

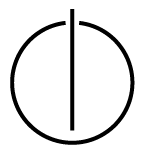
Fakultät für Informatik

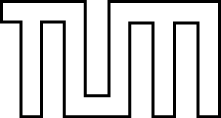
Der Technischen Universität München

Bachelorarbeit in Wirtschaftsinformatik

**Implementation of a Bluetooth touchpad based on Android OS**

Nikolay Kostadinov





Fakultät für Informatik

Der Technischen Universität München

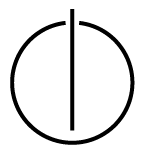
Bachelorarbeit in Wirtschaftsinformatik

**Implementierung eines Bluetooth Touchpads**

**auf Basis von Android OS**

**Implementation of a Bluetooth touchpad based on Android OS**

|  |  |
| --- | --- |
| Author: | Nikolay Kostadinov |
| Supervisor: | Prof. Dr. Uwe Baumgarten |
| Advisor: | MSc. Nils T. Kannengießer |
| Submission Date: | 15.10.2011 |



Ich versichere, dass ich diese Bachelorarbeit selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 15. Oktober 2011 Nikolay Kostadinov

I assure the single handed composition of this bachelor thesis only supported by declared resources.

München, den 15. Oktober 2011 Nikolay Kostadinov

Acknowledgements

First of all I would like to thank my advisor M.Sc. Nils Kannengießer for his outstanding support, supervision, useful suggestions and encouragement which was invaluable for the successful completion of this work. Thanks to Prof. Dr. Uwe Baumgarten for giving me a chance to work on this and other marvelous projects. Thanks to everyone from the chair for operating systems in the TUM as well.

Furthermore, I would like to thank my family for general education and ongoing support. Last but not least, I want to thank my more than talented girlfriend Evgeniya Sofronieva, who turned my slobbery hand-drawings into beautiful graphics.

Abstract (English)

Smart phones are gaining popularity both in the corporate and the entertainment sectors. They are gradually becoming a universal device, able to complete a variety of different tasks and fit into various use case scenarios. This work concentrates on realizing a single scenario and presents a completely new way of using a mobile phone for remote control of notebooks, computers and other Bluetooth-enabled devices.

The aim of this work was to develop a touchpad by using the Android OS as a platform. The touchpad application running on Android phone is able to connect to other systems over the Bluetooth radio technology. By using a set of standard supported drivers, the application provides an input service for the user that is not less powerful than the capabilities of ordinary input devices such as mouse and keyboard. The project not only fulfils this goal, but also introduces an extensible framework, which is extremely easy to implement by developers willing to unleash the power of the Bluetooth communication in combination with the widely supported drivers for input devices.   
 The open source Android operating system has established its place as the most popular operating system, designed to power smart phones and other mobile devices. Devices running this freely distributed OS are less expensive, than other devices with similar hardware specifics. For its openness, user-friendly concepts and developer-friendly software development tools, it became the platform of choice for this project.

Abstract (Deutsch)

Smartphones werden immer populärer sowohl in der Unternehmens- als auch in der Unterhaltungsbranche. Sie werden allmählich zu einem universellen Gerät, das in der Lage ist, zahlreiche Aufgaben zu erfühlen. Deswegen findet es auch in vielen Anwendungfälle einen Platz. Diese Arbeit konzentriert sich auf die Realisierung von so einen Anwendungfall und präsentiert eine völlig neue Art und Weise, wie die Fernsteuerung von Notebooks, Rechnern und anderen Bluetooth-fähigen Geräten, mit Hilfe eines Mobiltelefons betrieben werden könnte.

Das ursprüngliche Ziel dieser Arbeit war es, ein Touchpad auf Basis von Android OS zu entwickeln. Die Touchpad-Anwendung, die aufs Android-Handy läuft kann sich mit anderen Systeme mit Hilfe der Bluetooth-Technologie verbinden. Durch die Verwendung von einer Reihe von Standard-unterstützten Treiber bietet die Anwendung den Nutzer zahlreiche Eingabemöglichkeiten, die nicht weniger mächtig sind als diese, die von üblichen Geräten wie Maus und Tastatur angeboten sind. Allerdings, erfühlt das Projekt nicht nur dieses Ziel. Es wird ein Framework vorgestellt, welche von den Entwicklern sehr einfach zu implementieren ist. Damit können sie Applikationen entwickeln, die sowohl die Vorteile der Bluetooth-Kommunikation, als auch der breit unterstützten Treiber für Eingabegeräte ausnutzen.

Die Open-Source-Betriebssystem Android hat sich in der letzten Jahren als die meistgenutzte Betriebsystem etabliert, die speziell für mobile Geräte entwickelt ist. Geräte, auf die dieses freies OS läuft, sind meistens billiger als andere Geräte mit vergleichbaren Hardware-Spezifikationen. Android ist offen und bietet benutzerfreundliche Konzepte, sowie entwicklerfreundliche  Software-Entwicklungstools. Deswegen ist Android die natürliche Wahl für diesen Projekt.

List of Figures

[Figure 1: Mouse functionality 8](file:///G:\work\EclipseWinRepo\%20android-bluetooth-touchpad\BTTouchpad\thesis\BA.docx#_Toc304398452)

[Figure 2: Graphics pad concept 10](file:///G:\work\EclipseWinRepo\%20android-bluetooth-touchpad\BTTouchpad\thesis\BA.docx#_Toc304398453)

[Figure 3: Frequency hopping pattern [3] 13](#_Toc304398454)

[Figure 4: Bluetooth stack on PC Host and HID [6] 15](#_Toc304398455)

[Figure 5: Bluetooth stack architecture [5] 16](#_Toc304398456)

[Figure 6: SDP connection scheme [5] 20](#_Toc304398457)

[Figure 7: Service record 21](#_Toc304398458)

[Figure 8: Service attribute [5] 22](#_Toc304398459)

[Figure 9: Attributes in different service classes 22](#_Toc304398460)

Contents

1. Introduction
2. Bluetooth technology
   1. Common Bluetooth devices
      1. Mouse
      2. Keyboard
      3. Graphics pad
      4. Bluetooth devices
      5. Bluetooth radio
   2. Bluetooth software stack
      1. Bluetooth protocol architecture
         1. Overview
         2. Logical Link Control and Adaptation Protocol
         3. Host Controller Interface
         4. Link Manager Protocol
         5. Link Control
         6. Radio Frequency
      2. Service Discovery Protocol
         1. Overview
         2. Service record
         3. Record handle
         4. Service attribute
         5. Service class
3. Introduction

As Android becomes more and more popular, third-party developers are producing an increasing number of Android applications. The applications are small and useful programs utilizing different combinations of hardware features. Although, most of the mobile devices running Android are supporting Bluetooth communications, very little is done for realizing the vision that a mobile phone could be used as an universal remote control, that could connect to virtually any notebook, computer or other type of device supporting the Bluetooth technology and a standard set of drivers for input services, such as the HID drivers.

The main problem is the missing support for these drivers. The developer would have to dive deep into the lower levels of the operating system’s architecture and write programs interacting directly with the Bluetooth stack as part of the operating system’s core. The input service that would be provided by the application must be described according to the HID protocol. Information on how to do this is also spare. Then the resulting service description has to be inserted in the registry of an existing SDP module, which is responsible for making it publicly available, so the computer could find it and read it. Since the computer and respectively the user are aware of the service, a communication channel over Bluetooth must be established, so the service would be utilized.  Although, Bluetooth communication is generally supported by APIs that are part of the Android framework, developers are not provided with access to the lower level communication protocols, such as L2CAP. However, if communication on the higher levels is possible, there must be a way of accessing the protocols beneath.

The Bluetooth touchpad project is presenting a solution for each of these problems. A service description for a Bluetooth mouse and keyboard is defined and passed to the SDP server of the Android operating system. On the other hand, the resulting application is able of establishing communication by using the required protocols, although this feature is not officially supported. The solutions are packed in an extensible framework, which could be easily implemented by developers and used in other projects. The Bluetooth touchpad app, implemented on top of the framework is abstracting from the mouse and keyboard input specifics and thus provides a number of totally different input capabilities by using the phone’s motion sensors, the phone’s display and even the voice recognition API.

In this work, first the Bluetooth technology and also some specifics of common Bluetooth input devices are presented. The SDP is also explained, since it is playing an important role in the Bluetooth stack. Afterwards, the Android platform and the development phone used in the project are briefly introduced. After building up this foundation of knowledge, the implementation of the framework is reviewed in detail. In the last chapter results of quality tests are provided and the application’s performance is measured.

1. Bluetooth technology

In order to implement a Bluetooth touchpad based on the Android OS, one must first understand what Bluetooth is and how it works. In this chapter, first some common Bluetooth devices are introduced and then the communication technology behind them is reviewed in detail. Understanding both the hardware specifics of the devices and the architectural model behind Bluetooth is crucial in order to implement a new Bluetooth device from scratch.

2.1 Common Bluetooth devices

The task of realizing a Bluetooth touchpad on a mobile phone, running Android OS, could be moreover approached as simulating the behavior of a wireless Bluetooth mouse and keyboard. Subsequently, in order to complete this task, the functionality of both, as well as their physical capabilities and structure must be considered. In the following section the hardware specifics of regular corded mouse and keyboard, as well as a graphics pad device are successively discussed. Then the hardware capabilities of such wireless devices and the underlying Bluetooth technology are further reviewed. Finally, the process of physical connection and identification of Bluetooth devices, as well as some communication security concerns are briefly looked through.

2.1.1 Mouse

The main goal of the modern mouse is to translate the motion of your hand into signals that the computer can use as an input method. A simple, standard featured mouse consists of two buttons (left and right) and a scroll wheel, which could also act as a third button. Furthermore, the mouse motions on a flat surface are translated into the motion of a cursor on the computer’s display.



Figure : Mouse functionality

As shown on the figure 1 above, the mouse functional capabilities could be shortly described as the following three basic user interactions:

1) Pressing left/right button

2) Scrolling the wheel up/down

3) Mouse motion on surface

A mouse consists of several sensors that could handle and translate the user interaction into specifically formatted data, which is then sent to the computer for further processing.

* + 1. Keyboard

The modern computer keyboard originates its design from the mechanical, non-electric typewriters invented in the 19th century. Today, it is used to type text and numbers into computer programs, where the interpretation of key presses is left to the underlying software programs. Keyboards often have different or additional keys depending on the manufacturer or the operating system they are designed for. However, the different keyboard’s keys have similar size and shape. Furthermore, they are placed in a similar pattern, no matter what language is represented. The user interaction consists of pressing a single or a combination of keys at the same time. The keyboard reports all key presses to the operating system by sending them as specifically encoded data.

2.1.3 Graphics pad

A graphics pad, (also called drawing tablet) is modern computer input device that enables the user to hand-draw graphics, similar to the way a person can draw images with a pencil on paper or with fingers and paint on canvas. The ability to detect some or all of the pressure of the stylus and representing them on the computer display is considered to offer a natural way to create computer graphics. Figure 2 below is showing the concept of the device.



Figure : Graphics pad concept

Similarly to the mouse, the pad is able to capture the movement of the stylus or the user’s finger on its surface and translate it into the motion of a cursor. Since the behavior of this device is similar to the behavior of the mouse, the functionality of this device is also included in the Bluetooth touchpad implementation.

2.1.4 Bluetooth devices

Other than a regular mouse or keyboard, a wireless device is not using a cable connection for sending the data, but radio frequency technology. Radio frequency devices consist of two components: transmitter and receiver. The transmitter is placed in the device and is able of sending radio signal that encodes information about the user’s actions. In addition, the receiver is connected to the computer and is respectively accepting, decoding and passing the information to the computer’s operating system. Bluetooth is one of the most popular radio frequency technologies that wireless mice and keyboards use. Bluetooth is shortly described on the main page of its vendor- the Bluetooth Special Interest Group[1] as “short-range communications technology that is simple, secure, and everywhere. You can find it in billions of devices ranging from mobile phones and computers to medical devices and home entertainment products. It is intended to replace the cables connecting devices, while maintaining high levels of security.” (Bluetooth Basics [2]) Indeed, the fact that Bluetooth receivers can accommodate multiple Bluetooth peripherals at the same time is one of the main reasons why the technology has established its status as one of the most popular wireless standards ever.

2.1.5 Bluetooth radio

Almost all of the electronic devices today utilize radio frequencies (RF) to communicate with other devices. In order to avoid conflicts during communication, different devices use different frequencies. One of the benefits of the radio frequency technology is that it does not need a clear line of sight between the transmitter and the receiver. Unlike the infrared based communication technology, used for example in TV remote controls, the wireless signal can pass through barriers such as furniture or walls. What is more, the RF technology provides variety of other advantages for the wireless devices - the RF transmitters and receivers are very inexpensive, tiny and light weight. Furthermore they require low power and can therefore run on batteries.

Bluetooth is one the most widely used RF technologies. It allows a large number of different devices to connect to each other such as: phones, printers, notebooks, tablets etc. Bluetooth devices usually have a range of 5 to 10 meters and operate in the 2.4 GHz range by using RF. One Hertz (Hz) indicates thousand cycles per second or thousand electromagnetic waves per second. Subsequently one Megahertz is one million and one Gigahertz (GHz) is one billion cycles per second. [3]

In order for two Bluetooth devices to establish communication channel and transmit data, they must be “paired”. Pairing indicates the process of determining a common frequency and also a common communication code, resulting in a communication channel. Consequently pairing makes it possible to filter out interference from other RF devices. There are several methods of pairing, depending of the type of device and its manufacturer. If both devices have display, which is the case when pairing an android phone and a Bluetooth capable computer, the “Numeric Comparison” is usually used. A 6-digit numeric code is shown on each display and the user is asked to compare the numbers to ensure they are identical. Once the comparison is successful, one could confirm the pairing and data transfer between both devices may start. If the user has confirmed on both devices and performed the comparison properly, this method provides significant protection from one of the most common attacks - “man in the middle”. [4]

On the other hand, devices with limited input capabilities, such as Bluetooth mice and keyboards either require the user to enter a pin, which is predefined and usually easy to guess (“0000” or “1234”) or they do not require any user interaction at all. Obviously, this type of pairing does not provide protection against “man in the middle” attacks. As a consequence, a Bluetooth touchpad realized on an Android phone provides better protection then an ordinary Bluetooth mouse and keyboard, since it provides possibility for numeric comparison as part of the Android operations system.

In addition, Bluetooth devices use encryption schemes to encrypt data in unreadable format, as well the frequency-hopping method. This method causes the two Bluetooth devices to automatically change frequencies. Frequency-hopping “divides the band into 79 channels (each 1 MHz wide) and changes channels up to 1600 times per second”. (Bluetooth security mechanisms [3]) Every Bluetooth device has a physical clock responsible for this frequency change. Therefore, in order to establish a communication channel the devices needs to synchronize their clock and their frequency hopping pattern, a piconet is created. The concept of frequency hopping pattern is shown on figure 3 below. [3]



Figure : Frequency hopping pattern [3]

A piconet consists of master and between one and seven slaves. The master is responsible for setting the clock time and also the hopping pattern. Slaves, on the other hand, accept the master’s settings. Moreover, a Bluetooth device could be master in only one piconet, but a slave across multiple piconets. Generally, this frequency hopping technology strengthens the security on the Bluetooth protocol, because “any device not belonging to the piconet is unable to participate in communications by sending or listening to the data exchanged because it does not have access to the frequency hopping sequence. “(Bluetooth security mechanisms [3])

2.2 Bluetooth software stack

After taking into consideration the hardware specifics of Bluetooth mouse and keyboard, the next logical step of the process of implementation of Bluetooth touchpad on an Android device is to gather deep understanding of how exactly the Bluetooth technology works not only on the physical, but also on the higher levels. In the following chapter the Bluetooth stack architecture is discussed in detail and each abstraction level in the protocol stack, relevant for the implementation of the touchpad, is separately reviewed. Furthermore, a special attention is given to a service layer protocol- Service Discovery Protocol (SDP). The SDP is thoroughly discussed in the second subsection, since a good understanding of the protocol is required when implementing an input service via the standardized Human Interface Device protocol.

2.2.1. Bluetooth protocol architecture

The main goal of the Bluetooth technology is to replace the cables, connecting portable and also fixed consumer electronic devices. The head advantages of Bluetooth are outlined in the Bluetooth Core Specification as “robustness, low power, and low cost”. And furthermore, “many features of the core specification are optional, allowing product differentiation”, creating the foundation of an open standard. (Bluetooth Core Specification v2.0, Vol. 1, Part A, p. 13 [5])

2.2.1.1 Overview

Bluetooth has a layered architecture consisting of variety of protocols with different level of abstraction. The low-level core protocols are defined by the Bluetooth Special Interest Group organization. In time additional protocols from other organizations and vendor bodies have been adopted and all together have resulted in an open specification for a radio system that provides the network infrastructure to enable short range wireless communication. Although, Bluetooth stack implementations tend to vary across different vendors, protocols like LMP, L2CAP and SDP are considered mandatory and found in each stack realization. In addition, other protocols such as the HCI and RFCOMM have established as universally supported. [5] The Bluetooth protocol stack could be logically divided into two separate protocol stacks - the “controller stack” and the “host stack”. In general, the controller stack is implemented in low cost silicon device that contains a microprocessor and the Bluetooth radio. On the other hand, the host stack, which is responsible for the higher level data is either implemented as a part of the operating system (Bluez is the Linux Bluetooth software stack and is part of the Linux kernel) or is additionally installed (Widcomm is Bluetooth stack for windows developed by Widcomm Inc, and must be separately installed). The host and the controller stack are connected through the HCI pipe, providing standardized communication means between them. Alternatively, in some integrated devices such as Bluetooth mice and keyboards both host and controller stacks as well as HCI are naturally run on the same microprocessor. [6]

Figure 4: Bluetooth stack on PC Host and HID [6]

The graphic visualization of figure 4 illustrates scenario of communication between a PC host and HID as an example implementation and is described in the HID Specification: “The host is a personal computer and has the upper layers of the Bluetooth software running on its native processor and is connected to a Bluetooth radio module via a transport bus such as USB. The HID in this example has its firmware embedded with the radio firmware, running on the same CPU, for the lowest possible cost implementation.” (HID Specification, p. 20 [6])

Of greater interest for the implementation of Bluetooth touchpad based on Android OS are the four lowest layers, described on figure 5 below, as well as one common service layer protocol - SDP (Service Discovery Protocol), that is not shown for clarity. In the following, the role and specifics of the low level protocols of the Bluetooth stack, as well as their core functional blocks are separately explained.



Figure : Bluetooth stack architecture [5]

2.2.1.2 Logical Link Control and Adaptation Protocol [5]

The Logical Link Control and Adaptation Protocol (L2CAP) is the first layer of the host stack and thus provides connection-oriented, as well as connectionless data services to the higher level protocols by supporting protocol multiplexing, segmentation and reassembly of packets. It allows applications or upper level protocols to send and receive data packets up to 64 kB. One of the main tasks is to handle segmentation and reassembly of packets. The L2CAP layer contains two architectural blocks, which are using the L2CAP protocol to communicate- channel manager and L2CAP resource manager.

According to the Bluetooth Core Specification, the channel manager is “responsible for creating, managing and destroying L2CAP channels for the transport of service protocols and application data streams”. (Bluetooth Core Specification v2.0, Vol. 1, Part A, p. 24 [5]) In order for a L2CAP channel between two Bluetooth devices to be established, the channel managers of both devices must communicate with each other by using the L2CAP protocol and connect their endpoints to the appropriate entities. The channel managers then have to interact with their corresponding local link managers and create new logical links and configure them to provide QoS for the specific type of data, which is to be exchanged.

The functionality of the resource manager is outlined in the Bluetooth Core Specification as “managing the ordering of submission of PDU fragments to the baseband and some relative scheduling between channels to ensure that L2CAP channels with QoS commitments are not denied access to the physical channel due to Bluetooth controller resource exhaustion”. (Bluetooth Core Specification v2.0, Vol. 1, Part A, p. 24 [5]) For instance, this behavior is required because the Bluetooth controller, implementing the controller stack, does not have infinite buffering capability and neither has the HCI pipe limitless bandwidth. Additionally the L2CAP Resource Manager is also able to “carry out traffic conformance policing to ensure that applications are submitting L2CAP service data units within the bounds of their negotiated QoS settings.” (Bluetooth Core Specification v2.0, Vol. 1, Part A, p. 24 [5]) Since the Bluetooth data transport model assumes “well-behaved” applications, it is left for the developer to deal with this problem, when implementing on the top of the Bluetooth protocol stack.

Understanding the L2CAP protocol layer has crucial importance when implementing a HID and respectively realizing the Bluetooth touchpad on Android device, since the channel connection on the L2CAP level is the highest level on which communication is taking place. Although, a deep understanding of lower abstraction level and the protocols from the controller stack is not required for the implementation, these are briefly explained, since they are the basis underneath the L2CAP communication.

2.2.1.3 Host Controller Interface [5]

The Host Controller Interface provides standardized communication between the host and the controller stacks. The interface is responsible for creating both stacks independent in such manner, that each of them could be swapped with minimal to none adaptation. There are several HCI standards, each using a different hardware interface to transfer the same data packets between host and controller stack. PCs, for example, use USB (Universal Serial Bus), mobile devices such as phones, PDAs and tablet computers, on the other hand, use UART (Universal Asynchronous Receiver Transmitter). Conclusively, the HCI transport layer provides a common device driver interface to USB, UART and others by abstracting away transport dependencies.

2.2.1.4 Link Manager Protocol [5]

A Bluetooth device’s Link Manager Protocol (LMP) is responsible for the link setup between Bluetooth units. It also carries out the authentication and encryption by generating, exchanging and checking the link and encryption keys, as well as controlling and negotiating of baseband packet sizes. There are two entities with different functional tasks using the Link Manager Protocol- Device Manager and Link Manager.

The device manager is responsible for the general behavior of the Bluetooth device by handling all the operations of the Bluetooth unit, that are not related to data transfer, such as “ inquiring for the presence of other nearby Bluetooth devices, connecting to other Bluetooth devices, or making the local Bluetooth device discoverable or connectable by other devices.” (Bluetooth Core Specification v2.0, Vol.1, Part A, p. 25 [5]) Naturally, in order to perform all this tasks the device manager has to gain access to the lower baseband layer and communicate with the Baseband Manager.

The link manager, as outlined by its name is responsible for “creation, modification and release of logical links” and same as the device manager accomplishes this task by communicating with the corresponding link manager in the other Bluetooth device by using the link manager protocol. Moreover, the Link Manager is controlling some transport attributes such as “enabling of encryption on the logical transport, the adapting of transmit power on the physical link, or the adjustment of QoS settings for a logical link.” (Bluetooth Core Specification v2.0, Vol. 1, Part A, p. 25 [5])

2.2.1.5 Link Control [5]

The Link Control layer enables the physical RF link between Bluetooth devices to form the piconet by using both circuit and packet switching. Two important architectural blocks are located on the Baseband and Link Control layer - the Baseband Resource Manager and the Link Controller.

The Baseband Resource Manager is responsible for providing access to the radio medium and has two major functions. According to the Bluetooth Core specification it has a “scheduler that grants time on the physical channels on the physical channels to all of the entities that have negotiated an access contract. “ The other function is respectively to negotiate the access contracts with the corresponding entities on the higher level. An access contract is “a commitment to deliver a certain QoS that is required in order to provide a user application with an expected performance.” (Bluetooth Core Specification v2.0, Vol. 1, Part A, p. 25 [5])

The Link Controller is responsible for both encoding and decoding of Bluetooth data packets on the physical channel. “The link controller carries out the link control protocol signaling (in close conjunction with the scheduling function of the resource manager), which is used to communicate flow control and acknowledgement and retransmission request signals.” (Bluetooth Core Specification v2.0, Vol. 1, Part A, p. 26 [5])

2.2.1.6 Radio Frequency [5]

The radio frequency layer is always the lowest layer in every implementation of the Bluetooth architecture. It builds up the physical channel and is accordingly responsible for transmitting and receiving packets of data. As a result, everything in Bluetooth runs over the RF Layer, defining the requirements for the Bluetooth radio transceiver.

2.2.2 Service Discovery Protocol

The Service Discovery Protocol (SDP) is used to exchange information about services, which devices are providing for each other. Implementing the Bluetooth touchpad includes providing a service of mouse and keyboard input on the Android device side, which the computer side has to discover and consequently use. It is indeed, the SDP that makes it possible for the service, hosted on the phone, to be discovered by the PC host. For this reason, SDP is reviewed in detail.

2.2.2.1 Overview [5]

In the Bluetooth protocol stack SDP is bound directly to L2CAP. The L2CAP data channels are used to transfer information between two Bluetooth devices, about the services one of them is providing. In particular, client applications are enabled to discover the services, provided by server applications. Furthermore, clients can gain access to the attributes of these services, such as the type/class of service and other information needed to utilize the service. Figure 6 below shows the simplified connection scheme of SDP client and server.



Figure 6: SDP connection scheme [5]

The SDP server maintains a list of service records. Each service record is responsible for exactly one service and contains its specifics and information needed for its utilization. A client can retrieve a single service record or all records maintained by the SDP server with a single SDP request. However, a separate connection must be established if the client decides to use the service. SDP is only responsible for discovery of services and providing the necessary information about them, but not for their utilization. For example, while the SDP request and response are exchanged via the L2CAP layer of the Bluetooth stack of both devices, a service might require a connection on higher level, such as RFCOMM.

Bluetooth devices are allowed to have only one SDP server and only one SDP client. Depending on their general purpose, some devices do not need SDP server and have only client or the opposite. If multiple applications are providing services on the same device, a single SDP server is responsible for making the service record of each of them available to the SDP clients. In the same way, multiple client applications can use a single SDP client to send requests to one or more SDP servers on their behalf.

2.2.2.2 Service record [5]

In order to explain the meaning of the service record, the Bluetooth Core Specification gives a short definition of a service: “A service is any entity that can provide information, perform an action, or control a resource on behalf of another entity. A service may be implemented as software, hardware, or a combination of hardware and software.” (Bluetooth Core Specification v2.0, Vol. 4, Part B, p. 118 [5]­) The entire information about a service, provided by a Bluetooth device, is maintained by the SDP server as service record. Furthermore, the service record presents different information aspects about the service by using service attributes. Figure 7 shows the service record’s structure.



Figure 7: Service record

2.2.2.3 Record handle [5]

For instance, an attribute mandatory for all service records is the service handle. This is a 32-bit unique within the SDP server identifier for the service record. As outlined in the Bluetooth Core Specification, “if SDP server S1 and SDP server S2 both contain identical service records (representing the same service), the service record handles used to reference these identical service records are completely independent. The handle used to reference the service on S1 will be meaningless if presented to S2.” (Bluetooth Core Specification v2.0, Vol. 4, Part B, p. 119 [5] ­) To put it more simply, the service record handle is used to reference a specific service within a SDP server/client. This is particularly important for the task of implementing the Bluetooth touchpad. Since the Android device does not have a service record describing mouse and keyboard input service, one should be added in the SDP server, when the service is provided by certain application. When the application and correspondingly the service is not active any more, we use the predefined record handle to specify which service record, the SDP server should remove from its list.

2.2.2.4 Service attribute

Each service attribute describes a unique characteristic of service and consists of attribute ID and attribute value as shown on the figure 8.



Figure 8: Service attribute [5]

The attribute ID is precisely described in the Bluetooth Core Specification as “16-bit unsigned integer that distinguishes each service attribute from other service attributes within a service record”, and further “identifies the semantics of the associated attribute value.” (Bluetooth Core Specification v2.0, Vol. 4, Part B, p. 120 [5] ­) Moreover, all attribute IDs for a service are specified by a service class definition, which respectively also assigns the meanings of the corresponding attribute values. Thus, the attribute value is represented as data element with variable length. In general, any type of data element is permitted for the attribute value, as long it is specified in the service class and a corresponding ID is assigned for it.

The roles of attribute ID, attribute value and service class could be easily demonstrated by the following example, shown on figure 9:



Figure 9: Attributes in different service classes

The service class A defines the value for the attribute with the ID 12345 as a string, containing the name of the service’s author. Every service, instance of A (A1, A2 …) will have an attribute with ID 12345 and its value will be specifying the author. Service, which is instance of B might also have attribute with ID 12345 that has totally different semantics.

2.2.2.5 Service class

Similarly to the object-oriented philosophy each service is an instance of a service class, which provides the definitions of all attributes in the service. The attribute definition consists of unique within the service class numeric value for the attribute ID on the one hand, and description of the semantics and format of the attribute value, on the other. Although, different service classes might define different semantics for the value of attributes with the same attribute ID (as shown in the example on figure 9), there are some attributes that are common to all services. For example, the ServiceClassIDList attribute is found in every service. It represents a list, the first entity of which is the UUID. The UUID is universally unique identifier for the service that is guaranteed to be unique across all space and time.

For the most part, a service record contains attributes from several service classes that are related to each other in such manner, that each service class is a subclass of another service class. As simple as it is, “a service subclass definition differs from its superclass in that the subclass contains additional attribute definitions that are specific to the subclass.” (Bluetooth Core Specification v2.0, Vol. 4, Part B, p. 122 [5] ­) Further following the principal of inheritance, in order to define a new service class that is a subclass of an existing service class, one must only define the additional attributes, which are specific to the new service.

The ServiceClassIDList is of significant importance, since its value is the first that is being examined when processing a service record. The ServiceClassIDList attribute of a service is a list containing the unique identifier of the service, followed by the unique identifiers of all the superclass services. The identifiers in the ServiceClassIDList attribute are listed starting from the most specific class and ending to the most general class.

The Bluetooth core specification gives an illustrative example, representing the ServiceClassIDList attribute value of a color postscript printer with duplex capability:

DuplexColorPostscriptPrinterServiceClassID,

ColorPostscriptPrinterServiceClassID,

PostscriptPrinterServiceClassID,

PrinterServiceClassID

As previously outlined, the first UUID in the list represents the unique identifier of the service in question, followed by the UUIDs of gradually becoming more general services.