Mobile Applications with J2ME and JSR82: Bluetooth enabled Java Applications for Mobile Phones

Student: Borja Gomez Zarceño
Mentor: Dr. Stephan Rupp

Institute of Commucication Networks and Computer Engineering

Abstract

Bluetooth came in so as to make the use of home electronics more friendly to users. It permits any kind of electronics equipment to establish their own connections, without cabling an without direct actions from the user. Java language programming and platform appear to join Bluetooth technology to be the base of next generation wireless applications. Developers have built a platform providing a run time environment for embedded systems upon small devices. This paper gives and overview of Bluetooth and introduces the Java platform for developing Mobile Applications. This Java platform (J2ME) and the Java Application Programmer Interface for bluetooth (JSR 82) will be explained.

1. Bluetooth

There is a variety of ways to connect electronic devices: a cable connecting a processing unit to a display, a data cable connecting a mobile telephone to a PC, radio waves connecting a cordless telephone to a base unit, infrareds do with a remote control to a TV.

Bluetooth appears to be a different way to build up connections between devices placed in proximity. This radio frequency technology can be thought as a cable replacement technology that will not only replace cables, but be the base to develope next generation wireless applications, which will be built upon this technology. It offers a variety of other services, apart from connecting devices, and it creates opportunities for new usage models.

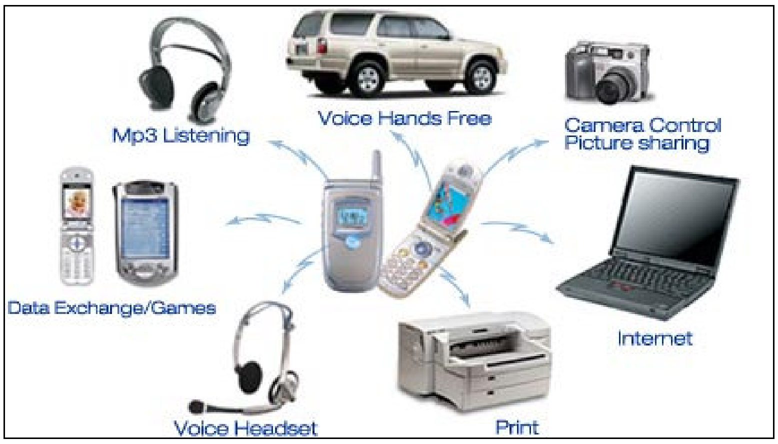

Figure 1.1 Bluetooth connections

Among the bluetooth characteristics some of them are remarkable: bluetooth is wireless; automatic; cheap; deals with data and voice; signals are omnidirectional and a device can send to multiple devices at the same time; signals can go through walls; bluetooth is secure, it uses frequency hopping what makes it difficult for the communication data to be intercept; it is standardized, governments have arranged a standard about the ISM band so that it is possible to use devices wherever.

Bluetooth technology is to be used for the next applications: *file transfer*, the content of one mobile phone can be dragged into another one; *ad-hoc networking*, network spontaneously form networks that persist as long as they are needed; *device synchronization,* data handled in one device is updated in all devices; hands-free packages for accessing the telephone service while driving; *peripheral connectivity*; *mobile payments,* a bluetooth enabled mobile phone can communicate with a bluetooth enabled machine to buy something and pay for it.

1.1 Topology

Bluetooth devices are grouped in piconets. Each piconet consist of a single master and different slaves. A master and a slave build a point-to-point communication. If there are more than one slaves, a point-to-multipoint connection is established. The master is the unit which initiates the communication. A device in one piconet can connect to a device belonging to another piconet, forming a scatternet. As we can see in the figure, in a scatternet, a master in one piconet can be a slaved in other piconet.

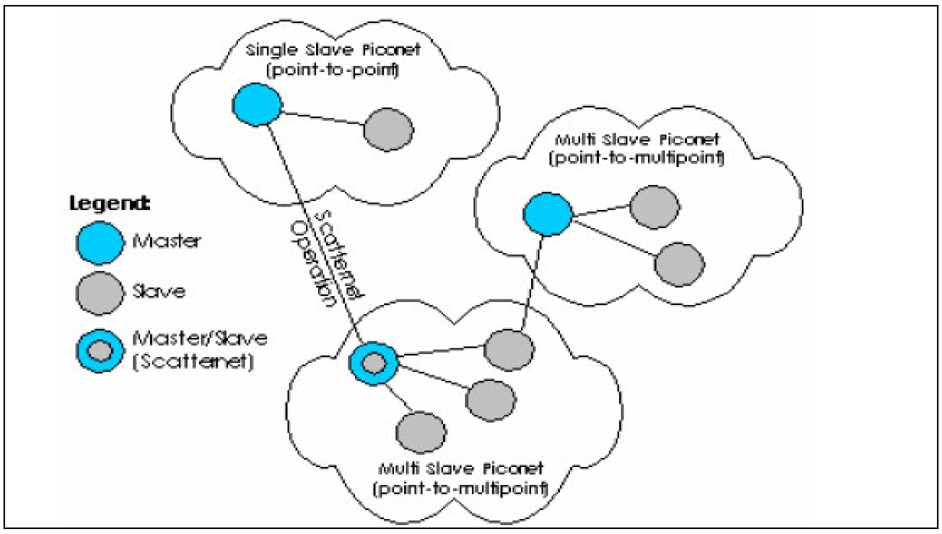

Figure 1.2 Bluetooth Network Topology

Bluetooth uses a time division multiplex scheme for transmission, dividing time into slots. To support full-duplex transmission, the master device uses an even-numbered slot, while slaves use an odd-numbered slot.

1.2 Protocol Stack

The bluetooth specification ensures that all the devices supporting this technology are able to communicate with each other no matter the part of the world where they are. A well defined protocol stack is then defined. A layer level view of the architecture of the Blueetooth technology is shown as follows

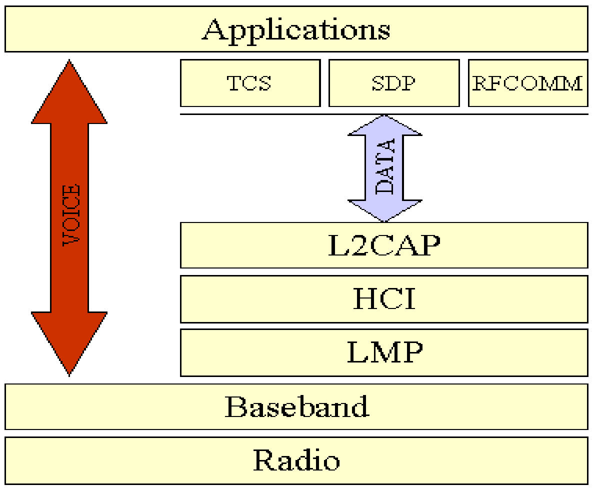

Figure 1.3 Protocol Stack

1.2.1 The Radio Layer

The radio layer is the physical wireless connection. The modulation is based on fast frequency hopping, allowing to avoid interferences with other devices using the same band. The frequency band from 2.402 GHz to 2.480 GHz (ISM band) is divided into 79 channels, each one 1MHz of bandwidth. The transmission frequency hops from one channel to another about 1600 times per second.

1.2.2 The Baseband Layer

The baseband layer controls and send data packets through the radio link. It does provide a transmission channel for data as well as for voice. This layer provides two types of links: A *Synchronous Connection Oriented* (SCO) link for voice and an *Asynchrorous Connectionless Link* (ACL) only for data . To ensure data integrity, when there has been a failure in a packet transmission, ACL support retransmission. In SCO there is no retransmission of packets.

SCO links are point to point symmetric connections, where time slots are reserved for each direction of transmission. The master transmits in one time slot, whereas the slave is allowed to answer back in the next immediate time slot. At most, there can be, for a master, three SCO links at the same time either to a single or to multiple slaves. On the other hand, in ACL links there is no reservation of time slots.These links se the time slots that are not being used by SCO connections. These links support point to multipoint transmissions, in which the master may address specifically a slave or not (broadcast messages). In the former case, only slaves addressed may respond during the next time slot.

1.2.3 The LMP, HCI and L2CAP Protocols

The Link Manager Protocol (LMP) makes use of the services from the layer below, that is, from the links set up by the baseband layer to establish connections and to manage the piconets.

The Host Controller Interface is the layer providing an interface between software and hardware functions. From there to the top of the protocol stack, functions are implemented by sofware. It is a driver interface for the physical bus connecting these components. Sometimes this layer is not required.

The Logical Link Control and Adaptation Protocol (L2CAP) negotiates QoS. As well, this layer receives application data and changes it to Bluetooth format. It provides connection-less and connection-oriented services to upper layers. This is done with multiplexing capabilities in order to differenciate the protocols above: SDP, RFCOMM and TCS.

1.2.4 RFCOMM, SDP and TCS protocols

The RFCOMM protocols is a transport protocol emulating a RS232 serial ports. It permits up to 60 different connections between bluetooth enabled devices.

The Service Discovery Protocol allows applications running over bluetooth to find out services and service characteristics registered in other bluetooth devices. The TCS define the interface needed to connect and disconnect a call, including signalling the devices to participate in the connection.

1.3 Bluetooth profiles

To assure interoperabilty among bluetooth devices, provided by different vendors and manufacturers, a bluetooth device must conform to a particular profile, which defines capabilities and roles for applications, defining mandatory options and parameters for each protocol of the stack. A profle can be seen then as a vertical sight of the protocol stack. Some of these profiles are slightly described as follows

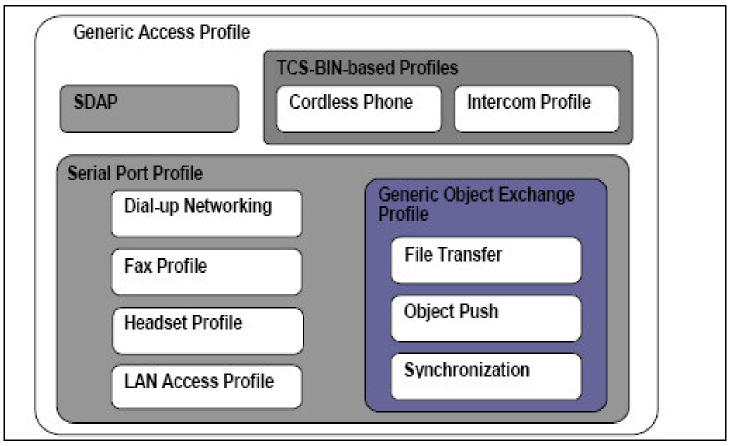

Figure 1.4 Bluetooth Profiles

1. The Generic Access Profile (GAP): for defining connection procedures, as well as link management and device discovery. That is, for describing the use of the lower layers of the protocol stack (LC and LMP).

2. The Service Discovery Application and Profile (SDAP): for defining characteristics in order to discover services around and to bring available information about these services. It defines protocols and procedures to be used by discovery applications in devices using the service discovery protocol.

3. The Serial Port Profile (SPP) for establishing connections.

4. The LAN access profile: for defining how to access a LAN and PPP procedures.

2. J2ME

The Java2 MicroEdition platform joins the Bluetooth technology to be one of the most exciting offerings in the wireless industry. J2ME was defined by Sun Microsystems to meet the new needs of Java developers working on consumer and small embedded systems. J2ME itself is not a specification, but a group of them defining how Java technology is upon devices with few resources compared to a PC. This platform is portable, so applications follow the Java philosophy "once written run anywhere". It appears as a tool to let us write custom applications and run them on mobile Bluetooth enabled devices.

Devices using this Java Platform are: PDAs, cell phones, television set top boxes, remote telemetry units… and other embedded devices. These are heterogeous devices regarding processor power, memory, persistent storage and user interface. It is difficult to provide an optimal functionality for all these embedded devices due to this heterogenity. There was a first division of the devices into two sections considering the resources above mentioned, but not taking into account the function or use of the device. For the latter reason, there was a subdivision into classes of devices.

There are two J2ME configurations corresponding to the former division above mentioned: Connected Device Configuration (CDC) and Connected Device Limited Configuration (CDLC), each one providing a JVM (Java virtual machine) and different libraries and APIs in order to create a run-time programming environment for a group of devices. Configurations provide a common denominator subset of Java suitable for the characteristics of devices to which they are intended.

Within a configuration, there is still much heterogeneity in aspects such as user interface, function and usage. Configurations do not specify the user interface toolkit and persistent storage APIs. This is the task of profiles.

A profile is a set of JAVA APIs for a particular class of device, considering its function, such as mobile phones. It is built over the platform provided by the correspondent configuration. It is meant to couple this configuration. The Foundation Profile and the Mobile Information Device Profile (MIDP), correspond to CDC and CDLC respectively. For example, MIDP 2.0 joins CLDC to provide a run-time environment.

J2ME defines a small core API defines a core API to be implemented in every J2ME compatible device, deployed in different configurations. The next sets fo Java classes are to be in every configuration: java.util and java.lang