Efficient implementation of voice channels over CAN networks

F. Baronti, D. Lunardini, R. Roncella, R. Saletti,
Department of Information Engineering, University of Pisa, Italy.

Abstract-The Controller Area Network is widely used as fieldbus in many distributed control systems. In addition to control information, a voice communication running on the same bus may have several advantages. This paper deals with an efficient implementation of a voice channel over an already designed CAN network. In particular, the solutions to minimize the band occupation and to cope with the large variability of the time taken to transmit a CAN message are described. A proof-of-concept demonstrator, consisting of two CAN stations provided with audio capabilities, has been realized and tested in an existing distributed control system designed and installed in a high-end motorcycle. Finally, in order to fully demonstrate the feasibility and effectiveness of our approach, the system performance under several bus load conditions has been characterized.

Introduction

CAN networks are widely used in many applications in which several microcontroller-based sub-systems (stations) need to communicate in an efficient and reliable way. CAN is a broadcast bus designed to work at speeds up to 1 Mb/s [1]. Data are wrapped in messages, from 0 to 8 bytes long, to which an unique identifier in the network is assigned. The identifier field of a message determines which kind of information is carried, and the priority during the bus contention. The main application fields are automotive, as well as industrial process control, building automation and maritime systems. We note that in most of these application environments a voice communication over the bus could often be desirable [2], [3].

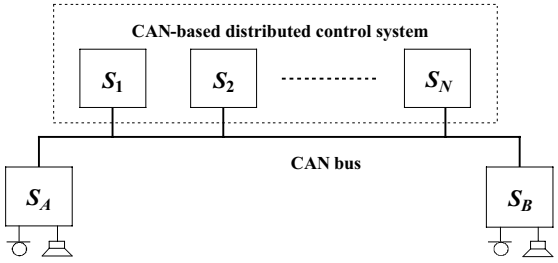

Figure 1. Implementation of a voice channel over a CAN network.

This goal can advantageously be achieved connecting additional stations, provided with the capability of interfacing a microphone and a speaker, to the same network responsible for the communication of the already designed distributed control system. Figure 1 shows the modified network configuration in which the stations S_1-S_N implement the control application, while the stations S_A and S_B provide operators with the chance of talking between two points of the bus line.

However, some issues must be faced to guarantee an intelligible speech communication when the traffic due to the already existing control system is present in the bus. First of all, the audio message overhead should not impair the control system real-time performance. In particular, the minimization of the band required to implement the voice channel and the variability of the time taken to transmit a CAN message (i.e. the time interval between the instant the message is ready to be transmitted and the one it is effectively received by the destination stations) are major topics to be faced.

These issues have already been addressed in [2]. The proposed system exploits a sophisticated voice compression algorithm to obtain a very low band occupation, but it needs a high computational power at the same time. Beside this, the feasibility of the voice channel has been proven only without additional traffic running in the network. Aim of our work is to efficiently implement the voice communication channel with a

negligible amount of computational power so that it could be established using the same microcontroller employed in the control system [3], and to characterize the performance of the designed system when the bus is loaded by other messages.

The paper is organized as follows. The first section gives details of the implementation of the voice channel, while the second one presents an analytical analysis of the system performance. Finally, the experimental results are described in the last section.

Voice channel implementation

Realizing a continuous full-duplex voice channel over a CAN network means that two stations (S_A and S_B in Figure 1) send the audio data coming from their local microphone on the bus, and get the audio samples to be reproduced from the bus too. From now on, we will refer to the units responsible for the implementation of the speech communication as voice stations. Giving more details, each voice station has the function of coding the local microphone signal, and wrapping the converted audio data in CAN messages to be transmitted as soon as the bus becomes free. At the same time, the unit must receive the messages carrying the audio samples originated in the other side of the voice channel, and decode them in an analog signal to be played by the local speaker.

From the bus load point of view, each voice station continuously transmits a 8-byte message, with a period depending on the adopted speech coding. If a traditional *A-law* or μ*-law* PCM (Pulse Code Modulation) speech coding with a 8 kHz sampling frequency is used, the transmission period is equal to 1 ms, which leads to a net band occupation of 128 kb/s [4]. Such a value is rather high in a CAN network and cannot be tolerated in most of the control systems, so the use of a more sophisticated voice compression algorithm is necessary. However, the reduction of band due to the compression is paid with an increase of the hardware and software complexity of the voice station. A very good tradeoff between used bandwidth and resources needed for the compression algorithm implementation is provided by the ADPCM (Adaptive Differential Pulse Code Modulation) speech coding, which allows to halve the band occupation with respect to the PCM coding, maintaining a good audio quality at the same time [4]. In fact, the ADPCM coding is implemented by off-the-shelf audio codecs, which can be used together with the low cost microcontrollers generally employed in the control units. As a consequence, the hardware of the voice station has been realized by simply adding an ADPCM codec to the control station described in [3].

Moreover, the lowest possible priority has been assigned to the CAN voice messages, so that the worst-case response time of the higher priority messages remains as it is [5] and the control system performance is not affected by the insertion of the voice messages, which are transmitted only when some free bandwidth is available. However, the low priority of the voice messages has some implications in the voice channel realization. In particular, we cannot rely on the time equidistance of the voice messages, because their transmission can randomly be delayed by higher priority messages. Therefore, two buffers must be provided on both sides of the voice channel, as shown in Figure 2. The transmitting buffer has the function of accumulating the data coming from the codec (ADC) at a continuous fixed rate when the bus is busy, while the receiving one feeds the codec (DAC) when no audio data are coming from the bus.

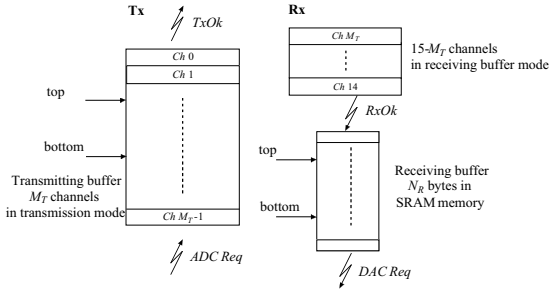

Figure 2. Structure for the data buffering.

It is worth noting that the larger the buffer dimensions are, the lower the audio communication sensitivity to a long period of bus unavailability is. The microcontroller used in our application provides 256 bytes of SRAM and integrates a CAN controller

with 15 message objects (Ch_i, i = 0..14), each of them independently configurable in transmission, reception, and buffer reception modes [6]. Hence, a very efficient data buffering implementation lies in using M_T message objects as a circular buffer to store the samples, coming from the ADC, until the bus becomes free and they can be transmitted. This approach has two main advantages. First, building the CAN message directly on the message object mailbox permits to save SRAM memory to be used as receiving buffer. Second, as soon as the message object is filled with 8 bytes, the contention for the bus access can immediately start avoiding the time overhead due to moving the audio samples from memory to the CAN controller.

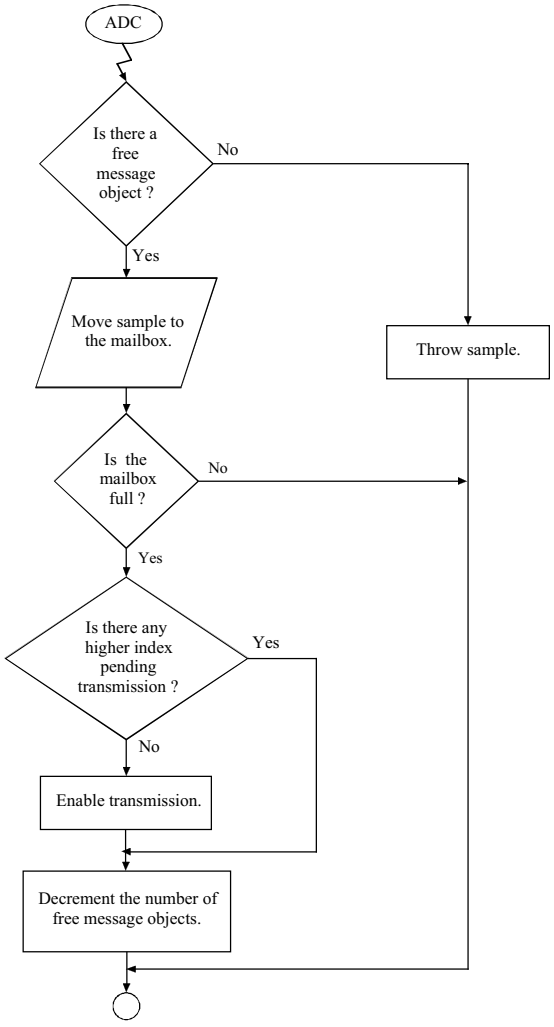

Figure 3. Flowchart of the interrupt routine triggered by an ADC request.

This data structure is handled by means of two interrupt routines the execution of which is triggered by an incoming audio sample from the codec ADC and by the success of a message transmission. The flowcharts of these two routines are reported in Figure 3 and Figure 4, respectively. We must observe that if more than one message object is enabled at the same time, than the message built in the lowest index object is always transmitted earlier [6]. Therefore, some tricks to avoid a time inversion of the audio samples have to be adopted. In fact, it is sufficient to enable a ready message object only when there are no pending transmissions related to message objects with higher index. Otherwise, the last enable message object will overcome the older ones causing a time inversion of the audio data.

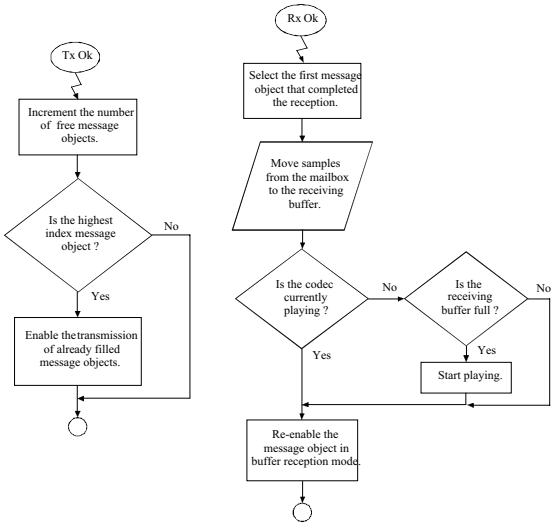

Figure 4. Flowchart of the interrupt routines which manage the CAN controller requests.

On the receiving side, 15-M_T message objects are used to receive the audio messages. We observe that more than one message object is necessary to collect a burst of voice messages, which may occur after a long period of bus unavailability. For a network bit rate less than 1 Mb/s, 4 message objects are sufficient, while one additional object is needed when the bit rate is set to 1 Mb/s. Once the reception of a voice message is completed, the related routine, shown in Figure 4, moves the audio samples from the mailbox to the receiving buffer, the dimension of which is set to N_R = 88 bytes, so the transmitting and receiving buffers are symmetrical. Since the presence of a high priority traffic on the network can lead to long periods in which the bus is unavailable and no audio messages can

be transmitted, the reproduction of the audio samples is started only when the receiving buffer becomes almost full, i.e. it is filled with 80 bytes and there is room available only for an additional audio message. The flowchart of the interrupt routine responsible for the reproduction of the audio samples is reported in Figure 5. Finally, we note that the traditional solutions adopted to manage a difference between the clock frequencies of the two voice stations [2] are hardly applicable in presence of a large variability of the time taken to transmit a voice message together with the reduced size of the receiving buffer.

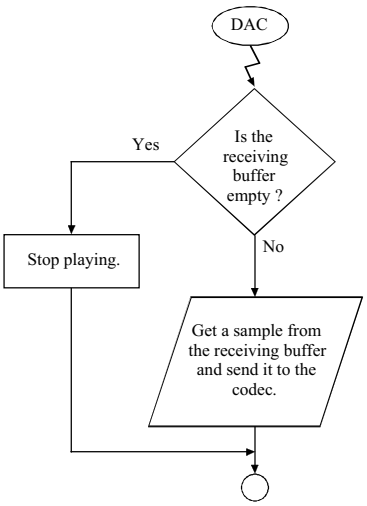

Figure 5. Flowchart of the interrupt routine triggered by a DAC request.

Performance analysis

We start evaluating the workload *WL* of the bus related to the implementation of one full-duplex voice channel. Since each voice message carries 16 audio samples (4-bit for each ADPCM sample), its transmission period T_v is 2 ms, assuming that the sampling frequency of the audio signal is equal to 8 kHz. In order to obtain the bus load, we need the time C_v taken to transmit a voice message. According to CAN specifications [7], the total number of bits of a standard data frame (11-bit identifier), carrying 8 bytes in the data field, is 111, as shown in Figure 6. Because of the bit-stuffing coding, the actual size of the voice frame could differ from the one calculated above. The number of stuff bits, added by the bit-stuffing coding, can be obtained by the sum of those originated in the header part of the message (Start of frame, Arbitration field and Control fields), which are known a-priori, and those originated in the payload and CRC field, which, instead, depend on the data transmitted.

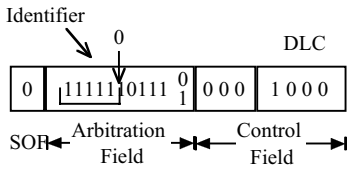

Figure 7. Header parts of the two voice frames.

Assigning the lowest possible priorities to the two voice messages results in the bit patterns of the voice frame header parts shown in Figure 7 (because of the electrical characteristics of the bus, the smaller the identifier is, the higher the priority is. In addition, the CAN standard forbids identifiers with the seventh most significant bits equal to 1). Therefore, 1 stuff bit is always originated in the header part, while the remaining 79 bits exposed to bit-stuffing can lead to further stuff bits in the range from 0 to 19. Consequently, the size of a voice message can vary between 112 and 131 bits. So, C_v is equal to 131 T_{bit} in the worst-case, where T_{bit} is the nominal bit time. However, the bus load

$$WL = 2\frac{C_v}{T_v} = \frac{262\,T_{bit}}{T_v} \qquad (1)$$

obtained in the worst-case scenario is really pessimistic [8]. A more realistic

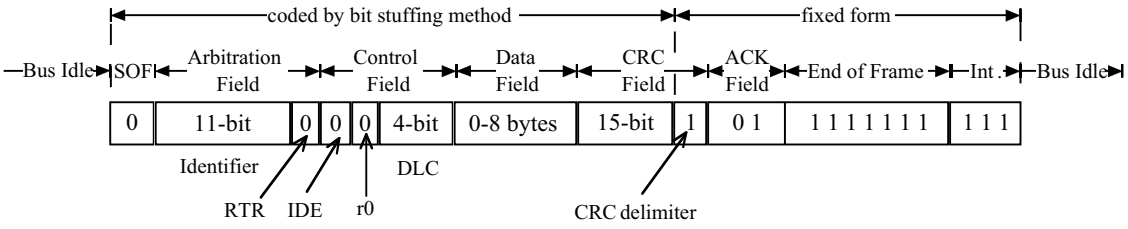

Figure 6. Standard data frame layout.

value can be derived according to the work of T. Nolte et al. [9], who have calculated the probability of having a given number of stuff bits in the data and CRC fields. Their analysis is based on the assumption of independence and equal probability of 0 and 1 in each bit position, which is a fairly good approximation in our case. It follows that the average time taken to transmit a voice message is:

$$< C_v > = 114.5\, T_{bit} \qquad (2)$$

From Eq. (2), we obtain the average bus load which is reported in Table 1 for different values of T_{bit}.

T_{bit} (Bit-rate)	Bus load
1 μs (1 Mb/s)	11.45%
1.25 μs (800 kb/s)	14.32%
2 μs (500 kb/s)	22.91%
4 μs (250 kb/s)	45.81%
8 μs (125 kb/s)	91.62%

Table 1. Bus load related to voice messages versus network speed.

Once the workload of the original control application is known, Table 1 permits to evaluate if the addition of a voice communication is in principle possible. As an example in a 1Mb/s CAN network, a voice channel can be established at the expanse of about 12% of the entire bandwidth. However, other phenomena must be taken into account [10], [11]. Indeed, since the voice messages have the lowest priority, their transmission is delayed by any other message. In that case, the voice messages ready to be transmitted are accumulated in the transmission buffer which eventually overruns. On the other hand, the receiving buffer is progressively emptied. Both these situations can lead to the loss of some voice samples, especially when the higher priority messages are enqueued in long bursts. Therefore, fixed the buffers dimensions, there is a maximum extent of the bus busy time beyond which same samples are lost causing a degradation of the audio quality. Figure 8 shows a condition in which a burst of higher priority messages fills the bus for T_{busy} seconds, so the lowest priority voice message must wait T_w seconds to be transmitted. When the audio messages transmission is not delayed, the transmitting buffer is normally empty and the receiving one is filled with 80 bytes, which means 160 ADPCM samples. Therefore, since the audio samples are played with a rate of 8 kHz, the maximum extent of time in which no audio messages are received cannot be larger than 20 ms. During this time the audio data are accumulated in the transmitting buffer. In order to avoid the buffer overrun, the maximum waiting time T_w must satisfy the following relation:

$$1 + \left\lceil \frac{T_w}{T_v} \right\rceil \le M_T \qquad (3)$$

As the waiting time is related to the bus busy time by the following relation:

$$T_w = T_{busy} + \left\lceil \frac{T_w + T_{bit}}{T_v} \right\rceil C_v \qquad (4)$$

from Eq. (3) and (4) we obtain a second bound for the maximum extent of time in which the bus cannot be available for transmitting audio messages. The calculated values of the maximum acceptable T_{busy} are reported in Table 2 for

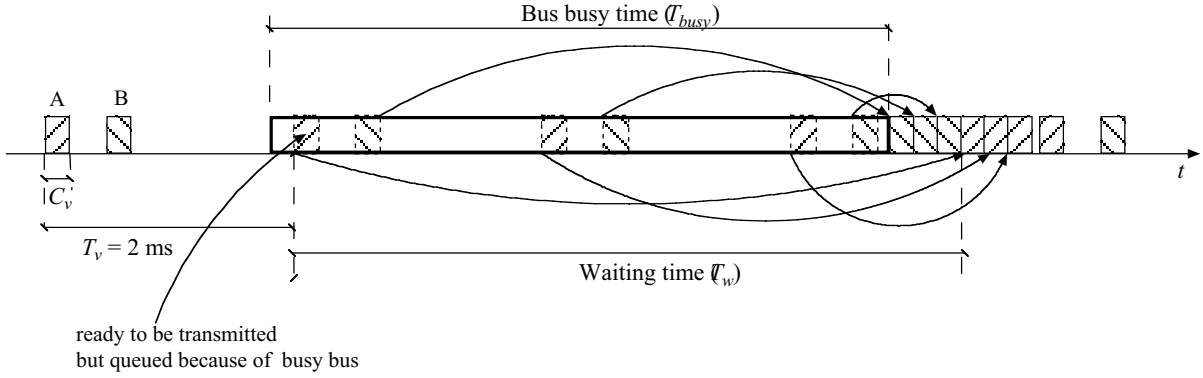

Figure 8. Bus condition in which the periodical audio message transmissions *A* and *B* are delayed because the bus is filled by higher priorities messages.

different network speeds.

T_{bit}	C_v (worst-case)	T_{busy}
1 μs	131 μs	16.8 ms
1.25 μ	163.75 μs	18.3 ms
2 μs	262 μs	17.38 ms
4 μs	524 μs	15.76 ms
8 μs	1.048 ms	10 ms

Table 2. Maximum extent of bus busy time versus network speed.

The lower value of T_{busy} for a bit rate of 1 Mb/s compared with the one obtained for 800 kb/s, can be explained bearing in mind that the transmitting buffer consists of 10 message objects for the former, while 11 objects are used in the other cases.

Experimental results

A proof-of-concept demonstrator, consisting of two stations connected to a microphone and a speaker, has been realized and fully tested on our target application [3]. The test results demonstrate a good quality of the speech communication, while the original system continues to correctly run. Furthermore, in order to prove the effectiveness of our approach in a more general way, the characterization of the designed system under several operating conditions has been carried out. In fact, a 3-station CAN network, shown in Figure 9, has been set up, in which two stations S_A and S_B are responsible for the audio communication, while the third has the function to periodically inject messages into the bus. In particular, every *Burst repetition period* seconds the *Traffic Injector* station fills the bus with a message burst, which creates a time interval of *Burst length* seconds in which the bus is not available for the audio

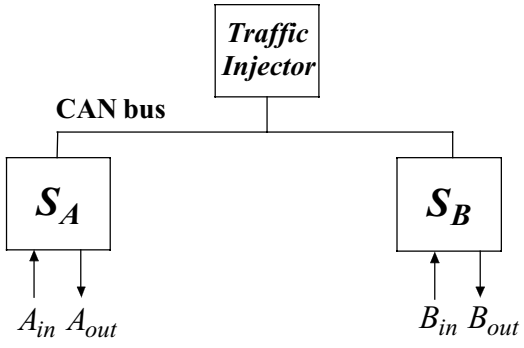

Figure 9. Set-up of the experiment.

messages transmission. This situation is illustrated in Figure 10. Varying the repetition period and the number of messages injected each time, several bus load conditions can be produced. We note that the message traffic so obtained is a good approximation for many control systems in which the great amount of messages is periodic, and in particular for those driven by a cyclic scheduler.

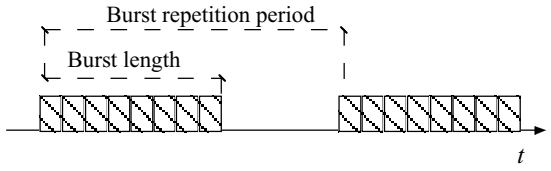

Figure 10. Message traffic generated by the *Traffic Injector* station.

In order to evaluate the performance of the voice channel, the two voice stations have been fed by the same sinusoidal wave ($A_{in} = B_{in}$) and the output waves (A_{out} and B_{out}) have been sent to an oscilloscope to show the correct regeneration of the input signal on both the receiving stations. It is important to note that an input signal like a sinusoid is a very pessimistic case, since every missed sample leads to a degradation of the reproduced signal. The experiment has been carried out for different bus loads by varying the *Burst repetition period* and the *Burst length*. Figure 11 shows the experimental results when the audio messages are the only messages running on the network, while Figure 12 shows the opposite situation in which the bus is completely filled by both the audio messages and the messages

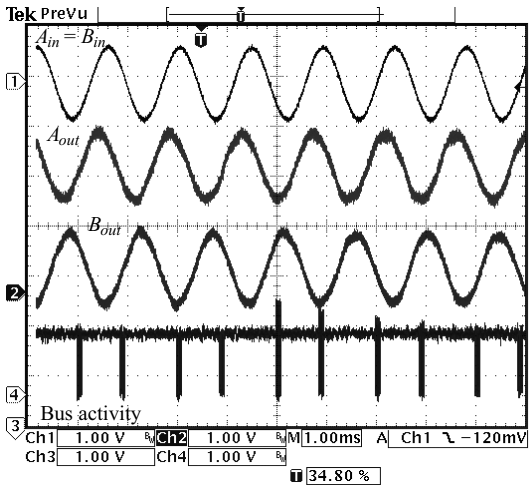

Figure 11. Experimental results for an unloaded bus.

generated by the *Traffic Injector* station. It is to be noted that the sine waves are finely reproduced at both sides of the CAN bus in both the cases of light and heavy traffic, thus demonstrating the functionality of the 2-way audio link.

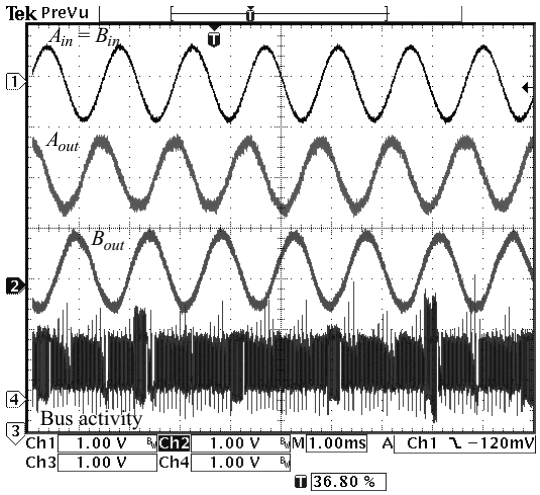

Figure 12. Experimental results in a heavy traffic bus condition.

Figure 13 summarizes all the experiments carried out. It shows the maximum *Burst length*, once the *Burst repetition period* has been fixed, which allows the sinusoidal wave to be perfectly reproduced on the receiving sides, for different network speeds. The dotted lines represent the bandwidth expressed in percentage occupied by the traffic injector station. It is worth noting that the points correspond with a very good approximation to the analysis pointed out in the previous section, both for the band occupation and for the maximum bus busy time allowed. As an example, let us consider a CAN network with 500 kb/s bit rate. We note that the voice channel can be established at full audio quality if up to 77% of the overall band is used by the control application (this confirms the theoretical expectations of Table 1), provided that the requirement of bus unavailability time less than about 17.5 ms is satisfied, as expected by the analytical analysis reported in Table 2. Finally, the same experiments have been carried out replacing the sine generator with a microphone and the oscilloscope with a headset. The speech communication remains intelligible also for bus loads slightly beyond the lines reported in Figure 13.

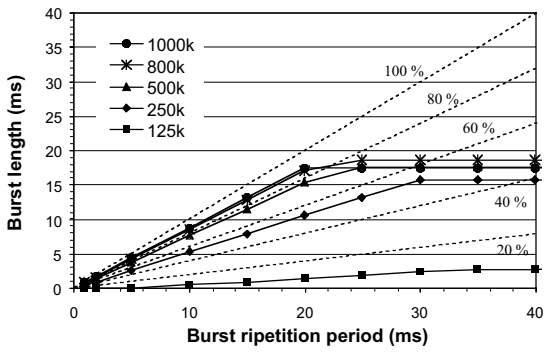

Figure 13. Voice channel performance versus the bus load for different network speeds.

Conclusion

The efficient implementation of an audio communication over a CAN control network has been presented. The goals of minimizing the band occupation and of coping with the large variability of time taken to transmit an audio message have been achieved keeping the hardware complexity very low. In addition, the requirements that the original network must satisfy in order to accommodate a 2-way audio communication capability have been calculated. Then, a proof-of-concept demonstrator has been realized and successfully tested on a CAN network already designed for a high-end motorcycle. Finally, the performance of the voice channel under several bus load conditions has been measured. The experimental results demonstrate that the audio communication is still possible also when the bus is completely filled by messages, provided that a given value of bus unavailability time would not be exceeded. Besides, a very good matching between the analytical analysis and the measured results has been found.

References

[1] Road Vehicles, “Interchange of Digital Information – Controller Area Network (CAN),” *International Standard Organisation (ISO), Standard-11898*, Nov. 1993.

[2] R. H. Arnold, “Investigation of Voice Channels in CAN Systems,” *Proc. 4th International CAN Conference ICC’97*, 1997.

[3] F. Baronti, R. Hippoliti, D. Lunardini, S. Mangraviti, R. Roncella, R. Saletti, “Low-Cost

Can-Based Communication System for High-End MotorScooter," *SAE Paper 2002-01-2154, Automotive Transportation Technology Congress, ATTC'02*, Paris, July 2002.

[4] "G.710-729 Terminal Equipments - Coding of Analog Signals," *G-series ITU-T Recommendations*.

[5] K. W. Tiondell, A. Burns, and A. J. Wellings, "Calculating Controller Area (CAN) Message Response Times," Control Engineering Practice, vol. 3, n. 8, pp. 1163-1169, 1995.

[6] "T89C51CC01 Enhanced 8-bit MCU with CAN controller and Flash," *Technical data sheet Atmel Wireless & Microcontrollers, Rev. D*, Dec 2001.

[7] "CAN Specification Version 2.0, Part-A and Part-B," *CAN in Automation (CiA)*, Am Weichselgarten 26, D-91058 Erlangen, htpp://www.can-cia.de/, 2002.

[8] T. Nolte, H. Hansson, and C. Norström, "Probabilistic Worst-Case Response-Time Analysis for the Controller Area Network," *Proc. 9th IEEE Real-Time and Embedded Technology and Application Symposium*, 2003.

[9] T. Nolte, H. Hansson, and C. Norström, S. Punnekkat, "Using Bit-Stuffing Distributions in CAN Analysis," *IEEE/IEE Real-Time Embedded Sysems Workshop RTES'01*, December 2001.

[10] L. Rauchaupt, "Performance Analysis of CAN based systems," *Proc. 1st International CAN Conference ICC'94*, 1994.

[11] B. P. Upender, A. G. Dean, "Variability of CAN Network Performance," *Proc. 3rd International CAN Conference ICC'96*, 1996.

Author 1 F. Baronti
Company Department of Information Engineering, University of Pisa
Address Via Caruso, 2
Phone +390502217629
Fax +390502217522
E-mail f.baronti@iet.unipi.it
Website http://vlsi.iet.unipi.it

Author 2 D. Lunardini
Company Department of Information Engineering, University of Pisa
Address Via Caruso, 2
Phone +390502217629
Fax +390502217522
E-mail d.lunardini@iet.unipi.it
Website http://vlsi.iet.unipi.it

Author 3 R. Roncella
Company Department of Information Engineering, University of Pisa
Address Via Caruso, 2
Phone +390502217669
Fax +390502217522
E-mail r.roncella@iet.unipi.it
Website http://vlsi.iet.unipi.it

Author 4 R. Saletti
Company Department of Information Engineering, University of Pisa
Address Via Caruso, 2
Phone +390502217648
Fax +390502217522
E-mail r.saletti@iet.unipi.it
Website http://vlsi.iet.unipi.it