Android real-time audio communications over local wireless

Román Belda, Pau Arce, Ismael de Fez, Francisco Fraile and Juan Carlos Guerri

Instituto de Telecomunicaciones y Aplicaciones Multimedia,
Universitat Politècnica de València,
8G Building - access D - Camino de Vera s/n - 46022 Valencia (Spain)
Corresponding author: robelor@iteam.upv.es

Abstract

This paper describes an Android mobile application that allows voice communications through short-range wireless networks, mainly Bluetooth and Wi-Fi. The application is able to replicate as close as possible the behavior of a two-way radio device. The application is designed to receive audio streams from multiple devices simultaneously and to send them. The main design considerations of the application, such as audio recording and playing, audio coding or data transmission, are explained through the paper.

Keywords: Wireless networks, Android, data transmission, audio coding.

1. Introduction

Application distribution platforms for mobile devices, such as the App Store or the Android Market, have promoted the appearance of an ever-increasing number of applications, drastically changing the landscape of the market. Although Apple pioneered this change, the convenience of the Android platform for both device manufacturers and application developers has established the Android Market as one of the most relevant application distribution platforms. Due to the fast growing number of Android terminals worldwide, application developers target this platform seeking for a massive number of potential users.

There are many different categories of applications such as games, infotainment or access to social networks. Users can even find applications that supersede the native communication applications of the phone, such as messaging, audio or video communications. The success of this kind of applications highlights the fact that although phones allow users to text or call each other, there are many use cases for which the native communication applications just do not fit.

Regarding voice communications, there are different use cases traditionally addressed by other kinds of devices different from mobile phones. Two-way radio transceivers allow peers to establish a voice communication without needing additional infrastructure, provided that the devices are in range. This is very convenient in different scenarios, especially where there is no coverage of cellular networks or just when users cannot afford maintaining an ongoing call alive for unlimited periods of time. Smartphones have great connectivity thanks to technologies such as Wi-Fi or Bluetooth, but they are not provided with voice applications allowing users to use them as two-way radios with these technologies. This is the objective of the application hereby presented, Android Intercom: turn Android terminals into two-way radios using their connectivity capabilities in the best way possible.

The rest of the paper is structured as follows: the next section explains the main design considerations of the developed application, putting special emphasis on the different wireless networks. Section 3 explains the main characteristics of the Android Intercom application. Finally, some conclusions are presented.

2. Design considerations

This section describes the different design considerations taken into account during the development of the application. Specifically, the next subsection shows the general

Smartphones have great connectivity but they lack of direct real time voice communication.

architecture of the system, in which the main blocks of the application are presented: audio recording and playing, audio coding and data transfer. These blocks are explained in the following subsections.

2.1 Architecture

Fig. 1 shows a general architecture of an audio transmission application. It shows a typical chain of an audio flow. Firstly, audio is digitalized and recorded. Later, the recorded data are usually packetized in samples of 20 or 30 ms. These samples are encoded to minimize the transmission bit rate and finally are transmitted. In the developed application, data transmission is made through wireless networks, mainly Bluetooth and Wi-Fi. When the audio samples arrive at the listener device, the samples pass by an audio decoder, which decode the data and send them to an audio player module where the audio is reproduced.

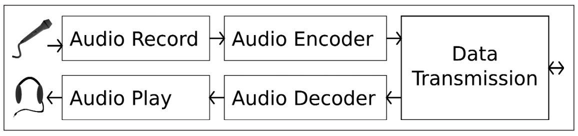


Figure 1. *System architecture.*

The application hereby presented is bidirectional, that is, the different devices work as transmitters and as receivers. The figure shows a typical transmission from one device to another.

Moreover, Fig. 1 also reflects the main points to take into account in the development of the application. Firstly, the audio recording and playing blocks allow to capture and playback audio, respectively. Audio needs to be encoded and then decoded through the corresponding blocks and finally, the audio must be transmitted over a channel. An extended analysis of each block is given below.

2.2 Audio recording and playing

In the development of the audio recording and playing blocks, it must be taken into account that in Android the major part of applications run on a dedicated instance of the Dalvik Java Virtual Machine (JVM). This characteristic makes applications very sensitive to Garbage Collector pauses, and that is why all Android multimedia Application Programming Interface (API) actually relies on native code [1].

Regarding audio, there are two Java-level APIs available:

- A high level API represented by MediaPlayer and MediaRecorder classes. This API has been built for encoding [2], decoding and accessing media content. However, this API does not provide methods to access streams in a real-time manner, which makes it unfeasible to develop applications that require real-time audio processing.
- A low level API represented by AudioTrack and AudioRecord classes. These classes are Java wrappers of the "libmedia" built-in native API.

Furthermore, two additional multimedia APIs have been added on newer Android versions. They are native APIs and are based on popular cross-platform multimedia systems. The APIs are:

- OpenSL ES™ 1.0.1. Starting from Android API-level 9.
- OpenMAX AL™ 1.0.1. Starting from Android API-level 14.

These native APIs are targeted to applications that mostly run native code. Before these libraries appeared, applications like games, which usually run native code such as Open GL, had to control audio from Java code instead, which resulted in unnecessary overhead.

All mentioned audio APIs work over the sound driver system. But there is no direct access to the sound system. In fact, there was no a standardized sound system. Only the latest Android version (Ice Cream Sandwich) has established ALSA (Advanced Linux Sound Architecture) as the default sound system.

The lack of a good audio system from the beginning has caused the fragmentation of sound implementations, most of them with very poor performance regarding sound latency matters [3]. As Android developers explain at last Google I/O [4], most of the latency is introduced by drivers and chipsets. Therefore, the solution to the problem is likely to come in the form of a porting guide for manufacturers, rather than a software patch.

The application presented in this paper uses the Java low-level audio API, since native APIs do not noticeably improve the latency. At this point, the API allows to record and playback unencoded audio waveforms represented in pulse-code modulation (PCM) format. For the application under study, the characteristics of the recorded audio are: 8 kHz sampling frequency, 16 bits per sample and a single channel.

Furthermore, as the objective of the application is to cover as much as possible use cases, three methods for activating audio recording and sending have been developed: static, manual and automatic. The first one consists of an audio amplitude threshold that sends any sound that is higher than the selected threshold. This threshold is selectable by the user at any time by dragging the marker. The second one is a push to talk (PTT) mechanism that uses volume or headset buttons to activate audio recording and sending. The last input method analyzes audio samples, background noise and instantaneous amplitude in order to detect human voice. With any of these three methods, the application is able to reduce both noise and bandwidth usage. As the application could be used in a wide variety of use cases, the suitability of each method will depend on the specific situation.

2.3 Audio Coding

The main goal of coding a media stream is to reduce the amount of transmitted data making the most of the intrinsic characteristics of the media (e.g. correlation between contiguous samples). On the other hand, wireless technologies used for local communications offer bandwidth rates that are more than enough to transmit several uncompressed audio streams. In this scenario, the question of power consumption arises: there is a trade-off between data compression, which reduces dramatically the bandwidth needed, and the processing consumption of the encoding and decoding processes.

Moreover, another matter to take into account is the use of wireless communications and coverage ranges. Bluetooth and Wi-Fi dynamically change their transmission rate based on the signal to noise ratio. This entails that the transmission rate is the lowest at the border of the coverage area.

There are many codecs successfully compiled for and used on the Android platform. Due to the Java and JVMs characteristics, it is much more efficient to use C or C++ codec versions through JNI (Java Native Interface) code. In this project, we use Speex [5], a widely-used open source audio codec for voice applications. Speex can generate constant or variable bit rates but the variable mode requires the use of floating point arithmetic, which is not usually present on mobile device processors.

Table 1 shows the available Speex encoder modes. A description of the resulting audio quality, as well as the computational costs, is given for each mode.

We are using audio coding for this application due to the perspective of a better performance on coverage borders and the low latency added by the coding and decoding process on mid-range device, usually less than 5 ms. When using the Speex codec we have to choose a mode that entails the quality level. As shown in Table 1, Speex mode 5 gives a good audio quality, compression rate and algorithm complexity.

2.4 Data transfer

As mentioned above, the application uses Bluetooth and Wi-Fi as wireless networks to carry out data transmissions. Each one of these networks has its own characteristics and implementation details, which are discussed in latter sections. In both cases it is necessary to define a protocol to transport the encoded audio and control the management messages. For this task we use Protocol Buffer [6]. This is a compiler that generates Java or C++ code from a protocol definition language. An example of how an audio packet is defined in Protocol Buffer language is shown below.

```
message AudioMessage {
  message AudioSample {
    required int32 seq_number = 1;
    required bytes audio = 2;
  }
  required int32 user_id = 1;
  repeated AudioSample audio_sample = 2;
}
```

This code is used to serialize and parse to and from binary streams. It defines a packet with a 32-bit identification (user_id) and an undefined number of audio samples (audio_sample), which in turn is composed by a sequence number (seq_number) and the audio bytes (audio).

Using Protocol Buffer as shown allows us to define a complete protocol seamlessly, optimized and extensible.

2.4.1 Bluetooth

Implementing a two-way radio application over Bluetooth [7] for Android devices presents many challenges. Firstly, Android devices need to be discoverable in order to pair devices and for that it is required that the user owning the device to discover sets it in discoverable mode. This is necessary for Android versions previous to 4.0, where there is an option to set the device into always-discoverable mode. From now on, we will suppose that all devices that want to communicate over Bluetooth are already discovered and paired. If the target device is not paired it is possible to access the system Bluetooth

Mode	Quality	Bit rate (bps)	mflops	Quality/description
0	-	250	0	No transmission (DTX)
1	0	2 150	6	Vocoder (mostly for comfort noise)
2	2	5 950	9	Very noticeable artifacts/noise, good intelligibility
3	3-4	8 000	10	Artifacts/noise sometimes noticeable
4	5-6	11 000	14	Artifacts usually noticeable only with headphones
5	7-8	15 000	11	Need good headphones to tell the difference
6	9	18 200	17.5	Hard to tell the difference even with good headphones
7	10	24 600	14.5	Completely transparent for voice, good quality music
8	1	3 950	10.5	Very noticeable artifacts/noise, good intelligibility

Table 1. *Speex modes characteristics.*

Android Intercom allows placing real-time calls over Bluetooth and Wi-Fi.

settings from the settings section of the application to pair them. These needs to be taken into account in the design of the user interface, so that the process of establishing a connection is as simple as possible.

Pairing is necessary for establishing secure connections. Secure connections were the unique possibility on Android versions 2.3.3. Since that version it is possible to establish unsecured connections to unpaired devices but those devices still need to be discoverable and the feature of being always discoverable, as commented before, it is only present starting at Android 4.0. To keep compatibility with the mayor number of devices only secured connections are used by the application.

Once the devices are paired, the only Bluetooth profile that is available in the public API is the Bluetooth Serial Port Profile (SPP). The Bluetooth SPP provides a RFCOMM connection. A RFCOMM connection provides a reliable transmission, error detection and flow control, which are not well fitted for time dependent data. Note that it would have been preferable to be able to access to Advanced Audio Distribution Profile (A2DP). AD2P is based on Generic Audio/Video Distribution Profile (GAVDP) and allows the distribution of high quality real-time audio streams.

To reach a good performance on real-time voice communications, a queue system on transmission and reception has been developed to avoid RFCOMM real-time drawbacks as much as possible.

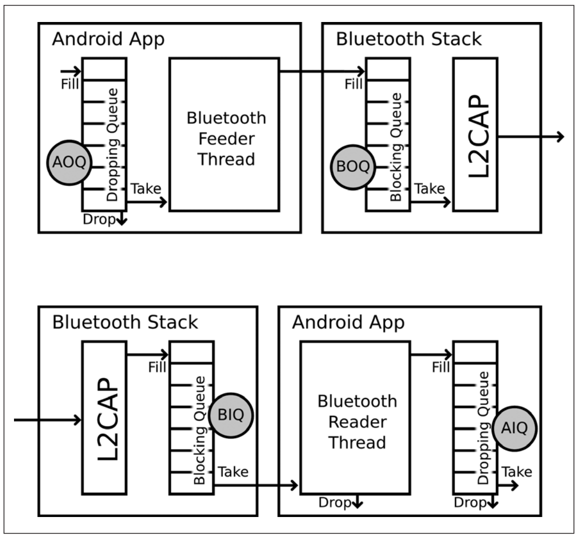


Figure 2. *Bluetooth communication queues.*

Fig. 2 shows the four kinds of queue used. Bluetooth Output Queue (BOQ) and Bluetooth Input Queue (BIQ) are system queues that cannot be controlled at application layer. These queues are designed to avoid undesirable effects of the RFCOMM protocol.

Regarding the other two queues, in a favorable transmission scenario, Bluetooth links offer much more bandwidth than needed. In this scenario, Application Output Queue (AOQ) and Application Input Queue (AIQ) are only acting as an audio record and playing jitter reduction/avoidance buffers.

When Bluetooth links become weak BIQ empties and BOQ fills because the audio encoder bit rate is higher than the connection capacity. In this scenario, samples added to the BOQ cannot be removed and will be transmitted if the Bluetooth stack does not fire a time out exception. Consequently, the AOQ will be filled and will drop samples in a First In First Dropped (FIFD) manner. As pointed out above, encoding audio will help to avoid get into this situation because of the considerably narrower bandwidth needed by it.

When the Bluetooth links become strong again stored samples from the BOQ and from the AOQ will arrive to the receiver in a burst because, as we have already said, Bluetooth links offer much more bandwidth than the required by a coded audio stream. On the receiver side, the Bluetooth reader thread will try to keep the BIQ as empty as possible and will drop any sample that has arrived out of time. Typically, the samples are stored at BOQ.

After a sample burst all the queues but the AIQ are in its initial state. To adjust the size of AIQ the pitch of the sample consumer player can be modified in order to let it act as a jitter buffer.

2.4.2Wi-Fi

Real-time communications are very sensitive to packet delay and jitter. Usually, when losses occur, it is better to ignore lost audio frames, or even drop stale packets, than just wait for retransmissions. In this sense, transport protocols that are not connection-oriented are more suitable for this kind of transmissions. Within the TCP/IP protocol stack, UDP is able to encapsulate audio packets without providing any retransmission mechanism, which would cause interruptions and undesired effects in a real-time communication. Nevertheless, it is worth noting that MAC layer in Wi-Fi communications, which uses 802.11 standard, has its own mechanism for acknowledging packets at lower level, so additional considerations have to be taken into account, as explained below.

Moreover, commercial devices do not present a homogeneous behavior regarding the way they manage UDP packets. Most of them block incoming broadcast traffic, and those who admit broadcast packets cannot receive them in standby mode. This is due to the energy-saving mechanism of the modern smartphones, which switch the wireless card almost off while the device is locked and the screen is off. Android version 3 and above provides the functionality to maintain the wireless card fully operative even though the screen is off. Anyway, experimental tests have shown that this issue occurs only when receiving and there is no difficulty when sending broadcast packets.

Broadcast messages are commonly used for discovering peers in the same network and for broadcasting general information about the nodes. Sometimes, broadcast traffic (or multicast) is used in point-to-multipoint communications (from one to several). In order to solve the aforementioned problems of broadcast packets attached to the heterogeneity of commercial smartphones, we have designed a hybrid unicast-broadcast communication mechanism, which detects at first if the device supports broadcasting.

Mainly, the developed system consists of two phases: peer discovering and audio communication. The neighbor discovering process uses broadcast packets unless the device does not manage to find any node in the network. In that case, unicast scanning starts to find new peers.
Every user must be able to listen to other users in the network so audio communication should be carried out using broadcast packets to profit the point-to-multipoint nature of the system. However, another mechanism is needed for those devices that cannot receive broadcast traffic. Therefore, these devices inform of these limitations in the discovering process so other nodes can be aware. Broadcast audio communication will be used between those nodes that support it whereas unicast audio communication will be used for those nodes that do not.
Regarding packet acknowledgement in Wi-Fi networks, unicast and broadcast transmissions differ. The former is acknowledged through ACK packets at MAC layer, regardless of the transport protocol used. In contrast, the latter cannot use acknowledgements due to the point-to-multipoint nature of broadcast transmissions. This fact motivates the use of a redundancy mechanism against losses for broadcast packets, but not for unicast transmissions. In this sense, a Forward Error Correction (FEC) mechanism has been implemented, which adds parity information to the transmitted data. Therefore, every broadcast packet transports an audio frame and two parity frames that could be potentially used to correct any previous lost audio frame.

3. Android Intercom Application

3.1 Wireless scenarios

There are two scenarios for the application based on the wireless technology to use: the Bluetooth scenario and the Wi-Fi scenario. Both of these approaches have their advantages and disadvantages.

On the Bluetooth scenario devices have to select the other devices to connect to. The need of selecting the devices to connect faces up to the two-way radio concept but it is enforced by the Bluetooth needs of discovering and pairing. For performance reasons the total number of devices interconnected on Bluetooth mode is limited to three. This limit has been obtained experimentally using the Google Nexus One as test device. On tests, four devices have never been able to talk simultaneously without causing voice interruptions.

Android Intercom is being successfully used by thousands of users worldwide.

Once every device has selected the others the application should handle all events of the Bluetooth connections to seamless control connection losses, discoveries and reconnections. When a user wants to speak the others every sample should be sent to each device connected to.

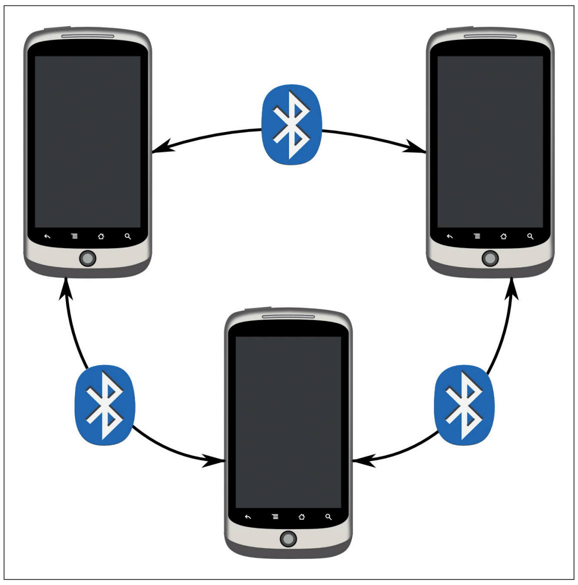


Figure 3. *Bluetooth scenario.*

On the other hand, the Wi-Fi scenario is more similar to a two-way radio channel where the BSSID replaces the radio channel. Also, broadcast Wi-Fi packets are more suitable for one-to-many communications than many one-to-one. The limit of the wifi scenario is farther than our device limit. It has been tested with up to ten devices simultaneously without problems.

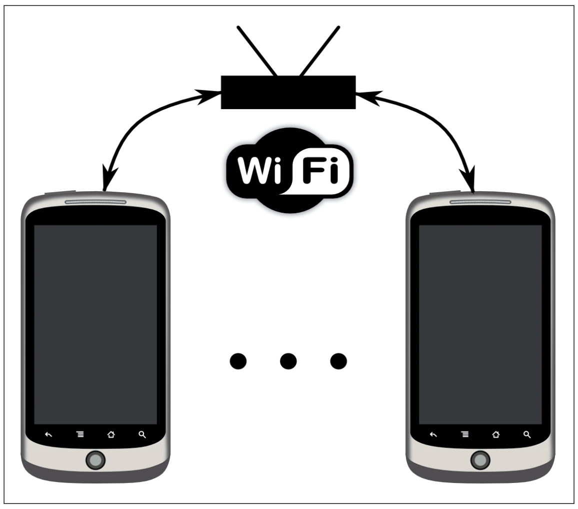


Figure 4. *Infrastructure Wi-Fi scenario.*

There are two Wi-Fi scenarios depending of the working mode. On the infrastructure mode every packet is transmitted through the access point (Fig. 4). On ad-hoc Wi-Fi mode there is no access point and devices communicate directly (Fig. 5). Despite its versatility, Android platform segregates the ad-hoc mode and only few commercialized devices support it. Mesh networking on ad-hoc Wi-Fi mode has been discarded because of its power consumption issues [8].

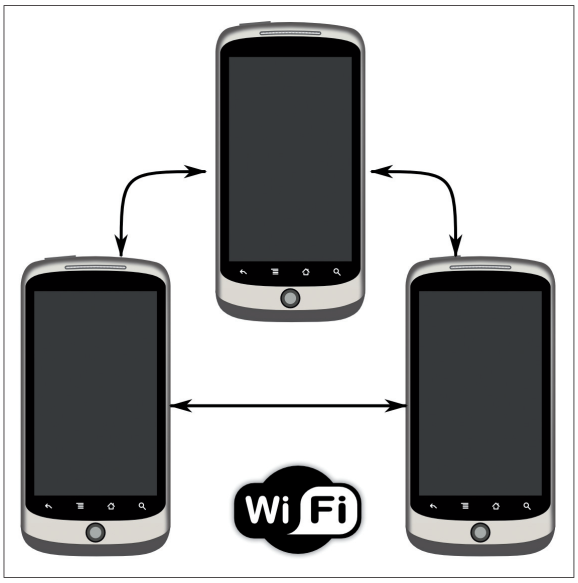


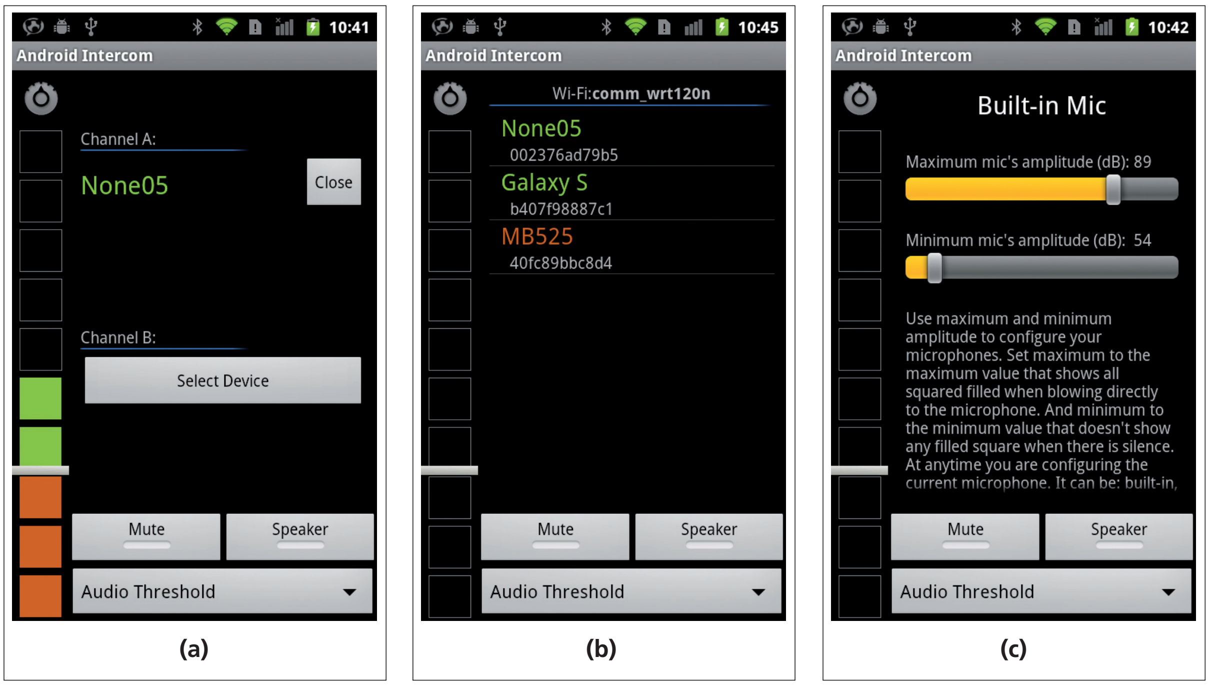
Figure 5. *Ad-hoc Wi-Fi scenario.*

3.2 Android Intercom

The developed application has been published on Google Play (the most popular online store for Android applications), with the name of "Android Intercom", and has reached more than 50.000 downloads since its publication. The local scope of communications limits its use cases in number but greatly improve its value when there is not Internet access. Motorcycle helmet-to-helmet communications, in-building communications and baby monitoring are the top three use cases based on the number of feedback comments received. Fig. 6 shows three screenshots from the application. In Bluetooth mode (Fig. 6a) users have two channels available to create point-to-point connections to other users. Wi-Fi mode let users find other devices in the same network and talk to them all together (Fig. 6b). Both modes show the audio recording level and threshold, depending on the algorithm of voice detection employed. Moreover, through the microphone preferences screen (Fig. 6c), users can adjust the sensitivity of the microphones connected to the device (e.g. built-in, mini-jack, Bluetooth), since each kind of microphone has its own variability.

The application can be downloaded free of charge at Google Play store. The QR code on the Fig. 7 contains the url of the download page.

Figure 7. *QR code to access the download page at Google Play store*



Maximum mic's amplitude (dB): 89

Minimum mic's amplitude (dB): 54

Use maximum and minimum amplitude to configure your microphones. Set maximum to the maximum value that shows all squared filled when blowing directly to the microphone. And minimum to the minimum value that doesn't show any filled square when there is silence. At anytime you are configuring the current microphone. It can be: built-in,

Figure 6. *Android Intercom application: Bluetooth mode screen (a), Wi-Fi mode screen (b) and microphone preferences screen (c).*

4. Conclusions

In this paper we have presented the key points of the development of an Android application for real-time audio communication over local-range wireless technologies. Furthermore, we have discussed about design considerations and technical problems that emerge when implementing algorithms and communication protocols in real-world devices. Thus, we have developed workaround solutions for efficient voice transmission over Bluetooth and Wi-Fi with the tools that Android public APIs offer.

The developed application is being successfully used by thousands of users worldwide and for many different use cases. Following improvements will include Wi-Fi Direct as emerging wireless communication technology. As mentioned, Android has avoided using p2p networking through ad hoc wireless networks. Instead, it has promoted using the new Wi-Fi Direct specification, which enables p2p relationships. On each relation, one of the peers is acting as access point and the other as a normal client. This new approach to p2p wireless networking is very promising for our application, which will improve usability and simplicity.

References

[1] M. Song, W. Xiong and X. Fu, "Research on Architecture of Multimedia and Its Design Based on Android," in Proc. of Int. Conf. on Internet Technology and Applications, Wuhan, China, Aug. 2010.

[2] Android Codecs, available: http://developer.android.com/guide/appendix/media-formats.html

[3] Android issue 3434, available: http://code.google.com/p/android/issues/detail?id=3434.

[4] Google I/O 2011: Fireside Chat with the Android Team, available: http://youtu.be/gfiYUL2exT8.

[5] P. Srivastava, K. Babu and T. OSV, "Performance Evaluation of Speex Audio Codec for Wireless Communication Networks," in Proc. Wireless and Optical Communications Networks 2011, Ghaziabad, India, 2011.

[6] G. Kaur and M.M. Fuad, "An Evaluation of Protocol Buffer," in Proc. of IEEE SoutheastCon 2010, pp. 459-462, Charlotte-Concord, NC, USA, Mar. 2010.

[7] M.J. Morón, R. Luque, E.Casilari and A. Díaz-Estrella, "Minimum delay bound in Bluetooth transmissions with serial port profile," IET Electronic Letters, pp. 1099-1100, Vol.44, no.18, 2008.

[8] A. Iera, A. Molinaro, S. Y. Paratore, G. Ruggeri and A. Zurzolo, "Making a mesh router/gateway from a smartphone: Is that a practical solution?," Ad Hoc Networks, pp. 1414-1429, Vol.9, 2011.

Biographies

Román Belda was born in Alzira (Valencia), Spain. He received the Computer Science degree from the Universidat Politècnica de València (UPV), Valencia, Spain, in 2004. He is currently working towards the M.S. in telematics at UPV. He also currently works as a Researcher at the Multimedia Communications research group (COMM) of the Institute of Telecommunications and Multimedia Applications (iTEAM). His areas of interest are mobile applications and multimedia transmission protocols.

Pau Arce was born in Valencia, Spain. He received the Telecommunications Engineering degree and the M.S. degree in Telematics from the Universitat Politècnica de València (UPV), Valencia, Spain, in 2005 and 2007, respectively. Currently, he works as a researcher at the Institute of Telecommunications and Multimedia Applications (iTEAM) UPV, where he is working toward the Ph. D. degree. His research interests include multimedia QoS, routing on wireless ad hoc networks and performance evaluation of computer systems.

Ismael de Fez was born in Valencia, Spain. He received the Telecommunications Engineering degree and the M.S. degree in Telematics from the Universitat Politècnica de València (UPV), Valencia, Spain, in 2007 and 2010, respectively. Currently, he is a Researcher at the Multimedia Communications research group (COMM) of the Institute of Telecommunications and Multimedia Applications (iTEAM), UPV, where he is working toward the Ph. D. degree. His areas of interest are file transmission over unidirectional environments and file encoding.

Francisco Fraile was born in Murcia, Spain. He obtained a degree in Telecommunication Engineering from the Universitat Politècnica de València (UPV) and the M. Sc. Degree in microwave engineering from the University of Gävle in 2004. Since then, until 2010, he has worked as a Research Engineer for the Swedish company Interactive TV Arena. In 2006, he joined the Multimedia Communications research group (COMM) of the iTEAM Institute, UPV, where he is working toward his doctoral studies as an industrial Ph.D. student. His area of interest focuses on networked electronic media.

Juan Carlos Guerri was born in Valencia. He received his M.S. and Ph. D. (Dr. Ing.) degrees, both in telecommunication engineering, from the Universitat Politècnica de València (UPV), in 1993 and 1997, respectively. He is a professor in the E.T.S. Telecommunications Engineering at the Universitat Politècnica de València, where he leads the Multimedia Communications research group (COMM) of the iTEAM Institute. He is currently involved in research and development projects for the application of multimedia to industry, medicine, education, and communications.