call and adjust volume) was envisioned. As the telephony profiles (including those noted above along with dial-up networking and fax) were further developed, it became evident that a richer set of telephony control functions was desirable and a working group was formed to define these capabilities for the protocol stack.

With the initial recognition of a need for minimal audio control functions early in the SIG's history, it was at first supposed that these simple operations would be accomplished via AT commands using the RFCOMM serial port abstraction (recall that RFCOMM was fairly well defined even at this early stage). Thus was born the TCS-AT specification. This specification was intended to describe how standard AT commands could be mapped over the Bluetooth protocol stack and to define any new AT commands required for Bluetooth wireless communication. TCS-AT was designed to support legacy applications that send and receive AT commands over a serial port (most likely using a serial cable). TCS-AT of course specified the use of RFCOMM as the serial port replacement. As the specification progressed, it became apparent that there was very little need for any new AT commands specific to Bluetooth environments (only two new AT command responses were identified as being useful enough to propose specific definitions for Bluetooth TCS-AT). Thus the TCS-AT specification became a short reference that described how to use AT commands in the Bluetooth protocol stack, and its definition was absorbed into the profiles that use AT protocols (namely headset, fax and dial-up networking).

In the meantime a binary, packet-based telephony control protocol was also being defined within the Bluetooth protocol stack. Called TCS-Binary (or TCS-BIN), it was adapted from an existing ITU-T specification, Q.931 [ITU98]. As in other cases, the SIG's adoption of existing standards provided benefits for the protocol stack, in this case including the capability for robust telephony control operations in a standardized manner. In early 1999 it was observed that the likely future direction for telephony control applications was along the lines of the TCS-BIN (ITU-T) style, and it was further observed that TCS-BIN provided all of the functions necessary for all of the telephony-based profiles. Finally it was also observed that the TCS-AT specification did not provide significant new functions specific to Bluetooth environments and primarily specified a method by which legacy applications might use standard AT commands over RFCOMM as a means of cable replacement. Thus TCS-BIN subsumed TCS-AT as a separate protocol in the stack. The SIG decided to remove TCS-AT as a separate specification, although the functions were not removed; only the name was.

Thus the version 1.0 specification does not mention TCS-AT,[1] although several applications in fact do use RFCOMM as a serial transport for AT commands in cases where a modem service supports such a configuration. Indeed, the headset, fax and dial-up networking profiles use AT command telephony control. With only TCS-BIN being explicitly mentioned in the specification, all further references to TCS herein imply TCS-BIN.

The TCS Protocol Examined

In addition to what the specification calls TCS supplemental services (including caller identification information and dual tone multi-frequency [DTMF] tone generation), TCS defines three major functional areas:

- Call control
- Group management
- Connectionless TCS

Each of these is explored below. The majority of the more than 60 pages of specification devoted to TCS deals with the detailed syntax and semantics of TCS-BIN, which are not reproduced here. Instead we highlight some of the important features and nuances of TCS-BIN in the protocol stack.

TCS Call Control

The TCS call control functions serve to set up calls that subsequently will carry voice or data traffic. TCS acts as a state machine, performing the operations necessary to progress a call from one state to the next, and tracking the resulting state. When making calls, these operations might include such things as setting up the call, including dialing information; establishing and confirming a connection; and disconnecting when the call is complete. For received calls, the states and transitions include call presence (ringing), call acceptance and connection establishment and termination. Much of the TCS chapter of the specification is devoted to a full explanation of these states and their transition operations; the appendix to the TCS chapter of the specification details these states and transitions in comprehensive state diagrams.

1. Actually there is one "leftover" reference to TCS-AT in the Bluetooth Assigned Numbers appendix of the specification, the last remnant of TCS-AT's former existence as a separately described protocol. As defined there, the value could be used to indicate a device's support for AT command telephony control.

The telephony control functions can operate not only in a point-to-point network topology but also in a point-to-multipoint configuration. The multipoint environment is relevant, as pointed out in the specification, for incoming calls when numerous phones all need to receive the incoming ring signal and control information. In this case, TCS uses multipoint signaling to alert all the telephones of the incoming call; it can then establish a single content channel (where the voice or data traffic will flow) with the telephone that answers the call.[2] TCS does not deal with the content that is subsequently streamed over the channel but only with the call control functions that occur on the control channel.

Unlike RFCOMM, in which a single instance of the protocol layer is multiplexed, the specification indicates that multiple instances of TCS may be executed at the same time to handle multiple calls (recall that Bluetooth wireless communication permits up to three voice channels simultaneously over the baseband). Multiple instances of TCS simply use multiple L2CAP channels.

TCS Group Management

Group management functions use the concept of a *wireless user group* (or WUG). Such a group can use the TCS group management functions to allow for groups of devices to take advantage of some special functions that TCS enables. These functions include a method for one device to make use of the telephony services of another device in the group; a way to manage group membership (called *configuration distribution*); and a way for two slave members of the group to use the TCS protocol to establish a direct connection (called *fast intermember access*).

Group management is useful in telephony applications to enable the provision of the sorts of telephony functions that many users expect, such as multiple telephone extensions, call forwarding and group calls. In addition, group management can help to accomplish parts of the three-in-one phone profile by permitting phones to join a WUG (thus enabling a cellular phone to be used as a cordless phone) and to directly communicate with other TCS devices (thus permitting the intercom or "walkie-talkie" function).

A WUG is just a group of devices that all support TCS. The specification makes special provisions for security within the WUG by allowing

2. The need to transmit ring signals simultaneously to multiple telephone handsets was a primary motivation for including group abstraction and management and connectionless channels in L2CAP. These features could certainly be utilized in other future scenarios, but in version 1.0 they are used only in the context of TCS-BIN.

the WUG master to distribute keys used specifically for communications within the WUG, including communication with the master and separate communication (using a different key) with other WUG members as is done with the fast inter-member access described below.

One device in a WUG can request to use the telephony services of another device in the WUG; TCS calls this an *access rights request.* A handset might request the use of the telephony services of a base station to make a call, or an access rights request might be used to transfer a call from one TCS device (such as a handset or headset) to another.

Configuration distribution is the TCS-BIN method for managing the membership of the WUG. Again using the concept of a WUG master that maintains all of the information about the WUG, TCS-BIN defines a protocol for the WUG master to send updated WUG configuration information to each WUG member, each time that configuration information changes. For example, this might be used to inform all WUG members that a new member has joined (or that some member has left) the WUG. Among other applications, this feature could be used to support the three-in-one phone profile by advising WUG members (perhaps stationary handsets and base stations in a home) that a new member (say, a mobile phone brought into the home) has joined the WUG. Thus the mobile phone's presence is known and it can contact the base station (to act as a cordless phone) or it could directly contact other phones in the WUG (to act as an intercom).

Fast intermember access is a facility by which any two WUG members can quickly establish a connection with each other. This feature makes use of the fact that two members already belong to a WUG and have already established connections with a common WUG master. Thus all WUG members are already in a single piconet, all using the same hopping sequence established by the WUG master's clock. Furthermore, via the configuration distribution noted above, all WUG members can know about all other WUG members. Because all of this information is already known, it can be leveraged to establish a connection with another WUG member more quickly than such a connection could be established from scratch. With fast intermember access, a WUG member uses the configuration information to determine another member with which it wishes to establish contact. It forwards this information to the WUG master, which in turn contacts the target WUG member. That member then responds to the WUG master, includes its own clock offset information in the response, and then places itself into a page scan state. The master forwards the clock offset information to the requesting WUG member, which can then very quickly use this

information to establish a connection with the target member by paging that member (which is now in page scan state to accept such pages), the result being a new piconet, consisting initially of the two devices. This scheme takes advantage of the other features that are already in place for a WUG to enable quick direct connection between any two devices in that WUG to support, for example, the "walkie-talkie" function of the three-in-one phone profile.

Connectionless TCS

Finally, TCS-BIN also provides a way for devices to exchange call signaling information without actually placing a call or having a TCS call connection established. This is called connectionless TCS. Connectionless TCS provides a sort of "sideband" in which devices within a WUG can send messages to each other without having to have a TCS connection established between them. What sort of messages might these devices want to send? The specification defines only a single message format for connectionless TCS called *CL Info*. CL_Info messages in turn can contain only two types of information: audio control, used to specify information about microphone gain and speaker volume settings, and company information, which is the common TCS way to allow any information not specified in a standardized TCS format to be interchanged. Thus it can be seen that connectionless TCS could be used to manage the audio settings of all members of a WUG as well as to communicate product-specific features, defined by the manufacturer, among all of the devices from that manufacturer in the WUG. Such use of connectionless TCS might allow, for example, advanced telephony features to be used between a base station and a handset from the same manufacturer that might not be available to handsets from another manufacturer (although through standard TCS operations the basic telephony functions would be expected to work within the WUG with all handsets).

Bluetooth Audio Development

There was no audio working group per se within the SIG. Audio has been an inherent part of Bluetooth wireless communication since its inception and thus has always been integrated into the fundamental design of the protocol stack. Audio (voice or other audio) is carried over SCO links at the baseband layer. These basic SCO links were already defined early in the SIG's history, shortly after it was publicly

announced (the addition of multiple SCO connections to support multiple voice channels was introduced in mid-1998).

This evolution of Bluetooth audio mirrors its situation within the stack: it is not a distinct protocol layer but rather a fundamental part of the technology. Audio essentially is integrated into the baseband. Owing to a few specific considerations for audio in the protocol stack we discuss it as a separate topic. And as noted above, due to audio's affinity with telephony, for pragmatic reasons we discuss it in this chapter.

Bluetooth Audio Examined

A quick scan of the specification searching for audio information is likely to locate only Appendix V, "Bluetooth Audio." This appendix contains information interesting mostly to audio and sound engineers, including such things as recommended sound pressure, loudness and audio levels. Although important, it is not the fundamental information about how to deal with Bluetooth wireless audio traffic. That information, as might be expected from preceding discussions, is actually found in the "Bluetooth Audio" section of the Baseband chapter of the specification.

While audio in Bluetooth wireless communication need not be used exclusively for voice, its design is optimized for voice content. Sound tends to be continuous for periods of time and is thus isochronous, or time limited. The transmission rate for Bluetooth audio traffic is set at 64 Kbps, chosen to be sufficient for normal voice conversations. While the communication of other audio media (say, music) over Bluetooth audio links is not precluded, the design is not based upon such audio traffic; it clearly is centered around voice traffic.

Two types of encoding schemes are specified for Bluetooth audio. The first is pulse coded modulation (PCM) with either of two types of logarithmic compression (called A-law and μ-law) applied. PCM audio with these compression types is well known and widely used for general audio, including things like short sound clips. The second audio encoding scheme is continuous variable slope delta (CVSD) modulation. The characteristics of typical voice conversations, which have a more predictable continuity than general audio (music, for example), make a delta-slope prediction more efficient. CVSD generally is also more tolerant of communication errors. Thus CVSD, in general, is a more effective and efficient (and thus generally preferred) method to use for Bluetooth audio communication; we observe once again that this is an optimization for voice versus other forms of audio.

The specification says little else about audio as a topic unto itself. The remainder of what needs to be specified (and what implementers and others may wish to understand) about audio can be gleaned from a study of the baseband protocol, including the SCO packet structure and timing designed specifically to support audio traffic simultaneously with data traffic. This information is found in the baseband chapter of the specification and in Chapter 6 of this book.

One final note about audio: it should be clear that Bluetooth audio as described here and in the specification is digital isochronous (effectively streaming) audio traffic that operates directly over the air-interface using the baseband protocols. Of course audio information can also be encoded in a digital packet-based format[3] using local recording and playback. Such digital audio information clearly could be transmitted over Bluetooth links using the L2CAP layer of the protocol stack, but this is quite different from what we refer to here as Bluetooth audio.[4]

Audio and Telephony Control Usage

Several families of telephony applications are possible. TCS-BIN is intended to support applications that realize the Bluetooth telephony-based profiles: cordless telephony and intercom. These are the only two profiles technically classified as telephony profiles, based upon their usage of TCS-BIN. Such applications are expected to use TCS-BIN directly, as depicted in Figure 10.1.

Other sorts of applications also might be considered in some respects to be telephony applications; these include dial-up networking, fax and headset profile applications. In volume 2 of the specification these profiles are considered to be part of the serial port profile family; this is because the telephony facets of these applications tend to use the programming model of AT commands over a serial port (RFCOMM in the Bluetooth wireless communication case), as described earlier in this chapter. Although not TCS-BIN based, we also consider these applications in general to be telephony applications, and they are depicted in Figure 10.1 as such (that figure shows telephony applications using both TCS-BIN and AT commands over RFCOMM).

3. Such as WAV and many other fundamentally similar representations.
4. For one thing, it is not truly isochronous, at least not in a streaming, over-the-air fashion. For another, most encoding schemes for such digital packet audio are designed so that many types of audio (music, sound clips and so on) can be effectively rendered, rather than optimizing the audio content for one primary use such as voice.

Legacy applications are likely to use AT command telephony control, since this is the typical programming model in the world of serial cables. New applications developed specifically to make use of Bluetooth wireless links, though, are encouraged via the specification to make use of the TCS-BIN protocol, which provides a robust set of telephony control functions based upon an existing standard that has been adapted for the Bluetooth stack.

Telephony and audio, particularly voice audio, play important roles in the Bluetooth stack. With their associated applications (which may involve both computing and telecommunications devices in some profiles and usage models), they provide a distinguishing feature of Bluetooth wireless communication.

Part 3

THE BLUETOOTH PROFILES EXAMINED

Having examined the Bluetooth core specification in Part 2, we continue in Part 3 with an examination of volume 2 of the specification, known as the profiles. We begin in Chapter 11 with an overview of the profiles and their interrelationships, including the rationale for our own grouping of the profiles in this book. The remaining chapters then explore each of the published profiles in logically related groups. Chapter 12 discusses the fundamental generic access and service discovery application profiles. Chapter 13 explores the profiles that deal with telephony functions (cordless telephony, intercom and headset), while Chapter 14 describes the family of serial port related profiles (basic serial port, object push, file transfer and synchronization). Finally Chapter 15 examines the profiles related to networking, namely the dial-up networking, LAN access and fax profiles.

Part 3, like Part 2, is designed to summarize important information from the Bluetooth profile specification, making that information more accessible and understandable. Drawing on our experience in the Bluetooth SIG, we attempt here to expose the motivation and rationale for key elements of volume 2 of the Bluetooth specification.

11

The Bluetooth Profiles

Volume 2 of the specification consists of the profiles. These are intended to promote interoperability among many implementations of the protocol stack that is specified in volume 1. In this chapter we discuss the general nature of the profiles–the rationale for generating them, their history and evolution and the interrelationships among them. Each of the version 1.0 profiles is then examined in more detail in subsequent chapters.

The Version 1.0 Profiles

Profiles spring from usage cases. Chapter 3 discussed the usage scenarios that were part of the development of the version 1.0 specification (although not all usage cases resulted in a corresponding profile for version 1.0). Many of the profiles can be grouped together based upon the shared elements of their usage scenarios; this is how the profiles are grouped into chapters in this part of the book. The profiles also can be grouped into families based upon their technical underpinnings; in many cases these map in a straightforward manner to usage scenario relationships, although in other cases the technical relationship (for example, of fax to headset) is not so obvious. Furthermore, the version 1.0 specification contains additional profiles that do not directly embody usage cases. These include the generic access profile, the service discovery application profile, the generic object exchange profile and the serial port profile. Each of these profiles can be considered a *transport profile* which defines a common basis of shared characteristics

upon which other *application profiles* can be built (or, in object-oriented terminology, these profiles are classes from which other profiles, or subclasses, inherit).

Figure 11.1 depicts the version 1.0 profile families based upon their technical relationships. A similar figure appears in multiple places in volume 2 of the specification. That figure depicts these relationships in a "nested container" sort of representation, while Figure 11.1 uses an object hierarchy representation, but both diagrams convey the same information. The abbreviations in parentheses in the figure introduce the shorthand nomenclature that is used in the remaining chapters of Part 3 to name the profiles. These abbreviations are used for convenience; in some cases (but not all) they are consistent with similar abbreviations for the profiles that are used in the specification.

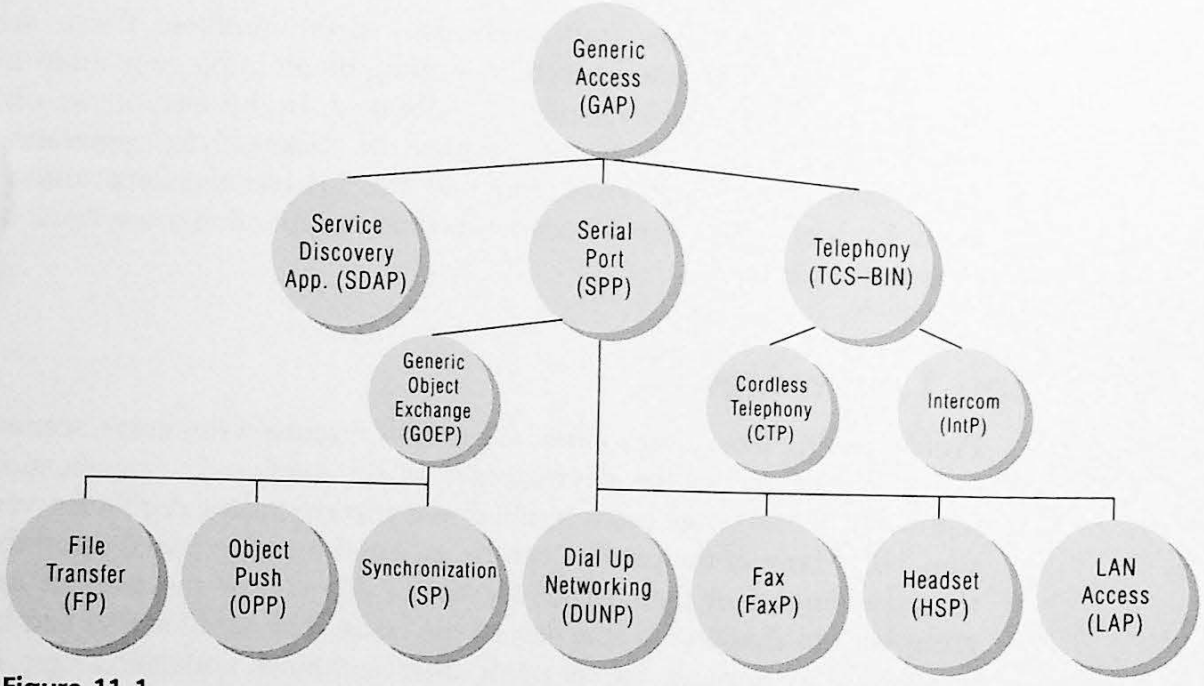

Figure 11.1
The Bluetooth version 1.0 profile families, based upon protocol stack relationships, in class hierarchy representation.

This chapter describes each of the profiles according to the relationship shown in Figure 11.1. The remaining chapters of this part of the book, though, examine the profiles in logical groupings based upon the relationships of the services that each profile provides. There are multiple ways of viewing several of the profiles and thus there are multiple ways to group the profiles. This chapter presents one such grouping,

that of the technical elements shared among profiles (which is consistent with the version 1.0 specification, although it does not propose any overt grouping at all other than including a different representation of the figure shown above). Figure 11.2 depicts the logical, service-based grouping that we use in discussing these profiles in subsequent chapters. One example of these multiple viewpoints is that of the headset profile. As depicted in Figure 11.1 in the profile hierarchy, headset is a derivative of the serial port profile. However, from a functional or services point of view the headset profile could be considered to be part of the telephony profile group, as shown in Figure 11.2, and thus we include it in Chapter 13 along with cordless telephony and intercom. Another example is that of the fax profile, which from a technical perspective is also a serial port profile derivative. Our treatment of the fax profile in this book, though, is alongside that of the dial-up networking and LAN access profiles, as part of what we consider a "networking" category,[1] a group which the specification does not address.

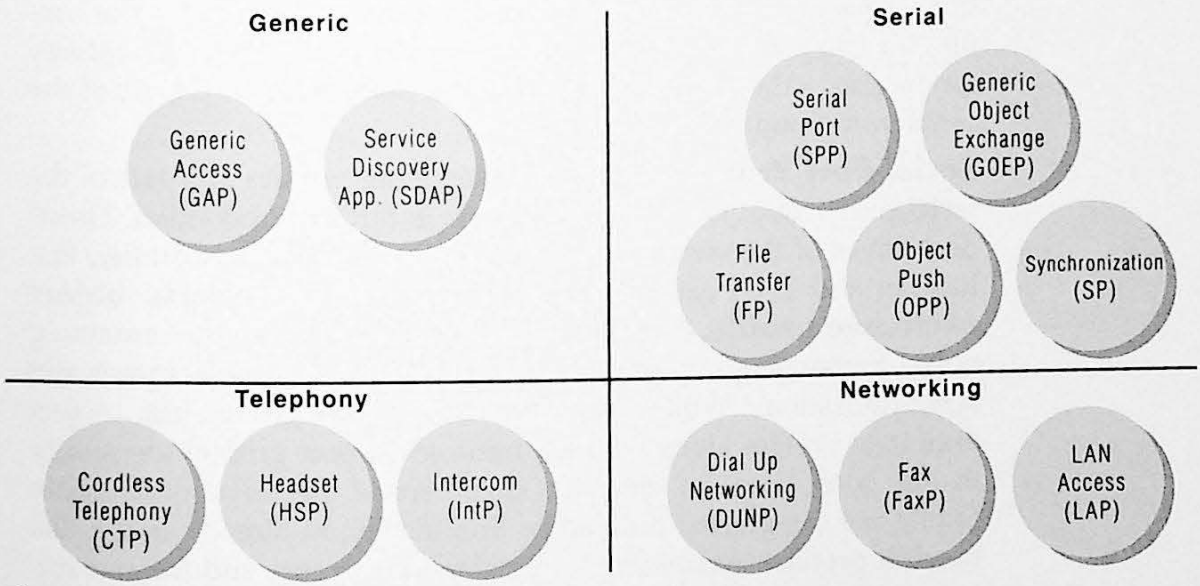

Figure 11.2
The Bluetooth version 1.0 profiles, based upon logical service-based groupings.

1. While it may not be obvious how a fax profile fits within a networking category, consider that the fax usage scenario, like dial-up networking and LAN access, uses a Bluetooth device as a "gateway" to a wide area network for data communication. Chapter 15 further elaborates this point.

This discussion and the figures are intended only to point out that there are multiple ways in which to categorize the version 1.0 profiles. The choice of which method to use is probably best made depending upon the purpose and circumstances surrounding the need to group the profiles in the first place, and most distinctions are not likely to be drastically different. However, because the number of profiles is expected to grow over time, with the SIG already planning to add quite a few new ones (as discussed in Chapter 16), the act of grouping the profiles becomes more advantageous, and the different ways of doing so should not be confused.

Generic Profiles: This group is composed of the generic access profile, which is the root of all profiles, and the service discovery application profile, which provides a framework for mapping from the application layer to the SDP layer of the stack.

Telephony Profiles: In the object hierarchy view, these are the profiles that imply the use of TCS-BIN for telephony control functions. This group includes the cordless telephony and intercom profiles (in Chapter 13 we consider the headset profile in the telephony group also, although from the technical perspective it is part of the serial port group).

Serial Port Profiles: All of the remaining profiles are part of the serial port group. However, this group is further subdivided. Direct derivatives of the serial port profile are the dial-up networking, fax, headset and LAN access profiles; as well as the **generic object exchange profile**. The latter in turn is the parent of the remaining object exchange group profiles, namely file transfer, object push and synchronization. While we treat this latter group (the object exchange profiles along with the basic serial port profile) identically in this book, some of the other members of the serial port profile family are categorized differently. Specifically, as already noted, the headset profile is examined in the telephony group, and the remaining serial port profile family members–fax, LAN access and dial-up networking–are treated as a networking family, which is examined in Chapter 15.

Part 3 Chapter Organization

In each of the remaining chapters of Part 3 we examine a group of profiles as described above. Each chapter is organized such that it addresses:

- the common elements of the profiles in that group;
- the history of each profile; and
- an examination of each profile, including how it maps to the protocol stack and how applications might use it.

In addition, wherever possible, we offer insight about the rationale, design thought process and future possibilities of the profiles based upon our participation in the SIG when the profiles were being developed.

12

The Generic Profiles

Our examination of the profiles begins with two that we call the *generic profiles* because they are fundamental to Bluetooth wireless communication. The *generic access profile*, or GAP, is the basis for all other profiles. It describes the fundamental operations necessary for two Bluetooth devices to establish communication, including the discovery of devices, link establishment and configuration and security considerations. The GAP is tightly coupled with the group of Bluetooth transport protocols described in Chapters 6 and 7. The *service discovery application profile*, or SDAP, describes fundamental operations necessary for service discovery, an operation often performed soon after a communication link is established. The SDAP maps directly to the service discovery protocol (SDP) described in Chapter 8.

Most Bluetooth devices are not expected to implement all of the profiles. A data access point, for example, probably would not implement telephony profiles, while a headset device is unlikely to implement networking profiles. However, nearly all devices are expected to implement both the GAP and SDAP; in fact, it is mandatory for all devices to comply with the GAP. Also, the use of SDP over the group of Bluetooth transport protocols follows the corresponding guidelines outlined in SDAP. These two profiles do not describe a specific usage case per se but rather define basic functions that are necessary (or at least highly desirable) for all devices.

Relationships

The relationship between these two profiles may not be evident upon a cursory examination of the specification. Each deals with considerations that are mostly relevant to different layers of the protocol stack. The GAP largely addresses lower-layer protocols involved in the most basic Bluetooth communication functions, while the SDAP focuses primarily on items of interest to higher-layer applications. However, these two profiles also share many traits.

Neither the GAP nor the SDAP addresses a specific usage case. Many of the other profiles, like synchronization or cordless telephony, deal with specific, concrete end-user scenarios. For the most part, the topics covered in the GAP and SDAP are not visible to an end user (with some of the application interactions excepted), but these profiles define procedures and protocol usage that are necessary to accomplish all of the other profiles. Indeed, this is why the GAP is the basis for all other profiles and why its inclusion in Bluetooth devices is mandatory. Similarly, the SDAP defines methods for exercising protocols for the purpose of service discovery, and service discovery is a component of nearly all of the other profiles. In some respects, the GAP and SDAP both define "invisible" operations that are necessary for all other profiles and thus enable those other profiles.

Even though the GAP and SDAP are concerned primarily with low-level communication functions, each also has an application facet and thus some degree of visibility to an end user. While much of the interaction these profiles specify can occur in an automated fashion, mostly hidden from the user, some operations might involve the end user as well. Examples include "browsing" applications[1] that present information about devices and services within proximity, and the presentation to the user of various options available, such as the degree of security to be used for a particular communication link. In such cases a user interface or application programming interface could be associated with these profiles. Nevertheless, the device and service discovery and connection management is always performed in accordance with the specifications of the GAP and SDAP.

The overall profile creation process started late in the development of the specification, when it was realized that application interoperability could be best achieved through a formally specified way to facilitate it. Moreover, the GAP and SDAP were the last profiles to start

1. One example is the "Bluetooth Piconet Minder" application described in Chapter 8.

being developed. The reason for their late start is that they both establish a basis for all other profiles, and hence their need became apparent only after common patterns emerged within the other profiles. The SIG realized that the Bluetooth community would be served better if these common patterns and procedures were grouped into separate documents for ease of reference. This reasoning led to the development of the serial port and object exchange profiles as well. More details on the development of the generic profiles are given within each profile presentation, starting with the GAP discussion that follows.

The Generic Access Profile

The Generic Access Profile, or GAP, forms a common basis for all Bluetooth profiles and hence for the common interoperable usage of the group of Bluetooth transport protocols. One of its important contributions is its definition of a standard set of terminology. Chapter 8 of the GAP contains four pages of standard terms with crisp definitions. Having this common vocabulary helps to remove ambiguity in all of the profiles, which is a key element in enabling interoperable implementations–and interoperability is the overarching goal of the profiles in the first place.

With this common terminology in place, most of the GAP is devoted to defining the procedures necessary to establish Bluetooth connections. This includes device and name discovery, inquiry procedures, pairing and bonding (explained below), and link, channel and connection establishment. For all of these considerations, the GAP provides common and standard procedures, in some cases including flowcharts. The importance of defining the fundamental communication operations cannot be overstated: without well-defined, interoperable methods for basic communication between devices, none of the other profiles could be realized.

GAP Development

Because the GAP is the root of all of the profiles, it is natural to assume that it was the first profile developed. In fact, the opposite is true: it was one of the last to be defined by the SIG. To understand why, one must understand the history of profile development in the SIG.

As described in Chapter 1, the SIG's software working group was organized into task forces that focused on developing one protocol or a

set of related protocols. In January 1999 the topic of profiles was first discussed in depth within the SIG. Profiles were suggested, and later adopted, as a means to formalize the Bluetooth usage scenarios and to ensure interoperability among multiple implementations of the protocol stack. Thus the initial set of profiles was based upon the same usage cases that drove the marketing requirements document, which in turn drove the development of the protocols. The responsibility for developing the profiles initially fell to the same task forces that developed the corresponding protocols. For example, the headset and cordless telephony profiles were developed by those who defined the TCS-BIN protocol; the serial port profile was written by the same people who defined RFCOMM; and the object push and synchronization profiles were developed by the group that specified the IrDA interoperability protocols. Later the SIG formed an interoperability working group, separate from the software working group, to focus exclusively on the profiles and related issues, although the participants in the interoperability working group in many cases were still those who helped to develop protocols in the software working group.

In the months preceding the creation of the interoperability working group and the efforts to develop usage-scenario-oriented profiles, an activity began within the software working group to develop standardized *man-machine interfaces* (MMI). An MMI task force, chaired by author Bisdikian, was formed in November 1998. Its goal was to enhance the user experience and interaction with Bluetooth devices through standardized usage and connectivity procedures, nomenclature, and graphic user interfaces (where appropriate), following the paradigm of GSM cellular phones. However, the variety of devices that could be capable of Bluetooth wireless connectivity far exceeds the variety of GSM phones; hence the SIG realized that the development of standardized procedures for all conceivable Bluetooth devices could be too restrictive. Thus, with the creation of the interoperability working group, the activities in the MMI task force merged with those of the interoperability group.

As the initial set of profiles progressed and matured, it became evident that each of the profiles made assumptions about underlying transport protocol layers. Because there was no end-user scenario that dealt specifically with low-level communication issues it was not initially evident that a profile was needed to address those topics. Meanwhile the SIG had begun to discuss security issues in earnest. These two developments resulted in the realization by the SIG that a profile was needed to address the common communication elements. By May 1999 the

framework of the generic access profile was in place. Considering that the specification was published the following July, it should be evident that a great deal of work was required in a short amount of time to complete the GAP.

GAP Examined

The GAP is concerned principally with three items: *dictionary*, *connectivity*, and *personalization*. The dictionary consists of a collection of terms and their definitions. These terms appear in the specification, both in its core and profile parts, and the dictionary provides the foundation for their unambiguous use throughout the specification. Connectivity consists of the operations performed by devices that allow them to connect, or not, and authenticate, or not, with other devices. Personalization consists of the elements that identify and customize Bluetooth devices, like their user-friendly names and PINs. For the last two items, the GAP provides terms that can be exposed at the user-interface (UI) level, whenever applicable.

This chapter focuses on the connectivity aspects of the GAP, which are used by all other profiles. They include the connectivity modes, the security modes, and idle mode procedures. The GAP also has a section on link establishment procedures that summarizes the sequence of operations used to establish Bluetooth links and L2CAP channels; these procedures are highlighted in Chapters 6 and 7 and thus are not discussed further here.

Connectivity Modes

As described in the baseband portion of Chapter 6, a device may enter inquiry scan mode or page scan mode, either to be discovered by other devices that transmit inquiries or to be connected with other devices that transmit pages to the device, respectively. The baseband specification does not state the conditions under which a device performs inquiry and page scans and thus it does not specify when a device allows itself to be discovered or connected. The HCI specification, described in Chapter 7, includes HCI commands sent by a host to the host controller, and from there to the link manager and link controller, that instruct the latter to enter the various inquiry and page states. Thus, it becomes a matter of a user-level (or, more generally, application-level) defined policy as to when a device enters any of these states. Similarly, a user-level defined policy also determines whether or not devices shall *pair* (authenticate) with each other. This is an important

point: user-level decisions determine the degree to which a given device can be discovered, connected to and paired. One concern often expressed about Bluetooth technology is that all Bluetooth devices will automatically communicate with each other at any time, but this is a misconception. Users, or user-level applications, set the connectivity policies that determine which devices can communicate with each other, and when. These policies could be fixed by the manufacturers of Bluetooth devices or could be configurable by their users. Thus, device manufacturers could use the connectivity policies as a way to differentiate their products.

The GAP defines the policies for device communication establishment and categorizes them into *discoverability* modes, *connectability* modes and *pairing* modes.

Discoverability Modes

A Bluetooth device is said to be *discoverable* if it allows itself to be discovered by other Bluetooth devices. In particular, a discoverable device executes inquiry scans regularly and responds to inquiries sent by inquiring devices. There are three levels of device discoverability:

1. *General discoverable mode:* At this discoverability level, a device enters inquiry scans using the general inquiry access code (GIAC), which is the inquiry access code (IAC) generated from the specially reserved lower address part (LAP) '0x9E8B33' of the 48-bit Bluetooth address, as described in Chapter 6. In this mode, a device responds to all inquiries and thus it always can be discovered by all other inquiring devices.
2. *Limited discoverable mode:* At this discoverability level, a device enters inquiry scans using the *limited inquiry access code* (LIAC), which is the IAC generated from the specially reserved LAP '0x9E8B00'. In this mode, a device may respond only to the inquiries that contain the LIAC and thus it may be discovered only by other devices inquiring using the LIAC.
3. *Nondiscoverable mode:* At this discoverability level, a device does not respond to inquiries and thus other Bluetooth devices cannot discover it.

When a device is discoverable, it must always enter the general discoverable mode, even if it enters the limited discoverable mode. The latter mode may be entered in parallel with or sequentially to the general discoverability mode, as described in Chapter 6. A discoverable

device must enter inquiry scans no less frequently than every 2.56 seconds and must remain in inquiry scan for at least 10.625 milliseconds.

Connectability Modes

A Bluetooth device is said to be *connectable* if it allows itself to create Bluetooth links with other devices. In particular, a *connectable* device executes page scans regularly and responds to pages sent to it by paging devices.

A *nonconnectable* device does not respond to pages and thus it cannot create links with other devices.

The discoverability and connectability modes might be set independently of each other[2]; however, a device that is only discoverable and not connectable may not be very useful.

Pairing Modes

A Bluetooth device is said to be *pairable* if it allows itself to be authenticated by another Bluetooth device, meaning that it can play the role of a claimant during an authentication transaction. Furthermore, a pairable device, in addition to accepting *LMP_au_rand* PDUs, must accept an initial authentication request received from a verifier in an *LMP_in_rand* PDU, as discussed in the LMP section of Chapter 6.

A *nonpairable* device responds to an *LMP_in_rand* PDU with an *LMP_not_accepted* PDU, signifying that the device is not willing to pair with any new devices.

Security Modes

Security operations in Bluetooth devices ultimately relate to device authentication and possibly link encryption. Recall that the former is a mandatory feature of Bluetooth devices while the latter is not.[3] Three levels of security relate to the "depth" of the security safeguards imposed upon communicating devices:

1. *Security mode 1:* A device that operates in this mode does not have any security barrier. In particular, it never acts as a verifier and thus never sends *LMP_in_rand* or *LMP_au_rand* PDUs.
2. *Security mode 2:* A device that operates in this mode places a security barrier at the L2CAP layer. In particular, it does not initiate any security transaction prior to receiving a request to establish an

2. There is a host controller interface command that enables inquiry and page scans independently of each other.
3. However, some profiles do mandate support for and the use of encryption.

L2CAP channel using the *L2CAP_Connection_Request* signaling command, as described in the L2CAP section of Chapter 7. This security mode allows a flexible security model for Bluetooth devices in which security barriers in a remote device can be raised based upon the particular service that a local device requests from the remote device.

3. *Security mode 3:* A device that operates in this mode places a security barrier at the link manager layer. In particular, it does not initiate data communications involving the upper transport and higher layers prior to authenticating the device with which it is to communicate. In other words, authentication occurs before the transmission and receipt of the *LMP_setup_complete* PDU, as discussed in the LMP section of Chapter 6.

Note that a device may be in one and only one security mode at any given time. For example, a device in security mode 3 cannot authenticate other devices selectively; instead it authenticates all devices that attempt to establish a link with it. A flexible security architecture for Bluetooth devices is described in [Muller99]. It forms the basis for the security modes included in the GAP.

Idle Mode Procedures

While the connectivity and security modes are associated with activities that a Bluetooth device follows to react to incoming stimuli (such as inquiries, pages, *L2CAP_Connection_Requests*, and so on), the idle mode procedures relate to the device that sends the stimuli. These procedures include general and limited inquiry, name and device discovery and bonding.

The general and limited inquiries are used to discover devices in general or limited discoverable mode, respectively. The device discovery process returns, among other things, the user-friendly name of discoverable and connectable devices. Note that requesting the name could involve just the LMP layers in two devices without involving the hosts.

Bonding is a pairing procedure executed for the purpose of creating a link key between devices and storing that key for future use. In *general bonding*, bonding is combined with additional communications such as accessing higher-layer services. In *dedicated bonding*, a device connects with a pairable device with the sole purpose of creating a bond between the devices without involving upper-layer transactions.

The GAP concludes with a brief description of a service discovery procedure used to search for services supported in remote devices. This

procedure follows the guidelines for service discovery found in the SDAP, which is presented next.

The Service Discovery Application Profile

As noted in Chapter 8, service discovery is expected to be a key component of most Bluetooth applications. Nearly all of the profiles include a service discovery element. Like the GAP, the Service Discovery Application Profile, or SDAP, provides a common and standard method for performing service discovery using the Bluetooth protocol stack.

Unlike most other profiles, the SDAP describes a standard service discovery application model and also defines abstractions of service primitives that in some respects resemble application programming interfaces (APIs). Even though the SDAP deals with the SDP middleware layer protocol and thus addresses some of the "invisible" operations described earlier, it is aimed primarily at application writers. It is the only profile with "application" in its title and the only profile to suggest API-like primitives. As explained in Chapter 8, these primitives could be mapped to platform APIs in a straightforward manner.

SDAP Development

Both authors have a special interest in the SDAP. Author Bisdikian served as editor of the SDAP portion of the specification, conceived the original idea for the SDAP and contributed most of its content, and author Miller chaired the service discovery task force responsible for delivering the SDAP.

As with the GAP, the need for an SDAP was not originally evident, and thus the SDAP was also developed late in the specification cycle. Not until January 1999, when most profiles were already underway, was the question raised regarding whether or not a service discovery profile was needed. By March of that year the idea of an application profile for service discovery was accepted and the SDAP development proceeded.

The development of the SDAP is rooted in the fundamental assumptions that led to the formation of the SIG itself: the diversity and number of devices that would be capable of Bluetooth wireless communication and the diversity and number of services available through these devices would steadily increase. To keep a semblance of order in the expected sea of devices and services available to a user, it was rec-

ognized that a standardized procedure should be created that would allow the user of a device to locate and identify services.

The SDAP does not describe how the service discovery itself is performed; it relies on SDP for this task. Rather, SDAP describes how an application that uses SDP should be created and should behave. In particular, it defines the functional characteristics of such an application through a set of service primitive abstractions, detailed below. Furthermore, the SDAP defines how other profiles and applications in general must use the group of Bluetooth transport protocols to carry SDP_PDUs when they need to execute SDP transactions. This latter item was an expansion of the SDAP's original scope. All of the "nongeneric" profiles contain an SDP section that provides a list of parameters for the protocol stack that leads to the particular application covered by the profile. These protocol stack parameters include ones like the RFCOMM data link connection identifier (DLCI) needed to reach, say, the PPP layer in the LAN access profile. These parameters are carried as service attributes within SDP_PDUs. However, these other profiles do not specify how the SDP layer could use the group of Bluetooth transport protocols to carry these SDP_PDUs. Since the latter process should be identical for all profiles, one idea was to include it in a generic profile like the GAP. However, the GAP does not focus on the transport of data with a source or sink above the L2CAP layer. Moreover, the SDP specification itself does not contain the dependencies of SDP on the group of Bluetooth transport protocols. Even though this may seem like an oversight, it was a deliberate choice. The Bluetooth service discovery protocol, although tied to systems that utilize Bluetooth wireless communication, is in principle a transport-independent protocol. Hence, the SDP specification focuses exclusively on the SDP transactions themselves and the various SDP_PDUs that are used, as well as the type and form of information that is carried in them. The SDP specification does not particularly focus on how these transactions are carried over the Bluetooth air-interface. Ultimately, SDAP became the only (and natural) point of reference for describing how the SDP layers use the group of Bluetooth transport protocols to carry the SDP_PDUs to each other.

SDAP Examined

The SDAP is unique among the application-oriented profiles, like the file transfer profile, the LAN access profile, and so on. These other profiles describe how the complementary parts of a user-level application

running on two (or more) devices work together to support a particular usage scenario. The application in SDAP needs to be present in only one device. This application interacts with the SDP layer of the stack in the device where it resides to initiate SDP interactions with one or more SDP layers in other devices, so as to learn about services in those other devices. Upon the arrival of responses from the other devices, the service discovery application can make those results available to the user of the device that initiated the transaction(s).

Often, service discovery in non-Bluetooth environments is performed by broadcasting inquiries for services or inquiries for locating directories of services. In the latter case, when a service directory is found, it is then contacted to find out about services that are registered there. In a Bluetooth piconet, broadcasts are entirely unidirectional in that they are exclusively directed from the master to the slaves of the piconet. Furthermore, broadcast transmissions are not recoverable in that they cannot be retransmitted following an error in their transmission. Thus, service discovery in a Bluetooth piconet does not use a broadcast model. Service discovery in Bluetooth piconets is closely associated with device discovery. Service discovery is executed only between fully identified pairs of devices and only after they have discovered each other and have created a Bluetooth link (up to and including an L2CAP connection) between them.

According to the SDAP, devices participating in service discovery may have either of the following roles:

- *Local device:* This device implements the service discovery application, like the service browsing application referred to in Chapter 8. It also implements the client portion of the SDP layer. A local device initiates SDP transactions as shown in Figure 8.3.
- *Remote device:* This device is contacted by a local device to inquire about services. A remote device implements the server portion of the SDP layer. It responds to SDP transaction requests from a local device. To produce its responses, the remote device maintains, explicitly or implicitly, a database[4] that contains service records for the services available via the remote device.

4. The specification does not define the format of this database, leaving that choice to implementers. Moreover, this need not be a "database" in the classic sense; it is simply a collection of information maintained by the SDP server in a format suitable for the device.

Even though some devices may act only as local or as remote devices, these device roles are, in general, temporary and meaningful only when an SDP transaction between two devices is under way. A device can be a local or remote device at different times or even at the same time, depending upon when it creates service inquiries or responds to them. The SDAP device roles bear no relation to the baseband roles of master and slave, as the latter roles are meaningless above the link manager layer. A local device could be either a master or a slave in its piconet, as could a remote device.[5]

The SDAP is the only profile that uses the term application in its title. However, the SDAP does not define any particular application. Such a definition would be very much platform dependent and possibly too restrictive for application developers, neither of which is a desirable objective for a specification. However, the SDAP specifies the services that a service discovery application should provide to its users to be useful. These services are summarized in four service primitive abstractions. These primitives could be mapped to an appropriate set of APIs based upon the underlying software and/or hardware platform in which an SDAP application is instantiated. An example mapping of these primitives to the Salutation APIs is given in [Miller99]. These primitives are:

- *serviceBrowse:* This service primitive is utilized when a local device wants to perform general service searches, referred to in Chapter 8 as service browsing. These searches might take the form of queries about what services in general or what services of type S are available, if any, via a selected set of remote devices. This application-level service primitive results in SDP_PDU transactions initiated by any one of the three basic request SDP_PDUs presented in Chapter 8: *SDP_ServiceSearchRequest*, *SDP_ServiceAttributeRequest*, or *SDP_ServiceSearchAttributeRequest*. Optionally, the *LMP_name_request* PDU could also be sent to learn the user-friendly name of the remote device.
- *serviceSearch:* This service primitive is utilized when a local device wants to perform searches for a specific type of service. The search could take the form of queries about what services of type *S* with attributes A1 and A2 are available, if any, via a selected set of remote devices. Similar to the previous primitive,

5. Often a local device acts as master of a piconet, since typically it would be the device that desires to create connections with other devices and search for and use services on them. However, this does not mean that a local device must be a master to perform service inquiries. A slave device could equally well initiate such inquiries.

this application-level service primitive results in SDP_PDU transactions initiated by any one of the three basic request SDP_PDUs previously mentioned.

- *enumerateRemDev:* This service primitive is used when a local device wants to search for remote devices in its vicinity. The search can be restricted with a class of device (CoD) qualifier to search only for devices belonging to the specified class. This application-level service primitive results in the local device executing inquiries to learn about devices, primarily any new devices, in its vicinity. If, in the future, CoD-specific *dedicated inquiry access codes* (DIACs) are standardized, then the inquiries for this primitive could be generated according to these DIACs. Otherwise, the *generic inquiry access code* (GIAC) is used and the local device needs to filter the received responses according to the information included in the CoD field in the FHS BB_PDUs received from the responding devices.
- *terminatePrimitive:* This service primitive results in the termination of the operations invoked by the previous primitives.

The first and second primitives above relate directly to transactions involving SDP_PDUs. The third primitive could be satisfied merely by requesting that the device enter the inquiry mode with the sole purpose of searching for any other devices in the inquiry scan state (as described in the baseband section of Chapter 6). The last primitive simply terminates any ongoing actions resulting from the use of any of the other primitives.

The SDAP requirements on the Bluetooth stack are straightforward. To implement this profile one needs to use nothing beyond the default settings for all the protocol layers below the SDP layer. In particular, devices that connect for the sole purpose of performing service discovery do not need to authenticate each other or encrypt their Bluetooth link.[6] The SDP transactions are carried over the ACL baseband link between the devices. At the L2CAP layer SDP transactions are carried over connection-oriented channels configured to carry "best effort" traffic.

An additional distinction of this profile is that it identifies the conditions under which L2CAP channels carrying SDP traffic are torn down. This is so because SDP does not define a session or a transport

6. This does not imply that device authentication and/or link encryption should not be performed. This statement simply implies that the SDAP imposes no security precautions for its execution. Security is as much a usage scenario requirement as a user-level settable policy, as discussed in the GAP. Security precautions may still be taken for reasons outside the scope of SDAP.

protocol for carrying SDP_PDUs. SDP itself is in essence a connectionless protocol. To run the connectionless SDP request/response transactions, the L2CAP layer needs to maintain the L2CAP channel that carries the transactions at least for the duration of the transaction. For efficient use of the transmission resources, the L2CAP channel between an SDP client and an SDP server should be maintained even longer. Ultimately, it is the application on behalf of which SDP transactions are executed that should open and maintain, for as long as necessary, the L2CAP channel for SDP transactions between two devices.

Summary

In this chapter we have highlighted the two generic Bluetooth profiles: the GAP and the SDAP. The GAP describes the connectivity and security modes of operation for a device that permits it to discover, be discovered by, and create trust bonds and Bluetooth links with other devices. These modes of operation are user- (or application-) settable device policies that specify how the device should behave relative to other devices with which Bluetooth communication might ensue.

The SDAP describes the functional characteristics of a service discovery application. Furthermore, and equally importantly, the SDAP describes the way that the SDP layer must use the group of Bluetooth transport protocols to carry SDP transactions. This aspect of the SDAP also forms the basis for executing service discovery within the other profiles.

13

The Telephony Profiles

Continuing our examination of the profiles, we now visit those that are based upon telephony functions. We include here the cordless telephony, intercom and headset profiles. As described in Chapter 10, the fax and dial-up networking profiles also make some use of telephony protocols, but we consider these part of the networking group that is covered in Chapter 15. All three telephony profiles discussed here primarily address voice telephony functions. They all target telephony devices (mostly mobile phones and headsets) and thus they exist largely for use by telephones.[1] In fact, the intercom and cordless telephony profiles instantiate two different aspects of the three-in-one phone usage scenario introduced in Chapter 3, although the cordless telephony and headset profiles explicitly address the use of computer audio in addition to telephone audio.

These telephony profiles, then, are expected to be implemented in many mobile telephones and other telephony equipment used with phones, like headsets and voice access points. All are intended to carry voice traffic, and in fact it is this common element that caused us to group them for this chapter's discussion.[2]

1. We suppose that other types of devices could implement these profiles. There is no reason that, say, a computer could not provide intercom profile function if it had the appropriate voice and TCS-BIN support. But in general the telephony profiles center around telephones.
2. These profiles could have been referred to as "telephony audio" or "voice telephony" profiles but we opted for the briefer yet still descriptive "telephony profiles."

RELATIONSHIPS

Voice telephony is a common element shared by these three profiles, but from a technical perspective (refer back to Figure 11.1) they inherit from two different profile families. The cordless telephony and intercom profiles are part of (and indeed all of) the TCS-BIN family (even though there is no TCS-BIN profile per se) while the headset profile derives from the serial port profile. Yet from an end user's perspective, the most visible feature of all of these profiles is the ability to route and process telephony-grade audio traffic using Bluetooth wireless communication.

The cordless telephony and intercom profiles are both part of the three-in-one phone usage case. Recall from Chapter 3 that the three types of usage for a mobile telephone in this scenario are as (1) a standard cellular phone; (2) an intercom or "walkie-talkie"; and (3) a cordless phone using a cordless base station, or voice access point. Standard cellular phone operation is addressed by protocols like GSM, CDMA, CDPD and others, using a wide area radio in the handset; Bluetooth wireless communication is not used for the standard cellular phone operation. Standard cellular phone usage is not discussed further.

The profiles for the remaining two aspects of the three-in-one phone usage model, both of which use the TCS-BIN protocol, are unusual among the version 1.0 profiles in that they define only part of a usage case. Most profiles (fax, dial-up networking, synchronization and so on) map one-to-one to usage cases. But in the case of the three-in-one phone usage case, there are two separate profiles–cordless telephony and intercom–that define the two separate parts of that single usage case that are relevant for Bluetooth wireless communication. In this case there are good reasons to separate the two distinct functions. While they can be combined to realize the three-in-one phone scenario, cordless telephony and intercom also can be of value individually, and a robust implementation of cordless telephony is much more involved and complex than is an implementation of intercom. As we will see below, cordless telephony often involves advanced functions of TCS-BIN, while intercom communications can be much simpler in comparison. Therefore it could make sense in some devices to implement just an intercom function without cordless telephony function. By separating these functions into individual profiles, such devices can conform to the intercom profile, which is useful in its own right, without implementing the full three-in-one phone usage case, which would require additional cordless telephony support.

While the intercom profile is simple in comparison to cordless telephony, the headset profile is intended to be even more straightforward and less complex–it is perhaps the simplest of all the version 1.0 profiles. This is by design, since Bluetooth headsets are generally expected to be simple, low-cost accessory devices; thus the requirements of the headset profile need to be minimal to enable such lightweight devices. This is one reason that the headset profile, even though it is involved in audio telephony, belongs to a different profile family than do the cordless telephony and intercom profiles. Recall from Chapter 11 that the headset profile is a derivative of the serial port profile. The SIG felt that requiring wireless headsets to implement the TCS-BIN protocol (as the cordless telephony and intercom profiles call for) would be too burdensome for headsets. For the version 1.0 headset function, only audio transfer (which occurs directly over the baseband link) and some very simple control functions are needed. TCS-BIN is excessive for these simple control functions for headsets, so a minimal set of AT commands was chosen for headset control. These commands can be used over the RFCOMM virtual serial port as described in Chapter 10. Thus headsets need only contain a minimal implementation of the RFCOMM protocol, so there is no requirement for them to implement the TCS-BIN protocol.

So we see that even though these three profiles exist on two different branches of the protocol-based "profile family tree," their common bond is that of supporting some form of voice telephony. Each is concerned with the rendering and transporting of audio traffic (SCO packets) over the Bluetooth air-interface.

The Cordless Telephony Profile

As already noted, the cordless telephony profile, or CTP, defines the "cordless phone" facet of the three-in-one phone usage case, but more generically it defines cordless telephony. The CTP not only allows a cellular telephone to use Bluetooth technology for short-range wireless voice communication, but it also addresses handsets that exclusively use Bluetooth wireless communication to act only as cordless telephones. These telephones are not cellular phones; they are solely cordless handsets for use with a local base station. The CTP also includes computers that could accomplish cordless telephony using Bluetooth wireless communication, through support for the TCS-BIN protocol

and audio traffic management using the microphone and speakers of the computer.

The cordless telephony usage scenario drove most of the requirements for the Bluetooth telephony control protocol. Cordless telephony introduces the notion of *terminal* devices and *gateway* devices and hence the requirement for control functions for these devices, such as group management. Furthermore, cordless telephony also introduces the need for reasonably sophisticated call control functions. For example, a base station needs to communicate call control information to and from a remote handset to enable the handset to receive ring tones for incoming calls and to dial through the base station for outgoing calls. Finally, advanced features found on many existing cordless telephones, like multiple handsets for a single base station, speed dialing, directory and caller identification information and so on drive the expectation that these same sorts of features can be accommodated in Bluetooth cordless telephony. Thus the functions required for cordless telephony helped to drive the selection of TCS-BIN, which is the primary protocol used by the CTP (recall from Chapter 10 that TCS-BIN provides call control, group management and connectionless TCS functions, all of which are important in satisfying the requirements outlined above).

CTP Development

Until March 1999 there was only a single three-in-one phone profile that encompassed both cordless telephony and intercom functions. As the profile development progressed, the SIG decided to split the cordless telephony and intercom functions into two profiles because, as we observe above, these functions might be implemented independently. Both profiles today still state that they apply for devices implementing the three-in-one phone usage case, but they also acknowledge that each addresses one specific function of that usage case.

Even though it is one of the more involved profiles, the CTP, at least in its incarnation as the three-in-one phone profile, was one of the first to be started and thus one of the first to mature and reach completed status.[3] This is due at least partly to the fact that the three-in-one phone scenario requirements had been studied for quite some time and had resulted in the selection of TCS-BIN as the telephony control protocol for this scenario. Thus the mapping of cordless telephony func-

3. Technically, nearly all of the profiles were completed, or formally ratified, at about the same time. But at that point some profiles had already been largely completed for some time while others were still being finalized.

tions back to TCS-BIN (which is what the vast majority of the CTP deals with) was rather straightforward.

CTP Examined

The three-in-one phone usage scenario heavily influenced the development of TCS-BIN in the protocol stack, and therefore the CTP makes heavy use of TCS-BIN. Most of the CTP is devoted to TCS-BIN interaction. A study of the TCS-BIN protocol is useful in understanding both the CTP and the intercom profile (described below). The CTP contains a detailed description of procedures and TCS-BIN messages used in cordless telephony applications; these are not repeated here. Instead we highlight some of the key aspects of the CTP, including the rationale for those design points.

The CTP first makes the important distinction of device roles, as either gateway or terminal devices. Nearly all of the remainder of the CTP is based upon this distinction. In general the gateway device can be viewed as the "server" of a piconet (and in fact is defined to be the piconet master in most cases), with various other devices, including cellular handsets, specialized cordless handsets and perhaps even computers or advanced headsets,[4] as "clients" of the local voice network (piconet). Figure 13.1 illustrates a cordless telephony piconet; a similar figure exists in the profile specification, but we include our own version, as we elaborate on the concepts and terminology shown here in following sections.

4. In this case, such headsets would not be the same as those developed to comply with the headset profile, which is based upon the serial port profile. Cordless telephony headsets would functionally resemble handsets and would need to comply with the cordless telephony profile (instead of or in addition to the headset profile).

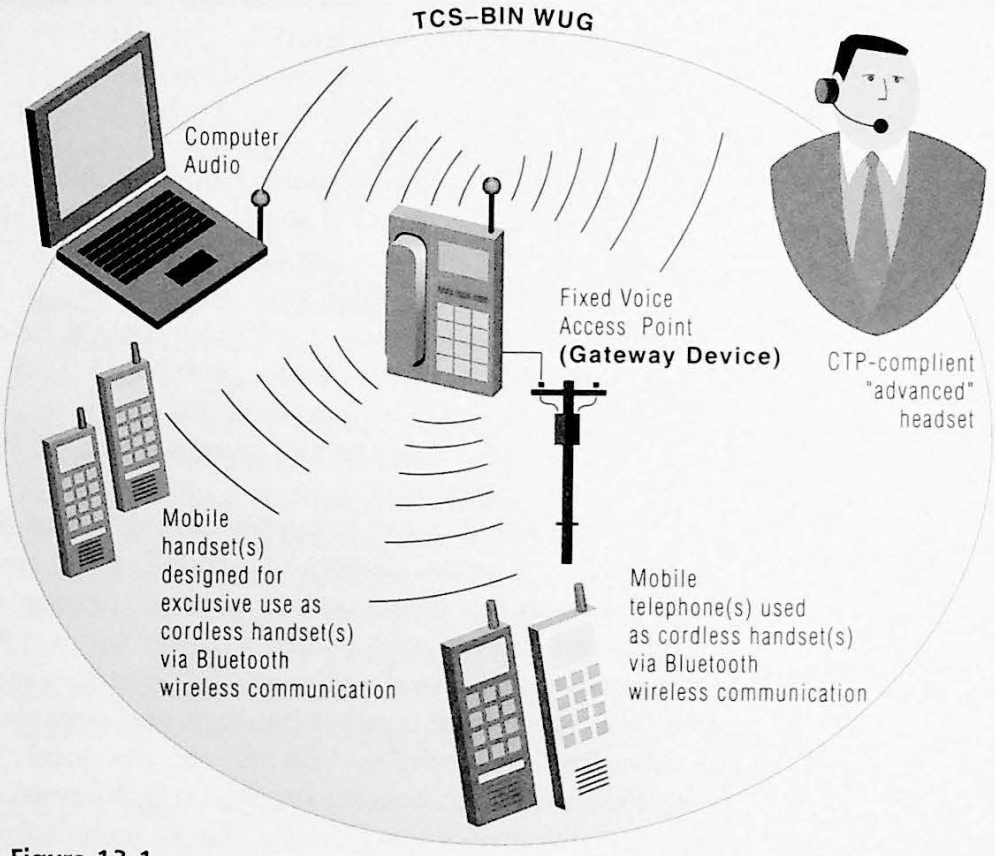

Figure 13.1
Cordless telephony piconet with gateway device and terminal devices.

Note that because the voice telephony network is a piconet, only seven slaves, or terminal devices, can be active at one time. This is still an improvement over many of today's cordless telephone systems, which often associate only one handset with a single base station. As the CTP points out, more than seven terminal devices are possible; as one might expect, this is accomplished through the use of parked slaves. Since the gateway is the master of the piconet, it would be responsible for managing up to seven active slaves versus some number of parked slaves in the case where there are more than seven terminal devices. To allow the development of gateways with varying features and complexity, managing more than seven terminal devices is not mandated (it is an optional capability and thus need not be implemented in every gateway device).

The CTP is one case in which the master-slave role switch described in Chapter 6 is used. The gateway device is the master of the piconet. New terminal devices (perhaps a mobile phone that is brought

into the home) are added to the piconet when the new terminal device pages the gateway device (base station). Once the gateway device accepts the page, the terminal device then, by default, becomes the master of that link. However, this is the opposite of what is needed for normal operation of this voice piconet, so a master-slave switch (role reversal) must be performed immediately. One alternative that might have been used, which could have removed the need for a master-slave switch, is a model in which the gateway device pages the terminal devices. For well-known terminal devices (say, those handsets that are always associated with the base station), this might be a reasonable method to use. However, for transient devices that might need only temporary access to the gateway (consider visitors to a home or office who might be granted temporary use of the gateway, or a public voice access point that permits devices in proximity to connect to it), the SIG chose a model in which the client, or terminal device, initiates a request to connect to a gateway device. This model takes into account the needs and desires of the user of the client device and seems preferable to a scheme in which a gateway attempts to communicate with every potential terminal device that might wander into proximity; many such devices might have no desire or even capability to participate in the voice network. Thus the process of joining the cordless telephony piconet is client initiated and therefore necessitates a master-slave role switch after a new member joins.

An important aspect of cordless telephony is security. The CTP requires that all devices in the voice piconet be authenticated. While one may wish to allow a trusted friend to have access to one's own gateway via the friend's handset, one certainly wouldn't want to offer this capability to any device that happened to be in range, since access to the gateway implies the ability to make telephone calls through it. The CTP allows for only trusted terminal devices to connect to the gateway.[5] The CTP also calls for encryption of all the traffic within the piconet. A common shortcoming of very early cordless telephone systems[6] was the ability of others to easily eavesdrop on the voice traffic trans-

5. Here is but one of many examples of the value of the common vocabulary of the GAP, discussed in the previous chapter. The CTP states that only trusted devices can connect to a gateway; without the GAP, which defines a trusted device, this term might be interpreted in different ways.
6. In the United States, 900 MHz cordless telephones (and similar systems such as baby monitors) without any encryption or spread spectrum capabilities were quite popular. With these systems it was easy to eavesdrop (intentionally or not) on conversations of neighbors. While many newer systems use spread spectrum, which can mitigate the eavesdropping problem, many older systems still remain in use.

ferred over the air. As noted in Chapter 1, the FHSS nature of Bluetooth communications adds one degree of protection from eavesdropping, but the use of the encryption inherent in the technology enables even more secure communication.

When a CTP piconet is formed, there exists a group of devices that all implement the TCS-BIN protocol (since this is necessary for conformance to the CTP). Recall from Chapter 10 that when such a group of devices is formed, a WUG (wireless user group) can also be formed to make use of the TCS-BIN protocol to provide additional features enabled by the protocol. In the case of the CTP, the WUG is employed to facilitate use of cordless telephony features in a secure manner. As would be expected, the gateway device acts as the WUG master (in addition to being master of the piconet). Because WUGs allow all participating devices to be known to and to interact with each other (an additional capability beyond the point-to-point master-slave communication), cordless telephony piconets have some unique advantages. For example, once a new device has been authenticated with the gateway (master) device, it need not authenticate individually with every other WUG member, since the master will by proxy authenticate the new device when it informs all the WUG members that the new device has joined. This could be useful if the device later initiated an intercom conversation (described below) with another member of the WUG. Not only would the device know about all of the other devices in the WUG, it could easily establish direct communication with any of them.

When a terminal device connects to the gateway, it establishes and maintains an L2CAP connection for as long as it remains in the piconet. So, when a call is made or received, it is not necessary to incur the overhead involved to establish a transport layer connection, which in some cases could take up to several seconds, to process the call. Only the TCS-BIN protocol needs to be initiated over the already existing L2CAP connection.

CTP Usage

Clearly the sorts of applications that implement the CTP would be telephony applications that manage telephone calls. While the specification does not define APIs, TCS-BIN establishes a well-defined functional interface for making and receiving calls and for transferring information like DTMF tones or caller identification data. Because TCS-BIN is based upon the ETSI Q.931 standard [ITU98], existing telephony APIs on many platforms should map easily to those of TCS-BIN.

The CTP adds more guidance to the telephony application developer by specifying which of these call primitives are mandatory and which are optional for Bluetooth cordless telephony. Applications on devices claiming to conform to the CTP must at least implement the basic set of functions described in section 2 of the CTP (these include on-hook, connection management, outgoing and incoming call management and others). Some more advanced features are optional. And like all profiles, vendors can add their own differentiating features beyond those specified in the profile. This is especially true in the case of the CTP, since it uses TCS-BIN, which includes the connectionless TCS feature described in Chapter 10. This feature directly enables vendor-specific features and extensions.

THE INTERCOM PROFILE

For convenience, we will continue our custom of shorthand notation for the profiles. In the case of the intercom profile we use the term IntP (rather than IP, which might be confused with Internet Protocol and IP networking). The IntP is the other profile that is based upon TCS-BIN, and it also defines the final aspect of the three-in-one phone scenario. Intercom, or "walkie-talkie" operation, generally is an easily understood concept, since it is a direct voice connection between two devices that many people have experienced. The intercom profile is thus unsurprisingly simple and straightforward.

With the intercom profile, two devices that both support TCS-BIN can make a direct voice connection using the Bluetooth air-interface, without any third-party carrier required. In the specification the IntP includes a figure that shows such a connection between two cellular phones. While this is the most obvious (and probably most common) situation, other devices that have audio and TCS-BIN support could also participate in the intercom usage model. As shown in Figure 13.1, specialized cordless handsets, CTP-compliant advanced headsets and computers might all include audio and TCS-BIN protocols and therefore could implement the IntP. In fact, one could build a true Bluetooth walkie-talkie that is dedicated solely to the intercom scenario.[7]

7. Although this is possible, it seems unlikely, since even with the optional radio the range of Bluetooth wireless communication is limited to 100 meters, which is less than that of existing walkie-talkies using other radio frequencies. The value of the intercom usage case is the utility it provides by adding another function to an existing device rather than enabling a new class of device.

IntP Development

As noted in the preceding discussion of the cordless telephony profile, the CTP and IntP were split from a single three-in-one phone profile during profile development. The IntP is really a special case of cordless telephony, and intercom calls are still referenced in the CTP, although in these cases they refer to the IntP for details. When the history of these profiles is known, it becomes quite evident that the CTP and the IntP are closely related. From a structural point of view, the two profiles mirror each other almost exactly.

Like the CTP, the IntP was one of the first profiles to be started and thus one of the first to be completed, or at least to reach a level of stability such that it was declared ready for publication.

IntP Examined

For the intercom usage case, the IntP indicates that there are no prescribed device roles. Unlike the CTP, where it is important to define a master (gateway) and slave (terminal) device, the devices in the intercom scenario are peers. Either device could be master of the piconet.

There could have been multiple ways to establish a direct voice link with Bluetooth wireless communication. The IntP chooses to make use of TCS-BIN for this scenario, so the intercom function is still very much a "telephone call" sort of operation. Data flows as SCO packets, which is the norm for voice traffic, and control is provided via TCS-BIN. This control might have been provided through some other means, but because TCS-BIN is used in the CTP, which is also part of the three-in-one phone usage model, the use of TCS-BIN for the IntP is natural. Furthermore, TCS-BIN's group management functions provide an environment in which it is relatively easy to establish an intercom call. Through the WUG enabled by TCS-BIN, each device in the voice piconet is aware of every other device. In addition, as a result of their authentication with the master, all of the devices are trusted by each other. Thus it is a straightforward operation for one device to locate and establish a communication link with any other device in the WUG (which overlaps entirely with the piconet) to perform an intercom call. The master need not be involved;[8] any device can directly page any other device to set up an intercom call. Note that this means that these devices temporarily leave the existing piconet to form their own new

8. Except to set up the intercom paging scenario, as described in Chapter 10.

piconet, but rejoining the original piconet is also made easier through the WUG/TCS-BIN group management functions.

One important aspect of the intercom usage case not addressed by the IntP is that of 10-meter versus 100-meter operation. As pointed out in Chapter 3, the intercom scenario with the standard 0 dBm Bluetooth radio is somewhat uninteresting. With the standard radio's range of about 10 meters, two parties who might use the intercom function are likely to be close enough to each other that they can talk without the benefit of radios. However, there certainly are situations in which intercom communications are useful even with a 10-meter range. Consider, for example, the parties being on different floors of a house or an office, or perhaps needing to communicate with the benefit of radios even when they have a line of sight between them (one example of the latter case given in Chapter 3 is that of audio/video technicians in a crowded auditorium during a conference presentation). However, intercom communications become even more interesting when the 20 dBm optional radio with its 100-meter range is employed. The ability to make a direct call to another device without using any third-party carrier, and thus without incurring any airtime usage charges, is quite attractive. Think about being able to use your mobile phone to contact your spouse or friend in their seat at a crowded sports arena while you are at the concession stand, without having to dial through your service provider. Other applications could include those where medium-range wireless voice communication is used today–security and maintenance workers, for example, who need to stay in communication within a local area such as a hotel or small campus area. In these cases, Bluetooth wireless communication might be used in place of other RF solutions, one benefit being that a single device could be used both for the local medium-range RF voice communication and for some other function (such as a cellular phone or computer usage), obviating the need to carry another device just for wireless voice communication.

IntP Usage

An IntP application is likely to be part of a general cordless telephony application. As previously noted, it seems unlikely that Bluetooth devices dedicated exclusively to functioning as intercoms or walkie-talkies will be prevalent. Thus it would seem unlikely that a separate application dedicated to intercom function would be developed. Since the IntP uses TCS-BIN, as does the CTP, we would expect that a cordless telephony application that implements the CTP could be easily

extended to also incorporate the IntP. In fact, while these two profiles need not be implemented together (recall that the SIG overtly chose to split them), in many cases, such as in support of the three-in-one phone usage model, it would make sense to do so.

As with the CTP, the IntP provides guidelines for application developers, including optional versus mandatory functions. As would be expected, the intercom function specified by the IntP is relatively simple and straightforward, and in fact for the most part is a subset of the TCS-BIN function that is needed to realize the CTP. This is another reason to expect the IntP to be implemented alongside the CTP in many cases–once the CTP application is complete, nearly all of the work required to realize the IntP would also already have been completed.

THE HEADSET PROFILE

For reasons previously stated, we consider the headset profile, or HSP, to be part of the telephony group of profiles, although it should be reiterated that the HSP is not directly related to the IntP or CTP. The HSP is a derivative of the serial port profile; it is not one of the TCS-BIN-related profiles. Nevertheless, the HSP also addresses voice traffic and its control.

In the CTP discussion we noted that one possibility for a cordless telephony device was a headset. Such an advanced headset would need to comply with the CTP, including support for all of the required parts of TCS-BIN. It would tend to resemble a telephone handset from a functional point of view and would probably be more sophisticated than the type of headset that the HSP deals with. A headset conforming to the HSP need not implement TCS-BIN at all; the simple telephony control functions needed for an HSP headset are accomplished using AT telephony control over RFCOMM. Thus the HSP defines a device that primarily serves as an audio peripheral to some other device (most popularly, but not exclusively, a telephone).

HSP Development

Like the other telephony profiles, the HSP was one of the first to be started and thus one of the first to reach stability for version 1.0 publication. The wireless headset, or ultimate headset as it is called in the Bluetooth usage model (see Chapter 3), has always been an important

scenario. The use of wireless headsets with mobile telephones was one of the driving forces behind the invention of the Bluetooth technology. Even though the HSP is not directly related to the CTP and IntP, it is evident from the structure of these profiles that all were developed by the same group of people.

Once the fundamental case of headset use with a mobile telephone had been covered, the profile was expanded to cover the computer device class. A Bluetooth headset could easily be used as the source and destination for a Bluetooth computer's audio traffic in the same manner as it is used with a phone, and the profile acknowledges this usage case. There was some discussion of the use of a headset with other devices (as noted in Chapter 3, future devices could include not only other types of phones but also other types of audio devices such as stereos, portable music players and so on). In version 1.0, the HSP does not address any headset usage other than with a mobile telephone or a computer; but since the specification defines a standard method for audio transfer and control, it is not expected that significantly different operation would be required for other types of devices. There is no particular technical obstacle that prevents the use of a Bluetooth headset with Bluetooth devices other than computers and telephones, but the HSP addresses just these latter two classes of devices in version 1.0 of the specification.

HSP Examined

Of all the version 1.0 profiles, the HSP targets the simplest sort of device. In fact, the driving requirements behind the HSP included the capability to develop a low-cost, simple and lightweight headset. If such a device is overburdened with functional requirements that need sophisticated software (which in turn requires more processing power and/or on-board memory along with the associated increase in power consumption), then a low-cost device becomes more difficult to realize. This is one reason that the HSP is based upon the serial port profile rather than the TCS-BIN protocol–TCS-BIN is robust and quite useful for telephony applications, but a rich and full TCS-BIN implementation could be relatively expensive as compared to a simple RFCOMM implementation. Moreover, TCS-BIN includes much more function than is needed for a headset as defined by the HSP. Note that the headset device of the HSP is quite different from the hypothetical CTP-compliant advanced headset depicted in Figure 13.1. The HSP describes a much simpler headset that uses AT commands over an RFCOMM link,

and the HSP-style headset is expected to be the more prevalent device. Figure 13.2 illustrates typical HSP operation.

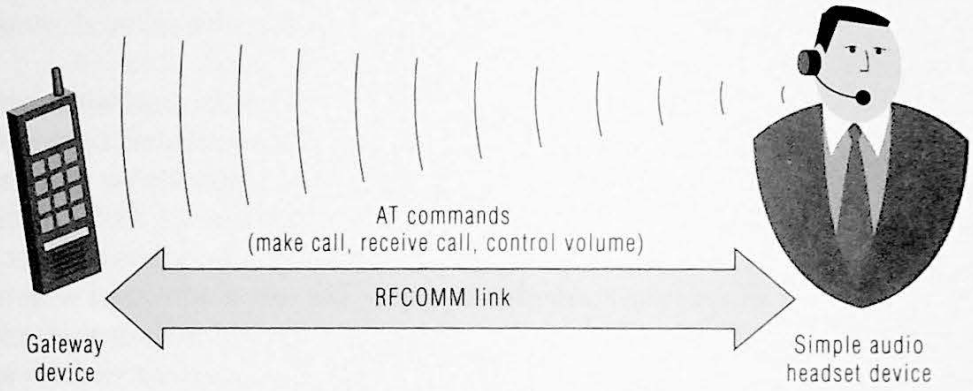

Figure 13.2
Typical headset profile operation.

The HSP does not prescribe any particular role for the headset or its associated (phone or computer) device–either can be master.[9] It defines a gateway device and a headset device, but unlike the CTP it does not designate either as master. There are only a very few control functions, including making a call (it is assumed that the headset has some minimal user interface, perhaps a button, to initiate a connection with the associated gateway device), receiving a call and (optionally) controlling volume. Unlike the IntP and CTP, these functions are controlled via AT commands (listed in the HSP) over RFCOMM rather than through TCS-BIN PDUs. While they may overlap functionally, they are entirely different means to similar ends.

The HSP does not mandate any level of security, leaving it up to the implementation as to whether or not a secure connection (including authentication and encryption) is used.

9. There was a significant amount of debate in the SIG about this design choice. There are good arguments for denoting the headset both as master and as slave, depending upon the specific usage–either the headset or the associated device could initiate the communication; consider both outgoing and incoming calls. The final resolution was to leave the master/slave roles unspecified in the profile, leaving them to the choice of the implementers.

HSP Usage

The headset is a specialized usage case. Most headset applications are expected to be embedded in headset equipment. Within a computer or a telephone, an application might support a headset peripheral, including the capability to route audio for incoming and outgoing calls to and from the headset and to remotely control the volume, if that feature exists.

While the HSP considers only headset function, the audio control and routing used for headsets probably could be generalized to other cases which are similar but use different hardware–things like the speaking laptop usage case described in Chapter 3 or audio routing to other external systems like those in an automobile.

14

The Serial and Object Exchange Profiles

We call this group the serial profile family, and it is composed of the serial port profile (SPP) itself along with the object exchange class of profiles. The object exchange group consists of the generic object exchange profile, the object push profile, the file transfer profile and the synchronization profile.

To be sure, many other profiles use the SPP; in fact, most of the version 1.0 profiles do. We include one such group in this chapter; other profiles that inherit from the SPP are treated in other chapters as part of other functional categories. In fact, each of the final three chapters of Part 3 includes at least one SPP-based profile. In this chapter we discuss the object exchange profile family and the serial port profile itself.

The SPP maps directly to the RFCOMM protocol and thus is used in many cable-replacement usage scenarios. Because so many of the version 1.0 usage cases employ RFCOMM, the SPP could be the most widely implemented and used profile of all in early Bluetooth device implementations. Even though the SPP itself does not embody a specific usage scenario, it enables many of them.

The object exchange profiles (generic object exchange, object push, file transfer and synchronization) are likely to be implemented in both computing and telephony devices. Wherever IrDA devices are used, the same applications are likely to apply in Bluetooth environments. Bluetooth technology provides a convenient way for devices like notebook computers to exchange files, and object exchange applications like electronic business card exchange are likely to be found wher-

ever an electronic address book is kept–on PDAs, mobile phones and notebook computers, for example.

RELATIONSHIPS

As shown in Figure 11.1 in Chapter 11, the serial port profile is the root of many profiles. Within the group of five profiles discussed here are two abstract, or "parent" profiles, from which others inherit. One of these is the SPP, the other the GOEP, with the latter descending from the former. The object push, file transfer and synchronization profiles all derive from the abstract GOEP, which addresses the common use of OBEX operations that apply to these three profiles. This group of profiles maps to the IrDA interoperability layers of the protocol stack.

In this chapter we discuss only a subset of the SPP family, but the RFCOMM serial port abstraction is also used for AT command telephony control in the dial-up networking and fax profiles (described in the next chapter) as well as in the headset profile (described in the previous chapter); the serial port is also used to enable a form of IP networking in the LAN access profile (also described in the next chapter).

THE SERIAL PORT PROFILE

The serial port profile, or SPP, is a transport protocol profile that defines the fundamental operations necessary to establish RFCOMM communications between two peer devices. Such a link is required for many of the concrete usage scenario profiles. In this respect the SPP is somewhat like the GAP, in that it describes how to establish necessary communication links that are in turn needed by other profiles. The SPP serves as a profile "building block."

SPP Development

We observed in Chapter 8 that the RFCOMM specification contains some elements that typically are found in profiles, so it seems as though the SPP was at least conceptually in development since near the beginning of the SIG's existence. Its completion was neither particularly early nor particularly late in the profile development cycle. The SPP was not always by design the basis for so many other profiles. Like the SDAP and GAP, it was not immediately evident that a profile relating to the RFCOMM protocol layer was merited. Even after it was created,

the SPP originally was not the basis for all other profiles that might use the RFCOMM protocol. For instance, in March 1999 there was still debate as to whether or not the object exchange profiles discussed in this chapter would use the SPP. At that time the GOEP (and its derivatives) all specified their own use of RFCOMM. The serial port profile existed but focused mostly upon transporting AT commands for the headset, fax and dial-up networking profiles and serving as a conduit for PPP for the LAN access profile. When it was observed within the SIG that the SPP might be a good basis for the GOEP, the differences between the GOEP and the SPP, in terms of specification and usage of the RFCOMM protocol and related stack layers, were identified. The SPP was then updated to accommodate the GOEP usage of the serial port abstraction as well. This is how the SPP came to be the foundation for so many other profiles.

SPP Examined

SPP defines peer device roles for serial communication. It does not define a specific device role for master or slave, nor does it define device roles for DTE/DCE devices (analogous to typical wired serial communication). The devices are peers, and it does not matter which is master and which is slave. In fact, the SPP just calls them "Device A" and "Device B," the only distinction being that Device A initiates the serial communication link. The SPP further states that even this distinction is of little consequence as far as the profile is concerned.

The SPP outlines the steps necessary to establish an RFCOMM emulated serial port connection; these are illustrated in Figure 14.1. Interestingly, these are the sorts of functions that one might expect to find in a Bluetooth adaptation layer of software as described in Chapter 5. That is, the SPP describes precisely how to use the Bluetooth protocols to establish a virtual serial connection; once this connection is established, serial data can flow across it. Thus for a legacy application that uses a serial port, the functions described in the SPP are just what is needed to replace a wired serial interface with a wireless one. Once the procedure outlined in the SPP is completed, the fact that a Bluetooth emulated serial connection is being used rather than a traditional wired serial connection should be transparent to an application. In fact, in the SDP service record of the SPP, the example service name shown is "COM5," an indication that the SPP could be used to set up emulated serial links for applications expecting to use "COM" port-style interfaces.

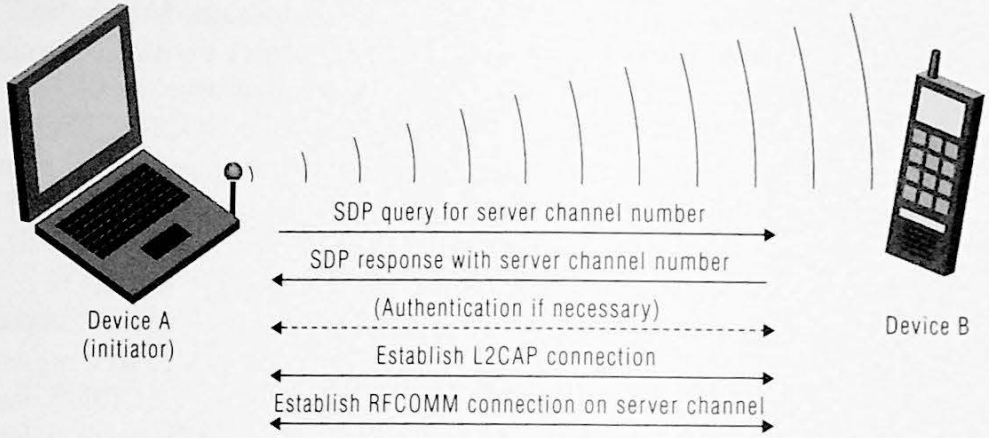

Figure 14.1
Typical serial port profile operation.

In addition to the link manager operations necessary to establish a baseband link between devices, SPP makes specific mention of two other protocol stack layers needed to establish the RFCOMM link. The first, L2CAP, has already been mentioned. RFCOMM communicates over L2CAP, and an L2CAP connection using the RFCOMM PSM value must first be established. The other protocol needed to establish an RFCOMM channel is SDP. SDP is used in setting up an RFCOMM link, to find the appropriate RFCOMM *server channel*[1] to use. Server channels are used to multiplex RFCOMM connections. SDP is used to choose the appropriate server channel, which might correspond to a given service (somewhat like "well-known port numbers" in TCP/IP). In any case, the service of interest must specify the appropriate server channel number to use to connect to that service. This channel is the one used in the resulting RFCOMM connection over which the SPP operates. After the server channel number is known, setting up the RFCOMM connection is straightforward: an L2CAP connection is established, over which the RFCOMM connection is established. Optionally, the devices might require some degree of authentication, and perhaps also encryption, prior to establishing the links between them. The SPP specifies that these security considerations are optional, because the SPP is a generic profile upon which others are built. Some applications that use the SPP may require authentication or encryption

1. Server channels are constructed using the DLCI values described in Chapter 8.

(which in turn requires authentication) while others may not. The SPP leaves it to the more specific profiles to describe security requirements.

SPP Usage

As an abstract profile, the SPP is more likely to be used by middleware than directly by applications. Bluetooth adaptation software might use the SPP to instantiate a virtual serial connection for an application that expects to use serial communications. The application need not know that the serial port is an emulated wireless one, so long as the emulation is sufficiently accurate.

Thus applications most likely will implement some other profile that makes use of the SPP–perhaps headset or dial-up networking. A device might support the SPP generically in middleware. But an application is likely to follow the standard procedures for a given platform to open, configure, make use of and close a serial connection, as opposed to specifically following the Bluetooth protocol stack methods for performing these functions. Thus it seems likely that some sort of platform middleware will transform the platform APIs into the corresponding functions of the SPP necessary to use RFCOMM over Bluetooth transports.

The Generic Object Exchange Profile

Like the SPP, the generic object exchange profile, or GOEP, is an abstract profile upon which concrete usage case profiles can be built. In this case the remainder of the GOEP family is the set of IrDA interoperability profiles, namely file transfer, synchronization and object push, each of which is examined below. The GOEP defines all of the elements common to these other three usage cases, including device roles, security considerations and how the OBEX protocol is used in general.

GOEP Development

The origin of what came to be known as the GOEP was the synchronization usage model. Since early in the SIG's history, synchronization was the driving force behind the use of the OBEX protocol and its related scenarios. Indeed, synchronization was one of the original usage models, and the group that produced the IrDA interoperability protocols and profiles in the SIG was called the synchronization working group during most of its existence.

The synchronization usage case drove the use of OBEX and IrMC; the use of these protocols in turn drove the additional usage cases of electronic business card exchange (represented in the object push profile) and object transfer (represented in the file transfer profile). Once the IrDA OBEX model was chosen for synchronization, it was evident that the other usage models enabled by the IrDA protocols applied to the Bluetooth protocol stack as well, so the relevant ones were adopted (resulting in the two additional profiles described in the sections that follow). Further, this selection spawned the whole idea of IrDA interoperability, which became the central theme of this family of profiles.

So in this case the evolution was from the specific category of synchronization to the broader category of IrDA interoperability, including object push and file transfer. During this generalization process the GOEP was developed. As the synchronization, file transfer and object push profiles progressed, it was evident that they shared a foundation of common elements, especially those related to the use of OBEX. These common elements were gathered into the GOEP.

GOEP Examined

The GOEP defines very specific device roles for all OBEX-related profiles. Unlike many of the other profiles, where the devices act as peers and there is little distinction between them, the GOEP and its derivatives define a client role and a server role. The client device is the one that pushes or pulls objects to a server, while the server is the one that provides the object exchange service, which allows those objects to be pushed to and pulled from it. At one level, this distinction might not seem clear: if objects are being exchanged, both sides may be pushing and pulling objects from the other, so a client or server role becomes somewhat ambiguous. However, in the OBEX model, there is a distinction. While both devices can indeed push and pull objects, it is the client that initiates the operation and locates a server with the desired object exchange service. Hence, the client and server roles are significant in the GOEP model. However, this does not imply any master or slave role. The client and server roles are relevant at the OBEX protocol layer, but they have no specified relationship to a device master or slave role; the GOEP client could be either a master or a slave device, as could the GOEP server.

The GOEP assumes a form of authentication called *bonding* (as defined in the GAP). To accomplish any of the object exchange usage models, the two devices engaging in the transactions must be known to

and trusted by each other. All of the object exchange profiles assume this trust relationship. Additional forms of security, such as encryption or application-level authentication (beyond that done at the Bluetooth transport level) are optional.

The GOEP defines the primitives for object exchange, two important ones being object push and object pull. These operations are used in different combinations under different sets of circumstances in all of the object push, file transfer and synchronization profiles. The simplest form of generic object exchange, one-way data transfer, is instantiated in the object push profile.[2] Bidirectional data exchange occurs in the file transfer profile, which can be considered to be a user-initiated object exchange, and in the synchronization profile, which acts as a rules-based object exchange. In addition to the fundamental OBEX operations, the GOEP defines how to establish and terminate OBEX connections and how to use common OBEX functions.

GOEP Usage

Like the SPP, the GOEP is not expected to be used directly by most applications. Instead, it provides a foundation for other profile applications. In fact, the set of IrDA interoperability protocols and profiles are intended to promote interoperability at the application layer for applications that can use both IrDA and Bluetooth transports. Thus there could be many existing applications that already implement file transfer, object push or synchronization functions using OBEX. These applications should run with little or no change from IrDA to Bluetooth environments.

New applications or middleware layers might implement the GOEP in support of accomplishing the more concrete object exchange profiles; however, it would be of little value to implement just the GOEP itself without at least one other concrete profile. The GOEP is a set of common elements that enable the other object exchange profiles, not a usage model unto itself.

The Object Push Profile

The object push profile, or OPP, is the simplest of the object exchange profiles. As its title implies, it essentially defines a one-way object trans-

2. As noted in the OPP discussion below, this is not always strictly one-way transfer but it is based upon a model of pushing data.