The speaking laptop usage model is an example of extending the functions of one device by allowing it to "borrow" the capabilities of some other device. Speakerphone capability becomes available, using the speaker and microphone of the notebook computer, even if the mobile phone does not have its own speakerphone capability. Extensions to the speaking laptop scenario might enable other similar usage models where existing audio input and output could be used to supplement that of a mobile telephone. For example, the audio portion of a call could also be routed to a car's audio system in a vehicle with Bluetooth wireless communications.

The Automatic Synchronizer

The automatic synchronizer is an example of using proximity networking to add value by making an existing task easier to do. Personal portable devices empower people to have quick and easy access to information they can use in their daily lives. For that information to be most useful it needs to be kept up to date, but this personal information management data might be distributed across the many devices that a person could use. For example, new calendar entries or to-do list items might be entered on any of a notebook computer, personal digital assistant or smart phone; or these entries might be entered on a desktop computer and stored on a server in a network. Synchronization is the process of merging the data from two different sources based upon some set of rules such that the resulting data sets are identical (or at least reflect identical information). A common example is synchronizing a personal digital assistant with a desktop or notebook computer. Today this often is performed using special serial cables and software that may be unique to the device. Standard protocols and object formats in the specification allow data on one device to be synchronized with data on any other device, whether they be PDAs, notebook computers, smart phones or even networked data accessed through a data access point.

The "automatic" part of this usage model is enabled by proximity networking. Today synchronization is almost always a conscious effort–it involves connecting a serial cable and pushing a button, or pointing two infrared-capable devices at each other and launching an application. With Bluetooth wireless communications it is possible for two devices to automatically synchronize whenever they come within range of each other. For example, a personal digital assistant carried in a person's pocket could automatically synchronize with that person's desktop computer whenever she walked into her office. Clearly it should be possible to configure the devices as to when and how to automatically synchronize, and to ensure that devices synchronize only with other known devices and data sources and not with just any random device; the specification does offer mechanisms that could be used to do this. Figure 3.8 shows examples of the automatic synchronizer. The figure illustrates how different devices in a personal area network (such as the mobile telephone and PDA shown) might automatically synchronize. The figure also shows how one of those devices (here, the PDA in the briefcase) might also synchronize with a desktop computer when those devices come within range of each other (in this case, when the PDA's owner walks into her office).

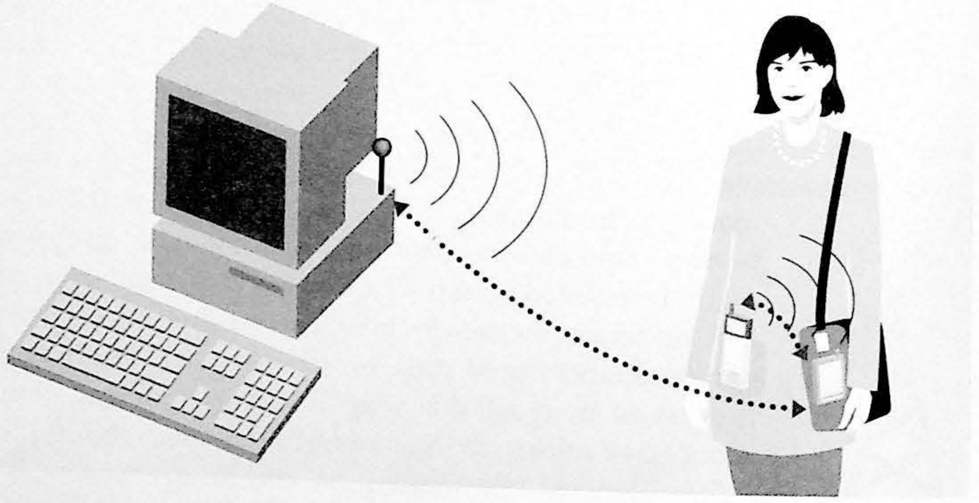

Figure 3.8
The Bluetooth automatic synchronizer usage model.

In addition to the convenience afforded by automatic synchronization, Bluetooth wireless communication removes the requirement for cables. By specifying a standard protocol and object formats for synchronization (these are adopted from IrDA, as detailed in Chapters 9 and 14), Bluetooth wireless communication makes it easier for any device to synchronize with any other device. This multidevice synchronization enables a person to use any convenient device to enter new appointments, to-do list items or other data.

The Instant Postcard

The instant postcard is another usage model that was discussed early in the development of the specification but is not formally part of the version 1.0 profiles. This is one of the few scenarios that involves a device other than a mobile phone or computing platform, although it is expected that over time many new usage models and profiles will be developed for additional device classes. The underlying concept is that of having a digital camera which can wirelessly transfer a photo image to some other device which could then e-mail the image to a recipient, thus creating a digital "postcard." Today many digital cameras use a serial cable to transfer photo images to a computer where they can be stored, catalogued, manipulated and distributed. As with the other version 1.0 usage models, Bluetooth wireless communication removes the need for a cable which in turn presents new ways to use a device, as pointed out below. Figure 3.9 shows how the instant postcard might be realized.

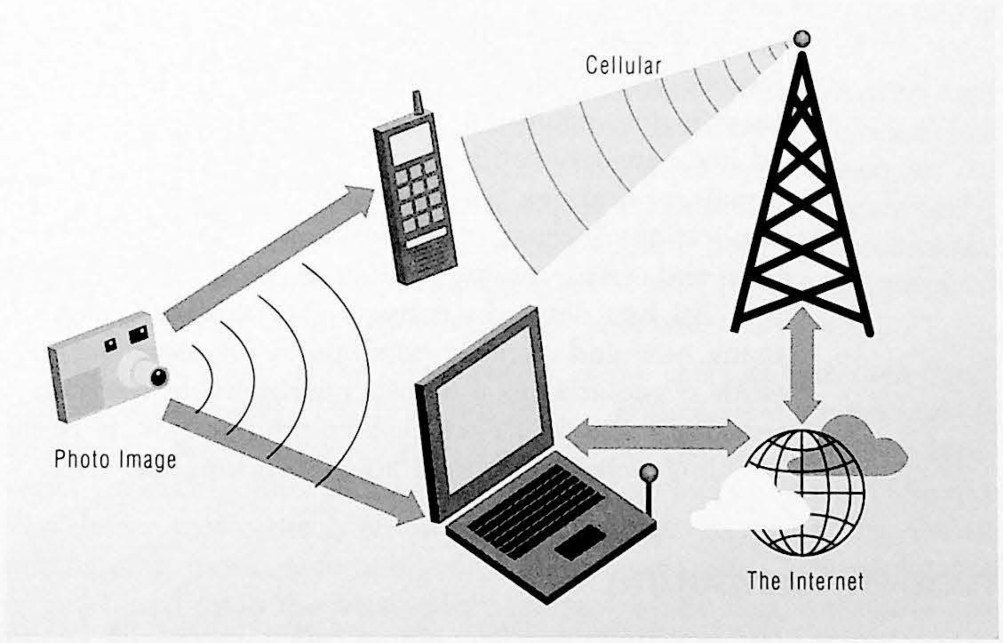

Figure 3.9
The instant postcard.

This scenario is useful not only for sending "postcard" type pictures to friends and relatives but also for commercial applications, such as real estate (transferring photos of newly listed homes to a central database), law enforcement (transferring photos of suspects or stolen vehicles, for example) and insurance (transferring photos of automobile damage resulting from accidents). In addition to replacing cables, Bluetooth technology also could change the way a digital camera is used, since photos need not necessarily be transferred to a computer. Many photos might be transferred directly from the camera to a mobile phone and then sent as an e-mail attachment without using a computer as an intermediary. In addition, the transfer of photos from the camera to a database or library could be accomplished in more of a real-time fashion, since no cabling is required.

Ad Hoc Networking

This usage model could be considered to be an extension of the interactive conference (file transfer) scenario. It is not specifically addressed in the version 1.0 specification but it does provide an illustration of usage cases that could be enabled in the future. *Ad hoc* networks are networks that form spontaneously; Bluetooth wireless communication is an enabling technology for these sorts of applications. The interactive conference usage model showed how objects such as electronic business cards or files could be exchanged in a conference room setting. When ad hoc networks can be formed among the meeting participants, additional applications become possible. Among these are collaborative applications such as real-time viewing and group editing of presentations and instant messaging among the meeting participants.

Ad hoc networks consisting of diverse types of devices present many new and exciting possibilities for usage scenarios. While more work is required to establish interoperable methods for general networking,[4] Bluetooth wireless communication is positioned to be an enabling technology for ad hoc networking scenarios.

Hidden Computing

Hidden computing (sometimes called unconscious computing) is one of the most exciting future applications for Bluetooth technology. While not directly addressed in the version 1.0 specification, hidden computing has been discussed at SIG events in the past and is an area ripe for

4. For example, issues with routing, name serving, address assignment and other topics all need to be addressed for effective ad hoc networking. Such issues are also relevant in forming Bluetooth scatternets, discussed in Chapter 6.

future exploration. The fundamental elements required for some forms of hidden computing already exist in the current specification, although the SIG has not developed profiles that describe how the various hidden computing applications might be accomplished in a standard and interoperable manner.

Hidden computing involves a class of applications in which devices that are not overtly being used by a person can still perform tasks on that person's behalf. We have already seen one example that could be considered a hidden computing application: the automatic synchronizer. In that usage model, a PDA "hidden" in a pocket or purse could synchronize with another device without user intervention. Several other examples have been described in the context of Bluetooth wireless communication. Among these are:

- A notebook computer "hidden" in a briefcase in a "sleeping" mode could be configured to awake periodically, receive new e-mail and send information such as new e-mail alerts (and possibly a short clip of the e-mail content) to a mobile phone. The user might then decide to browse e-mail using the mobile phone or process e-mail on the notebook computer.
- A mobile telephone "hidden" in a pocket or purse could be used by an appropriately configured notebook computer "hidden" in a briefcase to access a network in the manner described for the Internet bridge (dial-up networking) scenario. Once the computer is connected to a network in this fashion, network synchronization or transmission and reception of e-mail could be initiated, all without conscious user interaction with either device.

In early stages of the development of the specification, such applications were called "the briefcase trick." With proximity networking enabled by Bluetooth wireless communication, hidden computing applications abound. Other future possibilities might include the use of a hidden device that controls environmental settings (such as home climate and lighting, music, automobile driver's seat and mirror adjustments and so on) based upon the personal preferences of the user, automatic customer discounts applied at points of sale based upon a device tucked away in a pocket or suitcase, and automated identification and authentication when a person checks in with an airline or a hotel.

These sorts of scenarios are almost limitless. While hidden computing applications may not be fully realized for some time, the Bluetooth technology does offer a basis upon which industry innovators could build them.

4

Introduction to the Bluetooth Specification

Any good technical specification should answer several questions of "what?" for its readers. Some topics that a specification ought to address are:

- what is the product or technology?
- what is it designed to do?
- what is it composed of?
- what standards and metrics must an implementer meet?

Questions of "how?" typically are not addressed in a specification but rather are left to the judgment and ingenuity of the implementers. A specification usually does not address exactly how an instance of the technology or product is constructed using hardware or software modules or describe the precise methods used to ensure that standards and requirements are met.

The Bluetooth specification is no different from others in this respect. Even in its great magnitude (over 1,500 total pages in version 1.0B) the specification still focuses primarily on what an implementer needs to know to create products that use Bluetooth technology. One reason for the enormity of the specification is the breadth of the topics that it covers. The specification is not one that addresses only a radio or just a single layer of a software stack or a solitary interface; rather it addresses a combination of hardware and software that includes all of these facets and more, with broad applications and a diverse audience. The SIG deemed this approach necessary, given the many new concepts introduced with Bluetooth technology. However, the SIG adopted existing protocols where feasible; large portions of the specification deal with adapting these protocols to Bluetooth environments.

The SIG involved dozens, if not hundreds of people who spent over a year developing the first version of the specification. Rather than publish a first edition that was informational only, the SIG chose to ensure that the version 1.0 specification was sufficiently correct and complete to enable implementations to begin. While the initial publication was unsurprisingly imperfect (numerous errata have been published) and arguably can never be truly complete (since new applications of the technology will continue to evolve), this approach to producing a comprehensive first version of the specification was appreciated by many Bluetooth adopter companies.

This chapter explains the purpose, scope and structure of the specification and the relationships among its constituent parts. Because the specification is so broad and voluminous it seems unlikely that all readers will read it from cover to cover with equal interest in all of its diverse parts. Since the main body (Parts 2 and 3) of this book deals with the specification, its structure logically mirrors the specification's structure to a great extent. By explaining how the specification is organized, this chapter is designed to direct readers toward the chapters of this book, and hence toward the chapters of the specification, that are likely to be of most interest and relevance based upon the tasks they wish to accomplish or the knowledge they hope to gain.

Purpose of the Specification

Like most technical specifications, the Bluetooth specification is a response to marketing requirements. As previously noted, the SIG's Marketing working group originally generated a *marketing requirements document* (MRD), which is internal to the SIG and includes objectives and usage models which were the genesis of the specification. A core purpose of the specification is to define components that can be used to develop solutions that address these marketing requirements.

Among the objectives set forth in the MRD were those that now are key attributes of Bluetooth wireless communications: an open specification, unlicensed global use, low cost and interoperable solutions regardless of device manufacturer. In fact, each of the fundamental characteristics of the technology in the list in Chapter 1 has some basis in the marketing requirements document. In many cases, portions of the specification can be traced back to an MRD objective. For example, the objective of an open specification is realized with its public availability and royalty-free license; unlicensed global communications are achieved through the use of the 2.4 GHz spectrum; many of the radio's parameters (described in more detail in Chapter 6) are a result of design tradeoffs to specify a robust radio while meeting the objective of low cost; and the objective of interoperability is directly addressed in the more than 400 pages of profiles (volume 2 of the specification).

Many of the usage models and technical characteristics reviewed in Chapter 3 also were recorded first as marketing requirements. Most of these scenarios are described in the MRD and many of these survive largely unchanged today, although they have been refined and expanded. Initial outlines for the interactive conference (file transfer), synchronization, Internet bridge, three-in-one phone, ultimate headset and others all are included within the original MRD, although some of these scenarios originally were known by different names.

Scope

The SIG intentionally began the specification development by focusing first on cable replacement usage models and a basic protocol framework to support them. This philosophy resulted in the specification version 1.0, which defines a protocol stack that enables many important profiles, but the SIG has not stopped there. There is interest in many new applications and profiles; these will continue to be developed by the SIG and are likely to be published in future editions of the specification. Part 4 explores these possibilities further. By starting with the simplest and most fundamental usage models, the SIG was able to bound the version 1.0 specification scope.

The specification does not simply describe some existing implementation. Great care was taken during its development to ensure that anyone who has or can obtain sufficient skills and resources should be able to implement the specification. Recall that the specification was developed by a multicompany special interest group with a shared and stated objective to produce a truly open specification. The elements of the specification were developed to meet the objectives for the technology in a practical manner, not to match preconceived ideas nor a single company's viewpoint, experience or implementations. In fact, for most portions of the specification quite the opposite was true: the process was to draw upon the collective wisdom and experience of the company representatives to produce an initial version of some part of the specification. Hypotheses in the draft specification could then be tested at one or more companies via prototyping or some other means, with the results then fed back into the refinement process.

The many independent implementations of Bluetooth hardware and software by multitudes of vendors, many of whom were not part of the SIG's promoter group, seem to indicate that the SIG has performed well in producing a sufficiently complete specification. It is encouraging that many products with Bluetooth wireless communication are being produced on the basis of the version 1.0 specification.

In any work this large, of course, some errors and opportunities for misinterpretation are likely to arise. The Bluetooth specification is no exception. Following publication of the version 1.0 (or more properly, 1.0A) specification, numerous comments from adopters and others were received. Many of these comments dealt with portions of the specification that were unclear or for which multiple interpretations could reasonably be construed. In addition, minor errors that had slipped through even the diligent review of the SIG members were discovered. For each of these items–and there were dozens if not hundreds–the responsible working group within the SIG considered the comment and, if accepted, prepared an erratum document and corrected the error or clarified the wording in a corresponding change to the specification. The result was the publication in December 1999 of the specification version 1.0B, which is what most people mean when they refer to version 1.0 of the specification (and indeed this is what is meant by such references throughout this book). Of course even as new versions of the specification are generated, document maintenance must continue, and the SIG still deals with errata to the initial specification while developing new material for new versions.

The Specification's Structure

At the highest level the specification is split into two volumes: volume 1 is the core specification, which deals primarily with the protocol stack but also includes descriptions of related items such as testing and compliance; volume 2 is the profile specification. In this book these two volumes are examined in Parts 2 and 3, respectively.

For version 1.0, the core specification is by far the larger of the two volumes, weighing in at nearly 1,100 total pages. Volume 2, the profiles, is about 440 pages in version 1.0. The set of profiles is expected to grow more rapidly though, as new usage cases are formalized. A major portion of the SIG's work following release of the version 1.0 specification is focused on creating new profiles. So as Bluetooth wireless technology becomes more widely used in new industries with new applications, the continued creation of additional profiles is expected.

Volume One Structure

Within volume 1 the protocols are presented largely in a bottom-to-top organization. The specification begins with a short discussion of the radio followed by the baseband, link manager and L2CAP layers. Next are the higher layers: RFCOMM, SDP, TCS and IrDA interoperability protocols. The Host Controller Interface (HCI) is an interface to the baseband controller and link manager, but in the specification the HCI is discussed after all of the higher-level protocol sections (the HCI is a command interface rather than a protocol per se and its use may differ depending upon an implementation's design; thus it is discussed separately in the specification). Additional chapters that do not deal specifically with protocols include WAP interoperability, test mode, test control and compliance discussions. Finally the miscellaneous material is included in the appendices, although much of this material is important for many implementers and should not be overlooked. Among the topics covered in the appendices of volume 1 of the specification are audio (also discussed in Chapter 10 of this book) and Bluetooth assigned numbers.

Appendix VIII of volume 1 of the specification, Bluetooth Assigned Numbers, is a central area in which values defined by the SIG are recorded. This important part of the specification is not detailed elsewhere in this book, so we briefly discuss it here. The Bluetooth Assigned Numbers section of the specification defines values that are expected to change or evolve over time and must be relied upon and therefore registered. Included are the particular values assigned to key fields or structures that must be well known in several protocols. Examples include inquiry access codes and bit definitions for fields that describe device and service classes, used when establishing connections; channel and protocol values used in L2CAP; and several specific values defined for use with SDP. In this latter case these values represent particular services and attributes associated with those services that are required to accomplish the usage models (as described in the profiles) for version 1.0. This list is expected to grow over time as new usage models are adopted. Because it is difficult to predict what new services will be available, it is difficult to pre-assign values for all services;

thus the values are isolated in the assigned numbers registry so that the protocol specification itself need not be modified when instances of new services are developed.

Volume Two Structure

The organization of the profiles in volume 2 of the specification is quite straightforward. Each chapter is a single profile. For the most part, related profiles are grouped together, although the serial port profile seems to be oddly inserted in the middle of the telephony-based profiles. As in volume 1, the profile specification begins straightaway without any introductory or background material to set the stage. In Part 3 of this book we provide some context for the profile specification.

Part 3 of this book mostly follows the structure of volume 2 of the specification, grouping together the GAP and SDAP profiles, telephony-related profiles, serial port-related profiles and networking-related profiles. Although these profiles are not formally grouped this way in the specification, this approach is intended to aid understanding and is discussed further in Chapter 11.

Relationships

While initially it may not be evident that there is some coupling between the two volumes of the specification, there is in fact a correspondence between many layers of the protocol stack and one or more profiles. Because the profiles are intended to inform the reader about how to apply the protocols defined in volume 1 to realize an interoperable implementation of a particular usage case, these profiles tend to map to protocol layers in some fashion, although it is not always a one-to-one mapping. Each profile describes the associated protocol stack that it requires, as well as how to use and configure the appropriate protocol layers. Many profiles are somewhat attuned to certain protocols. For example, the generic access profile primarily defines how to use the baseband, link manager and L2CAP layers of the protocol stack. The service discovery application profile is tightly coupled with the SDP layer of the stack; the telephony-based profiles (intercom, headset[1] and cordless telephony) principally relate to the TCS protocol and audio traffic. The serial port-based profiles[2] (including object and file transfer and the serial port profile itself) and networking-based profiles (LAN

1. The headset profile does not directly use TCS; see the discussion in Chapter 13.

access, fax and dial-up networking) have some affinity with the RFCOMM layer of the stack, with the serial port-based profiles also being tightly coupled to the IrDA interoperability protocols.

So while there is not a direct mapping of the chapters of volume 1 of the specification to those of volume 2, the subject matter of corresponding parts of these two volumes does include related material. Therefore it is usually insufficient to read or write about just one portion of the specification in isolation. This is why this book covers both volumes of the specification in its main body; this is also why this book strives to explain the motivation for and relationships among the various parts of the specification. The following section suggests some methods that might aid in understanding those portions of the specification that are of most interest; if using this book as a guide to the specification, these same methods may be used to determine the focus areas herein, since the structure of Parts 2 and 3 of the book largely mirrors that of version 1.0 of the specification.

Guide to Understanding the Specification

The specification version 1.0 does not include any introductory information about its purpose, scope, structure or component relationships (aside from a table of contents). Its readers will find a title page, some notices and a master table of contents abruptly followed by the radio specification and remaining chapters. Readers will find no preface, no foreword and no organized background information. This is not necessarily a bad thing;[3] it primarily results from the specification's technical nature and direct approach to its subject matter. This chapter is intended to supply some of the information that is not found (or at least not explicitly called out) in the specification, thus making the specification more accessible and better preparing its readers to get the most from it.

While the most straightforward way to read the specification is to start with page 1 of volume 1 and continue reading all the way to the last page of volume 2, and while it is not our intent to discourage this method of reading, this may be impractical in many cases. We expect that many readers would have difficulty absorbing the tremendous detail contained in the more than 1,500 pages of the complete version

2. We group the serial port profiles as listed here; Chapter 11 further describes profile family grouping.
3. After all, it helps to create a market for books such as this one.

1.0 specification. Furthermore, many people probably do not need to learn all of the details of every protocol and profile, and even those who do may find it helpful to digest these details in logical groupings, one at a time. The specification is quite broad, covering everything from low-level radio details to application software considerations. The projected Bluetooth marketplace is expected to be just as broad, offering opportunities for many specialized products and skills as well as for general-purpose ones. Thus, depending upon interest, it may be beneficial to develop an individualized plan for delving into the specification (and into the main body of this book). The suggestions offered here are intended to aid in doing just that.

As partial remedy to the lack of introductory material in the specification, the SIG has published several white papers in addition to the specification. One of those white papers, *Bluetooth Protocol Architecture* [Mettälä99], is a useful overview of the contents of the specification and we recommend it as supplemental reading. This paper covers some of the same topics as Chapters 5 and 11 of this book and may serve as a good companion to that material.

Once introductory material (such as Part 1 and Chapters 5 and 11 of this book along with the cited white paper) is understood, many readers may wish to branch out to particular sections of the specification depending upon their interests and objectives. It is unrealistic to devise a reading plan to fit every audience's need, but this section suggests some general guidelines. It is hoped that most readers will be able to use one or more of these general classifications to achieve an understanding of Bluetooth wireless technology that is appropriate for their own situation.

For General Knowledge

Many readers, perhaps including students, teachers, consultants and others who have general interest in the Bluetooth technology and who wish to understand that technology in the context of their profession, may wish to read this book and the specification to gain general knowledge. Such readers may not need to read the specification from cover to cover since they do not need to learn every detail of the specification that might be required by someone implementing the specification.

Our suggested reading plan for general knowledge is to study the profiles after a general overview of the protocol stack as outlined below.

1. Read Part 1 and Chapter 5 of this book and the SIG protocol architecture white paper (cited above). This material helps to put the protocol stack in context.
2. Review or skim volume 1 of the specification using Chapters 6 through 10 of this book as an explanatory guide to the corresponding specification sections.
3. After reading Chapter 11 as an introduction to the profiles, study each profile in volume 2 of the specification, using Chapters 12 through 15 as an aid in understanding the related profiles.

From a Device Perspective

A great deal of the interest in the Bluetooth technology comes from those who are concerned primarily with implementing that technology in devices. Device manufacturers, software developers and original equipment manufacturers who build device components need to understand the details of the protocol stack. Readers who plan to implement Bluetooth wireless communication in devices, in whole or in part, should be prepared to study the core specification.

For these readers we suggest studying the core specification (volume 1) after becoming familiar with the technology basics, and then reviewing those profiles that are most relevant for the class of device being considered. An outline for this reading plan is:

1. Become familiar with the technology basics by reading Part 1 and Chapter 5 of this book and the SIG white paper already cited, along with other available Bluetooth technology overviews.
2. Read Chapters 6 though 10 of this book as a group in preparation for studying the core specification.
3. Thoroughly read all of volume 1 of the specification (or at least all of those sections that pertain to the implementation at hand).
4. Read Chapter 11 of this book to determine which profiles are relevant to the device or devices being created and then review those corresponding profiles in volume 2 of the specification in tandem with the corresponding chapters of Part 3 of this book. Software developers in particular may need to understand one or more profiles for use in certification and testing. The generic access profile and the service discovery application profile may be of interest to all implementers; other profiles may apply only for certain implementations.[4]

From a Solutions Perspective

Bluetooth wireless technology presents opportunities for many new solutions–not only those described by existing or planned usage models and profiles but also many sorts of new applications of the technology which will undoubtedly be invented in the future. Innovators who are driving these new solutions (especially, although not exclusively software developers and system architects) may need the fullest understanding of the Bluetooth technology. Those who are developing Bluetooth applications and solutions often will need to understand the details of existing profiles and also are likely to require a thorough understanding of the protocol stack. Knowing the capabilities and limitations of protocols is important for anyone setting out to invent new usage models.

While the reading plan for those who wish to be thoroughly immersed in the Bluetooth technology, including solutions developers, might be summarized simply as "read everything," a more practical outline may be:

1. Start with the typical background information already noted, namely Part 1 and Chapter 5 of this book, the SIG protocol architecture white paper and any other authoritative overview information.
2. Read the remainder of Part 2 of this book as a prelude to a study of volume 1 of the specification.
3. Thoroughly read and understand the core specification, referring back to the corresponding chapters of this book where necessary.
4. Read all of Part 3 of this book in preparation for scrutinizing volume 2 of the specification.
5. Study the profile specifications (volume 2) with Part 3 of this book at hand.

Solutions developers especially will want to keep abreast of new developments in Bluetooth wireless communication. For these people, continued reading of current articles in trade and professional journals is advisable, as is taking advantage of additional learning opportunities such as developers conferences.

4. For example, camera or keyboard developers are unlikely to be interested in telephony profiles, and developers of a simple pager may not be interested in fax or dial-up networking profiles.

Part 2

THE BLUETOOTH SPECIFICATION EXAMINED

This book continues with an exploration of volume 1 of the Bluetooth specification, called the core specification. Chapter 5 presents the overall Bluetooth protocol stack and describes the relationships among its layers. The remaining chapters of Part 2 examine each layer of the stack in turn, from the lowest layers to the higher layers. Chapter 6 discusses the Bluetooth radio, baseband, link controller and link manager layers. Chapter 7 explores the logical link control and adaptation protocol (L2CAP) layer and host controller interface (HCI). Chapter 8 discusses the RFCOMM and Service Discovery Protocol (SDP) layers, while Chapter 9 examines the protocols associated with application interoperability with the Infrared Data Association (IrDA) standard. Finally Chapter 10 describes the telephony control and audio protocols.

Part 2 is intended to make important information from the Bluetooth specification more accessible and understandable while explaining the motivation and rationale for key elements of the specification. Drawing upon our experience in helping to develop the specification, we attempt here to reveal its important elements rather than simply echoing information already available to specification readers.

5

The Bluetooth Protocol Stack

The core portion of the Bluetooth specification contains the protocol stack. This stack allows devices to locate, connect to and exchange data with each other and to execute interoperable, interactive applications against each other. In this section we present the major components of the Bluetooth protocol stack, highlighting the relationships among the various layers. The protocols are presented in more detail in following chapters.

The Protocol Stack Components

Figure 5.1 depicts the high-level components of the Bluetooth protocol stack. The elements of the stack (protocols, layers, applications, and so on) are logically partitioned into three groups:

- the *transport protocol* group;
- the *middleware protocol* group; and
- the *application* group.

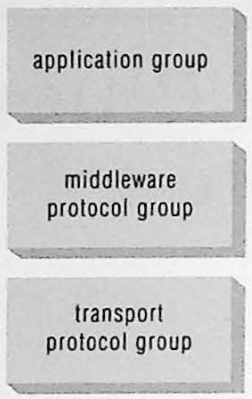

Figure 5.1
A high-level view of the Bluetooth stack.

Transport protocol group: This group is composed of the protocols designed to allow Bluetooth devices to locate each other and to create, configure and manage both physical and logical links that allow higher layer protocols and applications to pass data through these transport protocols. The protocols in this group include the radio, baseband, link manager, logical link and adaptation and the host controller interface.[1]

Middleware protocol group: Additional transport protocols needed for existing and new applications to operate over Bluetooth links comprise this group. The middleware protocol group includes both third-party and industry standard protocols and protocols developed by the SIG specifically for Bluetooth wireless communication. The former includes Internet-related protocols (PPP, IP, TCP, and so on), wireless application protocols, object exchange protocols adopted from IrDA and the like. The latter includes three protocols with a designed awareness of Bluetooth communications that facilitate a large number of other applications to run over Bluetooth links. A serial port emulator protocol called RFCOMM enables legacy applications that normally would interface with a serial port to operate seamlessly over Bluetooth transport protocols. A packet-based telephony control signaling protocol provides for advanced control

1. Strictly speaking, the host controller interface is not a communications protocol. However, the host controller interface specification defines formats for packets that cross a host interface and associations between these packets. For example, when packet A is transmitted, then packet B is expected in return. These formats and associations of packets are key elements of a protocol specification; hence, we group HCI with the transport protocols.

of telephony operations, such as group management and mobility support for cordless handsets and base stations. Finally, a service discovery protocol permits devices to discover each other's services and to obtain information on how to access those services.

Application group: This group consists of the actual applications that make use of Bluetooth links. These applications could be legacy applications that are unaware of Bluetooth transports, such as a modem dialer application or a web-browsing client; or they might be aware of Bluetooth wireless communication, for instance, applications that use the telephony control protocol for controlling telephony equipment.

Chapters 6 and 7 present in more detail the protocols in the transport protocol group. Chapters 8 through 10 discuss the middleware protocols developed and adopted by the SIG. Finally all of Part 3 of this book is dedicated to the various applications that the profiles define.

Before moving to the details of the various protocols and applications, we first discuss the key protocols in the transport and middleware groups and their relationships to each other.

The Transport Protocol Group

Figure 5.2 shows the organization of the protocols in the transport group. These are the transport protocols developed by the SIG to carry audio and data traffic between devices. In this chapter, the presentation of these protocols is made in "top-down" order, or from the point of view of a transmitting device where traffic passes from the upper transport layers to the lower layers. Clearly, traffic follows the reverse path in receiving devices. This illustrates an end-to-end data flow through the protocols of the transport group. In subsequent chapters, these protocols are presented in a "bottom-up" order, or from the point of view of a receiving device; this is the order followed in the specification as well.

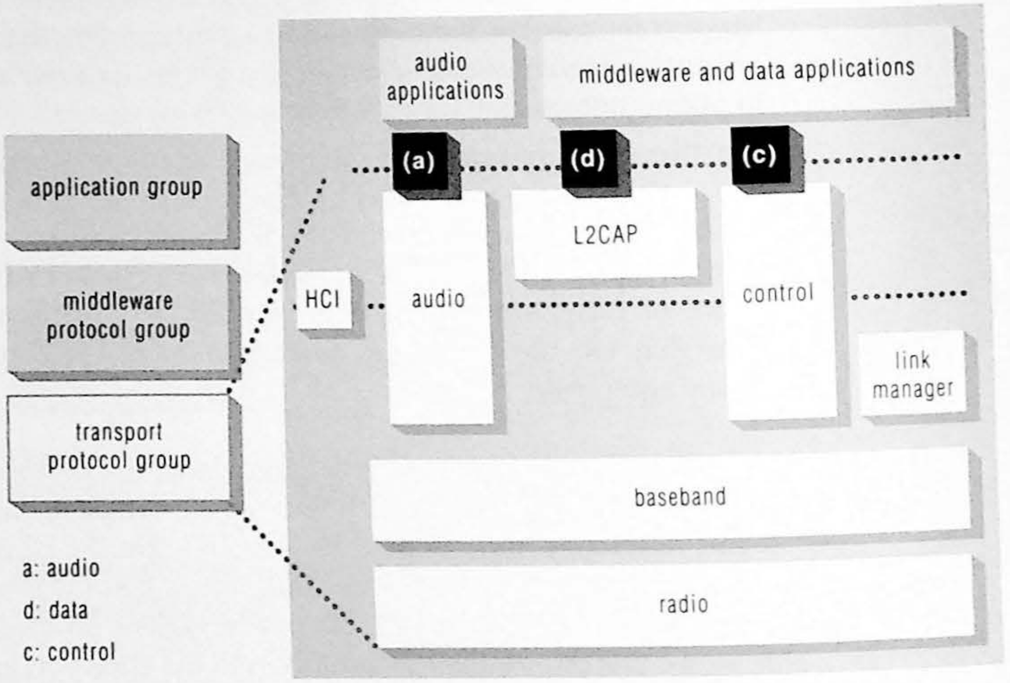

Figure 5.2
The transport protocol group stack.

The transport protocols support both asynchronous transmissions, for data communications, and synchronous (or periodic) transmissions, for telephony-grade (64 Kbps) voice communications. To maintain the high quality of service expected for audio applications, the audio traffic is treated with high priority. Audio traffic bypasses all of the intermediary protocol layers and is funneled directly from the audio application to the baseband layer, which then transmits it in small packets directly over the Bluetooth air-interface.

Before continuing our discussion of the transport protocol group, we observe that the protocols in the transport protocol group do not belong to the *transport layer* (layer 4) of the seven-layer OSI protocol model! With respect to the latter model, the protocols in the transport protocol group would fit best within the *data link layer* (layer 2) and the *physical layer* (layer 1). Collectively, the set of protocols in the "transport protocol group" form a virtual pipe that is used to transport data from one device to another across the Bluetooth air-interface. These protocols in principle define transports between communicating devices; hence, the naming choice for this protocol group. The term "group of Bluetooth transport protocols" is also used to further emphasize that reference is made to the transport protocols developed by the Bluetooth SIG rather than to a transport layer (OSI layer 4) protocol. Note that all of the protocols in this group are always needed to support the communication between Bluetooth devices. This is not true for any other protocol outside this group, even the ones that have been developed by the SIG, like RFCOMM.

The L2CAP Layer

Traffic from data applications is first routed through the *logical link control and adaptation protocol* (L2CAP) layer. The L2CAP layer shields higher-layer protocols and applications from the details of the lower-layer transport protocols. Thus, higher layers need not be aware of the frequency hops occurring at the radio and baseband level nor the specific packet formats used for transmission over the Bluetooth air-interface. L2CAP supports protocol multiplexing, allowing multiple protocols and applications to share the air-interface. It also enables segmentation of large packets used by higher layers into smaller packets for baseband transmission and the corresponding reassembly of those packets by the receiving device. Furthermore, the L2CAP layers in two peer devices facilitate the maintenance of the desired grade of service by negotiating an acceptable level of service. Based on the requested level of service, an L2CAP layer implementation may then exercise admission control for new incoming traffic and coordinate with lower layers to maintain the desired level of service. The L2CAP layer and its over-the-air protocol are described in more detail in Chapter 7.

The Link Manager Layer

Link managers in each device negotiate the properties of the Bluetooth air-interface between them using the *link manager protocol* (LMP). These properties include bandwidth allocation to support a desired grade of service for data (L2CAP) traffic and periodic bandwidth reservation to support audio traffic. Bluetooth link managers in communicating devices use a challenge-response approach for authenticating the devices. They supervise device *pairing* (creation of a trust relationship between the devices by generating and storing an authentication key for future device authentication) and encryption of the data flowing over the air-interface between the devices whenever needed. If authentication fails, the link managers may sever the link between the devices, thus prohibiting any communication between the devices. Link managers also support power control by negotiating low activity baseband modes of operation (introduced in Chapter 2 and detailed in Chapter 6) through the exchange of information about parameters such as the duration of the low-activity baseband mode. Link managers may request adjustments to the transmission power level for further power conservation as described in Chapters 2 and 6. The link manager layer and its over-the-air protocol are described in more detail in Chapter 6.

The Baseband and Radio Layers

The baseband layer determines and instantiates the Bluetooth air-interface.[2] It defines the process by which devices search for other devices and how they connect to them. The baseband layer defines the master and slave roles (described in Chapter 2) for devices–the device that initiates a connection process becomes the master of the link, while the other device becomes a slave. The baseband also defines how the frequency hopping sequences used by communicating devices are formed. It defines the rules for sharing the air-interface among several devices; these rules are based upon a *time division duplex* (TDD), packet-based polling scheme. It further defines how synchronous and asynchronous traffic can share the air-interface. For example, in synchronous transmissions, the master transmits to and/or polls a slave device periodically. The baseband defines the various packet types supported for synchronous and asynchronous traffic, as well as various packet processing procedures including error detection and correction, signal whitening,[3] encryption, packet transmission and retransmissions. The baseband layer and its over-the-air protocol are described in more detail in Chapter 6.

Note that the concept of master and slave devices does not propagate higher than the link manager. At the L2CAP layer and above, communication is based upon a peer-to-peer model and no special provisions are made for different actions in a master device or in a slave device.

Over-the-air packet transmissions are meaningless unless properly matched radio transmitters and receivers, or transceivers, are employed. The Bluetooth radio design includes several parameters designed to make it optimal for use with the Bluetooth protocol stack in short range wireless communications. The radio section of Chapter 6 gives details of the Bluetooth radio.

2. To make an analogy to a cabled environment, it might be said that the baseband determines the "shape" and "pin configuration" of the interface.
3. Whitening refers to signal scrambling and is discussed more fully in Chapter 6.

HCI Layer

The radio, baseband and link manager may be packaged together into a Bluetooth module. The module then attaches to a *host* device, enabling that device with Bluetooth wireless communication. In this configuration, the host contains the L2CAP layer and any appropriate portions of the higher layers of the stack. The module attaches to the host via some physical interface, called the *host transport*, such as Universal Serial Bus (USB), an RS-232 port or a UART. To enable the development of interoperable Bluetooth modules by different vendors, the specification defines a common interface for accessing the lower layers of the stack that reside in the module, independently of the particular physical interface that connects the host to the module. The *host controller interface* (HCI) allows higher layers of the stack, including applications, to access the baseband, link manager and other hardware registers through a single standard interface. Through HCI commands the module may enter certain modes of operation in which, say, an authentication operation or a device paging state may be performed. Through HCI events, higher layers of the stack can be informed of the results of a device inquiry operation, read the settings for the audio codec residing in the baseband, read the signal strength of incoming transmissions, and so on. Of course traffic, both synchronous and asynchronous, passes through the HCI as it is transmitted or received by the host, as well.

While the HCI layer typically resides below the L2CAP layer, it is not a required part of the specification. It has been developed for the sole purpose of enabling interoperability among host devices and Bluetooth modules, either of which could come from a variety of developers. Product implementations need not comply with the HCI specification to support a fully compliant Bluetooth air-interface. For example, in a tightly integrated, embedded system, an HCI may not exist at all or it may exist at a different point of the stack, perhaps above L2CAP, and it might have a form other than the one described in the specification. The HCI and the various host transport protocols are presented in more detail in Chapter 7.

The *control path* shown in Figure 5.2 is used to communicate control information between layers. For example, the L2CAP layer might notify the link manager of its quality of service expectations, or an application could communicate an end user's request for a lower power consumption mode. Typically, although not exclusively, the controls that are exposed to the higher layers (including to an end user) are for setting a mode of operation for the device that persists until that mode is again explicitly altered through an action that originates at a higher layer. For example, one may manually enable or disable authentication or encryption for a given device. A high-layer entity, like an application or a user, could place a device in a reduced power consumption mode, which would be translated to a control signal that a link manager understands and supports so it can act accordingly. Similarly, a device could also be placed in a discoverable mode, where the device responds to inquiries sent by other devices, or the device could be set to a private mode, in which the device responds only to connection requests from specific known devices, which also must be authenticated.

The control path is not explicitly described in the specification, but it is interwoven among the various protocols in the stack. Nevertheless, the HCI specification includes the bulk of the information that the control path carries. We honor this same approach, discussing the control signals carried through the control path via the HCI and the various protocols and modes of operation that these signals affect.

The Middleware Protocol Group

Figure 5.3 depicts the middleware protocol group. The middleware protocols make use of the underlying transport protocols and present to the application layers standard interfaces that may be used for communicating across the transports. Each of the middleware layers defines a standard protocol that allows applications to use a higher level of abstraction than would direct communications with the lower-layer transport protocols. The middleware protocols consist of:

- *RFCOMM*, a serial port abstraction;
- *Service discovery protocol* (SDP), used to describe available services and to locate needed services;
- a set of *IrDA interoperability* protocols adopted from the IrDA standard that enables interoperable use of IrDA-enabled applications; and
- a *telephony control protocol* (TCS), used for controlling telephone calls that might be used either for audio or for data.

Each of these protocols, along with Bluetooth audio communication (which is not a protocol per se but is considered part of the software stack) is described separately below and in detail in the corresponding chapters that follow.

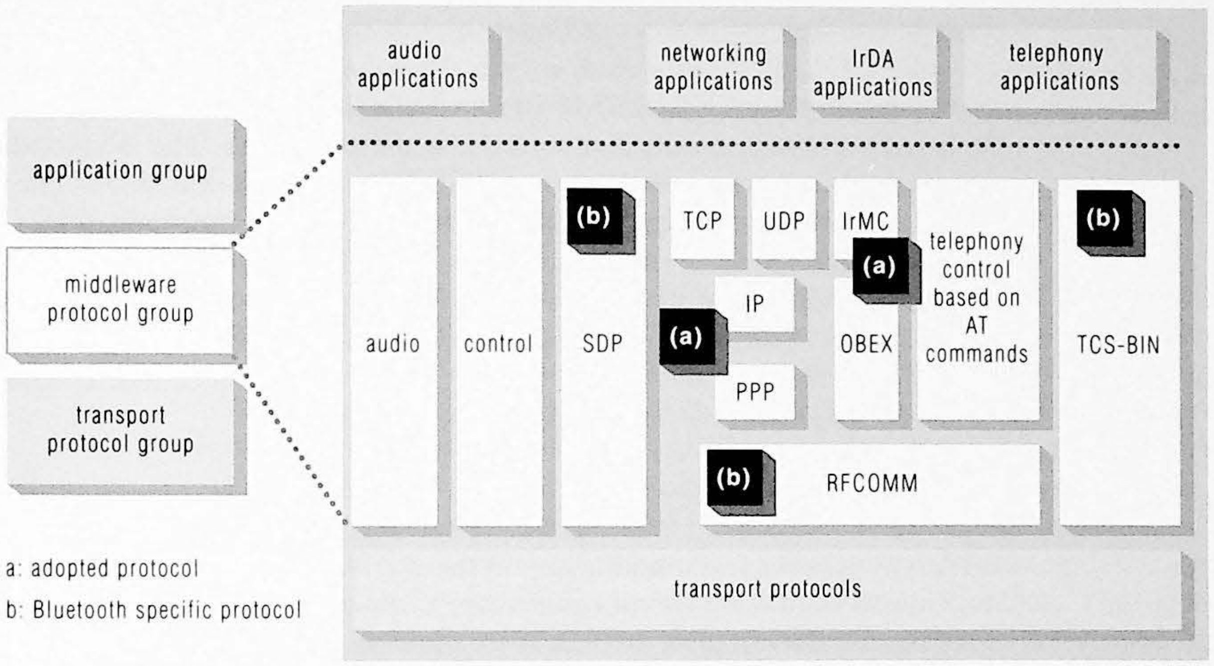

Figure 5.3
The middleware protocol group stack.

RFCOMM Layer

Serial ports are one of the most common communications interfaces used with computing and communications devices today. Most serial communication involves a cable for transferring data across serial ports. Since Bluetooth wireless communication is aimed at replacing cables, support for serial communications and related applications is an important feature for the initial set of cable-replacement usage models. Peer-to-peer file and object transfer, data synchronization and dial-up networking are some applications that commonly use serial communications (and associated cables).

To facilitate the use of serial communications over Bluetooth wireless links, the protocol stack defines a serial port abstraction called *RFCOMM*. RFCOMM presents a virtual serial port to applications; this facilitates the easy migration of applications modeled for cabled serial communications to the realm of wireless serial communications. An application can use RFCOMM very much like a standard wired serial port to accomplish scenarios such as synchronization, dial-up networking and others without significant changes (if any) to the application.

Thus the intent of the RFCOMM protocol is to enable legacy, serial port-based applications to use Bluetooth transports.

RFCOMM is modeled on the European Telecommunications Standards Institute (ETSI) TS 07.10 standard. This standard defines multiplexed serial communications over a single serial link. The Bluetooth specification adopts a subset of the ETSI 07.10 standard and also defines some adaptations designed specifically for Bluetooth communication.

Because serial communications are so prevalent in digital devices, the serial port capabilities that RFCOMM provides to applications make it an important part of the protocol stack, especially for enabling legacy applications based upon version 1.0 of the specification. Details of the RFCOMM layer can be found in Chapter 8.

SDP Layer

In some respects, instantiating any of the Bluetooth usage models might be viewed as making use of some set of services. A primary motivation for forming networks of devices is to allow those devices to communicate with each other and thus avail themselves of each other's services. In traditional networks such as Ethernet LANs, services such as file serving, print serving, name serving, bridges and gateways are provided by some set of devices (usually considered "servers") in the network so that other devices (usually considered "clients") can use them. In many cases the clients locate these network services through some static configuration; this configuration is often established and maintained by a system administrator who configures the client devices (or gives the necessary configuration information to the users of those devices, who in turn configure them).

For dynamic ad hoc networks such as those that could be enabled by Bluetooth wireless communications, though, this sort of static configuration is insufficient. Any two or more devices might begin communicating over Bluetooth links on the spur of the moment, and if these devices are to be able to make use of each other's services they require a more dynamic means of locating those services. Once a communication channel has been established, a next logical step in device communications might be to find out about services that are available to the device. This is what the Bluetooth *service discovery protocol* (SDP) addresses. SDP defines a standard method for Bluetooth devices to discover and learn about the services offered by other devices; symmetrically, it defines a way for devices to describe those services offered to other devices.

Service discovery is a key component in enabling end-user value in dynamic networks. The Bluetooth service discovery protocol is designed specifically to enable this function in an efficient and optimized manner within environments that exploit Bluetooth wireless communication. SDP is described in detail in Chapter 8.

IrDA Interoperability Protocols

Chapter 2 described infrared wireless communication and its resemblance in some respects to Bluetooth wireless communication. The Infrared Data Association (IrDA) has defined protocols for exchanging and synchronizing data in wireless environments. The SIG has chosen to adopt several of the IrDA protocols and data models because IrDA and Bluetooth wireless communication share some important attributes, usage scenarios and applications.

A fundamental requirement for exchanging data among devices is to define the format, both syntax and semantics, of that data. The *infrared object exchange* (IrOBEX or often, just OBEX) protocol developed by the IrDA is a session protocol for peer-to-peer communication. Among the applications in which OBEX can be used is the exchange of well-defined objects. Data objects such as electronic business cards (*vCard* format), e-mail or other messages (*vMessage* format), calendar entries (*vCal* format) and others can all be exchanged using the OBEX protocol. OBEX is the fundamental building block upon which the usage models of file transfer (object exchange) and object push are built. Additionally, *infrared mobile communications* (IrMC), another IrDA-defined protocol, enables the synchronization of these same objects.

The IrDA interoperability layers of the protocol stack are intended to promote interoperability at the application layer. They are described more fully in Chapter 9.

Networking Layers

Bluetooth wireless communication uses a peer-to-peer network topology rather than a LAN style topology. Nevertheless, the technology does make allowances for networking, in particular connecting to larger networks through a dial-up connection or via a network access point. The specification also discusses interoperability with the *Wireless Application Protocol* (WAP), a specification for wireless networking used by devices such as mobile telephones.

Dial-up networking uses the AT command layer of the middleware protocol stack, which is discussed below, to establish a connection to a network. In many cases, the network being accessed is one that uses the Internet Protocol, referred to as an IP network. Once a dial-up connection to an IP network is established, standard Internet protocols (such as TCP, UDP, HTTP and so on) can be used by the device (perhaps a notebook or handheld computer) that initiated the network connection to interact with the network.

A device might also connect to an IP network via a network access point as described in the LAN Access Using PPP Profile. In this case, a Bluetooth link connects the device to a network access point; the network access point is in turn connected to the larger (most likely, but not necessarily wired) network. The Internet *point-to-point protocol* (PPP) is used over the Bluetooth link to connect to the access point. As with dial-up networking, once the PPP connection is established, standard Internet protocols can be used to interact with the network. Access to a WAP network using a WAP gateway works similarly; the same sort of PPP connection is established to an IP network access point over which standard WAP protocols can be used to interact with the network.

It is worth noting that release 1.0 of the specification does not define a profile or an instance of the protocol stack that supports the use of Internet protocols such as TCP/IP directly over Bluetooth links. The only means of IP network access defined in version 1.0 of the specification is through the use of PPP as described above. While it certainly should be possible to directly operate an IP protocol stack using Bluetooth wireless communication as the bearer, the SIG has not defined an interoperable manner (that is, a profile) for such operation. In all likelihood, future revisions to the specification will address the direct use of Internet protocols with Bluetooth wireless communication.

TCS Layer and Audio

As previously noted, one key advantage of Bluetooth wireless communications is its ability to carry voice traffic as well as data. While the protocol layers discussed to this point exist primarily for use with data traffic, the Bluetooth *telephony control specification* (TCS) layer is designed to support telephony functions, including call control and group management. These operations are often associated with voice calls in which TCS is used to set up the call parameters; once a call is established, a Bluetooth audio channel can carry the call's voice content. TCS might also be used to set up data calls, such as would be used with the dial-up networking profile; in this case, the call content is then carried as standard data packets over L2CAP.

The TCS protocols are compatible with the International Telecommunications Union – Telecommunication (ITU-T) Q.931 specification. Because they use a binary encoding, these protocols are referred to in the specification as *TCS-BIN*. During development of the specification, the SIG also considered a second TCS protocol called *TCS-AT*. TCS-AT defined a modem control protocol (often called AT commands) that flowed through the RFCOMM layer. While AT commands over RFCOMM are indeed used for some applications, the specification does not define a separate protocol for TCS-AT. TCS-BIN is appropriate for some of the release 1.0 telephony-based profiles; applications that need to use AT commands over the RFCOMM serial interface are free to do so (as shown in Figure 5.3), but the specification does not define these AT commands as a separate protocol. Several of the version 1.0 profiles, including headset, fax, and dial-up networking, all use AT commands over RFCOMM rather than the TCS-BIN protocol.

The TCS-BIN protocol includes call control functions, group management functions and a method for devices to exchange call signaling information without actually placing a call or having a call connection established. Each of these facets of TCS is detailed in Chapter 10.

Audio, especially voice audio, is treated uniquely in Bluetooth communication. Because audio traffic is *isochronous*, meaning that it has a time element associated with it, voice audio traffic typically is routed directly to and from the baseband layer; it does not go through upper layers such as L2CAP.[4] Special baseband packet structures, called *synchronous connection-oriented* (or SCO) packets are defined for use with typical audio traffic. Bluetooth communication allows for up to three audio channels at one time (with some bandwidth left over for data traffic).

Bluetooth audio communication takes place at a rate of 64 kilobits per second (Kbps) using one of two data encoding schemes: 8-bit logarithmic *pulse code modulation* (PCM) or *continuous variable slope delta* (CVSD) modulation. Compression techniques called A-law and µ-law are applied for PCM audio.

Since voice is a primary application of audio communication (especially in devices such as smart phones that use Bluetooth wireless communication), audio is often equated with voice. Of course, voice is

4. Packetized digital audio could be carried as standard data packets using L2CAP, but this case would be treated as data traffic. In general, when we refer to voice or audio throughout this book we are speaking of the Bluetooth audio traffic carried directly over the baseband link in SCO packets.

not the only sort of audio traffic that can be carried by the Bluetooth baseband. So long as the audio stream can be sufficiently rendered at 64 Kbps, it can be transmitted and received over Bluetooth links. Thus, in addition to voice, Bluetooth audio channels could carry other forms of audio, such as some music[5] or short audio clips.

Because voice is an integral part and key attribute of Bluetooth wireless communication, it may be somewhat surprising that only a few pages of the specification deal directly with audio. This is not because audio is unimportant; rather it is mostly because audio traffic is carried directly over the baseband in packets designed for that purpose. Thus it is not necessary to include lengthy explanations of audio traffic – the format of the audio data is specified using existing standards – and that, for the most part, is that. It is, however, quite important to understand the baseband SCO packet structure and usage to achieve a fuller understanding of Bluetooth audio communication. Thus, while audio is explained in more detail in Chapter 10, it is also indirectly detailed in the baseband discussion of Chapter 6.

The Application Group

While some of the protocols that we refer to here as middleware protocols (especially IrDA interoperability protocols like IrOBEX or IrMC) might be considered by some to be application-level protocols, this is not what we mean when we write about the application group. The application group here refers to software that resides above the protocol stack as defined by the SIG. This is software that is supplied by device manufacturers, independent software vendors or others which exercises the protocol stack to accomplish some function that benefits the user of a Bluetooth device. Application software as defined here is depicted in Figure 5.4, which illustrates several possibilities for the organization of Bluetooth application software.

5. Bluetooth audio, being optimized for voice traffic, is not designed to carry music of high quality, such as CD-quality music. Support of music is likely to require the use of compression techniques.

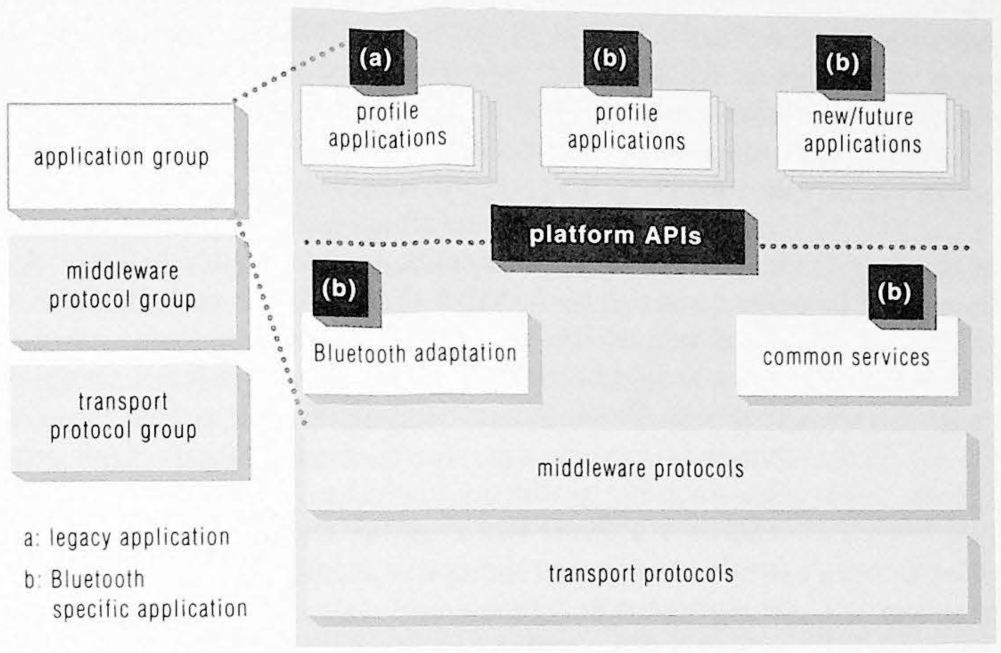

Figure 5.4
General view of the application group, with possible application organization, including adaptation layer and common application services.

Of most interest within the application group are those applications that instantiate the Bluetooth profiles. That is, given a Bluetooth protocol stack in a device, someone still needs to write application software to drive that stack to accomplish functions such as dial-up networking, file transfer, headset communications and so on. The SIG defines only the middleware and transport protocols for the stack; it does not define application protocols per se nor does it define application programming interfaces (APIs). Yet applications clearly are needed to accomplish the usage scenarios that are envisioned for Bluetooth wireless communications. To realize a Bluetooth profile, it is the added application code that uses the underlying protocol stack to supply value to an end user. While a profile defines how interoperable applications that address various usage cases are to be built, the look and feel of these applications is not defined in the specification. Application software developers have sufficient latitude to differentiate their products with added features or user interfaces, without violating the interoperability guidelines spelled out in the profiles.

It might be the case that some existing ("legacy") software that was designed for use with transports other than Bluetooth wireless communication can be employed using Bluetooth links. Because the SIG has defined layers of the protocol stack, such as the IrDA interoperability and RFCOMM layers, to support such legacy software, it could be possible for these existing applications to take advantage of Bluetooth links with little or no change. Examples include existing IrDA applications for object exchange or synchronization and dial-up networking applications. These applications all use existing protocols which are supported in Bluetooth environments, and typically they are designed to work over a serial port. Since the protocol stack includes the RFCOMM serial port abstraction, many existing applications of these and other types should be able to be mapped from IrDA or serial cable environments to Bluetooth environments in a straightforward manner, with minor (if any) changes to those applications. One way to accomplish this on some platforms might be to develop a thin layer of *Bluetooth adaptation software* that maps existing serial and other communications for the platform to the corresponding Bluetooth communications stack, as illustrated in Figure 5.4.

Another option is to develop new applications specifically designed to operate in Bluetooth environments. In cases where no existing application accomplishes a Bluetooth usage case, or when it is desirable to include application features that exploit unique capabilities in the protocol stack, a new application may be a good approach. When applications are developed specifically to leverage Bluetooth wireless communication for some platform, often it may be advantageous to develop *common services* for those applications. Such common services might include security services, connection management services, SDP services, and so on. These common application services might be realized with application-level code such as a security manager (see [Muller99] for a discussion of this topic), a Bluetooth management console (perhaps with associated user interface that allows a user to select devices and services in a piconet with which he wishes to interact), a common SDP client and server (again perhaps with a user interface for service searching and browsing), and so on. Figure 5.4 includes a representation of these common application services.

If the specification does not define APIs, how can standard applications for Bluetooth usage cases be developed? The answer lies in the profiles. Recall that profiles are developed to establish a base point for use of the protocol stack to accomplish a given usage case in an interoperable manner. Because Bluetooth wireless communication is expected to be supported in a plethora of device types, on various platforms, the specification of a single standard API that would be appropriate for the full range of devices and platforms could be quite difficult, to say the least. But since the SIG has chosen profiles as the means to establish interoperability among all of these devices, it seems that these same profiles can provide guidance for developing applications on the many platforms that are relevant for Bluetooth technology. As noted above, both legacy applications and applications developed specifically to use Bluetooth links could realize Bluetooth profiles.

When a technology is incorporated into a platform and results in the need for new APIs, those APIs are often best developed by experts on that platform rather than by experts on the technology. Thus the SIG has chosen not to specify Linux® APIs, Windows® APIs, Symbian™ APIs or any other APIs; instead the profiles define the necessary function to enable platform experts to develop appropriate APIs for use with Bluetooth applications on relevant platforms. So, even though APIs are not included in the specification, the profiles do indeed give direction to application developers. Some profiles do this more directly than others. The service discovery application profile, for example, describes potential application programming models and defines service discovery primitives that could in a straightforward manner map to APIs on a given platform.

Application development is not limited to software that instantiates profiles. As with devices, the landscape for applications is broad, perhaps even virtually unlimited. While the release 1.0 profiles define a base set of interoperable applications, and while new profiles will undoubtedly continue to be developed, these are not the only usage cases for which applications are expected to be written. Industry innovators are likely to develop many new uses for the Bluetooth technology and to create the application software to support these new usage scenarios. As more software continues to be developed to enable the existing profiles on multiple platforms, the APIs developed for and experience gained in that software development should provide a foundation for new applications. The opportunity for these new applications is tremendous.

6

The Lower Protocols of the Transport Group

This chapter and the following one present the protocols in the transport protocol group, from the radio up to and including L2CAP. In the Bluetooth specification, these protocols are detailed in over 600 pages. It is not within the scope of this book to recreate the algorithmic and implementation details of the specification. Instead, the key components of the protocols are highlighted. Readers who may want to learn about the protocols in more detail should supplement this reading with the study of the corresponding parts of the specification.

This chapter focuses on the lower-level functions (including the over-the-air protocols and information processing) in a Bluetooth system, which typically are performed within the Bluetooth hardware/firmware module as shown in Figure 6.1. The host I/O portion is covered in the next chapter in the presentation of the host controller interface.

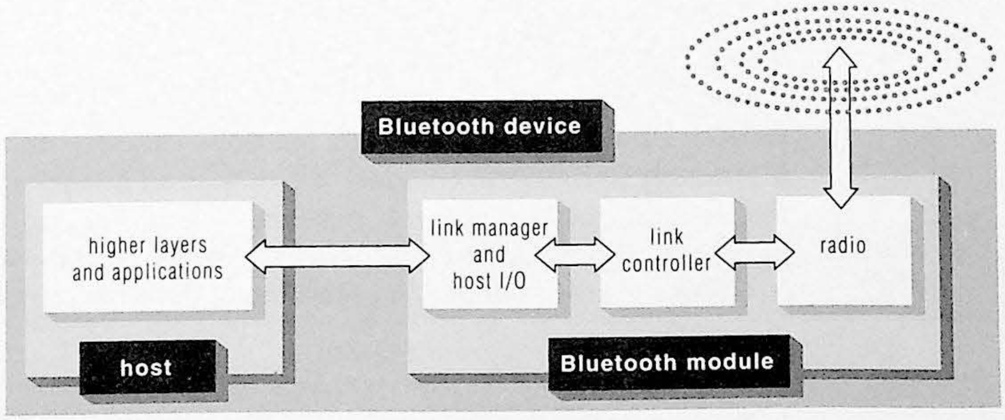

Figure 6.1
High-level functional composition of a Bluetooth module.

As shown in Figure 6.1, a Bluetooth *device*[1] represents the complete physical entity (such as a notebook computer, a digital phone, an information appliance, and so on) that contains applications that can communicate using the Bluetooth wireless technology incorporated in that device. Here we assume that a device is associated with one and only one transport protocol group implementation and air-interface. However, as mentioned in the previous chapter and further elaborated in the next one, the L2CAP layer supports the multiplexing of several higher-layer protocols, like RFCOMM, SDP, and so on, and hence middleware protocol stacks and applications, as well.

The design point of the Bluetooth transport protocols is dictated primarily by low manufacturing complexity, associated low cost and fast time to market. For this reason, a low-cost, easy-to-implement, frequency-hopping spread-spectrum radio solution was selected. In addition, the selection of a master/slave architecture for the baseband transmission was dictated by the ad hoc nature of the systems considered. In particular, Bluetooth piconets can form spontaneously among disparate devices with vastly different power and computing capabilities, unlike wireless LAN systems which are designed primarily to support a wireless extension of LAN connectivity services to relatively "power-unconscious" personal computers.

To support ad hoc connectivity with minimal (if any) state maintained by the Bluetooth devices, piconets are dynamically formed in isolation without the support of third-party (infrastructure) control signaling. For as long as needed, the master of a piconet serves as the control point for the communications on the piconet. For the duration of the existence of a piconet, the operation of a master resembles that of a base station of a picocellular system. Thus, the Bluetooth technology permits the spontaneous creation of a temporary picocellular system, where traffic is regulated by a spontaneously created base station, the master, that regulates the traffic to and from the other member of the picocell, the slaves. Recall from previous chapters that the Bluetooth technology permits the creation of multiple, ad hoc picocellular systems that remain operational even in cases of temporal and spatial overlapping. The use of master and slaves reduces the complexity of the design and thus the cost of deploying the Bluetooth technologies.

1. The terms *unit*, *node*, *station*, and so on may (and have) also been used instead of the term device. Insofar as possible, however, the use of these other terms is avoided. The reason for this choice will become apparent later.

The rest of this chapter highlights the Bluetooth radio; the link controller and the baseband functions it controls, like piconet creation and the medium access protocol; and the link manager that configures and manages the links between devices. In the reference implementation of a Bluetooth module, the radio and the link controller typically are implemented in hardware while the link manager is in firmware. As such, the radio and the link controller (with its baseband functions) were the first parts of the specification to mature, providing a stable enough hardware specification for chip designers to use in designing their early Bluetooth chipsets. The link manager specification originally was focused primarily on security-related transactions. The SIG's further work on the link manager specification has enriched it over time to take full advantage of the capabilities available in the baseband specification.

The Bluetooth Radio

The Bluetooth system operates in the 2.4 GHz *industrial, scientific,* and *medical* (ISM) band. The ISM bands are license-free bands set aside for use by industrial, scientific and medical wireless equipment. Regulatory authorities around the world have opened the ISM bands for use by low-power systems that can be operated without the need for a license but nevertheless under strict regulations. In the United States, these regulations are set by the Federal Communications Commission (FCC) and are detailed in the *Code of Federal Regulations part 15* [FCC99]; section 15.247 of the FCC regulations deals with the operational rules of *intentional radiators* operating in the various ISM bands including the 2.4 GHz band.

The 2.4 GHz ISM band is a globally available radio band, albeit not frequency harmonized. Table 6.1 shows the ISM frequency availability in the majority of countries around the globe.

Table 6.1
License-free frequency allocation in the 2.4 GHz band.

	2.4 GHz ISM band (MHz)	*Frequency channels (MHz)* $k = 0, 1, \ldots, m\text{-}1$	*LGB*[1] *(MHz)*	*UGB*[2] *(MHz)*
	2,400.0 – 2,483.5	2,402 + k; m = 79	2.0	3.5

1. *"LGB" stands for "lower guard band."*
2. *"UGB" stands for "upper guard band."*

The radio part of the specification consists mostly of a series of design specifications for Bluetooth transceivers, like in-band and out-of-band spurious emissions, frequency accuracy, co-channel and adjacent-channel interference, out-of-band blocking, intermodulation characteristics, and so on. The selection of the radio design specifications is driven by the requirement to allow the development of high-quality, low-cost transceivers that comply with the various 2.4 GHz ISM band regulations around the world.

The numerous design parameters plus the radio testing conditions are not repeated here. Readers interested in the Bluetooth radio design can find an abundance of information in the corresponding part of the specification. Since no new data-exchange protocols nor data-processing algorithms were developed for the Bluetooth radio, the radio part of the specification was the first to be completed. The errata to this portion of the specification are minimal, and the few that surface pertain mostly to the accommodation of new regulations or refinement of the radio testing procedures.

The Bluetooth transceiver is a *frequency-hopping spread-spectrum* (FHSS) radio system operating over a number m of 1 MHz-wide channels. While for the majority of countries $m = 79$, as shown in Table 6.1, regulations in certain countries may further constrain the license-free frequency spectrum for the 2.4 GHz ISM band. Thus, the Bluetooth radio (and the baseband described in the next section) design can accommodate two alternatives that operate over 79 or 23 channels, each one of which is 1 MHz wide. For frequency-hopping systems operating in the 2.4 GHz ISM band, the FCC part 15.247 regulations restrict the maximum peak output power of the radiator to no more than 1 watt (30 dBm). Moreover, at least 75 out of the 79 frequency channels must be used pseudo-randomly with a total residence time in each of the frequencies not to exceed 0.4 seconds within a 30-second period. A Bluetooth radio utilizes the maximum number of channels available, 79 in the United States, and it hops at a high rate, 1,600 hops per second, pseudo-randomly across all these frequencies to achieve high noise resilience. The use of *direct-sequence spread-spectrum* (DSSS) systems, which are also permitted in the 2.4 GHz ISM band, may be prohibitively costly for the low-cost requirement of Bluetooth radios.

Figure 6.2 summarizes the key operations in the Bluetooth radio. The figure also shows a pair of logical interfaces for carrying data and control information between the radio and the rest of the Bluetooth system. Here data relates to all information that is transmitted or received over the air. The control information controls the behavior of the radio.

On the transmit side, this includes the carrier frequency to which the transmitter needs to tune prior to transmission of a bit stream of information over the air and the power level that is to be used for this transmission. On the receive side, this includes the carrier frequency to which the receiver must tune for receiving a bit stream of information and (optionally) the strength of the signal being received. Power-supply and time-signaling lines are not shown in the figure. A standardized set of interfaces for the data and control information is not provided in the specification. Thus, chip designers and manufacturers can choose to integrate the radio component with the rest of the components of a Bluetooth module in the way they believe is best for a low-cost, power-efficient system. Table 6.2 summarizes some of the key operational parameters of the Bluetooth radio.

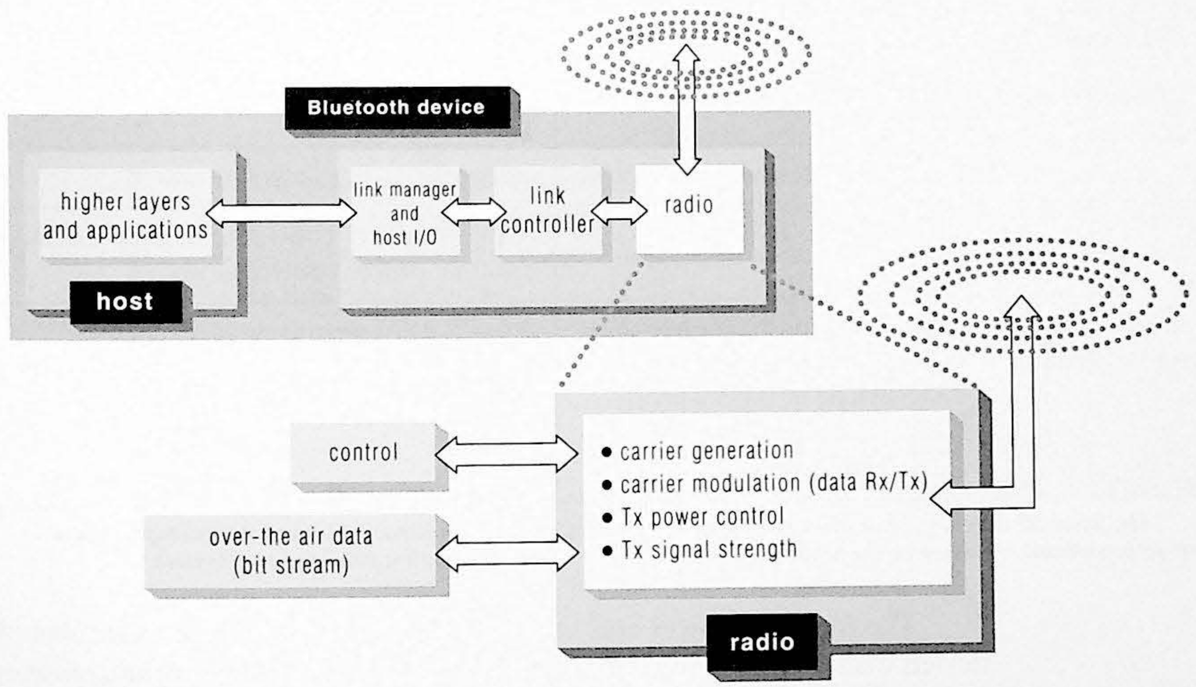
Bluetooth device
higher layers and applications
link manager and host I/O
link controller
radio
host
control
over-the air data (bit stream)
• carrier generation
• carrier modulation (data Rx/Tx)
• Tx power control
• Tx signal strength
radio

Figure 6.2
The Bluetooth radio operations.

Table 6.2

Key operational parameters of the Bluetooth radio specification.

Modulation	Gaussian frequency-shift keying (GFSK)	BT product[1]: 0.5; modulation index: 0.28 – 0.35
Symbol rate	1 Msymbol per second	using the binary GFSK, this translates into 1 Mbps raw link speed; bit transmission time: 1 μsec
Frequency-hopping rate	1,600 hops per second, typical	residence time: 625 μsec per hop
	3,200 hops per second for inquiries and pages	residence time: 312.5 μsec per hop
Transmit power	*Class 3*: 0 dBm (1 mW)	a typical Bluetooth radio; optional power control to below –30 dBm
	Class 2: 4 dBm (2.5 mW)	optional power control as above
	Class 1: 20 dBm (100 mW)	required power control to at least 4 dBm; optional power control as above
Receiver sensitivity	a Bluetooth receiver must attain a raw *bit error rate* (BER) of 0.1% with an input signal level of –70 dBm or lower	the –70 dBm sensitivity level shall be attained for any input signal generated by any compliant Bluetooth transmitter

1. The term "BT product" is not short for "Bluetooth product." It is a parameter describing the quality of transmitted waveforms expressed as the product of the bandwidth of the modulation filter and the bit time.

The transmit power and receiver sensitivity values are examples of design decisions that were made to reduce cost and power requirements for the Bluetooth radio. An IEEE 802.11 wireless local area network (WLAN) may use up to 30 dBm (1000 mW) of transmit power in the United States (lower power-levels in other parts of the world) and not less than 0 dBm (1 mW). Hence, an 802.11 wireless solution may not be suitable for many power-constrained personal, portable devices. The sensitivity level is also much lower than that of an 802.11 radio receiver[2]. This means that a simpler Bluetooth radio can be built at a reduced cost as compared to an 802.11 radio.

The Link Controller and Baseband

As discussed in the previous section, the radio deals with the acts of sending and receiving data over the air. Outside the scope of the radio's responsibilities are considerations such as what data to transmit and when, what data to wait for and when, and which carrier frequency and transmit power to use. These are the responsibilities of the Bluetooth link controller, described later in this chapter, that executes the baseband communication protocol and related processes.

Figure 6.3 summarizes the key functions of the Bluetooth baseband. These include piconet and device control functions like connection creation, frequency-hopping sequence selection and timing; modes of operation such as power control and secure operation; and medium access functions like polling, packet types, packet processing and link types. These items are detailed later in the chapter as we proceed through the operational phases of a Bluetooth device. The figure also shows a set of logical interfaces for carrying data and control information between the baseband and the rest of the Bluetooth system. For the same reasons mentioned in the radio section, the specification avoids specifying any standardized set of interfaces for the data and control information. Nevertheless, their presence, even on a "logical" plane, aids this presentation, since it allows us to concentrate on the baseband operations in isolation from the rest of the Bluetooth system.

2. The receiver sensitivity for an IEEE 802.11b radio receiver, which uses DSSS, is specified to be –80 dBm for a frame error rate of 8% for a 1024-byte MAC protocol data unit when a 2 Mbps DQPSK modulation is used. For more information on the IEEE 802.11 WLANs see [IEEE99]

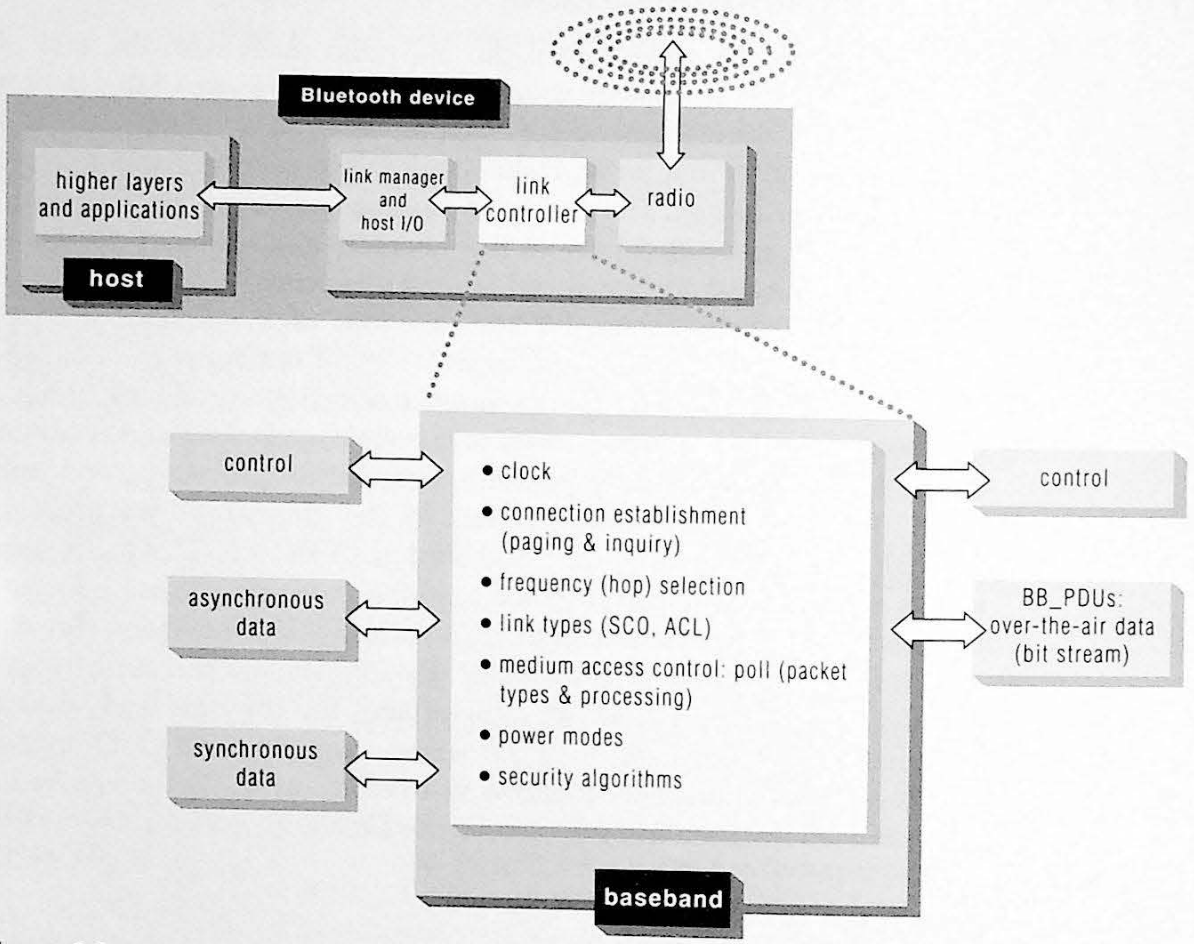
Bluetooth device
higher layers and applications
host
link manager and host I/O
link controller
radio
control
asynchronous data
synchronous data
• clock
• connection establishment (paging & inquiry)
• frequency (hop) selection
• link types (SCO, ACL)
• medium access control: poll (packet types & processing)
• power modes
• security algorithms
control
BB_PDUs: over-the-air data (bit stream)
baseband

Figure 6.3
The baseband functions.

Due to the vast and diverse areas covered by the baseband specification, the remainder of this section tries to combine the information flow in the specification with that of more traditional communication protocol standards and related articles. The discussion progresses stepwise, identifying the key components over which Bluetooth communications actually occur. We begin with the definition of a piconet over which Bluetooth devices can exchange information packets. We then describe fundamental helper elements that assist in the creation and maintenance of the piconet, such as the Bluetooth clock and frequency-hopping selection process. Using these helper elements, the presentation continues with the description of the procedures that Bluetooth devices follow to create and join a piconet. Finally, with a piconet in place, we focus on protocols and processes like medium access control, error control, security, power-saving operation and so on, that are exercised on the Bluetooth devices and on information packets sent and received over the piconet.

The Piconet

For devices to communicate with each other using Bluetooth wireless technology, they need to be part of a *piconet.* Simply stated, a piconet comprises a shared communications channel through which members of the piconet communicate. In the FHSS space in which Bluetooth radios operate, this communication channel consists of a well-defined sequence of frequency hops pseudo-randomly selected from the set of frequencies in Table 6.1 at a nominal rate of 1,600 hops per second. The members of the piconet are able to follow the successive hops of the frequency-hopping sequence in a synchronized manner. Piconets are formed as needed and endure for as long as participating devices need to communicate. The baseband operations define how a frequency-hopping sequence for a piconet is created, how devices learn how to follow it and hence join the piconet, and how to send and receive information packets communicated in a coordinated manner between devices in the piconet.

Bluetooth piconets are formed in a rather ad hoc manner. They are formed on demand among devices that want to communicate with each other without relying on the services of a dedicated support entity, such as a *base station* in a cellular network or a corporate or home WLAN. The baseband protocol establishes the rules according to which these ad hoc connections are established such that devices communicate in a coordinated and efficient manner.

The frequency-hopping sequences that define the communication channels in piconets are highly uncoordinated, or, to be more precise, they are created in a manner that makes them appear highly uncoordinated. Thus, owing to the frequency-hopping nature of transmissions in a piconet, multiple piconets may exist in time and space with minimal interference among themselves. Multiple piconets overlapping, at least partially, in time and space are referred to as *scatternets.* This opens the possibility of interpiconet communications when devices become members of multiple piconets.

Each piconet has one and only one *master* and one or more *slaves.* These roles are temporary ones and they are meaningful only while Bluetooth devices are members of a piconet. Certainly, Bluetooth devices may be built to operate only as masters or only as slaves, but this is a host application and usage scenario issue rather than a Bluetooth specification issue. The specification generally assumes that a Bluetooth device is capable of acting both as a master and as a slave, depending upon the role required to accomplish a given usage case. In the case of a scatternet, a device that participates in more than one piconet can be the master of at most one of these piconets, while it can be a slave in several of them. Through a detailed process, described later in this chapter, a master and a slave may exchange their roles. It is also possible for an "old" piconet based around an original master to be migrated to a new piconet with a new master. Piconet migration as well as communication across scatternets are not discussed in detail here, because they are not very mature processes in the version 1.0 specification, which does not define any usage scenarios that address communication across piconets. Nevertheless, the specification contains necessary considerations that could allow these processes to be accomplished.

The primary role of a master for a piconet is to define:

- which frequency-hopping sequence the members of this piconet shall follow;
- when frequency hops shall occur, thus defining the timing foundation for timed events in the piconet;
- which particular frequency is the "current" frequency; and
- which slave will be transmitted to and/or which slave will be permitted to transmit next (recall that transmission on the Bluetooth air-interface is through polling of the slaves).

The first three items are strongly associated with how a piconet is formed and maintained and how devices join it. The last item is associated with how transmissions occur in a piconet.

Figure 6.4 shows the operational states for a Bluetooth device. In the *connected* state, the device is a member of a piconet. On the other hand, when a device is not associated with any piconet or participates in no action that could result in its forming or joining a piconet, it is said to be in the *standby* state. The *standby* state is the default operational state for a Bluetooth device. In this state, a device typically idles with only its native clock operating in a low-power mode.

To move to the connected state, a device goes through the inquiry and page states, which are instantiated differently but in a complementary manner within a potential master and a potential slave. In the *inquiry* state, a device learns about the identity of other devices in its vicinity; these other devices must be in an *inquiry scan* state to listen for and subsequently respond to inquiries. In the *page* state, a device explicitly invites another device to join the piconet whose master is the inviting device; the other device must be in the *page scan* state to listen for and subsequently respond to pages. As Figure 6.4 shows, a device may bypass the inquiry state if the identity of a device to be paged is already known. The figure also implies that while a device is a member of a piconet, it may still perform inquiries and pages for additional devices to join this or some other piconet. In the latter case, a scatternet (multiple piconets overlapping in time and space as discussed in Chapter 2) eventually could be created.

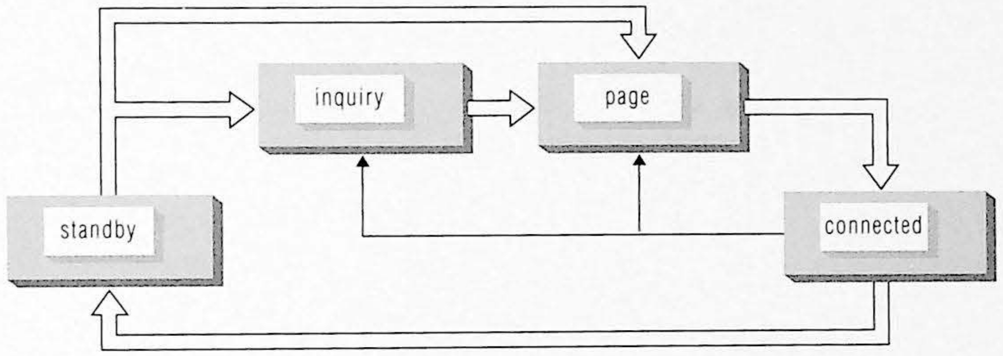

Figure 6.4
Operational states of a Bluetooth device.

To become a member of a piconet, a Bluetooth device needs to know how to recreate the frequency-hopping sequence that defines that piconet and which frequencies from this sequence will be visited and when. Also, to participate in communications over the piconet, the device needs to know how to formulate, read and write information packets on the piconet. All of these operations, as well as nearly every other operation in a Bluetooth device, are related to the following two *fundamental elements*:

- the Bluetooth device address
- the Bluetooth device (or native) clock

Any process in the Bluetooth baseband is intimately related to these two elements. Two baseband processes are notable and we refer to them as the *fundamental processes*, which are those that generate:

- the frequency-hopping sequence, and
- the access code.

These fundamental elements and fundamental processes are detailed below.

The Bluetooth Device Address (*BD_ADDR*)

The Bluetooth device address (*BD_ADDR*)[3] is the most static entity of a Bluetooth device. The *BD_ADDR* is a single 48-bit address electronically "engraved" on each device. A device's *BD_ADDR* is globally unique among Bluetooth devices. To guarantee uniqueness, a numbering authority assigns *BD_ADDRs*. The *BD_ADDR* is an IEEE 48-bit address, similar to the *medium access control* (MAC) address of IEEE 802.xx LAN devices.

The 48-bit address field, shown in Figure 6.5 from the least significant bit (LSB) to the most significant bit (MSB), is partitioned into three parts: the *lower address part* (LAP), the *upper address part* (UAP), and the *non-significant address part* (NAP). The 24 bits of the UAP and the NAP constitute the *organization unique identifier* (OUI) part of the address that is assigned by the numbering authority to different organizations. The LAP is assigned internally by various organizations. The various parts of the *BD_ADDR* are involved in nearly every operation of the baseband from piconet identification, to packet header error checking, to authentication and encryption key generation. These items are discussed in more detail later in this chapter.

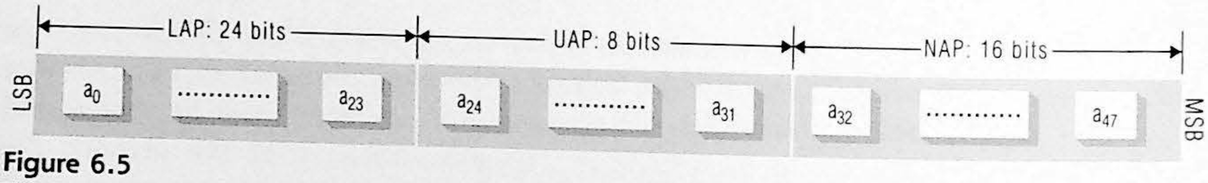

Figure 6.5
The Bluetooth 48-bit device address (*BD_ADDR*).

The Bluetooth Clock

Each Bluetooth device has a free-running 28-bit (native) Bluetooth clock. The Bluetooth clock is never adjusted and is never turned off. The clock ticks 3,200 times per second or once every 312.5 µsec, representing a clock rate of 3.2 KHz. Notice that this is twice the nominal frequency-hopping rate of 1,600 hops per second. The clock has an accuracy of ±20 ppm[4]. In low-power modes like standby, hold, and park (introduced in Chapter 2 and detailed later in this chapter), lower

3. The use of the notation *BD_ADDR* in the specification is the main reason for choosing the term "device" in Figure 6.1 over any of the other legitimate alternatives identified in footnote 1 of this chapter.
4. The abbreviation "ppm" stands for "parts per million" and it is a metric of clock accuracy, where the smaller the value the more accurate the clock.

power consumption can be achieved by using a low-power oscillator to drive the clock at a reduced accuracy of ±250 ppm. The Bluetooth clock is illustrated in Figure 6.6.

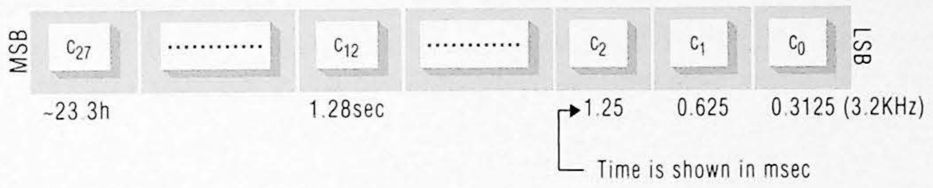

Figure 6.6
The Bluetooth native clock.

The Bluetooth clock wraps around just short of once per day. The Bluetooth clock plays a fundamental role in deciding when a device can or cannot transmit or listen for a transmission, and at which frequency and for what types of information packets it transmits or listens. A slave device uses the value of the Bluetooth clock of a master to accomplish piconet communications as discussed below. The significance of the time intervals shown in Figure 6.6 will become apparent as we proceed through the rest of this chapter.

The Frequency-Hopping Sequences

For devices to communicate with each other, they must transmit and receive on the same frequency at the same time. The *frequency-selection module* (FSM) contains the procedure for selecting the next frequency to be used under various operating conditions. The algorithmic and implementation details of FSM are beyond the scope of this book. They are covered extensively in chapter 11 of the Baseband part of the specification.

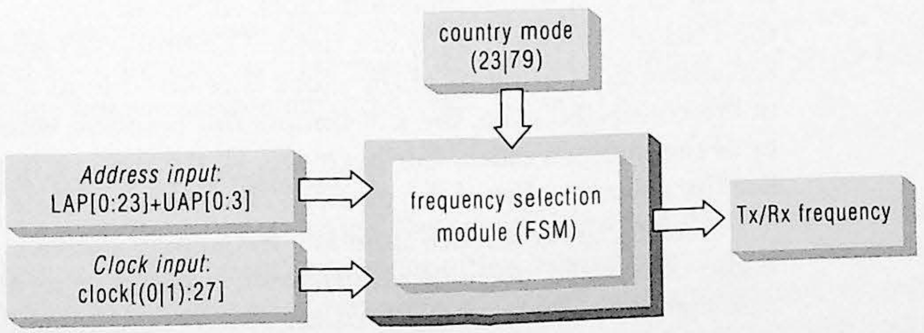
country mode (23|79)
Address input: LAP[0:23]+UAP[0:3]
Clock input: clock[(0|1):27]
frequency selection module (FSM)
Tx/Rx frequency

Figure 6.7
The frequency-selection module (FSM).

In Figure 6.7, the notation $(x|y)$ means that there are two permissible alternatives, x or y, only one of which is entered into the module based upon the circumstances. The notation $V[m:n]$, $m \leq n$, denotes the use of bits m through n of the bit field V; m is the least significant bit (LSB) of the two.

Depending upon the country of use, the FSM is set by the manufacturer to operate in a 23- or 79-channel frequency hop mode as explained in the radio section of this chapter.

Given the country frequency-hop mode, the *clock input* determines which frequency from the current frequency-hopping sequence is to be used, and when. In other words, the clock input determines the *phase* of the frequency-hopping sequence. The actual frequency-hopping sequence is determined through the *address input.* In each of the three active operational states shown in Figure 6.4, a different combination of clock and address inputs is supplied to the FSM, as explained immediately below.

Normal Piconet Operation

During normal piconet operation, the *channel-hopping sequence* is used. For the channel-hopping sequence, the address input to the FSM for all devices in the piconet consists of the 28 least significant bits, *BD_ADDR*[0:27], of the master's Bluetooth device address. The channel-hopping sequence has a very long period, but the (23 or 79) hop frequencies are distributed equally over short periods of time to satisfy the regulatory requirements for the frequency-hopping sequence described earlier in this chapter.

During normal piconet operation, a new frequency from the channel-hopping sequence is selected every 625 μsec. This interval is the period for bit c_1 of the Bluetooth clock shown in Figure 6.6; in this case, the LSB of the clock, c_0, is not used. The time, 625 μsec, between two successive ticks of bit c_1 of the clock is referred to as a *slot.* For devices in the connected state, the slot boundaries coincide with the ticks of bit c_1 of the master's clock. Furthermore, all the devices in the piconet utilize the current value of the master's clock to drive their FSMs.

During the residence time at a frequency, a device may transmit a single packetized piece of information, referred to as a *baseband packet data unit* (BB_PDU)[5]. BB_PDUs are strictly constrained within the resi-

5. The specification refers to the baseband PDUs as such or simply as packets. The BB_PDU nomenclature is introduced here for consistency with use of the terms PDU and packet elsewhere. Nevertheless, the term packet is used whenever appropriate for ease of reading.

dence time at a frequency. However, the residence time at a frequency may occupy multiple slots, thus permitting multi-slot BB_PDU transmissions to occur as discussed below. Following a multi-slot transmission, the next frequency selected is the one that would have been used if single-slot transmissions had occurred instead. That is, the frequencies from the channel-hopping sequence are selected on a slot basis, although the frequency hops themselves occur on a BB_PDU basis.

Page Operation

One device "invites" another device to join its piconet through the use of a page. The device issuing the page is called a paging device, and the device listening (or scanning) for pages is called the paged device. A paging device selects a new frequency at which to transmit a page every 312.5 μsec, which is the period of bit c_0 of a Bluetooth clock. During page scans, when a device listens for transmitted pages, a new listening frequency is selected every 1.28 seconds, which is the period of bit c_{12} of the clock. Note that a paging device changes frequencies at a much higher rate than a paged device. While the paged device uses its own clock to drive its FSM, the paging device uses its best estimate of the clock of the paged device to drive its own FSM. The paging device estimates the clock value of the paged device based upon the most recent communication between the devices. In the worst case, the paging device could use its own clock.

During the page operation, the *page-hopping sequence* is used. To generate this sequence, the paging and paged devices use the 28 least significant bits of the address of the paged device–that is, the LAP and part of the UAP of the address shown in Figure 6.5–as the address input to their respective FSMs. For each device, its page-hopping sequence is a well-defined, periodic sequence composed of 32 (resp.[6] 16) frequencies uniformly distributed over the 79 (resp. 23) frequency channels permissible in the 2.4 GHz band in various countries. The period of the page-hopping sequence is 32 (resp. 16) hops.

Inquiry Operation

Through inquiries a device "searches" for other devices in its vicinity. Just as in page operation, an inquiring device selects a new frequency at which to transmit an inquiry every 312.5 μsec. Inquired devices, executing inquiry scans, select a new listening frequency every 1.28 seconds.

6. The abbreviation "resp." stands for "respectively."

Note that an inquiring device changes frequencies at a much higher rate than an inquired device. Both the inquiring and inquired devices use their own clock to drive their FSMs.

During the inquiry operation, the *inquiry-hopping sequence* is used. To generate this sequence, the inquiring and inquired devices use the 28 least significant bits of the "inquiry address," referred to as the *general inquiry access code* (GIAC) LAP, as the address input to their respective FSMs. The inquiry address is a known, reserved 24-bit field equivalent to the 24-bit LAP of a Bluetooth address. The remaining four bits of the inquiry address (UAP[0:3]) are equal to hexadecimal '0x0'. The GIAC LAP is '0x9E8B33' and no Bluetooth device is allowed to have an address whose LAP coincides with the GIAC LAP. The inquiry-hopping sequence is a well-defined periodic sequence composed of 32 (resp. 16) frequencies uniformly distributed over the 79 (resp. 23) frequency channels permissible in the 2.4 GHz band in various countries. The period of the inquiry-hopping sequence is 32 (resp. 16) hops.

The Access Code

The access code is a 68- or 72-bit field prepended to each BB_PDU prior to its transmission over the Bluetooth air-interface. The access code serves a multitude of purposes including identifying the piconet, synchronizing on the incoming bit stream, aiding in establishing the proper DC-offset, and others. The focus here is the role of the access codes in identifying the state classification (inquiry, page, or connected) of transmitted BB_PDUs. The algorithmic and implementation details of the access code generator are beyond scope of this book. They are covered extensively in chapter 13 of the Baseband part of the specification. Figure 6.8 depicts an overview of the functions related to the access code.

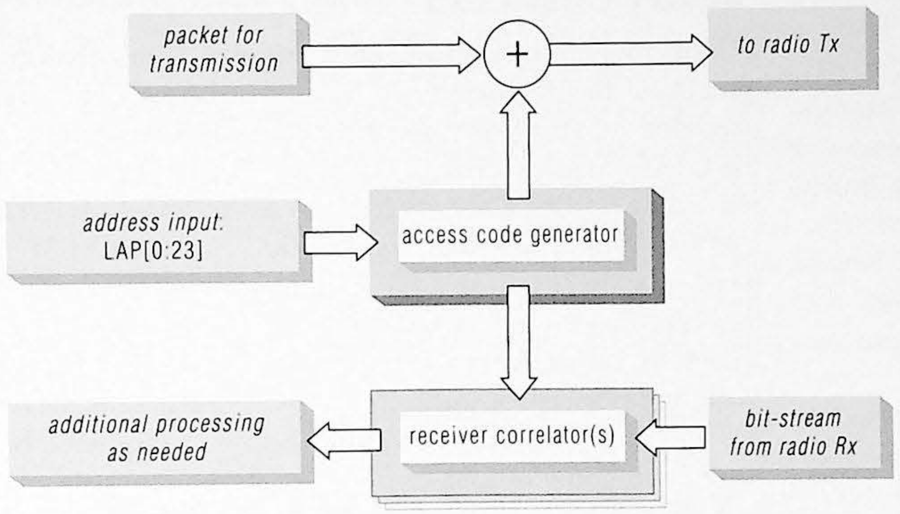

Figure 6.8
The access code functions; the "+" symbol above implies the prefixing of the access code in front of the transmitted packet.

To receive a transmission, a device uses a correlator, as shown in Figure 6.8, at its receiving end. The correlator can be tuned to particular access codes. If the correlator matches the access code of an incoming transmission sufficiently well, the receiver will continue receiving the incoming bit stream and pass it to higher layers for processing. Note that to conserve power, these higher layers may not be fully powered until the correlator informs them of an incoming message that has the proper access code prefixed to it.

As before, there are three classes of access codes that the device utilizes; each is described below.

Normal Piconet Operation

During normal piconet operation, each transmission on a given frequency of the channel-hopping sequence is preceded by the *channel access code* (CAC) generated using the LAP of the address of the piconet master. Only transmissions that contain the proper channel access code are received by a device.

Page Operation

During page operation, each paging transmission on a given frequency of the page-hopping sequence and each response to it are preceded by the *device access code* (DAC) generated using the LAP of the paged device's address.