Using IEEE 802.15.4 / ZigBee in audio applications

Authors:

Marcel Meli, PhD
Lecturer, Head of Wireless Group, University of Applied Sciences, ZHW (InES),
Technikumstrasse 9, CH-8400, Switzerland Marcel.Meli@zhwin.ch

Martin Gysel, Dipl. El. Ing. FH
Research Assistant, Wireless Group, University of Applied Sciences, ZHW (InES),
Technikumstrasse 9, CH-8400, Switzerland Martin.Gysel@zhwin.ch

Marc Sommerhalder, Dipl. El. Ing. FH
Büssingerstrasse 10, CH-8203 Schaffhausen, Switzerland
Marc.Sommerhalder@gmx.net

Abstract
Most of the current uses for ZigBee and IEEE 802.15.4 focus on control applications. However, there are other areas that will benefit from the standardisation, low cost and possibly low power of ZigBee/IEEE 802.15.4. This paper focuses on the use of ZigBee/IEEE 802.15.4 for audio applications. We will discuss the advantages and theoretical limits of ZigBee/IEEE 802.15.4 for this kind of applications. We will then present a design that we used as starting point to develop applications related to the transfer of audio data.

1. IEEE 802.15.4 / ZigBee

ZigBee and the underlying layers have been designed for low data rate applications, where low power consumption is of great importance. These standards have been described in detail in several documents, and the specifications are also available from the ZigBee and the IEEE organisations [5,7,14].
Some of the features are:

- 27 channels (1 channel of 20 Kbps in the 868 MHz band, 10 channels of 40 Kbps in the 915 MHz band, 16 channels of 250 Kbps in the 2.4 GHz band)
- DSSS modulation, CSMA-CA access.
- Low power modes and low stack complexity, making it suitable for implementation on low power and low cost microcontrollers
- Support for several network topologies with the possibility of connecting thousands of devices together.
- Fast connection time and support for Guaranteed Time Slots (GTS)

2. Audio needs

ZigBee and the underlying protocol IEEE 802.15.4 were developed with low data rate in mind. Low energy consumption is mostly noticed for applications where data rate is low, and devices can often go in power down mode. Why should these standards be at all considered for audio applications? Especially when standards such as Bluetooth and many proprietary solutions already cater for such needs?
There are several arguments that speak in favour of a ZigBee approach, at least for a certain category of audio applications:

- The data rate requirements needed for audio greatly varies with the application, and will range between tens of Kbps (low quality speech) and hundreds of Kbps (very good music quality), making ZigBee an alternative to consider for low-end / mid-end applications.
- Compression methods that are currently been used in the telephone (speech) or music industries could be applied to reduce the data rate. This will of course require the use of appropriate processing power for compression/decompression algorithms. Many algorithms with various degrees of complexity are available, with bit rate been

reduced in certain cases down to a few Kbit/s. Some applications already integrate the necessary computing power, or even the necessary compression algorithms.
- ZigBee / IEEE 802.15.4 modems have a low complexity, making it easier for the application engineers to integrate them in a solution.
- The potential low cost of ZigBee / IEEE 802.15.4 modems makes it likely that they will be deployed in large quantities, and the number of applications using this standard will increase. As this number grows, it is inevitable that for applications that already integrate a ZigBee modem and where audio communication is needed, one will consider the possibility of using that modem for audio.
- The network capacities of ZigBee are worth considering for low-end audio applications that need to cover a small area, and should have the flexibility and ease of deployment provided by a wireless system. The routing capabilities of ZigBee can be used in a similar way to VoIP systems.

3. Audio compression

Speech compression methods can basically be classified into 3 categories:
Waveform Codecs, source Codecs and hybrid Codecs

a) Waveform Codecs

These Codecs require a higher sampling rate, but are less complex. They can be classified in time-related Codecs (for example PCM, ADPCM) and frequency-related Codecs (for example SBC, ATC).

PCM –Codecs using $\mu - law \ or \ A - law$ will delivers 64 Kbit/s. This rate can be further reduced down to 16Kbit/s with the implementation of ADPCM. Speech orientated SBC Codecs typically deliver data rates between 16 Kbit/s and 35 Kbit/s.

b) Source Codecs

They work by modelling the speech source. Instead of audio data, parameters related to this model are transmitted. Some Vocoders work with data rates as low as 2 Kbit/s, and are used in applications where voice quality is not important.

c) Hybrid Codecs

They are a compromise between the 2 other categories. They deliver lower data rate than waveform Codecs, but a better quality than source Codecs. Typical examples are CELP-Codecs such as Speex (2 to 44 Kbit/s for an acceptable quality, but requires much computing power), RPE-Codecs such as GSM.

Many Codecs are also available for music.

Table 1 Some speech codecs

Speech Codecs	*Algorithm*	*Data rate (Kbit/s)*	*Sampling rate (Kbit/s)*	*Sample size (bit)*	*Comments*	*Ref.*
Waveform Codecs						
G.711	PCM	64	8	13/14	Telephony. $\mu - law \ or \ A - law$ used to compress the sample size down to 8 bits.	[13]
G.721	ADPCM	32	8	13 / 14	CCITT	[13]
G.723	ADPCM	24 / 40	8	13 / 14	CCITT	[13]
G.726	ADPCM	16 / 24 / 32 / 40	8	13 / 14	CCITT Includes G.721 and G.723, and can deliver 16 Kbit/s	[13]
G.727	ADPCM	16 / 24 / 32 / 40	8	13 / 14	CCITT. Similar to G.726 For packet-based systems.	[13]
DVI	ADPCM	24 / 32	8	16	From *Interactive Multimedia Association*. Less complex	[9]

					than the ITU-T ADPCM Codecs. Can be implemented on a 8-Bit microcontroller without FPU.	
G.722	SBC	64	16	14	Based on an SB-ADPCM algorithm. The Signal is divided into 2 bands.	[2]
Hybrid Codecs						
G.723.1	ACELP / MP-MLQ	5.6 / 6.3	8	16	VoIP.	[13]
G.728	CELP	16	8	8	Processing intensive.	[13]
G.729	CS-ACELP	8 /6.4 / 11.8	8	16	VoIP. The variant G.729A is compatible and less complex, at the expense of quality.	[13]
Speex	CELP	2 – 44	8 / 16 / 32	8 / 16	VoIP. An implementation exists for the DSPIC from Microchip, and can be used at low cost.	[23]
GSM 06.10	RPE-LTP	13	8	16	Less complex than CELP Codecs. Used in GSM.	[6]
iLBC	LPC	13.3 / 15.2	8	16	VoIP. Appropriate for systems with packet loss. As complex as G.729A, but better quality.	[8]
AMR	ACELP	4.75 – 12.2	8 / 16	13	Developed for 3GPP (3rd Generation Partnership Project).	[1]
IMBE	MBE	2.4 – 9.6			Less complex than CELP based Codecs.	[4]
AMBE	MBE	2.0 – 9.6			Less complex than CELP based Codecs.	[4]

Table 2 Some Codecs used in music

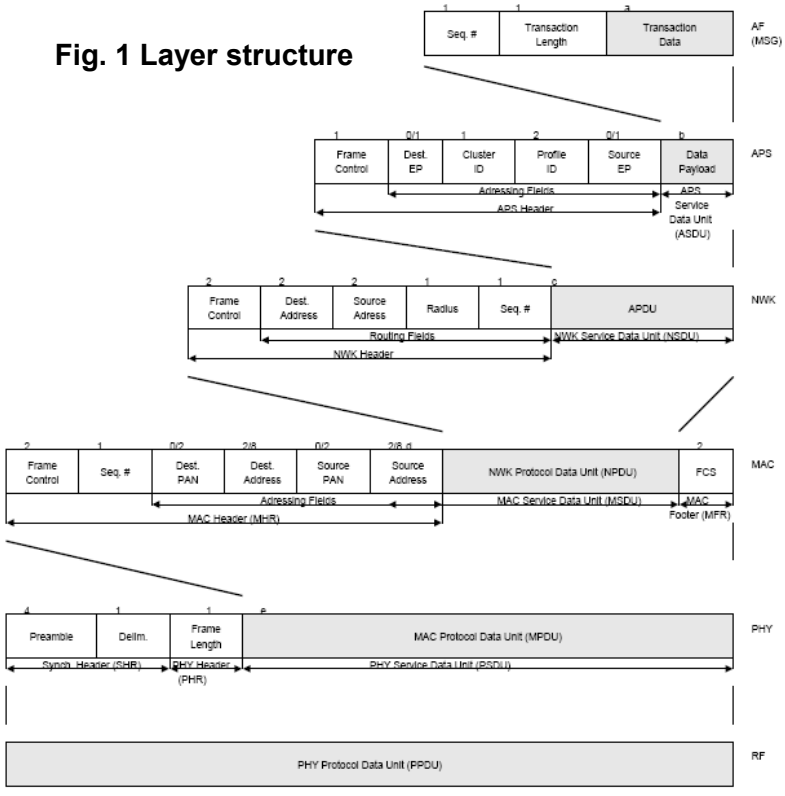
Music Codecs	*Data rate (Kbit/s)*	*Sampling rate (Kbit/s)*	*Comments*	*Ref*
ATRAG	48 – 292	48 / 44.1	Developed by Sony for the minidisk. Claims for the new version ATRAC3plus, to deliver with 64Kbit/s, the same quality as MP3 at 128 Kbit/s.	[12]
MPEG1 audio layer-2	8 – 448	16 - 48	MP2 is a SBC using 32 sub-bands. Implements a psychoacoustic model.	[10]
MPEG1 audio layer-3	32 – 320 / variable	32 - 48	MP3 also uses a psychoacoustic model.	[10]
AAC	variable	8 - 96	Defined in MPEG-2. Uses several compression methods that can be defined in a profile. With 96Kbit/s, it delivers about the same quality as MP3 at 128 Kbit/s	[10]
Musepack	3 – 1300	32 - 48	Based on MP2-Codec, but implements some other techniques.	[19]
Vorbis	16 – 500	8 - 192	Patent free. Quality is better than that of MP3 for the same bit rate, at the expense of more complexity.	[20]
Bluetooth SBC	variable	16 - 48	Used as standard in Bluetooth. Works with 4 or 8 sub-bands. Complexity is low.	[2]

4. Theoretical transfer limits

We will now have a look at the data rates that are possible with ZigBee / IEEE 802.15.4. We consider only the 2.4 GHz band.
The bit rate of 250 Kbit/s per channel can not be fully used for the payload (audio data). There is an overhead related to the way that the communication works.
Several mechanisms should be taken into account when estimating the effective transmission rate. ZigBee has a layer structure, and each layer adds additional header information, so that the payload at the end is less that the amount of information that is sent. The total amount of extra bytes depends on the layer from which the application runs, and on the operating mode of the communication system.

Working from the highest (ZigBee MSG) level, the maximum number of relevant bytes that can be packed in a frame is: $i_{max} = 127- (19 + h_{APS\_Addr} + h_{MAC\_Addr} + h_{MAC\_PAN})$
With headers of at least: 2 bytes for the AF frame, (4 + h_{APS_Addr}) bytes for the APS frame, 8 bytes for the NWK frame, (5 + h_{MAC_Addr} + h_{MAC_PAN}) bytes for the MAC frame.
h_{APS_Addr} is 2 for direct addressing, and 1 for indirect addressing.
h_{MAC_Addr} is 16 for long addresses, and 4 for short addresses.
h_{APS_Addr} is 4 for inter-PAN transfers, and 0 for intra-PAN transfers.
There are also 6 bytes for the PHY frame that contribute to the communication.

Fig. 1 Layer structure



Considering directly addressed intra-PAN transfer with short addresses and ignoring the delays due to the communication mechanisms, the maximum payload data rate is: *191Kbit/s* starting from the AF frame, and *218 Kbit/s* starting from the MAC frame.
The delays actually reduce the theoretical payload data rate. We deal here with the simple case of a Single Sender, and intra-PAN transfers with short address. Further loss of effective bandwidth is related to mechanisms necessary for Collision Avoidance (CA), Interframe spacing (IFS), the extra header of the GTS Frame when this mode is used.

Non beacon frames: (Fig. 10). Collision avoidance requires that a clear channel assessment (CCA) be performed before transmitting data. During this time, the channel is checked to insure that it is free. If the channel is not free, the transmitter should wait for a random time (random number of backoffs) before retrying. This random number is controlled by a variable BE, according to the function *rand* $(2^{BE} - 1)$.
interframe spacing depends on the length of the message transmitted. For messages with the MPDU less or equal to 18 bytes, a short interframe spacing (SIFS) (t_{SIFS} of at least 192 us) is used. Otherwise, a LIFS (t_{LIFS} at least 640 us) is used.

Two acknowledge signals (validation of received messages) can be used by ZigBee: One at MAC level and another at APS level. They will occupy the transmission channel, and therefore contribute to the total overhead. Collision avoidance and interframe spacing are not used for MAC acknowledge frames. ZigBee can use a second acknowledge signal at APS level. The use of acknowledge signals can be turned off (which will be meaningful in most audio applications).
A MAC acknowledge will require about t_{MAC_ACK}. If it is enables, the transmitter will wait at the most t_{wait} for the MAC acknowledge after every transmission.
An APS acknowledge is a normal data frame (from APS layer) lasts t_{APS_ACK} and requires a LIFS
The total time required for a message with a payload of *i* bytes, starting from the AF layer and considering that both MAC and APS acknowledge are disabled is:
$t_p + t_{IFS} + t_{CA} + t_{CCA} = T_B\ (8(h + i) + 4IFS(h,i) + 80\ rand(2^{BE} - 1) + 80)$
T_B is a bit period (4 microseconds) *h* is the total of header bytes needed until MAC layer, *i* is the payload at AF level , *BE* is 3 for CA enabled, and 0 if CA is disabled.
IFS(h,i) is 12 if *h+i* smaller or equal to 18, otherwise 40.
The function *y= rand(x)* gives a random value 0<= y <=x
A MAC acknowledge will add the extra time of: $t_{wait} = 216T_B$
The extra time for APS is: $t_{APS\_ACK} = T_B\ (8h + 80\ rand(2^{BE} - 1) + 224\ [+216]_{MAC\_ACK})$

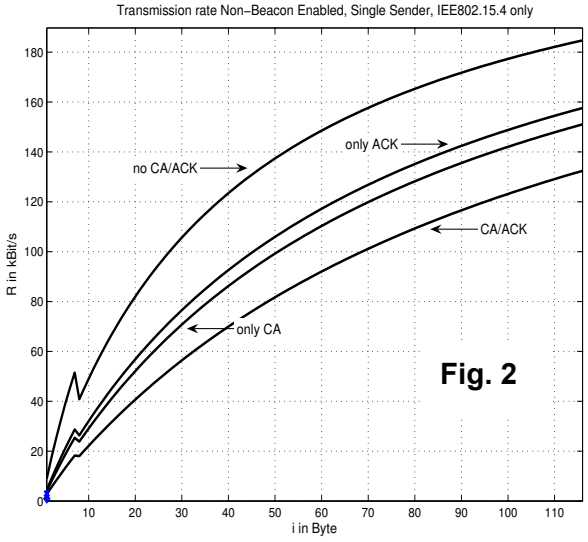


Fig. 2

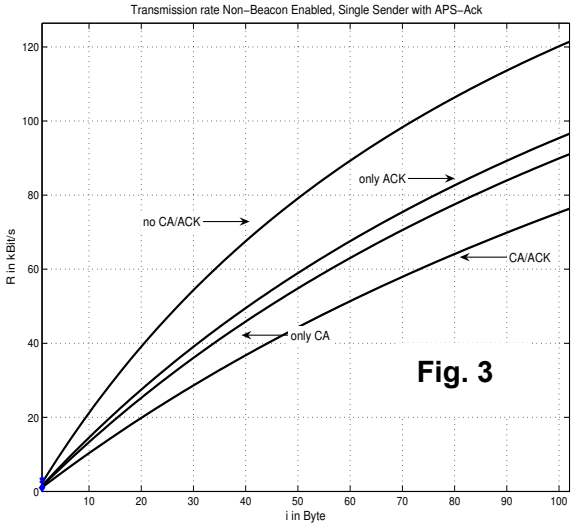


Fig. 3

The Data rate in Kbytes per second in function of the payload i is:

$$DataRate = R_B \frac{i}{i + p.h + 0.5.IFS(h,i) + m}$$

h = *24 [+1]* $_{direct\_adr}$ *[+2]* $_{inter\_PAN}$
If APS acknowledge used:
$m = 10\ [+35]_{CA}\ \ [+27]_{MAC\_ACK}$
$p = 1$
If APS acknowledge not used:
$m = 38\ [+70]_{CA}\ \ [+54]_{MAC\_ACK}$
$p = 2$

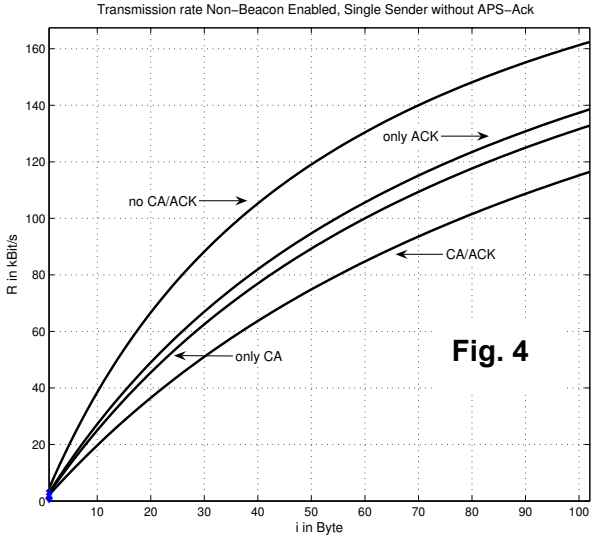


Fig. 4

These calculations will results in the curves shown above (Fig. 2, 3, 4), in the case of CA enabled, an average value is taken.
<u>Beacon frames (Fig. 8, 9)</u>
Similar computations can be made for Beacon modes (transfer with GTS or without GTS).

One should bear in mind that CSMA is not needed with GTS

- A maximum of 7 GTS are allowed
- The communication within a GTS must end 1 interframe spacing before the actual GTS is ended
- The minimal length of the CAP in normal cases should not be less than 440 symbol periods (that is 1760 TB)

The data rate is:

$$DataRate = R_B \frac{i}{480.2^{BO}} \; floor \frac{30n.2^{SO}}{i + h + 0.5.IFS(h,i) + 6[+27]_{MAC\_ACK}}$$

n is the number of basic slots that are used

BO is the MAC Beacon Order and *SO* is the MAC Superframe Order. Resulting curves for GTS transfer (Fig. 5) and non-GTS transfer (Fig. 6) are shown below.

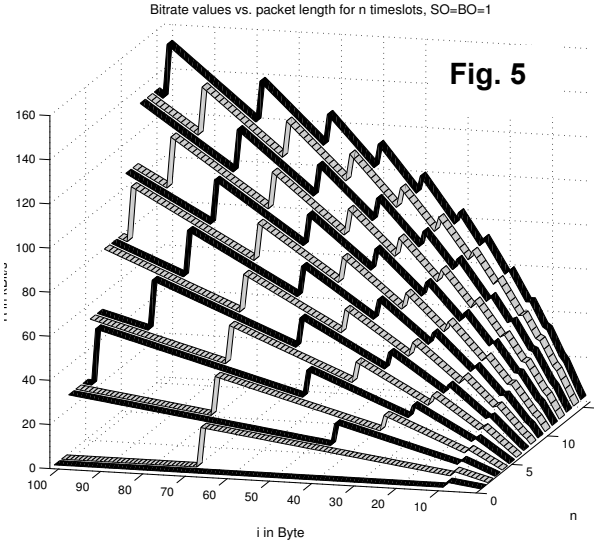


Fig. 5

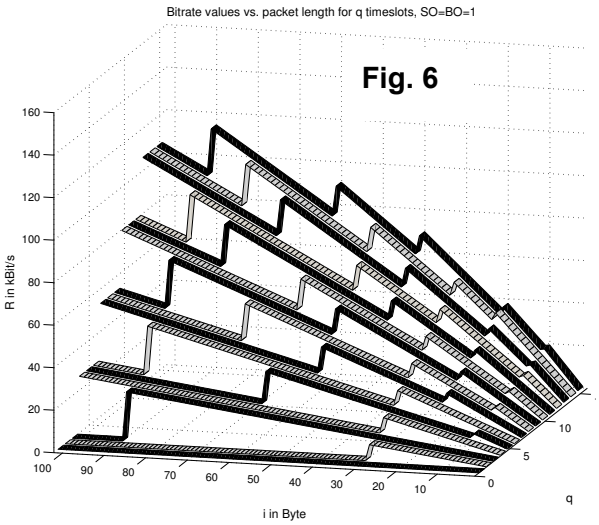


Fig. 6

5. Limits related to the processing power of sender and receiver

The transmission channel is not the only factor that will influence the final quality of the audio stream. The data also needs to be processed fast enough, otherwise there will be lost of data, or the transmission rate will have to be reduced. Since the data rate is already low enough, it will be better if the processing power of the sending and receiving stations does not constitute a bottleneck.

Both sender and receiver are usually controlled by a microcontroller. Typically, that processor will be used on the receiving end side for the following operations:

- Unloading the data received from the transceiver's memory into the microcontroller's memory
- Checking that the data received is error-free
- Processing the data (including decompression when needed)
- Sending the sample to the hardware the Codec, at the needed sampling rate.

Similar operations are needed on the transmitting side.

In both cases, the processing time will increase if a stack is implemented on top of the MAC layer (case of ZigBee). The use of DMA will allow some work to be done in parallel and reduce the load on the microcontroller.

The processing time can roughly be divided into 2 parts:

- A part that depends on the number of bytes sent/received. (This includes the transfer time between the transceiver's memory and the controller's memory, compression and decompression time)
- A part that is independent of the number of bytes that are processed.

The effect of the processing power can be approximated to a linear function. We can conclude that a faster processor and low software requirements will be better.

6. Limits related to delays and lost of packets

So far, we have assumed that the transmission occurred without errors. In practice, there are interferences that will affect the channels with more or less severity. Compressing and transmitting the data in packets also means that there will be delays. The severity of these elements depends on the application. Everything else been equal, we can basically say that the longer a data packet is, and the higher the delays and the probability that it will be disturbed. The effect of disturbances resulting in loss of data is more severe when the packets are larger. This means that errors occurring while transmitting larger blocks will tend to have more audible effects. This will introduce a packet length dependent limit for larger blocks of data.
In the case of speech applications, missing samples in the range of 10-20 ms will hardly be noticed. For real time speech application, delays should be kept below 200 ms.
Table 3 shows for ADPCM the resulting delays (sampling time + transmission time) for 2 different packet lengths. The overhead in transmission is not considered.

Table 3 Delays resulting from sampling and transmitting

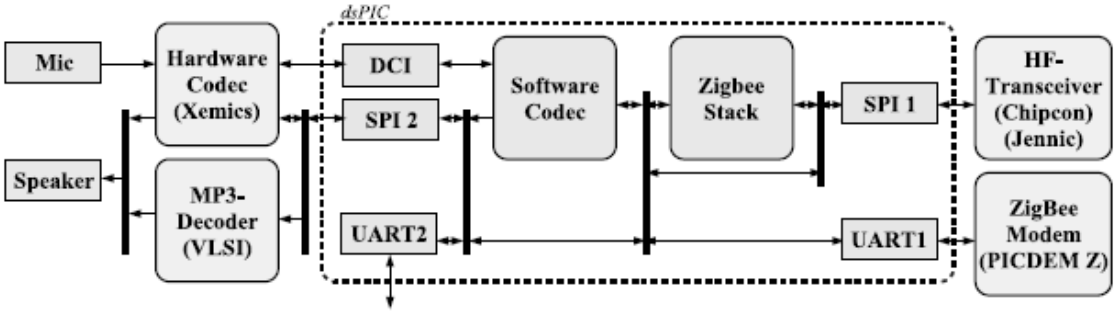
Coding	Sampling rate (KHz)	Bit per sample	Bit rate (Kbit/s)	Samples per packet	Sampling time per packet	Delay (ms)
Packet length = 102 bytes, transmission time of payload = 4.26 ms (2.4 GHz band)						
Comp. PCM	8	8	64	102	12.8	17
ADPCM	8	5	40	163	20.4	24.7
ADPCM	8	4	32	204	25.5	29.8
Packet length = 50 bytes, transmission time of payload = 4.26 ms (2.4 GHz band)						
PCM	8	8	64	102	6.25	8.84
ADPCM	8	5	40	163	10	12.6
ADPCM	8	4	32	204	12.5	15.1

It can be seen that for coding schemes with low bit rate, the delays are higher. The effects of packet loss will also be more important.
The delays will have to be corrected with the overhead related to the communication mode.

7. A set up for audio transfer

A flexible system allowing the connection of various ZigBee/IEEE 802.15.4 modems was built (Fig. 7)

Fig. 7 Block diagram of the hardware



Central to this set up is the DSP + Microcontroller. The 16-bit DSPIC has a processing power of 30 mips , is capable of executing some DSP algorithms [16], and can therefore be used to try some of the compression or decompression algorithms. It also implements a low end ADC / PWM combination that could be used for low quality audio.

The DSPIC is also rich in serial interfaces such as SPI, UART. It is therefore possible to implement several interfaces, in order to connect to various ZigBee / IEEE 802.15.4 modems. This makes it possible to evaluate different components.
On the software side, Microchip give access to the source code of its ZigBee stack as long as it is used on one of its processors [11,18]. This stack could be ported to the DSPIC if needed. A library including an implementation of the Speex Codec is also available for a low fee [17].
The Codec (XE3006) [22]. The very low power hardware Codec from Semtech (previously Xemics) is used for sampling the audio signal, and converting the samples back into analog audio. It can connect to the DSPIC using a fast serial bus.
For experiments with MP3, we included the VS1002, a device from VLSI [21]. It performs decoding of MP3 samples that would be sent from another platform.
Various modems can be connected to the system: the following were used:

- The ZigBee Kit from Microchip (PICDEMZ). This modem consists of a PIC18LF4620 microcontroller from microchip, running a ZigBee stack provided by the same firm, and a CC2420 transceiver from Chipcon [3].
- A module based on the Jennic JN5121 single chip ZigBee / IEEE 802.15.4 device [15]. The Jennic chip includes both a microcontroller and a transceiver. The 32-Bit processor clocked at 16 MHz runs the necessary stack, and also provides enough computing power or the application.

8. Results and conclusions

A version of the ADPCM algorithm was implemented and run on the DSPIC. A network consisting of a coordinator and a device was set up, and audio data from various sources were successfully streamed in half-duplex mode (mainly speech quality).
Using the PICDEMZ module (at ZigBee stack level), it was possible to stream audio samples in half duplex mode. Due to the computing limitations of the PICDEMZ processor, it was not possible to stream at more than half of the theoretical limit.
Measurements showed that packets were effectively being sent fast enough, implying that the low effective data rate achieved was related to the computing power limits and the demands of the stack. The RS232 serial link between the modem and the DSPIC were also found to be a weak link, requiring too many interrupts.
Using the Jennic modem for sending data at MAC level (network with 1 sender and 1 receiver), it was possible to approach the calculated theoretical limits of data rates, with more than 75% of the computing power remaining available for other purposes.
This shows that Zigbee/ could well be used for audio applications. A simple compression algorithm will help to reduce the data rate, and improve the system. The available bandwidth is sufficient for low-end applications. With proper compression, one could even reach better quality. It is however important to have enough computing power to ensure that the bottleneck is not on the stack (computing power) side, and that the communication channel is used efficiently. The low power advantages provided by ZigBee should not be at the forefront of the application, since the communication devices will frequently be active.
Important improvements can be obtained by working at the MAC level, since more bandwidth will be available. The extra computing power resulting from not using the ZigBee layers could be used to implement a simple compression algorithm for single-chip solutions like the Jennic device. This will reduce costs and size, and contribute to keeping the overall power consumption down.
Various modes can also be taken advantage of, in order to further reduce the overheads. However, these have to be tailored to the application.
Simple applications involving only 2 stations are obviously easier to implement. For applications with many stations, the computing power of a DSP is needed in order to reduce the bit rate to acceptable levels, and one should consider GTS modes.

Acknowledgement
We wish to thank the firms Jennic and Municon for their kind support in providing modules for tests.

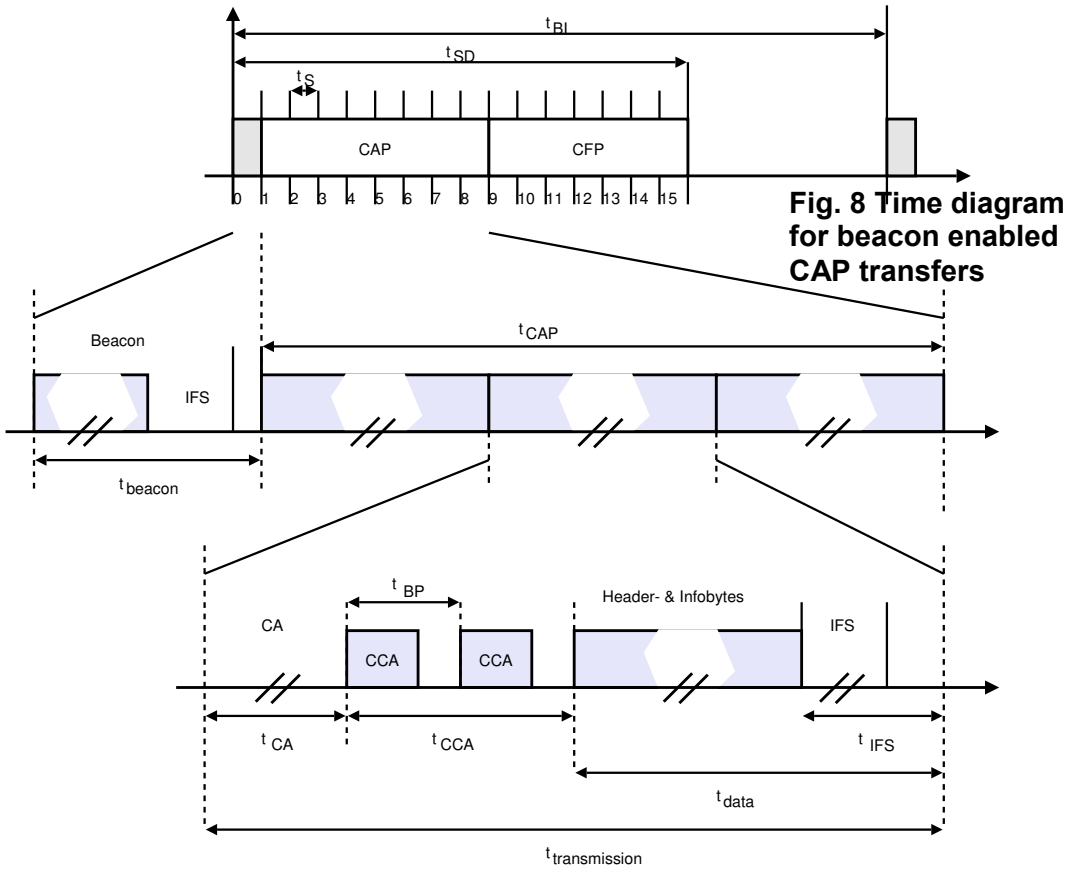


Fig. 8 Time diagram for beacon enabled CAP transfers

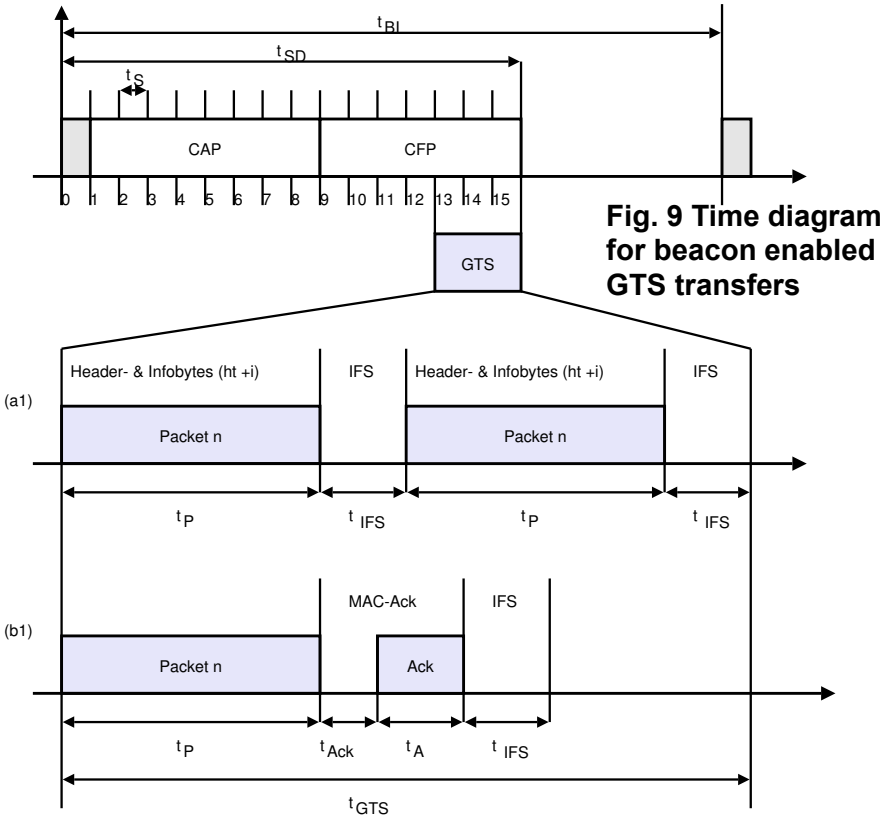


Fig. 9 Time diagram for beacon enabled GTS transfers

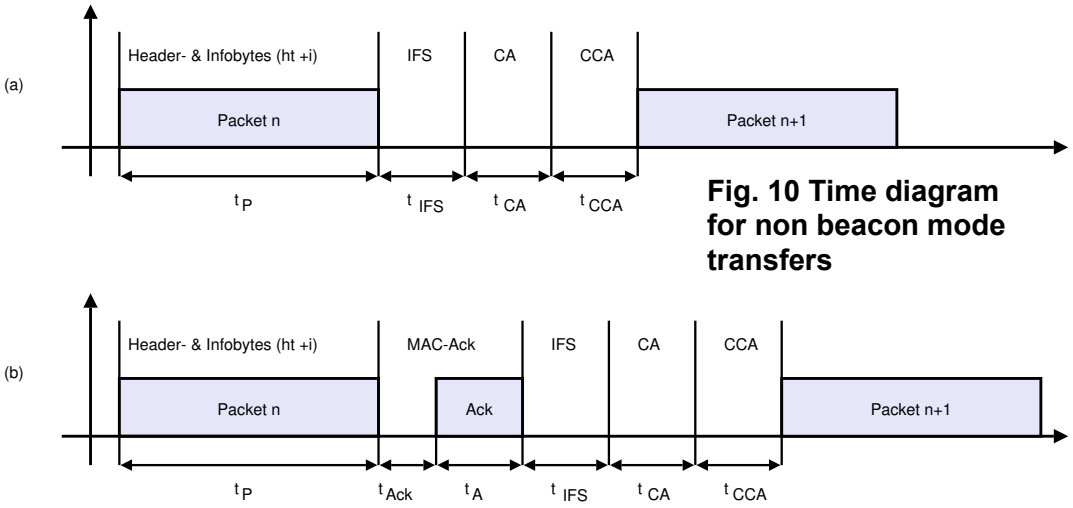


Fig. 10 Time diagram for non beacon mode transfers

References

[1] 3rd Generation Partnership Project. 3GPP Specification series. http://www.3gpp.org/ftp/Specs/html-info/26-series.htm, September 2005.
[2] Bluetooth Audio Video Working Group. ADVANCED AUDIO DISTRIBUTION PROFILE SPECIFICATION. http://www.bluetooth.org, September 2005.
[3] Chipcon. CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver, June 2004.
[4] Digital Voice Systems. DVSI Home Page. http://www.dvsinc.com/, September 2005.
[5] ZigBee Alliance. ZigBee Specification, June 2005. www.zigbee.org
[6] ETSI. Digital cellular telecommunications system (Phase 2+); Full rate speech; Transcoding. http://www.etsi.org, September 2005.
[7] IEEE. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), October 2003.
[8] iLBCfreeware.org Project. internet Low Bitrate Codec. http://www.ilbcfreeware.org/, September 2005.
[9] IMA Digital Audio Focus and Technical Working Gropus. Recommended Practices for Enhancing Digital Audio Compatibility in Multimedia Systems. http://www.cs.columbia.edu/~hgs/audio/dvi/, October 1992.
[10] Moving Picture Experts Group. MPEG standards. http://www.chiariglione.org/mpeg/standards.htm, September 2005.
[11] Nilesh Rajbharti. Microchip Stack for the ZigBee Protocol, AN965. Microchip, 2004.
[12] Sony Global. ATRAC3 Technology. http://www.sony.net/Products/ATRAC3/tech/atrac3/index.html, September 2005.
[13] ITU-T. G.x Recommendation. http://www.itu.int/rec/recommendation.asp?type=products&lang=e&parent=T-REC-G, September 2005.
[14] Zigbee alliance, white papers, slide presentations (www.zigbee.org)
[15] Jennic. JN5121 IEEE802.15.4 Wireless Microcontroller, 2005. (www.jennic.com)
[16] Microchip. dsPIC30F Family Reference Manual, 2004.
[17] Microchip. dsPIC30F Speech Encoding/Decoding Library User's Guide. Technical report, Microchip, 2005. (www.microchip.com)
[18] Microchip. Microchip Stack for the ZigBee., V1.0-3.3, September 2005.
[19] the musepack project. Musepack. http://www.musepack.net, September 2005.
[20] the xiph open source community. Vorbis. http://www.vorbis.com/, September 2005.
[21] VLSI. VS1002d - MP3 AUDIO CODEC, April 2005.
[22] Xemics. XE3005/XE3006 Low-Power Audio CODEC, 2005.
[23] Xiph.Org Foundation. Speex. http://www.speex.org/, September 2005.