需要认证而不用以加密形式发送，可以对其进行签名。第1l章安 奎0 185

对数据签名使用的是CMAC函数，定义在NIST特别出版物800\_38B e中。采用CSRK

和一个符号计数器作为此功能的密钥。该符号计数器是一个32位的值，在设备绑定期间，

每个消息从源设备发往目的设备时该计数器不断增加。设备完成绑定时，符号计数器重置为

零；无论设备是否断开，在每个新的数据包发送时自动加1。

11.2配对和绑定

为了保证低功耗蓝牙的绝大多数安全特征，必须完成两件事情：首先，设备必须互相配

对；其次，连接一旦加密，设备必须分配用于加密、保障隐私并对消息进行验证的密钥。只

要密钥被保存下来，设备就处于绑定状态了。

因此，要想了解安全是如何工作的，关键在于理解配对和密钥分配系统的工作原理。同

样，明确两个设备间的初始连接不同于二者之间的后续连接也很重要。

11.2.1配对

起初未提供安全性的两个设备如果希望做一些需要安全性的工作，首先必须彼此配对。

配对涉及两个设备的身份认证、链路加密以及随后的密钥分配，这会使两个设备在第二次重

连时的安全启动速度大为加快。

配对有三个不同的阶段：

口配对信息交换

口链路认证

口密钥分配

11.2.2配对信息交换

配对的第一阶段涉及配对信息的交换，该信息用于碗定设备的配对方式，以及确定在最

后的阶段将会分配哪些密钥。

需要注意的是，或许只是因为在配对期间采用了某种复杂的算法，设备配对就成了用户

排斥整个无线技术的阿喀琉斯之踵。试想，假如配对既繁琐又麻烦，用户很可能由于首次配

对受挫而将产品退回给商户。

低功耗蓝牙使用的配对过程和经典蓝牙的安全简单配对( secure simple pairing)相同。

每个设备首先确定它的输入和输出能力以及其他配对信息。输^和输出能力进自下列可能的

能力列表：

口无辖入无输出

口仅有显示屏

D显示屏十选择“是／否’

口仅有键盘

e参考NIST特别出版物E 口键盘+显示屏

为了确定应该使用这五个可能的值中的哪一个，该设备确定其输入和输出能力并将其提

供给一个矩阵，如下表所示。输入能力可以是“无输入”、选择“是，否”或者通过键盘输

入一个数字；输出能力可以是“无输出”或“数字输出”。注意，此处的数字输出指的是显

示六位数字的能力。

这五个设备的输入和输出能力通过使用配对请求《painng request)消息在设备之间进行

通信。

从设备发出的第一个安全消息即为配对请求消息。该消息不仅包含支持的输入输出能

力，还包括其他的配对信息，例如说明带外数据是否可用的比特位和认证的要求（如果有的

话）。它也包括正在请求的密钥列表，等到配对过程结束时再进行分配。认证的要求包括：

是否启用绑定，是否要求保护中间人攻击等。

为了响应配对请求，可以发送的消息包括配对响应（pairing response）或配对失败

【pairing failed）。

配对响应消息包含的信息与配对请求基本相同。然而，如果在请求中要求发送带外数

据，但响应却没有携带任何带外数据的话，则应当发送配对失败消息。事实上，配对失败消

息可以在配对过程中的任何时刻发送——例如当一个参数和应有值或期望值不匹配的时候，

从而为谩备提供一个中止配对的机会。

一旦交换了配对请求和配对响应消息，两个设备随后便进入到配对过程的第二阶段。

11.2.3认证

利用配对请求和配对响应消息所携带的信息，两个设备得以确定适合的配对算法。下表

列出了如何根据二者的输人和输出能力来确定具体使用的算法：

┏━━━━━━━━━━━┳━━━━━━┳━━━━━━━━┳━━━━━━┳━━━━━━━┳━━━━━━┓

┃ ┃ ┃ 显示屏+选择 ┃ ┃ ┃ ┃

┃ ┃但有显示屏 ┃ ┃ 仅有键盘 ┃ 无输^无输出 ┃键盘+显示屏 ┃

┃ ┃ ┃ “是／否” ┃ ┃ ┃ ┃

┃仅有显示屏 ┃ 正常工作 ┃ 正常工作 ┃ 辅^密朝 ┃ 正常工作 ┃ 输入密钥 ┃

┃显示屏+选择“是／否” ┃ 正常工作 ┃ 正常工作 ┃ 轱^密朝 ┃ 正常工作 ┃ 辅人密钥 ┃

┃但有铤盘 ┃ 输^密胡 ┃ 输人密锕 ┃ 输入密钥 ┃ 正常工作 ┃ 辖人密钥 ┃

┃无格A无输l出 ┃ 正常工作 ┃ 正常工作 ┃ 正常工作 ┃ 正常工作 ┃ 正常工作 ┃

┃键盘+显示屏 ┃ 辅^密钶 ┃ 输人密铜 ┃ lik^密钥 ┃ 正常工作 ┃ WA癣钥 ┃

┗━━━━━━━━━━━┻━━━━━━┻━━━━━━━━┻━━━━━━┻━━━━━━━┻━━━━━━┛

应当指出的是，一些貌似奇怪的组合仍然有可能引导用户去做正确的事情。例如，如果

设备A拥有显示屏和选择是／否的能力，而设备B拥有键盘和显示屏，那么设备A可以显第“幸安 空．187

示一个六位数字，设备B的用户看到该数字后在设备B上键人。该数字随后可作为TK值

使用。

一旦TK值被确定下来，便可执行一个看上去很简单、但却难以攻击的规程，用于防止

中间人攻击。设备各自生成一个随机数，并基于该随机数、1X值、已知的配对值（到目前

为止为设备地址）、配对请求和响应消息中的参数计算一个确认值，见方程（11-4）。

确认值- Erx（随机数I配对请求和响应消息设备地址） (11-4)

上述确认值能用于证实一点，即迄今用过的所有已知的参数和地址在两个对等设备是否

相同。这种做法能够有效地预防中间人攻击。

两个设备随即交换随机数和确认值；因此，它们可以检查确认值是否匹配随机散和所有

其他共享信息。有趣的是，确认值会在发送随机散之前进行交换。这样，攻击者在知道真正

的随机数之前，必须猜测对端设备将会使用，l2a种可能的随机数中的哪一个来计算确认值。

如果确认值不匹配随机数，说明某个地方出了问题，设备将发送配对失败消息终止配

对。如果确认值匹配随机数，说明两个设备具有相同的配对请求和配对响应参数、相同的地

址信息、相同的TK值和正确的随机数。

假设一切都确认无误，在认证中交互的随机数将用来计算STK．如11 2节所述。然后

该STK值在链路层加密规程中用于加密链路，如第7章所述，详见第710 3节。

11.2.4密钥分配

连接一旦利用了S7X加密，就可以分配所需的密钥了。这些密钥每次分配一个，固其

长度为128位，每一个数据包只能装进一个。

以下密钥可以分配：

口LTK

口IRK

口CSRK

由于从设备不具备安全数据库，因此LTK随着EDIV和RAND分配。从设备可以使用

这些信息直接为主设备生成LTK。

无论是主设备还是从设备都可以分配所有类型的密钥。这是因为就当前的莲接而言，拓

扑结构可能是A为主设备B为从设备；但在随后的连接中，拓扑结构有可能反过来．A变

成从设备而B变成主设备。因此，允许由主设备向从设备分配LTK．这样即使它们以不同

的拓扑结构重新连接，仍然可以快速实现重连。

一个必须考虑的问题是，在配对时使用的地址可能并不是设备的真实地址；配对期间可

以使用随机地址或私有地址。因此，分配两个设备的身份信息就变得非常有用，该信息可用

作数据库的唯一键值，使得未来设备间可以采用该身份信息进行连接，而非可能已经过期的

随机地址。11.2.5绑定

绑定真正来说属于通用访问规范的讨论范畴，然而，此刻也不妨考虑一下它的操作。绑

定指的无非是将密钥及相关身份信息保存到安全数据库当中。如果设备不保存这些值，它们

虽能匹配，但却不能绑定。只要当中某一个设备不保存，重新连接后，只有一个设备拥有

LTK，因此加密的启动将会失败。

为了避免这种情况，两个设备在最初配对时就会交换绑定信息，从而能够清楚地知道对

方是否保留了该绑定信息。如果对方设备不保存信息，那么一旦启动加密的尝试失败，主机

将试图再次配对。

11 3数据签名

当一个设备已经连接但尚未加密时，可以发送设有保密性的认证数据。要实现这一点，

第一次连接配对时应当交换CSRK。在此之后，只要没有具有保密性要求的数据需要交互，

就可以使用签名。

数据签名采用CMAC算法。该算法需用到进行身份验证的消息、一个签名计数器

( SignCounter)和用来验证发送方的CSRK，之后生成一个签名值。

消息认证是一个典型的属性协议的签名写入命令，由操作码、句柄和写入值构成。

签名计数器是一个32位的值，当每次发送一个新的数据包时必须递增，以踌止重放攻

击。如果相同的签名计数器出现在随后的数据分组中，有可能是攻击者莸得了以前的消息并

且试图通过重放消息玩一些花样。不幸的是，这类攻击者不过是在徒费周章，因为接收者会

丢弃所有的签名计数器小于下一个期望值的消息。应当指出的是，如果对端设备支持数据签

名，那么它们需要为每一个绑定的设备存储下一个期望的签名计数器值。

签名计数器值必须包含在消息中，这是因为接收者不清楚下一个收到的消息是否含有下

一个期望的签名计数器。设想一下将签名计数器用在打开和关闭车库门的场景。命令车库门

开启或关闭的消息不用保密，任何人从街上会看到汽车的出现，接着是车库门开启。如果你

某一天出远门，期间无意中多次按下车库门的遥控按钮，每按一下生成一个新的消息。当你

回家的时候，按下按钮来真正打开车库门，其签名计数器值可能会大干期望值。车库门本身

不会将其视作威胁，因为这个值之前从未被使用过。 第12章

通用访问规范

在任何一种已知的语言里都找不着“像机墙一样漂亮”这句习语，这可不是巧合。

——亚当斯道格拉斯(Douglas Adams)

核心规范的最后一部分是通用访问规范( OAP)。它定义了设备如何彼此发现、建立连

接以及如何实现绑定，同时描述了设备如何成为广播者和观察者，并且实现无需连接的数据

传输。最后，它定义了如何用不同类型的地址来实现隐私性和可解析性。

12.1背景

要理解低功耗蓝牙，最重要的是了解两个设备之间如何能发现对方、与对方协作以及不

断找到对方并彼此连接。这些都在OAP当中定义。为了便于说明低功耗蓝牙的工作原理，

让我们考虑下列典型的用户场景：

一位用户从商店回到家，带回了一个刚刚购买的新的心率带。她已经有一个支持低功耗

蓝牙的手机，因此打算把心率带连接到手机。此外，她最近还购买了大量的低功耗装置，包

括支持低功耗遥控器的电视机、低功耗的照明系统，还有一些用在她的电脑、鞋、手表上的

低功耗传礞器。她是如何配置心率带使之与手机一同工作的呢，

用户打开心率带的包装盒，翻开说明书，上面告诉她把塑料标签从心率带上拿掉，以便

启动设备。随后，说明书告诉她击发现心率带：在手机上打开蓝牙菜单，点击“添如设备”

按钮．然后瞧一瞧有哪些设备出现在屏幕上。

在设备列表中的顶部是地刚刚购买的心率带，附带了一个心率带的图标。她选择了心率

带，手机跳转到应用商店，显示了能够与此设备一同工作的应用列表。有些应用是免费的，

有些是收费的，有些是来自心率带的同一品牌。她选择了一个应用程序并将其安装在手机

上。几秒钟后，应用程序开始运行，并在屏幕上显示出心率的信息。用户出去跑了跑步测试

了一下该应用。 第二天，用户再次戴上心率带，启动昨天她下载的应用程序。再一次，心率的信息显示

在了屏幕上。就好像变戏法一样，它只要使用该应用，对应的设备就开始工作。

几天以后，这位用户的一个朋友向她推荐了另一个应用程序，他用该程序与自己的心率

带配合工作。于是她进入应用商店，搜索到该应用，下载然后运行。应用再次显示出心率信

息，但现在还包括额外的信息——用户锻炼的强度。虽然应用程序使用的还是来自同一心率

带的相同信息，但这次数据的使用方式有所不同。

这个例子表明，用户开始使用一种低功耗蓝牙设备可以非常地简单；同时，低功耗蓝牙

服务的灵恬性意睐着设备的使用不会依赖于某一个应用程序。

12.1.1初次发现

为了发现一个设备，必须扫描有哪些正在广播的设备。多个广播设备向多个扫描设备发

送数据包，构成一个多对多的拓扑结构。这里的问题是，这种方式还会扫描到那些可连接

的、但未必是可发现的设备。为了解决这个问题，除了数据之外广播报丈还包括一些标识，

用来显示设备是否为可发现的或是可连接的。

发现性包括两种类型：第一类为“有限可发现性”，由那些临时设置为可发现状态的设

备使用。例如，当设备第一狄接通电源时，它将进入有限可发现模式；或者如果设各有一个

专门的按钮，按下它时会令设备暂时进人该模式。设备不允许长期停留在有限可发现模式，

这是因为这一模式的目的是让设备从众多采用一般可发现模式的设备中脱颖而出。如果设备

都在有限可发现模式下维持很长一段时间，它们就无法凸显自己。因此，一个设备只能保持

有限可发现模式约30s。

第二类为“一般可发现性”，由那些最近没有执行过交互的可发现设备使用。例如，如

果一台计算机是可发现的，其他设备都可以找到并连接到它，但它最近没有启用过该功能，

一旦启用，它将进入一般可发现模式。一个寻找其他设备的设备通常会把一般可发现设备排

在设备列表的下方，这是因为较之有限可发现设备，它们的紧迫性相对较低。

确定设备的发现性结合了两个方面的工作：扫描所有设备和过滤各设备广播的可发现标

识。向用户呈现设备列表时，也可以选择其他的过滤规则。例如，与设备的距离可用于决定

设备的排列顺序，离得近的那些排在列表的顶端，离得远的那些排在列表的底部。为此，将

广播报文的发射功率与收到广播报文时的信号强度相比较，从而计算传播的路径损耗。路径

损耗较小的设备通常会比路径损耗更大的设备离得更近。

另一种可能的过滤规则为：利用广播数据里的服务信息，根据设备韵用途进行隔离。比

方说，某用户正在寻找一个心率带，那么在本区域内支持这个特定服务的设备很可能就是用

户希望连接的设备。设备可以公开两类服务信息，或者公开其已经实现的服务清单，或者公

开其希望对方支持的服务清单。

因此，利用不可发现设备、有限可发现设备、一般可发现设备、路径损耗范围过滤和基

于服务的过滤，可以为用户提供一个直观的界面。如果用户正在与某设备交互，该设备将显

示在列表的顶部；如果设备逐渐靠近，它离列表的顶部也越来越近；如果设备支持正在搜索第12章通fll访目规范．19t

的服务，它们就会出现在接近列表顶部的位置。一个用户只需执行搜索，仅凭查看到表顶部

的条目就能有很大的可能性连接到正确的设备。

在设备发现过程的一个小问题是名称的发现。用户不喜欢看用十六进制表示的48位数

字；相反，他们喜欢看好读易懂的名字。为了让每个设备有一个名字，GAP定义了公开设

备名称的特性。设备名称也可以包含在广播数据或扫描响应数据之中。

为了获得扫描响应数据，需要用到主动扫描。换言之，假如发现设备想在屏幕上显示其

他设备的名称，通常需要执行主动扫描。这是因为设备名称是静态的数据，通常会被包括在

扫描响应数据中，而非广播数据中。然而，有些设备有太多的信息，需要广播告知它们无法

在广播数据或扫描响应数据里放人完整的设备名称；相反，他们只能包含部分名称，抑或不

包含名称。

在这种情况下，要获得完整的设备名必须与此设备建立连接并读取其名称。值得庆幸的

是，关于设备的名称有个众所周知的特性：设备名（device name），可以通过使用简单属性

协议的根据类型读取请求获得。

12.1.2建立初始连接

一旦得到了设备列表，并且选中了其中的某个设备，下一步便是与该设备建立初始连

接，即向厂播报文提供的地址发起一个连接。假如设备建立了连接，它可以在穷举对端设备

的所有服务和特性或者只查找它感兴趣的服务及其特性之间做出选择。

举个例子，拥有应用商店的电话或电脑会查找设备公开的所有服务。然后，这些服务信

息被发送到应用商店，商店中的支持这些服务的任何应用程序随之呈现给用户。因此．用户

不再仅仅使用与设备捆绑的应用，而是能够使用一个应用生态系统。有些应用是免费的，有

些需要花钱，有些由该设备的制造商开发，有些来自独立的软件公司。

另一种方法是，设备只执行服务发现，寻找一个服务或是一个非常有限的服务列表。倒

如，一台连接遥控器的电视只会寻找人机接口设备的服务或者可能的电池服务，仅此而

已。这意味着，即使设备支持的服务不止上述两个，电视也不会对它们执行服务发现。该

方法通常用于用户界面有限的设备，或者是连接到外围设备、只使用一组非常觯的功

能的设备。

服务发现最终将获得一个服务列表，包含对端设备公开的各种服务。从此，客户端便可

以使用这些服务。在应用商店模型中，由应用来完成接下来的特性发现和配置工作。特性发

现与服务发现类似，设备可咀枚举某一服务的所有特性，或者直接用一些服务必须公开的公

知特性。例如，电池暇务必须公开电池电压特性，即使没有对其执行特性发现，客户端也可

以直接读取电池电压这一特性。

12.1.3服务特性

对于心率带，特性的发现和配置可能更复杂。倒如，心率的服务可能只会公开心率值，

或者心跳的区间统计值以及统计的时间间隔，或者只是心跳的间隔而已。它也可以公开由心率传感器计算得出的一些其他信息。客户端可以按需挑选所要读取的特性。有些客户端可能

只对心率值有兴趣，而其他客户可能希望阅读所有信息。

为了实现设备间高效的数据传输，应使用通知或指示。如果客户端希望收到关于某特性

的通知或指示，必须在客户端的特性配置描述符当中写人信息，通过其启用所需的功能。于

是，服务器设备在必要的时候将发送相应的通知或指示。回到测量心率的例子，客户端可以

设置发送心率值的通知，无论心率变化与否，服务器都将每秒发送一次心率值。其他服务的

时间间隔可以更灵活，好比电池服务．只有当电量值改变时才发送通知。

因此，应用程序可以配置发出通知或指示的特性集，然后等待服务发送的数据。这意味

着即使用户更改了心率应用，新的应用将沿用以前的配置并且仍然能够收到心率通知。

121 4长期关系

大多数时候，一个外围设备仅与一个中央设备关联。比如你的接近标签与你的电话关联；

你的键盘与你的计算机关联；你的车库门开关与你的车库门关联。设备与设备相关联俗称“绑

定”。绑定包含两个步骤，使得寨不相识的两个设备能够彼此验证身份，并且分享机密。

要完成绑定，需要将两个设备设置为可绑定。不可绑定的设备（可能由于其已经与其他

设备绑走，且每次只能管理一个绑定）不能在广播中声称其为可绑定。

当两个设备均可绑定，并且其中之一打算进行绑定，这就近出了绑定过程的第一步。在

这之后，两个设备变换输入和输出能力，基于这些能力选择一个认证算法，然后彼此进行身

份验证，产生用于加密链路的STK（更多的信息参见第儿章11.1.7节）。

链路加密一旦完成，将执行第二步绑定过程。这一步涉及共享机密的变换，用于与之前

连接过的设备执行重连。通常，共享机密包括具有多种用途的密钥：长期密钥( LIX)用于

加密后续连接，身份解析密钥( IRK)用于解析私有地址，连接签名解析密钥(CSRK)用于

检查已签名的属性协议命令的签名。除此之外，还包括一些发送蛤主设备的、无需从设备存

储的身份信息。

12.1.5重连

有时候，设备完成了所有数据的发送，可能不愿再浪费能量保持连接而将其中断。比如

电灯开关会创建一个连接，发进“开灯”命令并随后快速断开连接。但另一些时候，连接可

能会保持非常长的时间，比如键盘可能一直保持与计算机的连接，直到计算机关闭之后才会

断开；当计算机重新打开，它需要重新连接到键盘。

重新连接既简单又难。在低功耗蓝牙中，如果一个设备打算在重连后成为从设备，它必

须通遗可连接广播事件进行广播。然而，它既可能是不可发现设备，叉可能是有限可发现

或一般可发现设备，还可能不允许任何设备与之连接。与此同时，要与广播设备建立连接，

主设备必须执行扫描并且向正在广播的特定地址发起连接。如果使用了白名单，那么该重

连设备必须位于扫描设备或发起设备的白名单之上（关于白名单的更多的信息详见第8章

8.4.10节）。12.1.6私有地址

设备如果使用私有地址广播，容易导致一些混乱的情况。私有地址是指周期性变化的随

机地址，比如每隔15分钟变换一次。这意味着，即使你发现了一个目前正处于广播态的设

备，20分钟之后，它可能换了个完全不一样的地址，届时你将难以判断那个设备是否还在

附近。乍一看来，这似乎是一个不可能解决的问题。

要解决该问题，需要执行三个步骤。第一步是在绑定时保存IRK密钥；第二步是使用

该密钥生成一个可解析的私有地址；最后，主设备必须扫描所有的设备，利用其所有的IRK

解析每一个地址。只有能够验明身份的地址才能进行连接。

可解析的私有地址是一类随机地址，包括三个部分。第一部分是固定模式的两位组，用

来表明该随机地址是可解析的私有地址。这就减少了扫描设备的运算负担，它们只需要解析

那些可解析的私有地址即可。第二部分是一个22位的随机散。第三部分是随机数与在绑定

时分享的IRK的哈希值。

将随机数和IRK相结合，意味着每一个使用私有地址的设备实际上拥有四百万个可用

地址作为指纹。

要连接使用私有地址的从设备，主设备必须扫描所有的可解析的私有地址，取出其随机

数，并结合每一个可能使用私有地址的设备的IRK，分别计算对应诙设备的哈希值。如果得

到的哈希值与可解析地址中的哈希值相匹配，那么有相当高的概率可以确定此设备即为密钥

对应的那台设备。

当然，万事无绝对。也有可能另一十设备拥有不同的IRK和相同的随机数，也能生

成完全相同的哈希值。然而，快速的连接和加密技术还能蟮进一步检查该设备正确与否。

IRK、用于加密的LTK以厦用于签名认证的CSRK在不同设备上各不相同，它们可以很快

地确认到底连接的是正确的设备，还是发生了哈希值重复的小概率事件，以致连接了一个错

误的设备。

私有地址存在一些缺点。最大的缺点在于，主设备每次收到一个可解析的私有地址，都

要依次结合各个IRK来执行暴力控验。如果主机知道许多私有的设备，可能就要耗费大量

的时间。在这种情况下．HCI的加密命令将会非常有用，尤其是在主机的计算资源比较有限

的时候。

私有地址的另一个缺点是，无法利用白名单来简化连接。连接到私有设备的唯一方法是

先扫描可解析的私有地址，计算其是否属于可以建立连接的私有地址，而后手动执行连接。

由于主机必须执行许多地址解析的工作，这无形中增加了主机的能量消耗。

12.2 GAP角色

低功耗蓝牙设备定义了四类GAP角色

口广播者

口观察者

口外围设备

口中央设备

广播者是发进广播报文的设备，通常广播一些服务数据给处于观察者角色的设备。广播

者必须有发射装置，但是不一定有接收装置。比如一个纯广播设备就只需装配一个发射装置

即可。

观察者是扫描广播者、并且将信息报告给应用的设备。观察者必须有接收装置，也可以

选装发射装置。

外围设备是利用可连接广播报文进行广播的设备。因此一旦连接，它将成为从设备。外

围设备必须同时拥有发射和接收装置。

中央设备是向外围设备发起连接的设备。因此一旦连接，它将成为主设备。中央设备也

必须同时拥有发射和接收装置。

一个设备可以同时支持多个GAP角色，比如既可以是广播者，又可以同时为外围设备。

在GAP内部有两个基本概念用来描述设备的行为，即模式(mode)和规程(procedu蜱)。

当—个设备被配置为按照某种方式操作一段较长的时间时，称为模式。当一个设备被配

置为在某一段有限的时间内执行某种特定的操作时，称为规程。

举个例子，当一个设备正在进行广播时，称其为“广播模式”。广播往往会持续较长时

间，也许是该设备的唯一用途。而当一个设备正在找寻广播者时，称其为“观察规程”。观

察往往持续一段较短的时间，用以构建用户界面或者寻找需要昀指定信息。

GAP定义了下列模式：

口广播模式

口不可发现模式

口有限可发现模式

口一般可发现模式

口不可连接模式

口定向可连接模式

0无向可连接模式

口不可绑定模式

。可绑定模式

GAP定义了下列规程：

口观察规程

口有限发现规程

口一般发现规程

D名称发现规程

口自动连接建立规程

口一般连接建立规程

口选择性连接建立规程

口直接连接建立规程

口连接参数更新规程

口终止连接规程

口绑定规程

要理解GAP．就必须理解上述模式和规程是如何相互操作的。例如，广播模式和观察

规程可能在逻辑上相互结合，以支持广播者进行观察操作。

12.3.1广播模式和观察规程

设备处于广播模式时，将使用链路层广播信道和广播数据包发送广播数据。使用观察规

程的设备能够获得该广播数据。

广播时，广播的数据必须具有正确的格式，如12 5节所述。

必须指出，一些设备可能只装备了发射装置，即为纯广播设备。它们通常不能在广播数

据中使用私有地址或者签名的数据。这是因为二者均需要密钥的相关信息，它们其可能在连

接建立后的绑定过程中进行分配，而这既需要发送装置又需要接收装置。当然，如果密钥可

以通过带外方式分配，那么仍然可以使用它们，不过这并非标准定义的方法。

12.3.2可发现性

在GAP中，只有外围设备才允许被发现。尝试发现外围设备的那些设备处于中央角色。

为了提供不同的可发现性，外围设备有三种工作模式：不可发现模式、有限可发现模式

和一般可发现模式。在GAP中，不可发现模式并不意味着不允许发送广播数据，而是指对

端设备即便发现了该设备，其用户界面也不会发出通知。按照GAP的定义，一个设备完全

可以在不被“发现”的情况下发送广播消息。

因此．GAP要求可发现设备必须在其广播数据里包括有关可发现性的信息，从而与那

些发送不可发现信息的广播设备区分开。为实现这一点，在AD信息标识（n鹕）当中包含

了2位（详见第12 5 1节），用来表明设备处于不可发现模式、有限可发现模式还是一般可

发现模式。

1不可发现模式

处于不可发现模式的外围设备不允许将AD信息标识设为有限可发现模式或是一般可发

现模式。除非设置其他信息，否则AD信息标识不必放在广播数据之中。不可发现模式是三

种可发现模式当中的默认值，主机需要执行命令才能将该默认模式变为其他任意一种模式。

2有限可发现模式

处于有限可发现模式的外围设备，应在AD信息标识中设置有限可发现摸式位(Bit O)．并且清除一般可发现模式位(Bit l)。如果用户最近与设备进行过塑互，比如首次打开设备、

按下连接按钮或者通过某个用户界面设置设备为可发现模式，则该设备应当进人有限可发现

模式。

有限可发现模式大概只能维持30s的时间。这意味着，假如另一个设备发现了有限可发

现设备，它几乎可以断定这是一个外围设备，并且刚刚执行过用户操作。因此，这极有可能

是用户目前希望连接的设备之一。

由于有限可发现模式主要用于打算被发现并且进入连接的设备，因此，我们强烈建议在

广播数据中包括下列有用信息，以帮助接收者构建用户界面：

口发射功率值AD，用于计算路径损耗，判断距离以实现排序

口本地名称AD，用于显示设备的名称

口服务AD，用于根据设备支持的功能对设备进行过滤

有限可发现设备应该按照合理的广播间隔执行广播，一方面便于快速生成用户界面，另

一方面减少低功耗蓝牙主设备的扫描时间。建议的间隔为250ms - 500ms之间。显然，可

发现的、想建立连接的设备应该允许来自任何设备的连接，因而不会使用白名单。

3 -般可发现设备

处于一般可发现模式的外围设备，应在AD信息标识中设置一般可发现模式位(Bit l)，

并且清除有限可发现模式位(BitO)。当谩备打算被发现时，应使用一般可发现模式。

腺下列不同点之外，一般可发现模式与有限可发现模式基本类似：

口一般可发现设备的可发现时间没有限制；有限可发现设备最多维持该模式30s。

口一般可发现设备建议的广播时间间隔更长，介于1.28s和2.56s之间；而有限可发现

设备介于250ms和500ms之间。

必须指出，发现一般可发现设备需要较长时间。遗其实是有意为之，在生成用户界面

时．最好是把有限可发现设备排在列表最上方，而把一般可发现设备放在它们后面。

同样，广播数据包也应该包含发送功率值以及其他用于构造用户界面的信息，以便连接

设备实现优先级排序，或是过滤需要的可发现设备。

4可发现规程

希望找到可发现外设的中央设备，可以使用有限可发现规程或一般可发现规程。除了基

于AD信息标识执行的过滤操作不同，这两个规程基本上是一样的。

如果广播数据包中没有AD信息标识，或者虽然存在该信息，但既不古有限可发现模式

位也不含一般可发现模式位，则视该广播者为不可发现设备，且不会被发现。

如果AD信息标识设置了有限可发现模式位，则该设备总是可发现的。无论使用有限可

发现规程或一般可发现规程均是如此。

如果AD信息标识设置为一般可发现模式值，仅当使用一般可发现规程时，该外围设备

才是可发现的。也就是说，这些外围设备不会被使用有限可发现规程的设备所发现。

如表12-1所示，一般可发现规程将发现所有可发现的外围设备，无论它们为有限可发现或一般可发现外围设备，而在有限可发现规程仅仅能够发现有限可发现的外围设备。

衰12-1可麓现性

12.3.3可连接性

在GAP中，只有处于外围角色的设备才可使用可连接模式。尝试连接外围设备的那些

设备处于中央角色，使用连接建立规程。

就可连接性而言，可连接的外围设备可处于下列三种模式：不可连接模式、定向可莲接

模式和无向可连接模式。然而，对中央设备而言可连接性更为复杂，存在四种不同的连接建

立规程：自动、一般、选择性和定向连接建立规程。

从OAP的角度来看，可连接性要比可发现性容易管理。链路层提供了两种类型的可

连接广播报文：ADV\_IND和ADV DIRECT IND，以及两种类型的不可连接广播报文：

ADV NONCONN IND和ADV SCAN IND。因此，主机总是可以根据它的连接模式选择

正确的广播报文类型。

1．不可莲接模式

处于不可连接模式的设备，在广播时不允许使用可连接广播报文类型。也就是说，它们

只允许使用ADV\_\_ NONCONN IND或ADV SCAN IND两种广播报文类型。不可连接模式

是默认模式，要将外围设备变为可连接，需要主机执行指定的操作。

2定向可连接模式

如果希望快速连接中央设备，外围设备应使用窟向可连接模式。该模式采用ADV

DIRECT二IND广播报文，不古主机的广播数据，正因如此，无法与可发现模式相结合。

由于定向广播报文的发送频率很快，该模式最多持续1.28s，之后控制器将自动停止广

播。在超时之后，不允许主机立刻启动定向可连接模式，否则可能严重干扰其他设备的广

播操作或连接建立操作。因此，建议在定向可连接模式之后选择无向可连接模式作为备用

方案。

ADV- DIRECT一IND广播报文既包含当前外围设备的地址，又包含中央设备的地址。也

就是说，外围设备必须曾经与该中央设备建立过连接才能获悉其地址。因此，我们假定该模

式只会用于曾经绑定过的设备。

3无向可建接模式

一个可连接的外围设备如果无需快速建立连接，或者打算尽可能地节省能量，则可阻使

用无向连接模式。该模式采用ADV IND广播报文。由于ADV IND广播报文包含AD信息

标识，设备处于可发现模式的同时，也可处于无向可连接模式。 直到主机切换为不可连接模式或者连接建立，外围设备将始终处于无向可连接模式。因

此，新连接一旦中断，设备假如还打算保持其可连接性，必须再次进人无向可连接模式。显

然，一个已经建立连接的设备不能再处于可连接状态，否则链路层将以从设备角色维持两个

连接，这不符合链路层的规范。

4．自动建接建立规程

自动连接建立规程用来向多个设备同时发起连接。对低功耗蓝牙来说典型的用户场景

是：一个中央设备的主机与多个外围设备绑定，只要它们开始广播，便立刻与其建立连接。

比如，当传感器设备有新的读数时，中央设备必须有能力在其广播开始时便与之进行连接。

为实现同时向多个设备发起连接，中央设备的主机首先必须生成白名单，其中包括应

与之建立连接的设备的集合，再根据白名单发起连接（关于白名单的更多信息请参考第8章

8.6.1节）。通常，主机使用已绑定设备集作为白名单。一旦找到其中的设备，使用定向可连

接模式或无向可连接模式的主机即可建立连接。如果还需要与其他外设连接，自动连接建立

规程将再次启动。

该规程有两个缺点：首先，只能使用一套基本连接参数；其次，不能连接使用私有地址

的设备。

由于连接参数由中央设备的主机决定，在发起连接时提供，因此不可熊为不同的外围设

备设置不同的连接参数。假如外围设备的类型大同小异，这种机制工作良好，至少从连接参

数的角度上看没什么问题。当然，连接建立后还可以对连接参数进行调整。可见，在使用该

规程时，有必要选择一套相对稳妥的连接参数。

在该规程中，私有地址同样会导致严重的问题。因为私有地址必须频繁、随机地改变才

能提供隐私性，任何时候要对设备即将使用的私有地址进行预测无异于绿术求鱼。正因为如

此，该规程不能用来连接那些使用私有地址的外围设备。

5 -般连接建立规程

要解决上一节中的问题，可以采用一般连接建立规程。与自动连接建立规程略有所不

同，该规程并不使用白名单，而是采用被动扫描寻找所有广播设备。

对于那些使用可解析私有地址的设备，它们的地址将与所有IRK进行匹配，以便找出

中央设备打算连接的那些设备。如果地址可以正确解析，主机将停止扫描，并将该私有地址

用于定向连接建立规程。

显然．如果设备收到的并非可解析私有地址，但该地址直接出现在了待建立连接的外围

设备列表之中，那么主机将停止扫描，并将该已知设备地址用于定向连接建立规程。

该规程的缺点在于，在主机发现广播设备和开始直接连接之间，福要一定的处理时间。

这意味着，可连接外围谩备至少需要发送两个可连接广播数据包才能建立连接。这是使用私

有地址的自然结果。

一般连接建立规程还需要主机处理控制器收到的所有广播报文，包括非可连接广播报

文，以及来自中央设备不感兴趣的其他设备的报文。这要比自动连接建立规程消耗的能量多得多。

6选择蓬接建立规程

选择连接建立规程用来同时向多个设备发起连接，但是设备的硅接参数各不相同。这解

决了自动连接建立规程里的单一连接参数问题。

执行选择连接建立规程，主机首先将待连接设备集故人白名单，而后启用白名单并开始

扫描。正在广播的外围设备只有在白名单上的才会被提交给主机，而该区域内的其他设备会

被控制器立即过滤。

对于那些白名单上的设备，主机可以从控制器收到其广播信息。它首先检查该设备是否

使用可连接广播数据包类型，如果是这样，主机就停止扫描，使用定向连接建立规程向该地

址发起连接。由于各个外围设备可能希望配置不同的连接参数，主机可以查找该外围设备期

望的连接参数，用于定向连接建立规程。

与一般连接建立规程相比，该规程也有类似的缺点，那就是最少需要耗费两个广播报文

才可建立连接。与自动连接建立规程相比，它解奂了单一连接参数的问题，不过投有解决私

有地址解析带来的私有性同题。

7定向连接建立规程

之前的许多规程提到了定向连接建立规程。大体上，该规程采用一套特有的连接参数与

指定设备建立连接。

该规程直接向单个设各地址发起连接，不再考虑白名单（更多信息参考第8章8.6.2

节）。选就解释了一般连接建立规程和选掉性连接建立规程复用该规程的原因：一旦获得了

设备的广播地址，直接调用该规程建立实际连接。

12.3.4绑定

正如可发现性和可连接性那样，绑定也定义了模式和规程。只不过模式只有两种：不可

绑定模式和可绑定模式；而规程只有一种：绑定规程。

1不可绑定模式

设备的默认模式为不可绑定模式。此时设备不接受绑定，不变换也不存储密钥。

2可绑定模式

设备如果想被绑定，必须处于可绑定模式。此时，它将接受对端设备的绑定请求。要进

入可绑定模式，在配对请求消息的认证需求中，应设置绑定位（更多关于配对的信息，请参

考第II章11.2.2节）。如果可能的话，设备将交换安全密钥并视情况存储它们。

3绑定规程

希望与其他可信设备进行绑定的设备可以使用绑定规程。当使用绑定规程时设备将发起

配对，并在配对请求消息的认证需求中设置绑定位。

具体而言，使用可绑定规程的设备应发起配对并设置绑定位，如果对等设备是可绑定的，它将回复响应并同样设置可绑定位。一切顺利的话，密钥将在链路加密之后进行相互分

发并被存储起来。密钥的分发和存储是设备已绑定的标识。

12.4安全模式

在配对时．认证的强度取决于选择的算法，只有足够有效的认证才能判断对端设备是否

名副其实。举个例子，投有输人或输出能力的设备由于没有办法证明自己，所以无法证明它

就是实际连接的设备。我们称这种情况为非认证配对。

其他的设备，比如键盘和计算机等能够通过带外的方式传输它们的身份信息，并用其来

执行认证。计算机在屏幕上显示6位数字，再要求用户在键盘上键入上述数字。二者于是都

获得了6位数字的信息，但任何窃听连接的设备不能通过侦听数据包的方式获得该共享数

字。上述方式称之为认证配对。

绑定时，分配的密钥应标记认证算法的强度。具体来说，设备如果使用非认证算法进行

配对，那么绑定时分配的密钥应当标记为非认证密钥；而如果使用认证算法进行配对，该密

钥应当标记为认证密钥。

在此之后，任何重连操作如果用到了上述打过标记的安全密钥，仍将枯用初始配对时采

用的算法强度。这样一来，在描述一个加密的连接时就可以采用“未认证的配对加密”或

“带认证的配对加密”来说明，即便配对在之前的连接中就巳经完成。

安全模式

GAP定义了两类安全模式和三个安全等级。第一类安全模式用于在连接内部提供不同

级别的加密，第二类安全模式用于提供不同级别的数据签名保护。这两类安全模式被服务用

于描述其所需的安全性级别，比如一些服务对安全性完全没有要求，其他一些服务则希望尽

可能提高安全性来防止窃听，确保身份认证需求和保密性需求得到满足。

安全模式和级别的定义如下：

口安全模式1等缎1：无安全性

口安全模式l等级2：带加密的未认证配对

口安全模式l等级3：带加密的认证配对

口安全模式2等级1：带数据签名的来认证配对

口安全模式2等级2：带数据签名的认证配对

当两台设备之间无需安全要求时，使用安全模式l等级l。这是链路的默认安全级别。

当要求数据机密性但是不要求认证（或难以进行认证）时，使用安全模式l等级2。当

然，在该安全级别下，也可以在加密链路上发送投有安全性要求的数据。

当对数据的机密性和认证均有要求时，使用安全模式l等级3。这是强度最高的安全模

式等级。当然，在该安全级别下，也可以在链路中传输那些只箍较低安全级别的数据。

安全模式2等级l用于既不需要数据的机密性又不需要认证的场合。 当不需要数据机密性但是需要身份认证时，使用安全模式2等级2。当然，在使用认证

的链路中也可以传输无需认证的数据。

如表12-2所示，在使用更高安全级别的链路中，可以传输较低安全模式和安全等级的

数据。举个例子，一个加密的且使用认证配对的链路能够发送任意类型的数据。另一个例

子，一个链路并未加密，但是拥有之前配对获得的认证密钥，则该链路可用于发送要求安全

模式2的数据；不过，倘若要发送要求安全模式l等级2或安全模式1等级3的数据，链路

首先应当进行加密。

表12-2安全楼式和级别

┏━━━━┳━━━━━━━━━━━━━━┳━━━━━━━━━━━┳━━━━━━━━━━━━━┳━━━━━━━━┓

┃ ┃ ┃ ┃ ┃加密链路，目从 ┃

┃ ┃ 来加密链路且未认证配对 ┃未加密链路但认证配对 ┃ 加密链路但未认证配对 ┃ ┃

┃ ┃ ┃ ┃ ┃ 证配对 ┃

┣━━━━╋━━━━━━━━━━━━━━╋━━━━━━━━━━━╋━━━━━━━━━━━━━╋━━━━━━━━┫

┃安全 ┃ ┃ ┃ ┃ ┃

┃模式1 ┃ 发送敷据 ┃发送技据 ┃ 发送数据 ┃ 发送数据 ┃

┃等级1 ┃ ┃ ┃ ┃ ┃

┣━━━━╋━━━━━━━━━━━━━━╋━━━━━━━━━━━╋━━━━━━━━━━━━━╋━━━━━━━━┫

┃安全 ┃ ┃ ┃ ┃ ┃

┃模式1 ┃ 加密并发进数据 ┃加密并发进数据 ┃ 发送数据 ┃ 发送数据 ┃

┃等fa2 ┃ ┃ ┃ ┃ ┃

┣━━━━╋━━━━━━━━━━━━━━╋━━━━━━━━━━━╋━━━━━━━━━━━━━╋━━━━━━━━┫

┃安生 ┃ 不可能——须通过认证配对 ┃加密井发迸数据 ┃ 不可盹——须通过认证配 ┃ ┃

┃模式l ┃修复 ┃ ┃对謦复 ┃ 发送散据 ┃

┃等级】 ┃ ┃ ┃ ┃ ┃

┣━━━━╋━━━━━━━━━━━━━━╋━━━━━━━━━━━╋━━━━━━━━━━━━━╋━━━━━━━━┫

┃安垒 ┃ ┃ ┃ ┃ ┃

┃模式2 ┃ 使用认证密钳签名数据 ┃使用认证密蜩签名数据 ┃ 发送数据 ┃ ┃

┃等级l ┃ ┃ ┃ ┃ ┃

┣━━━━╋━━━━━━━━━━━━━━╋━━━━━━━━━━━╋━━━━━━━━━━━━━╋━━━━━━━━┫

┃安全 ┃ 不可能——绷过认证配对 ┃使用认证密锕签名敦据 ┃ 不可能——须通过认证配 ┃ ┃

┃M 2 ┃任复 ┃ ┃对修复 ┃ 发送数据 ┃

┃等级2 ┃ ┃ ┃ ┃ ┃

┗━━━━┻━━━━━━━━━━━━━━┻━━━━━━━━━━━┻━━━━━━━━━━━━━┻━━━━━━━━┛

12.5广播数据

设备在发送广播报文时，必须遵循固定的广播数据格式或扫描响应数据格式。梏式指的

是一串广播数据结构。各结构的开始处均含有一个长度字段，表示该结构其余部分的字节长

度。紧接着长度的是广播数据类型宇段，通常为1个字节，也可能是两三个字节或更长。假

如不认识某广播数据类型，设备可以将其忽略并跳到下个结构。结构内的其他任何数据字节

都由数据类型决定。

例如，发送功率等级(饥power level)为1个字节的数据类型，位于长度字段和数据类

型字置之后。这意味着在广播报文或扫描响应包中，发送功率等级总共需要3个字节：一个

长度字节，一个数据类型字节，以及一个字节记录实际的功率等级。

广播数据可以是变长的。比如，本地设备名称可以从几个字节到数十字节不等。不过，

由于数据结构起始处的长度字殷约束了广播数据的长度，所以没有必要引人终止字节符。指定的数据也可能不完整，比如完整的数据超过了报文的限制长度，或者从数据包移除了那些

没有价值的部分。

对于那些可能溢出的数据，通常使用多种数据类型表示：一是可能的完整值，另一个是

缩减后的版本。这样一来，接收者可以确定数据并非完整值，从而采用其他方式获得数据

的剩余部分。比如，本地名称可能长于短广播报文的长度限制，不过运用OAP的设备名称

(device name)特性，仍然可以完成完整名称的读取。

12.5.1标识

标识（flags）AD是位字段的序列，可以是从O字节到若干字节的任意长度。在广播数

据之外的任意字节默认值均为“。”，这意味着，如果有需要的话标识宇段可以扩展额外的标

识位。接收到标识AD的设备假如发现其长度比预想的要长，可以截断该数据，并不会导致

任何问题。

低功耗蓝牙定义了如下标识：

口有限可发现模式

口通用可发现模式

口不支持BR/EDR

口设备同时支持LE和BR/EDR（控制器）

口设备同时支持LE和BR/EDR（主机）

有限可发现模式和通用可发现模式标识位与第12.3.2节的定义完全相同。

不支持BR/EDR标识位用来在建立连接前通知对端设备，本方不支持经典蓝牙，因此

必须使用低功耗蓝牙建立连接。这一蠃十分重要，因为两个双模设备之间通常不会建立低

功耗蓝牙连接。因此，双模设备必须检查该标识位，从而判断采用何种方式向对方设备发

起建接。

设备同时支持LE和BR/EDR标识位——一个关于控制器，一个关于主机——用于判断

对端设备能否在已有经典蓝牙连接的同时还可以发起低功耗连接。由于控制器和主机的能力

不同，在试图向同一设备建立两类连接前，必须检查上述两个标识位。

12.5.2服务

有多种不同类型的服务广播数据类型，每种类型公开了一个服务UUID列表。IJUID包

括两种大小：裁剪的16位值以及完整的128位值。有时设备不愿意公开所有支持的服务，

或者完整的服务列表可能很长，这类设备必须既支持完整的服务列表，又支持部分服务列

表。综上，共有下列四种不同的服务广播数据类型：

口16位服务rnnD完整列表

口16位服务UUID部分列表

口128位服务UUID完整列表

D 128位服务UUID部分列表

第12章通甩访r日授范‘203

12.5.3本地名称

本地名称广播数据类型共有两类：

口完整的本地名称

口裁剪的本地名称

当本地名称太长而无法放人一个广播报文中时，使用裁剪的本地名称数据类型。本地名

称是UTF-8字符串，允许截断。

12.5.4发射功率等级

发射功率等级广播数据类型是指传输该广播数据包时采用的功率值，长度为1个字节，

单位为dBm。

12.5.5从设备连接间隔范围

从设备连接间隔范围代表外围设备倾向的连接间隔。由于连接参数完全由主设备控制，

连接参数的范围通常在定向连接建立前就设置好了。利用从设备连接间隔范围，中央设备能

够获得外围设备倾向的连接间隔的参考值，用于发起连接。

间隔参数包括两个16位的值，第一个为最小连接间隔，第二个为最大连接间隔。二者

位于同一单元，正如用于发起连接的HCI命令使用的参数。

12.5.6服务请求

一些时候，外围设备想要使用中央设备的某个服务，但是中央设备却不想广播，或者外

围设备不愿意花费能量扫描所需的中央设备。针对这个问题的解决方案是引入一个服务列

表，包括了外围设备希望中央设备支持的服务。该服务请求为正在搜寻外围设备的中央设备

提供了连接能力，它们能够据此判断哪些外围设各最有可能包含和服务对应的客户端功能。

服务请求广播数据包括16位服务UUID或128位服务LrUID的部分列表。

12.5.7服务数据

设置服务广播时，使用的是“服务数据”广播数据类型。服务数据的起始两个字节是

16位rnnD，表示服务号，其他字节为实际的服务数据。

12.5.8制造商指定数据

最后一种广播数据类型为制造商指定数据。数据的起始两个字节为16位的公司标志，

接着是公司指定的数据。这便意味着任何公司都可以定义自己的广播数据结构，并将其放在

广播报文中。12.6 GAP服务

GAP定义了通用属性规范服务，为设备提供了一种确定信息的方式，包括自身的名称、

外观特性、如何涟接等。概括说来，服务提供了下列五项特性：

。设备名

口外观

口外围设备隐私标识

口重连地址

口外围设备首选连接参数

12.6.1设备名特性

设备名特性是一个UTF-8字符串，表示设备的名称。有趣的是，由于一个设备只允许

有一个设备名特性，直接使用接UUID读取请求便能快速地获得设备名，无需首先执行服务

发现或者特性发现。

12.6.2外观特性

外观特性是一个16位的值，用来列举设备的外观样式。在经典蓝牙的问询（inquiry）

阶段提供了一种可发现的设备类型宇段，既包含设备的外观，又包括设备的功能。然而事实

证明这种设计存在缺陷：由于一些制造商过滤了设备类型，从而导致了交互性问题。低功耗

蓝牙的解决方案是：避免过早进行信息过滤，首先集中重要的信息来构建用户界面。因此，

外观特性通常被用来在用户界面的设备旁边显示设备的图标。

该特性值为枚举类型，在编号分配文件中定义。该文件目前仍在更新，当需要分配新的

编号时，应加入该文件。

12.6.3外围设备隐私标识

外围设备隐私标识表示设备是否正在使用隐私。该特性既可读又可与，这一点便其在使

用时常常会令人困惑。

口假如该特性不存在，鄢么设备不支持隐私，始终使用公开的蓝牙地址。

口假如该特性存在但被设置为“禁用”，那么设备不使用私有地址而使用公开的蓝牙地

址。

口假如该特性存在并且被设置为“启用”，设备将始终使用私有地址。但是，这不意味

该设备一定能够用私有地址建立连接。

口假如存在重连地址（reconnection address）特性，建立连接时将使用重连地址，而非

广播地址。

也就是说，只有当存在重连地址并且隐私标识设为“启用”时，设备才能真正实现带隐

私的重连。 中央设备可能不希望和其他设备使用隐私，尤其是那些将外围设备隐私标识设为“禁

用”的设备。这种情况下，中央设备可以尝试写入该特性，尽管不一定能成功。

由于与外围设备绑定的中央设备通常不止一个，当与第二个或更多中央设备启用隐私

时，首个绑定的中央设备将无法重连。这是由于该中央设备将会向外围设备的公有地址发起

重连，而非上次读取的重连地址。

禁用隐私也可能导致类似的问题。然而，与上述发生功能紊乱的情形不同，中央设备能

够侦测到外围设备的广播报文使用了公有地址，从而判断其禁用了隐私，可向共有地址发起

重连。在重连时，它可以检查外围设备隐私标识，确认隐私已经被禁用。

因此，使用隐私的唯一的安全方法在于从外围设备建立连接起就一直让其保持“启用”。

时常改变隐私状态毫无裨益。

12 6 4重连地址

如上所述，在启用隐私时应使用重连地址。它用于外围设备与曾经绑定的中央设备进行

重连，且该中央设备知道该外围设备启用了隐私的情况。

使用时，重连地址应为不可解析的私有地址。这样一来，要想真正找到与之对应的外围

设备，只能是那些曾经与该外围设备进行了绑定且读取了其重连地址的设备。

将重莲地址纳人整个系统的初衷，是为了在启用隐私时实现能耗的优他。试想很如没有

这样一个地址，所有私有的外围设备将其能使用可解析的私有地址进行广播；而中央设备则

需要检查每个收到的广播报文与已知的IRK进行检验——该过程由于牵扯到主机，能量消

耗较高。相反，如果运用静态的非解析私有地址，只需将其放^白名单，控制器便能够有效

地建立与对端设备的连接，绕过了耗能的主机处理。

重连地址还有个有趣的地方，即必须在每次重连时更换以保护隐私。可见，每次中央设

备使用某一外围设备的重连地址进行重连后，都应写入一个新的重连地址，且这一操作应在

其他任何操作之前进行。

上述的写入过程有可能遭遇连接失败的情况。这可能是由于外围设备收到了写入请求，

但没法回复写入响应；或者外围设备压根就役收到写入请求。此时，中央设备应假定外围设

备既可能沿用旧的重连地址，也可能使用新写入的地址。因此，二者都应被添加至白名单。

12.6.5外围设备首选连接参数

许多外围设备是为了某个具体用例而设计的，为使其更好地工作，通常配有若干首选参

数。外围设备通过外围设备首选连接参数特性公开这些参数。因此，中央设备不用猜测这些

具体参数，而只需在首次连接时读取它们，并且尽快调整连接参数即可。同时，当中央设备

重连到外设时，如呆仍然保留了之前的参数，则可以在连接请求中直接设置，免去了连接后

修改参数的麻烦。

第四部分

应 用

第13章描述应用如何使用低功耗蓝牙与外围设备进行交互。

第14章阐述如何澧计外围设备以便最佳地利用低功耗蓝牙技术。

第15章说明如何对设计实施鉴定以便发布或者销售蓝牙产品。

第13章

中央设备

人既在又不在整十飓风的中一心。

——哈罗德·品特(Harold Pinter)

低功耗蓝牙的应用可以分成两类：一类为中央设备而设计，用于找到并与外围设备进行

交互，另一类为外围设备而设计，用于向中央设备上运行的应用提供信息。二者都需要相应

的知识，才能把前述的所有部件组装成一个完整的功能模块。本章将从中央设备的视角来审

楹应用，下一章则具体分析外围设备该如何设计。

13.1背景

中央设备对于低功耗蓝牙的应用而言至关重要。它们往往具有丰富的功能和复杂的用户

界面。本章不打算讲解如何为某类设备开发具体的应用，那些是平台开发人员的事情。不

过，本章会涉及一些需要考虑的重要事项，比如怎样优化用户体验、实现最优节能等。

13.2发现设备

中央设备的一个首要任务是发现其他的设备。要实现这一目标，既可以使用被动扫描，

也可以使用主动扫描。被动扫描是指中央设备被动地侦听来自外围设备的广播报文。主动扫

描是指侦听到外围设备后，中央设备向其询问更多信息。

如果中央设备仅需了解周围有些什么设备，或者它们携带了哪些信息，则应当使用被动

扫描。如果中央设备还需生成一个用户界面，比如一个屏幕或窗口，那么应使用主动扫描。

这是因为遁过主动扫描还能够得到一些跗加的信息，而这对于生成用户界面的设备列表程有

帮助。

通过扫描可以发现的信息不但有设备名称，还有用来识别设备的唯一标识。中央设备可#13t十央设备．209

以利用该标识连接到对端设备。此外，在扫描响应包中还可能包含一些服务相关的有用信

息，比如电池的电量或当前时间。

有些信息难以通过扫描立即获得，但可以在快速连接对端设备之后进行读取。打个比

方，设备的完整名称可能很长，无法放入单个广播数据包或扫描响应数据包。有必要的话可

以与该设备建立连接，再读取设备名称的剩余部分。一且这样做，中央设备上的应用程序必

须十分小心，应避免弄错服务或读错特性。

因此，应用程序首先要进行服务发现，找到那些需要的服务，比如OAP服务，进而

在该服务中查找设备名称的特性值，并直接使用按UUID读取特性值(Read Characteristic

Value by UUID)这一规程进行读取。GAP服务规格书要求每台设备只能运行一个GAP服务，

而该服务只会有一个设备名称的特性值。如果设备名太长，超过了一个属性协议的响应报文

所能携带的数据量，可以利用第一个响应报文中的句柄值来发送多个请求，从而得到设备名

称的剩余部分。

类似的规程可用于获取其他的有用信息，如GAP的外观特性值。

应当注意，使用被动或主动扫描时，应用不仅能够获得广播数据包的内容，还能在控制

器内得到这些数据包酌接收信号强度( RSSI)。用发射功率（Tx Power，可能在广播报文里

携带）减去RSSI值就能大致估算出路径损耗（path loss），进而判断本设备与中央设备的距

离，如下列公式所示：

路径损耗=发射功率接收信号强度

路径损耗如果很小，位于。- 20之间，那么设备之间的距离应该很近。反之，假如损

耗很大，比如大于70，则说明设备之间的距离很远。鉴于无线传输的特性，上述值应该取

若干秒内的平均值以消除多径干扰。这样一来，在用户界面上可以把路损最小的设备排在列

表的顶端，让用户优先看到那些距离较近的设备。如果有很多设备，用户可以通过向下拖动

列表来查看那些距离更远的设备。

还有一个注意事项，有些设备虽然在广播，但不一定是可发现设备。除非用户界面正在

显示广播服务数据，否则那些不可发现设备应当从设备列表中予以删除。

13.3连接设备

在之前的章节中提到了生成可发现的设备列表，用户可以从中选择某个（或几个）设备

进行交互。接下来的任务便是与之建立连接。

选择并连接到用户界面中的设备将会引发一个连接发起过程。然而，如果所选设备使用

的是私有地址，那么要特别注意该设备是否近期变更了私有地址。假如情况属实，发起的连

接将会超时，玎发现设备的列表也会重新刷新。设备也许还在原处，只不过出于保护隐私的

目的换了一个私有地址。不过，假如设备之间曾经变换过身份解析密钥(IRKs)并实施过绑

定，这种情况也不会有什么问题。中央设备能够自动更新密钥列表并解析新的私有地址，最终选出属于该设备的密钥，继而与之连接。

在发起连接时，首先根据两台设备的要求选择连接参数。通常外围设备有一个“客户端

首选连接参数”特性，以便向中央设备提供明确的提示信息。在初次连接时，这些信息还不

可用，因此初次连接时选用的参数需要在降低功耗和加快获得设备特性之间取得某种折中。

最佳参数集往往具有较小的莲接间隔，这样初次连接时能够很快地交换属性协议和链路

层的控制信息；此外还应有相对较大的从设备延迟，以便外围设备尽可能地节省能量。例

如，把连接间隔设为15ms - 30ms，从设备时延设为150ms，不但能以最高60Hz的连接间

隔快速获得外围设备的数据，也使得从设备能以6Hz的空闲频率工作。

对于中央设备上的应用所设定的参数，从设备可能会有不同的要求。中央设备应该总是

试图兑现这些要求，特别是在其读取了所有想要读取的数据之后。

一旦有要发送或接收的数据，中央设备随时可以改变连接延迟。

13.4这个设备能做什么

中央设备连接到外围设备后，会想知道这个设备到底能干点什么。为了收集这些信息，

它将以下列腰序执行四个规程：首要服务发现、关系发现、特性发现以及描述符发现。

第一个规程是首要服务发现，这些服务描述了设备的作周。举个例子，如果该设备装有

电池，首要服务将公开电池服务；如果该设备具有温度传感器，它将公开温度服务；如果该

设备具有电池内的温度传感器，第二个温度不会通过首要服务公开，原因是它可能混淆中央

设备的功能。首要服务只公开设备的首要功能。

接下来，由于中央设备已知的每个首要服务还能包吉其他服务，需要进一步发现这些关

系，包括延伸、组合、重用等。

一个设备拥有的服务集(set of services)不一定决定外围设备支持的规范集(set of

profiles)。没有什么好办法能够迅速确定设备支持的规范集。实际中往往会用一个较复杂的

算法，即用外围设备公开的服务集来匹配中央设各支持的规范角色，同时检铡角色的有效

性。这不是一项简单的工作，能做这件事的设备都应该具有丰富的资源，并且这当中的复杂

性在低功耗蓝牙中并没有考虑。

上述方法的好处在于，未来即使某个客户端规范的角色使用了外围设备的服务集，在制

造时也不一定非得把它设计成外围设备。这就构造了一个非常灵活的可扩展系统，适用于绝

大多数采用下载应用模型的中央设备。

只要发现了服务，就可以发现特性集及其描述符。由于低功耗蓝牙的服务没有版本号，

因此，要想知道是不是具有某个可选劝能，只有检查与该可选功能相关联的某个公开的特

性。作为选择，特性属性及其描述符也可用来区分可进行为，比如通知（notification）和客

户端特性配置(client characteristic configuration)描述符等。13.5通用客户端

可以制造一种完全通用的客户端，按照人类可读的格式来读取和显示特性值。这给予丁

中央设备一种能力——即使不理解外围设备的某个服务或特性，也能够将其展现给用户。通

用客户端包含两个层面：第一层可以直接使用外围设备的可用信息，第二层在可用信息的基

础上，通过互联网获得更多的信息。

处于第一层的通用客户端会发现外围设备上的所有特性，过滤掉不能直接读取的特性，

以及那些没有特性显示格式(characteristic preseatation format)描述符的特性。特性显示格

式描述符包括了把二进制数据变成人类可读值所需要的大部分信息。具体而言，使用格式宇

段和指数字段，能够决定如何把一个定点值转换成一个更加直观的数字；而使用单位字段，

能够获得一个UUID．并通过它解码上述数字对应的单位。

大多数物理量的单位可以从BIPMo的单位列表获得，其中包括了所有标准的SI单位和

一些常见的单位。最后，描述符中还包括一个命名空间：描述（namcspace:dcscription）对，

允许客户端显示更细的单位粒度。比如，把一个16位的无符号数转换成表示重量的磅值，

同时指出该重量来自某个悬挂称重器，而不是为丰辆称重的桥称或者浴室的体重计。

处于第二层的通用客户端可以显示最复杂的特性值。由于特性显示格式描述符的能力

有限，它只能代表所有可能的数据结构的一个很小的子集。因此，第二层的通用客户端不

依赣于该描述符，而是利用了每一个特性类型都有的一个唯一编号UnD。在蓝牙技术联盟

(SiO)o的开发者网站上可以查找数据格式的XMI，表示。

无论在哪个服务规格书中，每一个特性类型都必须有一个定义该特性的XML文件。这

些文件既可用于设备测试，也可以用来判断特性值的结构。XML文件能够为每一种数据表

示执行编码，包括枚举类型、位字殷、基于枚举值或位字置的可选字段，以及二进制和十进

制指数定点格式编码等。这些字驻也可以连接在一起，从而构成复杂的数据结构。

因此，一个连接互联网的通用客户端首先会获得可读的特性值，向网站执行一个简单查

询，下载描述特性的XML文件，最后再通过这个文件向用户显示数值。

13.6与服务交互

一旦确定与外围设备上的服务进行交互，中央设备将连接到该设备，并开始读写特性值

和描述符。上述操作通过属性协议进行，该协议本质上是一种无状态的协议。

属性协议不仅在连接时没有状态，在连接和连接之间也不保存状态——它根本兢不是一

个“会话协议”。为了绕开这一限制，所有的状态保持在应用层，应用程序可以就如何节约

能源做出明智的决策。接下来的章节将描述如何实现这一点。

e国际标准计量局，更多信息请查看http:/Ⅳ/WwW.bipma叫。

窜在b ttps：∥d eve LoperbI uet叫Ih。rai可吼查看所有可能的特性。13.6.1可读特性

最基本的服务只是简单地公开了一组可读的特性。就好像移动设备的信息服务仅仅提供

了若干个与设备信息相关的特性。基本服务用起来非常容易，只要客户端理解服务的某个特

性，就可以读取相应的值；对于长度较短的值只要一个读取请求，对于较长的值则需要多个

读取请求。

1可写特性

接下来，更为复杂的情况是服务既包含可读特性，又包含可写特性。链路损耗服务

（Link Loss Service）是一个很好的例子，它有一个可写的特性——报警级别(Alert Level)．

客户端可以写入该特性，当两个设备之间连接丢失时配置应该执行的操作。

具体而言，如果客户端在该特性中写入“无需报警”，那么当连接断开时，服务器将什

么都不做。如果客户端写入“轻微报警”．那么不管由于什么原因导致了连接丢失，服务器

将使用轻微的报警方式，提醒用户发生了情况。如果客户端写入“强烈报警”，当连接断开

时，服务器将使用各种支持的报警方式，如铃铛、口哨声、灯光闪烁等。

或许一些对于面向连接的协议更有经验的人可能会担心这样的问题：你是如何“启动”

一项服务，又是如何“停止”这项服务的呢？

要弄清楚上述问题，关键在于理解服务的状态是通过其特性对外公开昀。接着上面的例

子，在链路损耗服务中的报警级别特性决定了设备的行为。如果该特性的值为“无需报警”，

服务器将不执行任何操作，体可以认为这是一种“未连接”状态；如果特性的值为“轻微报

警”或“强烈报警”，服务器会在客户端断开连接时执行一些操作，你可以认为如果特性值

设置为其中之一时，即为“连接”状态。

当然，这意昧着如果客户想优雅地与服务器断开连接，但该服务器配置丁“轻微报警”

或“强烈报警”级别，那么客户端在断开连接前必须将该特性改写为“无需报警”。

还有些人也许会问，如果一台服务器在同一时间维持了多个连接，一个客户端在链路损

耗服务的报警级别特性中写入了“轻微报警”，但其他客户端投有这样做，那么，当连接断

开时究竟会出现些什么问题呢？答案是，完全没有问题！这是由于对每个客户端来说，各自

的服务特性值可以是不同的。假如客户端A写道“轻微报警”，在客户端B断开时并不会导

致服务器发出提醒，因为客户端B的值仍然是“无需报警”。

这种类型的服务和以下各节中所描述的服务有一个重要区别，即这些服务的特性既是可

读的又是可写的。也就是说，客户端可以随时检查服务的当前状态，而不用记忆之前都做过

些什么。这在客户端的应用程序意朴终止时（也许是因为程序调试导致）格外有用。重新启

动后，客户端可以通过读取相应的特性值来更新对方的状态。

13.6.2控制点

还有一种服务类型虽然不保存状态，但是客户仍然可以向该服务写入值。这看上去有点

奇怪，特别是回想起上一节中的服务似乎都会保持状态。一个服务怎么能既有一个可写特第13章十央设备◆213

性，又不保存这种状态？答案很简单：服务将立即使用写入的值，在其被消耗之后不必存储

该值。我们把这种特性称为控制点(control point)。

上文描述的链路损耗服务有一个可写的报警级别特性，可以决定两个设备断开时的行

为。但如果客户想让服务器立刻报警呢？当然，它可以为该服务写入适当的值，然后断开连

接，但这种做法具有破坏性。客户端上可能还有一些应用程序在使用其他服务，它们可不希

望连接突然就断了。有没有更好的方法？答案是肯定的，使用控制点特性就能实现。

该服务被称为即时报警服务(Immediate Alert Service)，其特性称为报警级别。没错，

这里的报警级别与用在链路损耗服务中的那个完全一样。特性不过是一种数据格式，在上述

情况下实际上是一个包含三个值的枚举类型，即“无需报警”，“轻微报警”和“强烈报警”。

行为是由服务决定的，而不是由特性决定的。

在即时报警服务中，报警级别特性是可写的，它将立即引发报警。因为报警是即时的，

因此该特性没有状态。不管写入什么值都会被立即消耗掉——用来触发报警而不会被保存。

因此，没有必要为这种特性设置可读属性，也没有保留状态的必要。

还有一种拄制点的类型，在下面的通知和指示一节将对其进行讨论。

基于控制点的服务类型所具有的优势在于，设备不用理会是哪个从设备在对控制点发号

施令，只需按照写入控制点的指令进行操作即可。

13.6.3状态机

接下来要介绍的服务类型公开丁一些可写控制点和一个或多个可读的特性，它们用于反

映一个状态机的状态。这里所谓的“状态机”就好比一部“机器”，它具有一个公开的状态，

并提供了一种从内部或外部改变状态的方法。基本上，状态机和之前描述的控制点的唯一区

别在于：状态机会将状态保存起来。你可以读取它，或者在它变化时通知客户端。

为帮助理解，让我们考虑一种时间同步状态机。它的当前状态可以是“什么也不干”，

或者是“搜索更准确的当前时间”。我们分别将它们定义为“空闲”和“搜索”状态，作为

状态机的公开状态。

接下来，我们需要一种方法控制设备的状态机。使用刚才介绍过的控制点就能实现，只

要将该控制点与状态机的外部状态联系起来即可。在时钟同步服务的例子中需要用到两个命

令：启动同步和取消同步。

这种方式有许多优点。比方说，通过状态机公开的状态特征，任何设备都可以询问状态

机的当前状态。这意味着，如果有三个客户端都要与服务器进行时间同步，他们鄯可以检查

后者的状态，只有当其处于空闲状态时才开始执行新的同步。

更实际地，每个设备都可以命令状态机启动同步。对于一个设计精良的状态机而言，即

使输入了无效的命令，它仍旧能够继续运行。举个例子，如果服务器的状态机正处于搜索状

态时，即使有多个客户端启动了同步的命令，状态机也不会出现问题。一种常见的做法是由

服务器向客户端发送错误响应，不过还有更好的做法：服务器通知客户端执行随机退避，之

后才能再启动同步。 状态机必须提供足够的鲁棒性，让所有的命令都有明确的状态转换。比方说，在时间同

步服务正处于空闲状态时，收到了客户端要求取消同步的请求，状态机应将其视为一个错

误，或者认为客户端并不清楚自己位于空闲状态，所以不予理会。状态机一般选择上述的第

二种处理方法，这样一来，无论客户端在何时执行何种命令都将是安全的。

然而．这种情况下产生了一个有趣的副作用：一个要求启动同步的设备并不能确保同步

执行完毕，这是因为该操作可能被另一个设备取消。作为一种替代方法，可以将同步服务定

义为一旦开始就必须执行完毕，只有启动服务的客户端才有权取消。

13.6.4通知和指示

服务公开的状态信息可能很快地改变，或者随机地发生了变换。对设备来说，采用轮询

的办法来探测状态变化是一种低效的方式。拿设备中的电池作为例子，不使用该设备的时

候，电弛电量可能每天仅改变一次；然而，如果该设备处于使用状态，电池电量可能每15

分钟左右改变一次。问题在于，客户端检查电池电量的频率设为多少才箅合适。如果每天检

查一次，一天下来客户端可能认为电池还是满的，但电池实际上只剩4%；如果每15分钟

检查一次，即使设备处于空闲状态，由于读取电池电量导致的额外能源消耗．电池电量的下

降速度会比应有的速度要快的多。

相反，另一种做法是对某个特性进行殴置，仅当特性值发生变化时才会告诉客户端。通

过这种方式，只有在状态实际改变时客户端才会收到通知。客户端可以等待外围设备的状态

发生变化，并在收到消息后有效地应对这些变化。

要做到这一点，客户端可以在服务器上配置特性来发送所需的通知(notifcation)。大多

数的通知发送由服务来定义，但其中一些还可以通过特性描述符进一步配置。

应该指出的是，某些特性不支持通知，只有那些有正确属性以及有客户端配置描述符的

特性才可以支持。

还应当指出的是，从服务接收数据的方式并非只有一种。通知能够在任何时候发出，并

且无需应用层进行确认。因此，必须意识到它们是不可靠的。很多时候这也许无关痛痒，例

如对某个状态机特性进行状态更新。但如果要读取一十血糖读数，服务器发送不可靠的数据

可万万不行。因此，客户端可以配置特性发送指示(indication)。指示和通知类似，唯一的

不同是指示需要客户端向服务器发送一条确认消息，告知数据已经被应用程序成功接收。

指示同样是利用客户端配置描述符来进行配置的。

13.7绑定

较之简单连接，一些设备需要维持更长时间的连接．还有些设备可能希望秘密地传输数

据，或者与曾经认证过的设备、而非当前区域里的可用设备通信。遗可以通过使用安垒模型

来实现，从根本上说是利用“绑定”。

如果客户端想与其他设备建立长期通信关系，首先应连接该设备，找到一些能用的服举I3幸十央设鲁0 215

务，然后启动一个安全连接。这些安全连接先是认证该设备为正确的设备，再对连接进行加

密阻确保机密性。最后，设备交换一些配对信息——这是关键的一步：如果客户端存储了该

配对信息，就与对端设备完成了“绑定”。

绑定的重要性在于，当客户端重新连接到该设备时，并不需要再次认证身份或交换配对

信息，荷只需将部分的绑定信息用于连接加密。于是设备之间便建立了一个认证的、保密的

数据连接。

绑定也提供了一些其他好处。设备绑定之后，服务器将为此客户端保存其配置数据。这

种模式将使重连设备立刻收到通知，而不需要重新配置服务器。

举个例子，如果在中央设备配置了一个特性，比如通知电池电量，那么每当其电量改变

时客户端就会得到通知。但是，如果客户端与服务器断开连接，服务器将无法发送通知。不

过，假如进行过绑定，当客户端重新连接到服务器时便可以立即收到通知。

这当中有一个地方值得注意：服务器可能忽略所有来自绑定设备的状态变化，而在重新

连接后询问客户端的当前状态。如果数据很少改变，这种方式无论对客户端还是服务器都是

一种低效的方式。该模型还假设客户端不会经常性地断开连接和重连，由于每次重连都要读

取所有状态位，重新连按的成本将显著提高。

低功耗蓝牙采取了另一种方法，让服务器不仅要记住配置了通知的特性，而且还要记住

客户端断开连接后该值是否已经改变。这就要为每一个可通知的特性增加一个关联位，当特

性值改变时，将其设为l。之后，当客户端重新连接时检查这些位，只有设置过关联位的特

性值才会向对方通知。

服务器针对绑定的连接保存配置信息，结果是客户端不再与服务器保持“连接”。相反，

客户端绑定井配置一个外围设备，执行完功能，接着断开连接，只有在必要时才会重新连接

该外设。有数据要发送时外围设备将变成可连接状态，中央设备根据需要重新与这些绑定过

的外围设备建立连接，并快速地接收数据。

中央设备甚至可以要求外围设备做一些事情。例如，中央设备向外围设备要求进行时间

同步，然后断开连接。一段时间以后，中央设备可以重新连接到外围设备，检查时间同步的

状态，尽管不保证同步完成。

这种方法的另一个主要优势是，客户端可以记住外围设备的属性句柄等信息。也就是

说，一且中央设备了解了外围设备的服务集，并且完成了配置，就能记住相关的属性句柄。

当中央设备重新连接到外围设备，它可以读取和写入这些属性句柄，而不用再做一欢服务扫

描。这减少了从连接到使用设备之间历需的时间。

13.8变更服务

正如上一节中，中央设备（客户端）能够记住或缓存不同连接的服务集和特性集。有些

设备将有能力改变或增加服务。例如，一台计算机或一部智能手机添加含有新服务的应用，

或者一个外围设备更新其固件。当发生这种情况时，客户端将不能读取任何属性，所有到该216．第四部舟应 用

服务器的请求都会失效。这保护了客户端不会读取到错误的属性。

伴随着错误消息，客户端还会收到来自GATT的服务变更（senwe changed）的通知。

由于客户端已经保存了这个属性句柄，当其收到通知后便可以执行适当的操作。

服务变更通知包括一系列已被更改的句柄。这意味着，如果设备只是增加或删除一个服

务，那么通知里将只含有这个服务的句柄范围。如果该设备改变了所有的服务，比如更新了

设备的操作系统，那么通知里将包括所有服务的句柄范围。

应注意，服务变更只与绑定的设备密切相关。如果中央设备没有和外围设备绑定，则不

能缓存属性句柄，也收不到任何服务变更的通知。换言之，对两个未绑定设备来说，客户端

必须在每次连接时刷新服务器的整个服务列表和特性列表。

在设计一个中央设备时．最关注的是它支持的功能。通常情况下，中央设备将实现一个

或多个客户端的角色，支持若干规范。要明白这当中的含义，必须就规范加以说明。

13.9.1定义规范

规范(profile)是针对给定用饲的设备功能描述。规范也称为配置文件，在其中规定了

角色。规范的不同角色作为设备生态系统的不同部分，保证了规范的正常运作。

每个规范角色定义了一套谩备必须实现的服务。有些规范角色不需要任何服务，有些只

需要一个服务，还有些则需要许多服务。通常情况下规范定义丁两个角色，比如一个报告者

角色和一个监视者角色。报告者角色只需一种服务，一般在外围设备上实现。监视者角色则

不需要任何服务，一般在中央设备上实现。

举个例子，接近规范(Proximity Profile)定义了两个配置角色：接近监视者(momtor)

和接近报告者(reporter)。接近报告者实现即时报警服务、链路损耗服务以及发射功率服务。

兰个服务相互独立，按照各自的服务规格书进行操作。在该规范模型中，这种服务之间的独

立性是实现未来的灵活性的重要保证。

接近监视者定义了客户端如何通过上述三个服务实现接近应用。例如，在发现了服务和

特性之后，监视者读取当前的发射功率特性值，获取报告者报文的接收信号强度，并比较二

者。于是，接近监视者可以判断两个设备之间的距离是否在可接受的范围内。如果这个距离

不能接受，接近监视者可以在即时报警服务中写人报警级别特性。

接近监视者也可以在链路损耗服务中写入报警级别特性。采用该方式，如果信号强度下

降非常快，以至于监视者还没来得及在即时报警服务中写入报警级别特性，连接已烃中断，

则接近报告者仍然可以以一种受控的方式发出提醒。

服务的组台让接近监视的应用成为可能。但即时报警服务也可以用于不同的场合，比如

通过在即时报警服务中写人报警级别特性，用户就能昭在界面上看出哪些是已经选择的设

备，不过客户端的这种行为将在其他的规范中定义。第13幸十央设鲁．217

13 9 2查找服务

一个规范需要做的第一件事，就是找到对端设备支持的服务。要做到遮一点首先必须发

现首要服务，具体来说就是要么根据服务类型查找首要服务，要么查找所有的首要服务。

这样一来，如果规范只用到某一服务，那么它可以只寻找需要的那个服务，而忽略设备

上的所有其他服务。这就好比要使用用户界面报警服务，只要找到该服务即可。

一个更复杂的规范可能需要发现多个服务。接近监视者这一规范角色就需要发现三个服

务。在搜索服务时，它可以和搜索单个服务那样一次查找一个，也可以一次找到所有的首要

服务，然后只存储那些需要的服务。

通常情况下，复杂的客户端会一次性地发现服务器的所有服务，将其缓存以便稍后使

用。然后，假如实现了某个规范的应用要求获得对端的所有服务，客户端将直接向其提供缓

存的服务列表。这种方案完全可行，因为即便服务器上的服务集发生变化，它会向客户端发

送一个通知，告知服务已经改变。

13.9.3查找特性

一旦发现了对端设备上的服务集，接下来的工作就是发现相关的特性集。和服务一样，

规范既可能只需一个特性，也可能需要许多特性。由于有的特性可能是可选的，因此，必须

在服务内搜寻它们，以检查其是否稃在。

客户端一般会寻找巳发现服务的每个特性。对于只有一个特性的最简单的服务，这种

操作快速而高效。此外，如果服务包含许多可进的特性，该操作能够找到服务实现的所有

特性。

13.9.4使用特性

发现特性之后就可以使用它们。例如，对于即时报警服务，接近监视者客户端可以在报

警级别特性值写入所需的报警级别。

应当指出，由于一些服务只有一个可读特性，特性的发现和特性值的读取可以组合到一

块，变成一个操作。举个例子，一旦发现了发射功率服务，发射功率特性可通过GATT规

程按类型读取特性（read characteristic by type）。这意味着刚才提到的单独的特性查找步骤

已经不再需要。

有些特性可以支持通知或指示。客户端假如要使用这些服务器的功能，必须先找到客户

端特性配置描述符(clieat characterislic configuration deecriptor)。这需要客户端在获得特性

值之后查找同一特性组的其他属性，以便找到所有特性相关的描述符。一旦发现客户端特性

配置描述符，即可向其写人正确的值，从而启用通知或指示。

13.9.5规范安全

最后，如果客户端打算断开连接以便未来快速重新连接，或者特性需要加密或认证键

路，那么客户端和服务器之间必须进行绑定。绑定通常是由规范中的客户端来发起的。 客户端可以尝试读取或写人服务中的任一特性值。只要其拥有读取或写入的正确权限，

服务器将会响应原有的特性值或响应写入的值；如果权限不正确，服务器会回应一个适当的

错误代码。

如果错误码表明客户端的安全性不足，客户端可能会尝试与服务器进行配对或绑定，以

获得正确的安全级别。如果只需进行一次信息的读取，配对通常就可以。如果需要一遍一遍

重连，则还需进行绑定。一旦设备获得了适当的安全性，连接将被加密，客户端可以重试先

前失败的请求。手是．。是的最前沿。

第14章

外围设备

——雅各布·布鲁诺斯基（Jacob Hronowski

率章介绍外围设备的设计，将从外围设备的视角综合来看前述各章的内容。

14.1背景

外围设备是低功耗蓝牙生态系统的命脉。中央设备通常会是电话、电视或电脑，而外围

设备则是为超低功耗实施了高度优化的定制产品。

外围设备大多围绕它们的电池、传感器、输入和输出装置来设计，之后才考虑无线技

术。外围设备的工作离不开与中央设备进行交互。交互的方式不外乎三种：广播数据；被中

央设备发现并与之建立连接：断开连接，然后在需要时建立连接。

14.2仅广播

有些设备具有一定量的数据，希望与许多其他设备共享。使用广播模式是一种最好的实

现方式，也是功耗最低的方式。举个例子，一个设备知道当前的时间，并愿意把它分享给每

一个位于该地区的其他设备，它可以广播时间数据，以便其他设备接收当前的时间。

一个只进行广播的外围设备可以非常省电。它不需要被连接或被发现——无需接受任何

来自中央设备的连接，也不用维持庞大的属性数据库以便被发现——它只是不断广播有用的

数据而已。

广播的间隔或频率是该类设备必须考虑的一个重要因素。假设当前的广播间隔为

iOOms．即每秒10改，但目前的时间只要求精确到秒，在一秒内反复广播浪费能量显然就

很不值当。因此，一些数据的广播间隔可以基于用户的等待意愿，即他们愿意为获得数据而等待多久的时间。

之所以说当前时间更适合用较大的间隔来广播，还有一个原因在于信息的观察者只会偶

尔侦听此类信息，比如每天一次或每周一次。因此，每iOs -次的广播间隔或许更加合理。

有趣的是，一个每天和广播者同步的挂钟只需要精确到当天之中最接近的那一秒就可

以。如果该挂钟每月走快半分钟，那么只要每天与精确的时间广播者同步一次，就能够消除

偏差。

14.3可发现

除了仅做广播的外围设备，所有其他的设备都是可发现的。可发现意味着外围设备的广

播目标是在此区域的正在扫描的所有中央设备。

可发现有两种类型：有限可发现性和一般可发现性。在与用户交互后，外围设备只能在

一小段时间内处于有限可发现状态，其他所有时候将会处在一般可发现状态。

举个例子，用户剐刚在外围设备中装上电池，此时将处于有限可发现状态。这将使新的

移动设备显示在中央设备（或许由同一用户持有）的用户界面的列表顶端。

再举个倒子，在按下设备上的“连接”按钮后，外围设备也会处于有限可发现状态。同

样的道理，用户希望该外围设备会在他按下连接按钮之后出现在设备列表的顶端。

外围设备并不总是处于可发现状态。比如一个外围设备与中央设备配完对，也许就再也

不能被发现了，或者只有苒次按下“连接”按钮时才能发现它。这种设计很常见，因为它带

来了两个好处：

首先，假如外围设备不需要被发现，就不用在其广播报文里放入与可发现性有美的信

息，比方说设备名称或当前的发射功率等。

其次，如果中央设备正在寻找和发现其他设备，通常只会显示用户感兴趣的外围设备，

而非当前区域的所有外围设备。由于外围设备并非始终能被发现，中央设备上显示的设备列

表就更易于管理，也更加有用。

14.4可连接

可发现和可连接彼此非常相似，它们者嫩用广播报文，但服务于不同的目的。可发现的

设备通常不与任何其他设备配对，而可连接设备通常与一个或多个设备绑定，并且只接受来

自这些设备的连接。

当设备处于可连接状态，它将接受来自发起设备的连接请求。从本质上讲，一个可连接

的外围设备能接受来自任何设备的莲接请求报文。

从功耗的角度来看，这种全盘接收的做法不无问题。任何中央设备可以与外围设备连

接，使得后者将花费大量时间进行协商，导致能量的浪费；更为重要的是，这还会阻止外围

设备绑定过的中央设备与其建立连接。 为了解决这个问题，外围设备上的控制器可以配置为其接受有限的若干中央设备的连接

请求。中央设备列表存储在白名单中，如果发起连接的设备不在其中，该请求将被忽略。这

样一来，外围设备只需要接收绑定过的、位于白名单中的设备请求，从而避免了能量的浪费。

接收已绑定的、位于白名单中的设备请求连接，先决条件在于设备自身必须执行过绑

定。在此之前，该设备将处于混杂模式，接受任何设备的连接。这通常在设备首次设为可连

接但是还未与任何设备绑定的时候才会发生。

14.5公开服务

进A连接，外围设备通常会公开一个或多个服务。每个服务都封装了外围设备中某个

组成部分的原子行为。服务通过服务器的属性协议公开，服务器上属性的集合统称为属性

数据库。

属性由通用属性规范(OAP)来组织，其结构如下：

属性数据库中的每个服务都始于一个服务声明，其中定义了服务的类型，紧随其后的是

隶属于该服务的各个属性。接下来又是下一个服务的服务声明。

在每个服务中可以包含一个或多个其他服务，在提供更复杂的行为的同时，避免了扩展

服务的核心功能。这就鼓励开发者去定义那些只能实现一些小块功能的服务。

打个比方，电油服务只能公开电池电量。但相机这样的设备可能有两个电池．一个用于

闪光灯，另一个用于相机机身。这些组件都可以有自己的服务，比如闪光灯服务和相机服

务。因此，相机服务既可以包括机身的电池服务，也可以包括闪光灯服务。而闪光灯服务将

包古一个单独的电池服务。

此外，在每个服务中可以有一个或多个特性。从根本上来说，特性是可以访问的一个

值。每个特性都有特性声明，用来说明数据到底是什么或者怎样可以访问它；特性也可以包

括额外的信息，描述它是如何被格式化或是如何配置的。

上述服务声明、服务包含、特性声明、特性值和特性描述符等都是属性数据库内的单个

属性。因此，外围设备只需要根据它想公开的行为决定需要公开的属性，然后等待客户端来

与这些属性交互。

14.6特性

特性是服务的基本建筑模块。一个特性即为—个客户端能够交互的值。特性值的格式由

特性的类型决定，并最终由一个XML特性规格书文件决定。

特性的行为不在该XMI。文件中定义，而是在服务规范中定义。因此．在一个外围设备

里可以有多个类型相同但行为各异的特性。

例如，在立即报警服务和链路损耗服务里都用到了报警级别特性，但行为却不尽相同。

在立即报警服务中，只要写人警报级别特性，外围设备将立即基于写入的值发出报警。不222．幕四部分应 甩

过，该特性无法读取，因为该值会立即被消耗掉，这就是所谓的控制点特性。

在链路损耗服务中，报警级别特性既可读也可写；如果中央设备与外围设备断开，后者

将基于断开时的特性值引发报警。在该服务里的特性值具有状态，并可以读取和写人。写入

的值被存储起来，在稍后的时间点可以读取。

14.7安全事项

外围设备公开的数据可能是一些机密信息或隐私信息，只能将其发送到那些通过了认证

的设备。为了确保这一点，属性服务器一旦认为请求达不到当前的安全级别，将拒绝任何有

关属性信息的请求。

举个例子．如果客户端请求读取电话中的未读邮件的数量，或者想要写入电话的振铃电

平，这些请求只有来自能够确保机密性的认证设备才可能被接受。这就要求外围设备先对请

求的中央设备进行认证，再与之绑定。一般来说，认证和绑定是由中央设备首先发起的。设

备一经验证，连接将被加密，客户端的请求可以被重新发送。

这种安全模式对于外围设备而言再简单不过了。外围设备的属性服务器每收到一个请

求，首先检查接受请求所需的安全权限。如果安全权限不足，服务器会发送一个错误响应告

知错误的内容，提示对方如何在重新发邀请求之前解决这个问题。相反，如果安全权限允

许．服务器将根据请求执行操作并发送一个响应。在任何时候，外围设备都不必主动地发起

任何安全的要求。

服务或特性所要求的安全性是由具体实现的服务规范来定义的。有些服务由于只是公开

公共信息，比如当前的时间，它们对安全可能没有要求。但有些服务则需要非常强的安全

性，尤其是那些公开私人的信息的服务。

14.8为低功耗优化

对于那些使用微型电池供电的外围设备来说，为了延长其使用时间，必须考虑为低功耗

操作进行优化。这包括确定最佳的广播间隔和连接间隔、优化属性的访问、选择是否保持连

接还是断开重连等。

为了便于理解低功耗是如何实现的，让我们考虑外围设备使用的一些典型状态(参见图

14-1)。首先映人眼帘的是关闭状态，显然，设备在关闭时不会消耗能量，或者只会消耗很

少的能量。当设备首次启动，外围设备将进入可发现广播状态，在该状态下能够被一十或多

个中央设备发现。

在某个时刻，一个中央设备将会连接到外围设备并与之绑定。绑定成功之后，外围设备

进入可连接广播状态。如果绑定没有发生，外围设备将回到可发现广播态，等待另一个中央

设备的连接和绑定。图14-1外围设备的典型状态

已绑定的外围设备将执行可连接广播，于是

外围设备。它们一旦重新连接就进入了连接状态

的间隔。

只有与之配对的中央设备才能连接到该

中央设备会根据自身的意图来管理连接

处在连接中的外围设备还需要明确一点：是该保持连接？还是该现在断开？其结果关乎

于中央设备多久段有执行过请求、外围设备当前在不在发送数据以及未来一段时间内有没有

数据要发送。

假如选择了断开连接，外围设备有可能进入下列两种状态：一是回到可连接广播，即周

期性地发送广播报文以便中央设备根据其需要发起重连；二是关闭广播，直到外围设备有数

据要发送时再使用定向广播与中央设备重新建立连接。

最后，别忘丁外围设备上还可能有个“连接”按钮，一旦按下，设备将删除当前的绑定

信息，重新变为可以被发现的设备。此时，任何与之相连的设备都将被断开，外围设备将返

回可发现广播状态。14.8.1可发现广播

在外围设备中，一个优化功耗的根本办法在于为广播和连接精心选择时间的间隔。一个

适合的间隔可能让电池工作好几年，而一个糟糕的间隔只能让它工作几个星期，这简直就是

天壤之别。不过，要做出正确的选择并不容易，当中有很多地方必须进行权衡。

一开始，外围设备通常会执行可发现广播，以便于中央设备发现自己。由于用户总是想

要掏出设备井希望其尽快完成连接，该操作耗费的时间一般很短，只占设备的整个工作寿命

的一小部分。一旦连接成功，该设备只会执行可连接广播。

位于可发现模式时，外围设备希望尽快被发现，以便为用户提供最佳的用户体验。设备

同时也会广播大量的附加信息，比如：发射功率——令中央设备可以根据远近对设备进行排

序；设备名——供中央设备构建友好的用户界面；此外还有设备支持的服务集——便于中央

设备根据需要对扫描到的设备进行过滤等。

综上，一个可发现的外围设备更倾向于使用较快的频率发送广播。将广播间隔设为

250ms有助于在发现速度和节能之间取得较好的折中。

14.8.2绑定

每当外围设备建立了连接，连接的时间间隔将由中央设备决定。中央设备应该设置合理

的连接间隔值，不过这一点无法保证。

处于可发现广播状态昀外围设备能够以相当快的连接间隔建立连接。快速的连接间

隔——比如介于7.5ms - 25ms之间的一个值——将会耗掉大量的能量，但它也意味着中央

设备可以很快发现外围设备提供的服务集和特性集，从而向用户提供及时的反馈，告知其如

何与外围设备进行交互。

如果连接间隔很慢（介于is - 4s之间），中央设备将耗费一段相当长的时间来确定如何

使用外围设备，而用户可能会认为设备不能正常工作。

一旦绑定完成，并且中央设备已完成对外围设备的分析，此时应该尽量减少连接的间隔

以节省能量。关于如何选择时间间隔的问题我们稍后舍讨论。中央设备可以开始使用该外围

设备公开的服务，并在某个时间点断开连接。

已绑定的外围设备一旦断开连接，将会进入可连接广播状态或者定向广播状态。下面的

章节将详细说明这两种情况。

14.8.3可连接广播

断开连接后的外围设备可以周期性地进行广播，以便让中央设备重新发起连接。在这种

情况下，广播需要综合考虑多种因素，既要保证中央设备的重连速度，又要考虑外围设备自

身的功耗。

有那么一些外围设备，它们不在连接中的时间要比处在连接的时间多得多。以心率带为

例，该装置只有在用户慢跑的一个小时内会保持连接。假设用户每个星期锻炼兰次，于是，

在一周之内心率带只连接了三个小时，而在其他的165个小时内则处于可连接广播态。对这类外围设备而言，使用较长的广播间隔带来的好处十分明显。

如果把可连接广播间隔设为is，那么中央设备就能在几秒钟内完成连接。这对于普通

用户来说是一个完全可以接受的时间。然而，假如外围设备还想让中央设备更快地连接，就

必须把可连接广播间隔设为0.5s或更少。

还应当指出，某些设备并不需要持续进行广插。让我们回到心率带的例子，假如用户把

它从身上取了下来，它其实并不需要执行可连接广播。可见，当用户没有佩戴设备时，完全

可以禁用设备的广播功能，从而极大地提高节髓水平。

相反，在用户戴上心率带时，可以针对广播分配更多的功率，比如将广播间隔设为

iOOms。这会给用户一种印象：一戴上心率带连接就能在瞬间完成，用户体验堪称卓越！显

然，如果中央设备此时暂不能向心率带发起连接，比如用户已经藏上心事带，但中央设备并

不在其通信范围之内，一种慎重的做法是降低心率带的广播速率，换一个更大的广播间隔来

节能。

14.8.4定向广播

一些情况发生时，外围设备可能会选择与中央设备直接连接。对外围设备而言，自事件

发生到通知中央设备之间的时间必须尽可能短。此时，最佳选择是使用定向广播。

定向广播对能量有较大损耗，此时的外围设备将以非常快的频章向某个中央设备发送大

量的广播报文。如果该中央设备正准备向外围设备发起连接，连接会立即建立，外围设备能

罄迅速地发送所需数据。

定向广播是外围设备连接中央设备的最快方式。包括连接建立和数据发送在内，最快能

在Jms内完成。

定向广播不允许设置广播间隔，这是因为广播报文必须每隔3.75ma在三个广播信道发

送一遍。挽句话说，每1.25ms外围设备就会发送一个定向广播报文，即每秒发送800个数

据包。

对于那些需要快速发送数据、但很少处于可连接状态的设备，定向广播将是最佳的模式。

14.8.5己连接

在连接中，中央设备对于外围设备所使用的时间间隔和延迟具有完全的控制权。外围

设备则有向中央设备提出要求的权利，可通过逻辑链路控制和适配协议( L2CAP)的信令信

道．向中央设备建议更适合的连接参数。

倒如，如果中央设备正在监视外围设备的距离，需要的连接间隔大约为每秒三次。假如

实际使用的连接间隔比这短，外围设备将被迫耗费更多的能量与中央设备同步，不过这对用

户来说毫无裨益；假如实际的连接间隔比这长，外围设备与中央设备同步变慢，以至于无{击

发现设备的移动并及时向用户报警。

涉及功耗的连接参数共有两个：连接间隔( connectioninterval)祁从设备延迟(slave

latency)。连接间隔是一个时间值，用来确定中央设备与外围设备之间的传输和同步频率，226．第四部舟应 月

它必须为1.25ms的整倍数。

连接间隔并不是最重要的因紊，从设备延迟对于外围设备的功耗而言更为关键。从设备

延迟是指从设备可以忽略的主设备连接间隔的数量，它是一个。- 500之间的整数。

举个例子，倡设连接间隔为12.5ms．从设备延迟是。，那么从设备必须每12.5ms侦听

一次主设备。这无疑会消耗大量的能量。如果保持12.5ms的连接间隔不变，但从设备延迟

变为1，于是从设备可以忽略一个连接间隔，但必须确保每隔一个间隔执行侦听。这样一来

其功耗将减半，但是仍然保留了以12.5ms间隔发送数据的能力。

从设备延迟不单单是从设备的一种节能手段，也决定主设备向从设备发送数据的时延。

打个比方，你有一个低功耗蓝牙键盘，上面的一个大写锁定指示灯需要在0.5s内完成开启

或关闭。如果连接间隔为12.5ms，则从设备延迟必须为39-从设备会跳过主设备发来的

40个包中的39个，即每500ma侦听一次。较之将从设备延迟设为。或l，该值将会明显提

升从设备的节能效率。当然，500ms不过是一个大概的时延，它假设信道是理想的，因此没

有考虑重传的因素。由于实际使用中的重传很难避免，时延会不时地变大。

应当指山，增大从设备延迟并不能无限地提高节能收益。如果数据流只从外围设备流

向中央设备，自然可以使用一个非常大的从设备延迟，但是选择更大的数所产生的收益将

会递减。

一个例子是，有一个将数据发送到计算机的鼠标，连接间隔设为i5ms，从设备延迟设

为500，于是鼠标只需要每7.5s侦听一次。然而，考虑这样的一个事实，即中央和外围设备

的时钟都会有一定的不精确性，最坏的情况下每台设备上的误差高达万分之五。换言之，每

过一秒，无论是中央设备或外围设备的时钟都可能会偏移半毫秒。

这意味着，由于在7.5s的时间内鼠标一直没有同步，它的时间概念可能与准确值相差

了3.75ms，主设备也一样，并有可能往相反的方向偏离。因此，总的不确定性将是7.5ms。

外设在算出主设备的传辅时刻后，必须提前7.5ms开始侦听，并持续到既定时刻后的

7.5ms，以便覆盖所有可能的偏离区间。这就是所谓的“窗口扩展（window widening）”。窗

口扩展将耗费额外的能量，比如在上述例子里，每7is无线电要持续打开7.5ms（期望值）。

如果从设备延迟仅为50，即鼠标每过750ms将侦听一次，此时的窗口扩展意味着从设

备必须提早0.75ms侦听。不幽所料，窗口扩展与侦听间隔的比率与上面的倒子完全相同。

外围设备也必须收发一些空报文，即当主设备发送一个80vs的报文时，从设备将回复

一个同样长度的报文。随着从设备延迟的增加，用于接收和发送数据包的时间变得微不足

道，从设备通过减低同步频率所节省的能量损失在了窗口扩展中。

其结果是，把从设备延迟设为小于300ms或大于is都行不通。低于这个范围，反复同

步消耗的能量会高于多等待一段时间消耗的能量；高于这个范围，窗口拓展的能量将极大地

影响节能效果，此时，更好的选择是能够提升用户体验的较低的延迟值。

14.8.6保持连接还是断开

外围设备进^莲接之后，实际用倒有可能要求数据以随机的间隔执行传输。具有这类用途的外围设备既可以保持连接，叉可以断开连接，随后等待有数据时再重新连接。

这里需要判断两个因素。首先．如果外围设备开始广播，中央设备能否在一个合理时间

内重连，让用户不至于察觉连接已经断开；其次，如果保持连接，应该使用多大的连接延

迟，电池寿命允许的连接延迟范围又是多少。

为了使外围设备做出更明智的决第，外围设备可以公开“扫描参数”服务，以便中央设

备向其告知扫描使用的参数。在中央设备连接到外围设备时，它会发现上述服务并且写入一

个延迟值，等到重新建立连接时再兑现该延迟。利用这一信息，再结合用户体验的要求，外

围设备可以确定是否该断开连接，以厦该何时断开。

举例说来，中央设备向外围设备的“扫描参数”服务写入iOOms的连接延迟值，如果

外设能够接收的延迟上限为250ms，那么只要没有数据发送，外围设备理论上可以随时断开

连接，因为它知道仅需iOOms对方就能继续发送数据。不过，虽然断开连接能够节能，等

到重建连接时还是会用掉相当的能量。假设最大延迟为iOOms，每次重连平均将花费50ms。

在此期间，外围设备每l.25ms发送一个176斗s的数据包，因此总的供能时间为8.64ms。

虽然听上去并不很长，但如果朴围设备的连接间隔为250ms，并且不发送任何数据，在

每一秒内无线电打开的时间其实只有短短640ps。这样一来，外围设备宁可与中央设备保

持13.5s的连接也不愿提前断开，否则耗费的能量将会更多。

然而不幸的是，实际情况远不止这么简单。外围设备如果要变为可连接，即使在它断

开时也需要进行慢速广播。假设外围设备的广播间隔为is，它需要每秒发送3个广播报文，

总共为504Vs。因此，外围设备既可以选择保持连接，每秒钟工作640p s，也可以断开连

接执行周期广播，然后在需要时迅速重新连接。

可见，即使断开连接，外围设备每秒也只节约了136yS的工作时间。这意味着外围设

备实际上至少应当保持一分钟的连接，在此之后断开连接才有意义。更糟的是：如果外围设

备希望中央设备能连得更快些，自身也必须广播得更快。而500ms的广播间隔本质上就会

比保持连接所稍耗的能量更多。因此，对于一些设备而言，始终保持连接可能是一种更好的

选择。

这即是说，一些外围设备没有必要实现“扫描参数”这一服务。外围设备一旦连接，就

保持在连接状态。不过这些都要取决于具体的用倒，或者外围设备的使用方法。

当然，如果一个外围设备公开了扫描参数服务，但中央设各并未向该服务特性写入有用

的值，那么外围设备必须凭猜测来确定何时应该保持连接，何时应该断开。这种猜测通常导

致一种简单的做法，即在上一次应用数据发送之后将连接保持一段时间，等到定时器超时随

即断开连接。这里定时器的典型值可能从几分钟到几小时不等。

14 9优化属性

总而言之，外围设备的典型应用是为自身的数据提供一个人口．以便其他设备访问。

这就好像心脏监护仪提供了佩戴者的心跳速率一样。最后一项针对外围设备的优化是数据传送。

通常，外围设备公开一个或多个服务，中央设备发现这些服务及其特性和描述符。然

后，中央设备读取、写入或配置这些特性，从而使用由外围设备提供的服务。但外围设备也

可以通过实现与特性有关的通知和指示，帮助自己节省能量。

特性和服务是通过属性来描述的，并且通过属性协议来访问。该协议使客户端（位于

中央设备）能够访问外围设备的属性，不仅支持属性的读取和写入，而且支持属性的通知

和指示。

以心率监测仪为例，假设该设备具有一个特征，将测量的心跳速率值每秒向中央设备

（如手表或手机）发送一次。该数据传输可以通过每秒钟读取一次特性值来实现。但是这需

要中央设备发送读取请求，同时外围设备发送读取响应。外围设备的无线电每秒有272“s

保持打开，用于接收请求和发送响应。

如果为该特性设置“通知”，那么无线电的打开时间将减少至232ys。此时的能源效率

要优于轮询的情况。当数据只能在随机时间产生，而不是周期性产生时，这种方式的优势更

加明显。在这种情况下，客户端的很多轮询往往是不必要的。

考虑一个设备，每秒监视一次传感器，但传感器的值极少变化，也许每30s才改变一

次。若采用轮询机制，每分钟内的读取请求和读取响应需要发送60次，怛该值在此期间可

能只改变一次。发送60个请求和60个响应需要16.32ms。但是，如果外围设备支持通知并

且中央设备配置了通知，无线电将其需要工作9.744ms。与轮询的方案相比，通知几乎降低

了一半的功耗。

指示和通知类似，但它在收到消息后，协议层会发送一个响应消息以确认该信息已被接

收。这比上述的通知更加耗能，大约为iOms略多的能量，但它也有好处：不会有数据丢失。

因为通知不在协议层进行确认，如果接收缓冲区已满有可能会被丢弃。指示的限制在于每次

只能发送一次，在接收到确认消息之前无法再次发送。

较之轮询方案，通知和指示的效率要高得多。通过为尽可能多的特性提供通知或指示，

外围设备可以帮助自己进一步节能，这也为中央设备提供了更多可配置通知或指示的特性。 第15章

测试和质量鉴定

想法必须要经受拴验，这是我们做事的初衰，否则它们仅仅停留在想法而己。想击和实

现之同往往有一道鸿沟。我有许多自认为不错的想法，但实际上都行不通。

——安迪·戈营沃西(Andy Goldsworthy：

在低功耗蓝牙产品发布之前，必须对其实现的服务和规范进行测试，才能够获得质

量鉴定。蓝牙技术联盟(Bluetooth SIG)的会员协议要求所有签署的成员必须遵守上述规

定。根据该协议，只要是符合公开规格书的产品，就可以运用或借助任何其他成员的知识

产权声明。这意味着，无论一款产品实现了几个蓝牙规范，只要其开发者签署了承诺声明

（Declaratdion of Compliance．DoC），都能获得免授权许可（royelty-free license）从而免费使

用该技术。承诺声明的巨大益处彰显无疑。不过，所有这一切的前提是设备必须通过测试和

质量鉴定。

如图15-1所示，质量鉴定程序涵盖了从最初的概念设计到最终的合格上市产品的多个

步骤。其中的部分鉴定过程需要进行测试；有一些步骤需要收费，还有些步骤要求对产品的

功能做出评判。

15.1启动项目

第一个步骤最为简单——启动一个项目。这需要你登录蓝牙SIG同站e bluetooth.org并

且开始一个新的测试计划生成器(Test Plan Generator．TPG)项目。你需要为新项目取一个

名字，并设置期望的鉴定日期。不要过于担心这些信息，在后面的步骤里你可以更改名称和

日期。

启动项目的好处在于——它是免费的，你想申请多少个都可以。如果想了解测试需要做

一些什么工作，你可以启动一个项目并进行配置，然后瞧瞧都有哪些测试的任务。您也可以

。耍创建一十新的项目，请访问http：Ⅳwww.blu etaom org/tp g/create\_proj e.t c fm。在任何时候删除一个项目，比如某个项目看起来不再可行，你可以把它从当前的括动项目列

表中予以删除。

图15-1蓝牙的测试和鉴定过程

在初始步骤中创建的项目将一直保持不变，直到最终完成质量鉴定。这意味着，尽管

可以免费创建项目，但是如果想让某个项目成为官方列表的合格产品，还必须支付一定的

费用。

要启动项目，进入bluetooth.org网站中的创建新项目(create new project)页面。你需

要准备的信息包括：预期的鉴定日、TPG发行版本、项目名称、产品类型以及硬件和软件

版本的描述。

对于低功耗蓝牙产品而言．TPG发行版车应该为4 0或更高。这样一来项目才能够使用

系统里所有的低功耗协议层，它们只有在4 0版的核心规格书中才被引入。

下一个要选择的是产品类型。有八种类型可供选择，取决于正在鉴定的组件或者子系统

的类型：

口最终产品这是一个可供大众消费的实际的产品。

口主机子蕞统产品该产品仅包台主机部分的核心规格书，如逻辑链路控制和适配协议\*15幸删诚和质量3电+23

( L2CAP)、属性协议（ATT）、通用属性规范(GATT)、安全管理(SM)和通用访

问规范( GAP)。这类产品需要与控制器子系统和一个或多个规范子系统相结合，才

能成为最终产品。

口控制器子系统产品该产品仅含有核心规格书的控制器部分，如无线电和链路层。这

类产品需要与主机子系统和一个或多个规范子系统相结合，才能成为最终产品。

口规范子系统产品这是一个实现了一个或多个规范和／或服务的子系统。该类产品需

要与主机子系统、控制器子系统以及可能的其他规范子系统产品相组合，才能成为最

终产品。

口组件子系统产品这是子系统的单独部分。例如，你的SM的实现可能会被单独列为

一个组件子系统产品。

口开发工具这是在蓝牙产品的开发过程中使用的一种特殊产品类型。这种类型的产品

不能出售给终端消费者，但是可以销售给其他成员，以协助其快速制造最终产品。

口喇试设备这是一种特殊类型的产品，专为需要获得免版税许可的少数测试设备制造

商而设计。由于测试设备可能需要做一些规格书之外的东西，才能真正用于产品测

试。为此，这些工具单独设立了一个特殊的产品类型。

关于产品类型，一件有趣的事情是如何把不同的类型结合起来。宽涯地来讲，各个部件

实际上可以组合在一起构成一个完整的最终产品，不过这种整体方案会极大地增加测试的复

杂性，带来很多问题。相反，如果对大多数组件分别进行测试，并列为独立的组件子系统，

那么将其组合成最终产品时就不需要任何额外的测试。

组件也可以组成控制器或主机子系统。这样一来，产品的所有设计部件可以构成一个列

表。比如，使用一个控制器芯片时，将它列为控制器子系统，以便与主机子系统相结合从而

获得最终产品。

15.2选择功能

鉴定过程的下一步是为你的产品选择支持的功能，它们会出现在汇总模板中。在这里，

你可以选择一套设备打算支持的核心层次、规范和服务。举个例子，一个低功耗的靠近装置

会在核心层次中选择RF物理层、链路层、L2CAP、ATT、SM和GATT，并且在主机部分

选择链路损耗服务、即时报警服务、发射功率服务等。

在此之后，你需要深入研究上述选择做出了哪些声明，看看打算实现的功能集是否与稍

后的台规声明相吻合。每一个声明的功能都需要运行一纽测试，在未收集相关测试证据之

前，你将无法签署承诺声明(DoC)。

在汇总模板里选择的每个协议层都附有详细的实现一致性声明（Implementation Confor-

mance Statement，ICS）。每个ICS包含多行记录，每一行显示了一个强制或可选功能，从

中你能判断出你的产品是否支持该功能。每个规范声明、服务声明以及核心规格书的协议层

声明均对应了一个单独的ICS。15.3 -致性检查

一旦填写完所有的ICS．系统会对选择的有效性进行检查。有许多有关依赖性的规则，

比方说某一个功能被选中，那么为了支持该功能还必须选择若干其他的功能。对于无效的选

项，对应的行将被标记为无效。当纠正错误之后可虬重新执行一致性检查。举个例子，某个

服务的某一特性有一个通知选项，如果你选择这一行，一致性检查将确保GArT有能力写

入特性描述符。这是因为假如没有这种能力，将不可能配置该特性以启用通知。

有时，选出的功能集会前后矛盾。如果它们不一致，需要对选出的功能集进行调整，并

且再次尝试一致性检查。只有所有的一致性检查顺利完成才可以移动到下一个步骤。

值得注意的是，用于新规格书的测试计划生成器( TPG)偶尔也会出错。如果你相

信自己是正确的，而TPG出了问题，可以向蓝牙认证管理员（Bluetooth Qualifcation

Administruor．BQA）提出，他们会安排解决TPG的有关问题。

15.4生成测试计划

无论是核心规格书的每个层次、服务或是配置文件都有一个相关的测试规范。在这些测

试规范里的每个测试都有一个唯一的标识符。许多的测试不必用在你的产品上，这取决于你

在上一步中确定的ICS酿置。因此，接下来的步骤是为你的产品生成测试计划，它涉及你

之前完成的ICS。测试计划包括所有需要运行的测试，用来证明你的产品已经实现了声称支

持的功能。

生成的测试计划可以从Web站点下载到本地计算机，这为产品的开发过程提供了一张

测试清单。产品一旦被研制出来，并且有了可用于测试的原型，在进入最终的质量鉴定之前

还有最后一步—创建—个合规文件夹。

合规文件夹用于存放所有关于产品的信息，包括产品的功能集、性能以及所有的测试证

据。合规文件夹可以是电子文档或纸质文件的集合。该文件夹必须安全地存储，同时对于每

个鉴定产品来说都便于访问。如果稍后在你的设备上发现问题，蓝牙技术联盟可吼审核该产

品的舍规性。因此，必须授予官方访问台规文件夹的权限以便其执行审计。

合规文件夹包含以下内容：

口产品信息包括有关产品的信息，如产品名称、产品描述、硬件和软件版本信息。它

还包括相关的用户手册或用户指南、框图和产品的技术说明、有关蓝牙组件是如何集

成到产品的设计说明等。

口测试信息涵盖所有的设备测试信息，包括生成的测试计划、含有每个测试结果的测

试声明、正式验证测试要求的测试报告以及任何与测试相关的额外信息，使得测试可第15章测试和质量婪定’233

在今后重复执行。

口设计信息描述了相关的蓝牙功能产品的详细设计信息。设计信息可以包括元素，如

电路原理图、印刷电路板布局、元件的布局以及用于描述物理外部组件及其容差的材

料清单；也可以包括一些其他东西，诸如软件设计文档，或者从软件版本控制系统中

精确提取软件所需的充分的信息等。

口实现一致性声明(ICS)这包括系统中的每一层、每个配置文件或服务对应的ICS文件。

在产品的出售或分销阶殷，合规文件夹必须安全地保存。只有当该产品退出市场超过一

年后才可以删除合规文件夹。

15.6鉴定测试

一旦原型具备了进行测试的条件，测试计划已经生成，并且台规文件夹也已经创建（用

于存储所有关于产品及其测试信息），那么就可以开始测试。

一些测试需要通过验证铡试仪才能完成。有三种类型的验证测试仪：射频测试仪用于

运行物理屡的鉴定测试；协议测试仪用于运行协议层测试；规范调整套件(Profile Tuning

Suite．PTS)用于运行规范测试。

每一项测试都属于下列四种类别之一：

OA粪A类测试是具有最佳定义的测试用例，必须由蓝牙鉴定测试机构（Bluetooth

Qualified Test Facility．BQTF】或蓝牙认可测试机构（Bluetooth Recognized Test

Faeility，BRTF）来完成。运行的铡试将生成测试报告，存储在合规文件夹中。

口B类B类测试由产品的制造商根据测试的要求执行。测试的设置、证据和结果都应

存储在合规文件央中。

口C羹c类测试也是由产品的制造商来执行，但对文档的要求显著降低。c类测试只

需声明它们已经完成了测试并且获得了测试结果，而不必提供试验正据。

OD粪D类测试是可选的。没有运行这些测试的硬性要求，也没有必要记录它们的执

行情况和测试结果，或者存储任何的试验证据。

D类测试可能显得没什么意义，然而随着测试的成熟和新问题的发现，所有的测试可以

在不同类别之间移动。例如，一个新的规格书可能会要求进行所有的c类测试，但随着测

试成熟度的增加，它们可能变为B类测试。同样的道理，B类测试也许由测试设备制造商负

责进行，当时机成熟时可能将它们移至A类。

如果某项测试出现了问题并且无法立即解决，可以将其移至D类。按理来讲SIG不会

再要求运行它们，然而应该记住，质量鉴定是证明你的产品符合相应的规格书，而测试用例

仅仅是检查合规性的有效途径之一。所以，即使某个测试用例有同题，产品奶然应当正确地

实现该项测试所要求的功能。

在所有的鉴定测试过程中，如果测试用例或期望的输出有误，可以采用测试用倒勘误

表。一项测试是否有效取决于核心规格书而不是测试规范。因此，有可能在执行A类测试234．幕四部分应 用

后，发现产品没有实现核心规格书的要求，

鉴定。这种情况并不时常发生，无论如何，

15.7鉴定设计

但在提供了所需的测试证据后还是能够通过质量

质检过程可视为一个有效的安全屏障。

一旦测试完成，台规文件夹应该包含一大堆的测试报告。接下来的步骤是设计鉴定。为

此，你需要一个合格设计标识符（Qualified Dcsign Identifier．QD ID）——一个关于你的设

计的唯一号码。QD ID并不是指一个最终产品，就这一点下面还会详细介绍；相反，一个合

格的外观设计可以用在多个终端产品中。一个具体的例子是设计一种可以打上不同品牌的产

品，每种品牌的包装各异；鉴定合格的设计允许使用蓝牙无线技术的部件被用于许多最终产

品。鉴定一个设计需要一定的费用，但将其用于最终产品时不会收取壤外的费用。

一旦获得QD ID，接下来需要上传测试声明到蓝牙技术联盟的网站。这个测试的声明

包括了所有的来自验证测试人员的测试报告，以及任何额外的C类或D类测试报告。此外，

规范调整套件报告应单独上传。系统将直接解析上述文件，井立即核实设备支持的规范集和

服务集。

15.8合规声明

蓝牙技术联盟一旦拥有了所有的设计相关信息，包括设备劝能、已经运行的测试以及测

试结果，就可以对该项设计进行鉴定。设计产品的成员公司必须完成一份承诺声明( DoC)。

该声明是一份具有法律约束力的文件，其中，该成员宣称其设计符合所引用的蓝牙规格书，

该产品实现了参考设计，并且符合流程定义文档规定的测试要求和归档要求。整个过程是由

鉴定程序参考文档（Qualification Program Rcference Document．PRD）所定义的。从本质上

讲，只要签署了这个承诺声明，可以说该成员就完成了一切程序要求的操作。

因为DoC是一份法律文件，它只能由有权处理公司法律事务的人来签署。这就要求该

公司的管理人员或经过授权的人员通过签名做出法律承诺。

DoC必须包括QD ID、产品信息、产品的功能集、引用的蓝牙规范、所使用的PRD版

本和测试规格书版本以厦用于确定判断测试集及其类型的测试用例参考表。

成员公司只要签署了DoC文档，该产品将被免授权许可覆盖，在产品上可以使用蓝牙

品牌。

已检验合格的产品必须将其QD ID明确标注在设备上或文档中，以便让任何人判断产

品的功能集和设计要求。

15.9产品名录

一旦成员公司的产品获得了QD ID．接下来是登记产品名录。此名录并不需要设直即公#15章刹试和质量鉴定0 235

开，因为这可能对本产品的发行营销计划产生影响，但产品必须在三个月内进行登记一只有

在加入产品名录后，产品的合规性才能被视为有效，因而务必在产品上市销售前执行这一步。

15.10结合组件

可以把多个产品名目组合成一个产品。例如，某个上市的最终产品可能用了一个制造商

的控制器芯片，另一个公司的主机协议栈，再使用了来自两个不同供应商的服务。因此，该

产品将会有五个QD ID: -个用于控制器，一个用于主机，两个用于服务，还有一个用于所

有QD ID的组合。

还应当注意的是，每一种合格的设计本身也可以是多个合格部件的组合。在前面的例子

中，主机在设计时每一层都需要单独的QD ID：-个用于L2CAP．一个用于ATT．一个用

于GATT．一个用于GAP，一个用于SM，还有一个用于该组合部件作为主机子系统。因

此，这个看似简单的最终产品将最终参考十种不同QD iD。

QD ID树是一个非常强大的概念，它允许封装的组件或子系统分别进行设计和鉴定，然

后作为一个完整的设计进行交付。这些组件或子系统可以在许多产品中重复使用，无需支付

任何额外的测试费用或名录登记费。对于使用许多现有部件的设备，用于创建新产品的咸本

可以忽略不计。推荐阅读

柚M快董嵌入式曩麓量垒馈计：基于开曩疆件r-Ib州

作者：RObT洲Is。n等5BN。978 7-111埘019-0定价．w．coi

嵌入式软件开发蔫解

作者．COlin waib 15BN．978-7¨1-44952-2盎价．79001

黔～

-弓5

置…目

聊㈣ 罗宾·海登( Robin Heydon)是蓝牙技术联盟名人堂的成员，他把之前分散

于多个标准文档的精华部分汇集起来，并就其来龙去脉提供了专业的说明，这些都

是实现高性能运行系统不可或缺的重要信息。作者首先回顾了低功耗蓝牙的设计目

标，并解释了该目标是如何推动关键契构设计的，以及介绍了低功耗蓝牙的创新应

用模型。之后，他深入地阐述了低功耗蓝牙的两个主要部分——控制器和主机的工

作原理，随后探讨了安全、规范以及测试、鉴定等一系列关键问题。这些知识是制

造“Bluetooth Smart”和“Bluetoolh Smart Ready”设备必不可少的。

本书是学习低功耗蓝牙官方标准文档不可或缺的参考手册，适合每一位正在考

虑、规划低功耗蓝牙产品开发，或者将该技术纳入工作体系的技术专家或决策者。

Robin Heydon毕业于英国曼彻斯特大学计算机科学专业，是CSR公司负责

蓝牙技术标准化的标准架构师，也是制定低功耗蓝牙技术标准的超低功耗工作组的

联合主席。自2007年起推动低功耗蓝牙技术标准的制定工作，因其贡献卓著．2010

年他入选了蓝牙技术联盟名人堂。

O

黪

g%导电跏{iH

www怒= 皿