Real time half-duplex voice calling over IEEE 802.15.4/Zigbee standard using Android platform

B. Sree Charan Teja Reddy1, Shah Palash Manish Bhai2, N. Sai Teja1, and G. V. V. Sharma1

1Electrical Engg. Dept., Indian Institute of Technology Hyderabad, {ee12m1010, ee11b022, gadepall}@iith.ac.in
2Electrical and Electronics Engg. Dept., National Institute of Technology, Karnataka, psp.palash@gmail.com

Abstract—This demo describes the implementation of a peer to peer real time half-duplex voice calling application in the android platform with IEEE 802.15.4/Zigbee as the underlying standard. In this demo we show the implementation of Speex voice codec on Android platform, interfacing TelosB motes with Android platform and radio communication between TelosB motes.

I. INTRODUCTION

Wireless Sensor Networks (WSNs) are primarily used for Health care monitoring, Industrial monitoring, Environmental sensing, Data logging etc. WSN motes consume less power and require minimal maintenance. These unique properties of IEEE 802.15.4/Zigbee standard can be exploited to enable short range voice communication over WSNs. The main application of this demo can be found in underdeveloped areas that suffer from power outages and poor economy. Such a network was implemented in [1] and tested. They implemented real time voice communication over a Zigbee network which covers an entire village. The main disadvantage of the network was that proprietary handsets were used which were expensive.

In this demo, as a first step towards developing a short range communication network which uses commercially available devices, we implemented peer to peer half-duplex voice communication over IEEE 802.15.4/Zigbee using cheap open source Android devices. Since Android devices have high speed processors, we can implement the necessary voice codecs for effective voice communication on them. Also, necessary hardware for voice input and output is built into the device, so no other hardware apart from Zigbee motes is required for implementation. We used Aakash tablets [2] running Android, Zigbee based TelosB motes and TinyOS [3] platform for programming the motes.

II. DEMO

The user handset in this demo is Aakash tablet interfaced with TelosB mote (See Fig. 1) using USB (Universal Serial Bus) On-The-Go (OTG) cable. Aakash tablet uses packet-less serial communication to communicate with the TelosB mote. The TelosB motes communicate through packet based radio communication. For this Demo, we developed an Android application for communicating with TelosB mote.

A. User interface layout of Android application

The user interface layout (see Fig. 1) of the application contains a text view, a text box and three user buttons. Text view gives the instructions of operation. Text box is to input the destination node id which is given to each TelosB mote. After writing the destination node id in the text box, click on Enter destination button. After that by clicking on Record button the voice call is initiated. Stop button is used to end the voice call.

B. Voice Recording

The voice data from the microphone is handled using AudioRecord Android API (Application Program Interface). The sampling rate is kept at 8 Kbps. The audio encoding format used is 16 bit PCM (Pulse Coded Modulation). The buffer size is 4096 bytes for the above mentioned sampling rate and recording format. The recorded data is written into a temporary file in frames of size 4096 bytes.

C. Voice Encoding

Speex voice codec [4] is used for compressing the voice data so that it can be transferred over low bit rate IEEE 802.15.4 standard. Speex encoder operates in frames. A frame contains raw voice data corresponding to 20 ms of voice. In Narrowband operation mode (8Khz) and for the parameters used in voice recording section, the frame size is 320 bytes (160 samples). These frames are read from the temporary file (created while recording voice) and given as input to the Speex encoder, which compresses the given frame and returns 38
bytes of encoded voice data. Now the 38 bytes of encoded voice data is written into a file, which will be later used by the Serial communication part.

D. Voice Decoding and playing

At the receiving end, 38 bytes of encoded voice data is received from TelosB. Every time 38 bytes of data is received, it is stored in receive buffer and this buffer is given as input to the decode method, which decodes the data and returns 320 bytes of voice data. These 320 bytes are stored in a dynamic buffer and every time after decoding 38 bytes, 320 bytes are appended to this dynamic buffer.

This dynamic buffer is used for playing the voice. Write and Play methods of AudioTrack Android API are used for this purpose. When 320 bytes are decoded for the first time, Play method is initiated and starts playing the voice. Write method is used for writing the data from dynamic buffer to Play method. As the dynamic buffer will be modified every 20 ms, the voice will be played in real time until the sender ends the voice call.

E. Serial Communication

Android device and TelosB mote communicate using serial communication at a baud rate of 115.2 Kbps. The micro USB port of Android device is connected to the USB port of TelosB mote using OTG cable. The USB port of TelosB mote is internally interfaced with FT232BL chip [5] which is used for conversion of serial data format to USB data format and vice versa.

FTDI device manager: The serial communication on the Android device is managed by using FT_device Android API and ftdi D2xx Java library. This API can be used for managing and configuring FTDI devices connected externally to the Android device. The FT232BL chip embedded on the TelosB mote is well supported by this API.

Sending serial data: FT_device API provides write method to send the encoded data to the microUSB port. To make the communication real time, we are using a thread for reading 38 bytes of encoded data from encoded file and writing it to the micro USB port every 20 ms. Along with this 38 bytes of data, an extra byte containing the destination node id is sent before 38 bytes, which will be used by TelosB for further processing.

Receiving serial data: TelosB mote sends data byte by byte to the micro USB port. These bytes are stored into a buffer queue. The getQueueStatus method provided by FT_device API is used to check the length of the buffer. When the length of the buffer reaches 38 bytes, these 38 bytes of data are read using Read method and loaded into a byte array. This byte array will be used by the decoding section.

F. Radio Communication

TelosB motes communicate through packet based radio communication. For this demo, TinyOS v2.1.2 platform is used for programming TelosB motes. TinyOS program handles Serial communication with Android device and Radio communication between two motes.

Mote to Android device communication: The TinyOS interface used is UartStream which provides Send method for sending data and an event Receivedbyte which signals the receipt of a byte. As and when Android devices starts sending data, Receivedbyte event is triggered which stores the received byte into a buffer. When the buffer size reaches 39 bytes, this data will be processed and used to form a packet and is sent over radio. For sending the received data from other mote, Send method is called. Send takes pointer to the stored received data and length of the data to be sent.

Mote to mote communication: TelosB uses cc2420 radio for communicating with other motes. cc2420 uses CSMA/CA [6] as default MAC layer. This communication is purely packet based. In this demo, the packet contains a variable for destination node id and a byte array of size 38 bytes for storing encoded data. Hence, the packet size is defined as 39 bytes.

After receiving every 39 bytes from USB port, a packet will be created. For sending the packet, we used Send method provided by AMSend interface. Send method can broadcast the packet or it can send it to a particular destination by writing the destination node id in the address field of the method.

For receiving data from other motes, AMReceive interface is used. This interface provides an event called Receive for handling the received packets. This event parses the received packets and extracts the data from it. The extracted 38 bytes of data is sent to Android device using UartStream interface.

III. Conclusion

In this demo we implemented real time half duplex voice calling on Android devices with IEEE 802.15.4/Zigbee as underlying communication standard. The user handset developed in this demo is Android device interfaced with TelosB mote, both are popular and cheaply available in the market. We can easily extend this peer to peer half duplex voice communication to full duplex voice communication over a network which can cover an entire village (which is proved in [7]). This application can also be extended to support reliable messaging [8] and file transfer as they need not be done in real time. This kind of network is likely to be very useful to the people in underdeveloped areas.

IV. Acknowledgment

We thank Department of Electronics and Information Technology, Govt of India for funding this research.

References