American Journal of Engineering Research (AJER) **2018**
American Journal of Engineering Research (AJER)
e-ISSN: 2320-0847 p-ISSN : 2320-0936
Volume-7, Issue-5, pp-107-112
www.ajer.org
Research Paper Open Access

Improving the Behavior of a Distributed Adaptive Cruise Control System by Reducing Transmission Jitter

Mouaaz Nahas
Department of Electrical Engineering, College of Engineering and Islamic Architecture, Umm Al-Qura University, Makkah, Saudi Arabia
Corresponding author: Mouaaz Nahas

ABSTRACT: *The Controller Area Network (CAN) protocol is inherently used in distributed real-time, resource-constrained embedded systems due to its deep roots in automotive as well as other industries. In CAN, bit-stuffing is employed in the physical hardware layer to synchronize the clocks of sending and receiving microcontroller nodes. Such a mechanism causes the frame length to vary depending on the sequence of data transmitted in the network and hence introduces timing jitter. To address this problem, a wide-range of software techniques have been proposed and found useful in practical embedded system applications. In this paper, we seek to verify the effectiveness of the software bit stuffing technique in dealing with transmission jitter and show how such an improvement in jitter behavior can help to improve the performance of a distributed CAN-based adaptive cruise control system used in modern passenger cars.*

Keywords - *Jitter, bit stuffing, scheduler, time-triggered, shared-clock, adaptive cruise control, hardware-in-the-loop, jerk*

Date of Submission: 18-04-2018 Date of acceptance: 03-05-2018

I. INTRODUCTION

The Controller Area Network (CAN) protocol is widely used in distributed embedded control systems [1], [2]. CAN uses "Non Return to Zero" (NRZ) coding for bit representation. Therefore, when a long sequence of identical bits has been transmitted, a drift in the receiver's clock may occur which might in turn cause a corruption in the received message. To avoid this scenario, the CAN communication protocol (in its hardware layer) employs a bit-stuffing mechanism. In bit-stuffing, whenever a sequence of five consecutive identical bits are found in a given frame, the sending node adds an additional bit of the opposite polarity to force a transition in the voltage level and hence avoid clock drift. At the receiver, such stuffed bits are removed to recover the original data [3], [4]. Despite the usefulness of bit-stuffing mechanism in CAN, it results in a variation of the frame length that becomes (in part) a function of the data contents. Such variation results in introducing a jitter in the timing of tasks which are due to execute simultaneously on different nodes in a CAN-based network. The presence of jitter may have a negative impact on the performance of many distributed embedded systems including control applications [5]–[13].

There has been a great deal of interest in addressing the jitter problem caused by bit-stuffing, and a wide range of jitter-reduction techniques were hence proposed. For example, Nolte and his colleagues [14], [15] proposed a technique in which the data section of each CAN frame is XOR-ed with the alternating bit-pattern 101010… etc. This technique was found useful for particular industrial-based data in which the probability of having bit value of "1" (or "0") was not 50%. A modification to this approach was carried out in [16], [17], where each byte is checked individually and will only be masked (i.e. XOR-ed with 10101010) if it is subject to CAN bit-stuffing. This technique was described as "selective byte-based XOR masking" and referred to as "Nolte C". Please note that Nolte A refers to the direct application of Nolte's method, and Nolte B refers to a "frame-based XOR masking" which was also found ineffective with pseudorandom data [16]. The implementation of Nolte C in practical systems demonstrated a jitter reduction of 20%. Further reduction in jitter was achieved by "software bit stuffing" (SBS) [16] and "eight-to-eleven modulation" (EEM) [18], where the complete data section is masked to avoid the possibility of CAN hardware bit-stuffing. Alternative techniques for dealing with jitter problem in CAN system are detailed in [19]–[25].

The impact of SBS on the performance of a distributed CAN-based "adaptive cruise control" (ACC) system was carried out in [10]. The system was built using "hardware-in-the-loop" (HIL) testing method [26], [27]. The ACC system considered had 10 nodes (one Master and nine Slaves). Each Slave node was responsible of controlling a different part of the vehicle as follows: front left wheel, front right wheel, rear left wheel, rear right wheel, radar interface, pedal interface, driver interface, throttle control and backup Master. The results show that, when employing the SBS technique, the control performance (measured by the integral of absolute error in velocity and in distance as well as the maximum jerk) was generally improved, except in the case of positive jerk. Such results were found promising for the system type considered in that study. In [28], the impact of SBS on the control behavior of a distributed CAN-based CarboNitriding heat treatment furnace system was investigated. The study revealed that a significant improvement in the control performance of the furnace (measured mainly by the integral of absolute error and integral of time-weighted absolute error) was achieved, particularly when a temperature disturbance was introduced.

This paper verifies the effectiveness of SBS by applying it to a more recent version of the ACC system that is compacted to 6 nodes (instead of 10). Reducing the network size in the design of a passenger vehicle may be required in some circumstances. Such a modification in the system architecture is expected to have an impact on the transmission jitter and jitter reduction levels resulting from bit stuffing (due to significant alteration in the task list run by each node as well as the contents of data exchanged in the network).

The paper is organized as follows: Section II provides an overview of the ACC system. Section III outlines the experimental methodology used in this study. The experimental results obtained are detailed in Section IV, and finally the paper conclusions are drawn in Section V.

II. Adaptive Cruise Control (ACC) System

In this section, we provide a brief description of the adaptive cruise control system and how it can be built on an HIL testbed facility.

2.1. Overview

The ACC is a relatively new technological development in the automotive field, and is said to reduce driver fatigue and the rate of auto accidents whilst increasing fuel efficiency [29]. The main function of the ACC is to control the speed of the host vehicle using information about the distance between the subject vehicle and any front vehicles (using Doppler radar), the motion of the subject vehicle itself and commands from the driver. Based on this information, the controller sends commands to the vehicle throttle and brakes to either regulate the vehicle speed to a given set value or maintain a safe distance to any leading vehicles (respectively). It also sends status information to the driver. The ACC concept is summarized in Fig. 1.

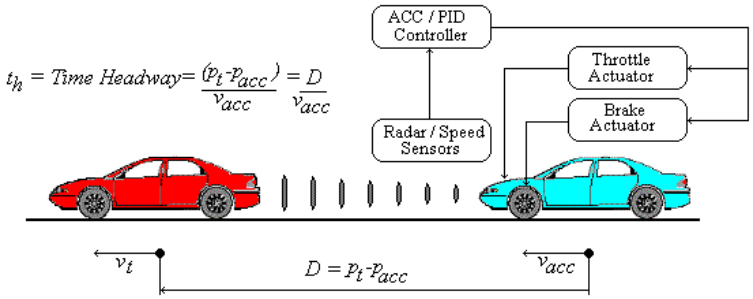

Fig. 1: An overview of the ACC system operation [10]

2.2. Testbed

The HIL testbed employed in this study has been previously described in detail [30]. Briefly, the simulation consists of a real-time representation of a motor vehicle travelling down a three-lane motorway, under realistic traffic conditions. It enables different embedded control system architectures to be assessed and quantitatively compared in a variety of realistic and repeatable scenarios. In this section, the hardware under test represents an ACC system.

In the present study, we create the software required to implement the control system as a series of communicating tasks, which are distributed across the multiprocessor architecture (6-node implementation). A schematic of the overall system, with the various inputs and outputs of each node, is shown in Fig. 2. Note that, as compared to 10-node system, we have only one node to control the front wheels and one node to control the rear wheels. Moreover, the driver interface and pedal interface nodes are combined into a single node, and no backup Master node is used in the system.

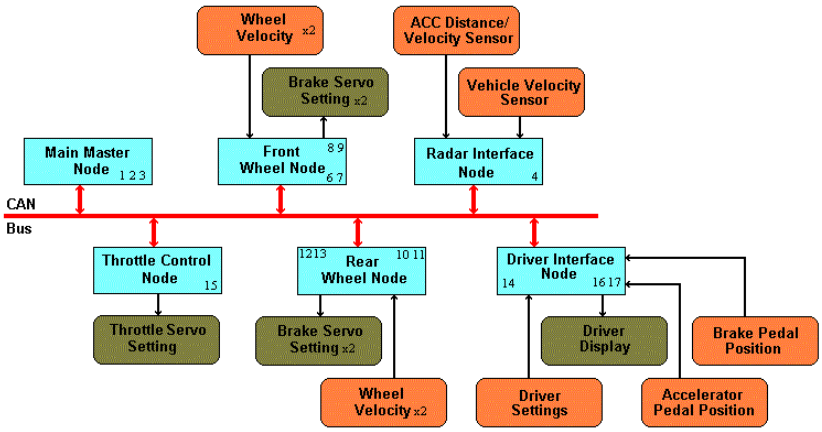

Fig. 2: The 6-node ACC implementation [31]

Each node used here is based on Infineon C167 microcontroller operating at a 20 MHz oscillator frequency. The nodes are connected via a CAN bus running at 500 kbit/sec baudrate, with the TTC-SCC protocol used to synchronize the schedulers on the communicating nodes [32], [33]. Each node executes its tasks using a TTC scheduler [34] and the scheduler code is written in C language [35]. The Keil C166/167 tool chain is used to develop and de-bug the software. The system uses a Tick interval of 5 ms and the main ACC control loop is iterated every 100 ms. The controller itself consists of an outer distance and velocity control loop, acting on the driver setpoints and radar information, which provides an inner control loop with an instantaneous reference velocity signal to track. A dual-output PID controller then actuates the throttle and brakes to equalize the actual vehicle velocity to this reference velocity.

III. Experimental Methodology

Three different data frames were considered:

- Original (with 5 bytes for real data in 'Tick' messages and 1 byte for Slave ID).
- Nolte C (with 5 bytes for real data in 'Tick' messages, 1 byte for Slave ID and 1 byte for coding information).
- Software bit stuffing (with 5 bytes for real data in 'Tick' messages, 1 byte for Slave ID and 2 bytes for stuff bits).

In each case, to compare the jitter, measurements were made from the Master and Slave ISR functions, using the following methodology. A pin on the Master node was set high (for a short period) at the start of the Master ISR function. Another pin on the Slave (initially high) was set low at the start of Slave's Interrupt Service Routine (ISR) function. The signals from these two pins were then AND-ed (using a 74LS08N chip) to give a set of pulses whose widths represent the transmission delays. These widths were measured using the National Instruments data acquisition card 'NI PCI-6035E' used in conjunction with LabVIEW 7 software.

To assess the impact of the different data coding methods on system performance, the Integral of Absolute Error (IAE) was measured. The IAE represents the error between the measured speed (or time-gap) and the reference, with the test duration T of 300 seconds. The IAE is defined in (1).

$$IAE = \int_0^T |e(t)|\, dt \tag{1}$$

The speed setpoint used for each velocity test was 70 MPH and each distance test was performed whilst following a lead vehicle at 50 MPH (distance setpoint of 33.53 m for a 1.5 s headway).

In addition to the velocity (and distance) error measurement, the rate of change of acceleration (defined as "jerk") was recorded to provide an indication of the control performance. Both positive and negative jerk were recorded over a 300 second test period in which the ACC system was put through a series of typical maneuvers. Note that the jerk levels form a significant factor in this system, and provide a sensitive measure of the system performance. Increased jerk levels, caused by large or uneven changes in the accelerator or brake pedal inputs, may lead to increased passenger discomfort, as well as increased wear in the powertrain components [36].

IV. RESULTS

This section presents the results of this study which include jitter and performance measurements. The performance measurements consists of IAE velocity, IAE distance, and the maximum positive and negative jerk.

4.1. Jitter measurements

Table 1 shows the basic timing measurements for the ACC system including the jitter values. The difference jitter is taken as the difference between the worst-case and the best-case latency values of the entire sample range. On the other hand, the average jitter is taken as the standard deviation of the total latency of the entire sample range.

Table 1. Basic timing results from the ACC study.

Test	Original	Nolte C	SBS
Min transmission (μs)	292.2	297.2	319.6
Max transmission (μs)	308.1	311.3	329.8
Average transmission (μs)	299.8	301.8	323.9
Difference Jitter (μs)	15.9	14.1	10.2
Average Jitter (μs)	3.9	2.3	2.0

Results in the table show that the levels of transmission jitter have decreased by approximately 11% and 36% as an effect of applying the Nolte C and SBS techniques (respectively) on the considered hardware platform. Such reduction in jitter is seen significant in practice. Remember that, here, all protocols use 6 bytes for data (including the Slave ID).

4.2. Impact on performance

Table 2 below shows the control performance measures of the ACC system when using the different data coding methods.

Table 2: Impact of Nolte C and SBS on the control performance of the ACC study.

Test	Original	Nolte C	SBS
IAE Velocity	43.54	43.28	41.88
IAE Distance	127.96	126.49	124.45
Max Pos Jerk (m/s^3)	2.04	2.03	1.84
Max Neg Jerk (m/s^3)	2.34	2.36	2.26

From the results shown in the table, we note that Nolte C had no real measurable impact on the control performance of the ACC testbed. In contrast, the SBS technique helped to improve the overall system performance. Moreover, we can clearly see that the levels of positive and negative jerk have been reduced by approximately 10% and 3.4% (respectively) when applying SBS.

4.3. Comparison with 10-node ACC system

When comparing the results obtained here with those of the 10-node ACC system [10], it is clear that a significant improvement in the average jitter and maximum positive jerk has been achieved. The improvement in IAE velocity and IAE distance are comparable. It is worth noting that the results presented for 10-node system were based on that the original frame sends 8-bytes real data. In contrary, the results presented here considers that the original frame uses only 6-bytes and the remaining two bytes were left empty (for a fair comparison with SBS technique that can only be applied to 6-bytes actual data). This may interpret the discrepancy in our results, where we would expect to see a better improvement in the other performance measures if the same number of real data bytes were transmitted in both cases.

V. CONCLUSIONS

This paper aimed to investigate the impact of a jitter-reduction technique, known as “software bit stuffing” (SBS), on the control performance of a distributed “adaptive cruise control” (ACC) system used in modern passenger vehicles. The results clearly demonstrated that the jitter reduction of 36% in the message transmission time had a significant impact on the ACC system performance by reducing the maximum positive and negative jerk.

The paper also compared the results obtained here with those obtained from a previously developed ACC system with different network architecture (i.e. 6-node versus 10-node). It was found that the modification of network architecture and hence the data patterns exchanged in the network also had an impact on the overall system performance. This was manifested by the improvement in average jitter and maximum positive jerk.

Note that it would be expected to achieve the same level of performance improvement as a result of employing EEM technique in the ACC system considered in this study. Further work suggests to compare the results from SBS and EEM with alternative data coding techniques meant to deal with jitter and clock synchronization problems in CAN-based distributed embedded systems (e.g. the studies referred to in the literature review).

ACKNOWLEDGEMENTS

Author would like to thank Dr Michael Pont (SafeTTy Systems Ltd, UK) for providing useful comments on the results presented in this paper. The HIL simulation model used here was developed by Dr Michael Short (Teesside University, UK) to whom the author is grateful.

REFERENCES

[1] M. Farsi and M. B. M. Barbosa, *CANopen implementation: applications to industrial networks*. Baldock, Hertfordshire, England; Philadelphia, PA: Research Studies Press, 1999.
[2] M. D. Natale, H. Zeng, P. Giusto, and A. Ghosal, *Understanding and Using the Controller Area Network Communication Protocol: Theory and Practice*. Springer Science & Business Media, 2012.
[3] Bosch, *CAN Specification Version 2.0*. Bosch, 1991.
[4] R. Obermaisser, *Time-Triggered Communication*. CRC Press, 2011.
[5] F. Abugchem, M. Short, and D. Xu, "An experimental HIL study on the jitter sensitivity of an adaptive control system," in *2013 IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA)*, 2013, pp. 1–8.
[6] F. Cottet and L. David, "A Solution to the Time Jitter Removal in Deadline Based Scheduling of Real-time Applications," presented at the 5th IEEE Real-Time Technology and Applications Symposium - WIP, Vancouver, Canada, 1999, pp. 33–38.
[7] Q. Huynh-Thu and M. Ghanbari, "Impact of jitter and jerkiness on perceived video quality," in *Proc. Workshop on Video Processing and Quality Metrics*, 2006.
[8] A. J. Jerri, "The Shannon sampling theorem #8212;Its various extensions and applications: A tutorial review," *Proceedings of the IEEE*, vol. 65, no. 11, pp. 1565–1596, Nov. 1977.
[9] P. Marti, J. M. Fuertes, G. Fohler, and K. Ramamritham, "Jitter compensation for real-time control systems," in *22nd IEEE Real-Time Systems Symposium, 2001. (RTSS 2001). Proceedings*, 2001, pp. 39–48.
[10] M. Nahas, M. Short, and M. J. Pont, "The impact of bit stuffing on the real-time performance of a distributed control system," presented at the Proceeding of the 10th International CAN conference iCC, Rome, Italy, 2005, pp. 10–1 – 10–7.
[11] F. M. Proctor and W. P. Shackleford, "Real-time operating system timing jitter and its impact on motor control," in *Intelligent Systems and Advanced Manufacturing*, 2001, pp. 10–16.
[12] F. Smirnov, M. Glass, F. Reimann, and J. Teich, "Formal reliability analysis of switched Ethernet automotive networks under transient transmission errors," in *Design Automation Conference (DAC), 2016 53nd ACM/EDAC/IEEE*, 2016, pp. 1–6.
[13] F.-Y. Wu and Y.-M. Chen, "Impact of PWM Duty Cycle Jitter on Switching-Mode Power Converter Efficiency," *IEEE Transactions on Power Electronics*, 2017.
[14] T. Nolte, H. Hansson, and C. Norstrom, "Minimizing CAN response-time jitter by message manipulation," in *Eighth IEEE Real-Time and Embedded Technology and Applications Symposium, 2002. Proceedings*, 2002, pp. 197–206.
[15] T. Nolte, H. Hansson, C. Norström, and S. Punnekkat, "Using bit-stuffing distributions in CAN analysis," presented at the IEEE Real-Time Embedded Systems Workshop, London, 2001.
[16] M. Nahas, M. J. Pont, and M. Short, "Reducing message-length variations in resource-constrained embedded systems implemented using the Controller Area Network (CAN) protocol," *Journal of Systems Architecture*, vol. 55, no. 5–6, pp. 344–354, May 2009.
[17] M. Nahas and M. J. Pont, "Using XOR operations to reduce variations in the transmission time of CAN messages: A pilot study," in *Proceedings of the Second UK Embedded Forum*, Birmingham, UK, 2005, pp. 4–17.
[18] M. Nahas, "Applying Eight-to-Eleven Modulation to reduce message-length variations in distributed embedded systems using the Controller Area Network (CAN) protocol," *Canadian Journal on Electrical and Electronics Engineering*, vol. 2, no. 7, pp. 282–293, 2011.
[19] G. Cena, I. C. Bertolotti, T. Hu, and A. Valenzano, "Fixed-length payload encoding for low-jitter controller area network communication," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2155–2164, 2013.
[20] G. Cena, I. C. Bertolotti, T. Hu, and A. Valenzano, "A mechanism to prevent stuff bits in CAN for achieving jitterless communication," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 1, pp. 83–93, 2015.
[21] M. M. Hassan, "Third Bit Complement (TBC) Mechanism to Reduce Bit Stuffing Jitter in Controller Area Network (CAN)," *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (An ISO 3297: 2007 Certified Organization)*, vol. 4, no. 5, 2015.
[22] T. R. Jena, A. K. Swain, and K. Mahapatra, "A novel bit stuffing technique for Controller Area Network (CAN) protocol," in *Advances in Energy Conversion Technologies (ICAECT), 2014 International Conference on*, 2014, pp. 113–117.
[23] S. K. Kabilesh and B. V. Kumar, "Design and simulation of modified selective XOR algorithm for payload attrition in CAN," in *2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2016, vol. 01, pp. 1–6.
[24] H. J. Lad and V. G. Joshi, "Hybrid message conversion technique to reduce jitter in CAN based distributed embedded system," in *Emerging Technology Trends in Electronics, Communication and Networking (ET2ECN), 2014 2nd International Conference on*, 2014, pp. 1–6.
[25] K. R. Priyanga, K. Venkatesan, and others, "A fixed length payload encoding for can," *International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)*, vol. 3, no. 12, 2014.
[26] M. Bacic, "On hardware-in-the-loop simulation," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, 2005, pp. 3194–3198.
[27] M. Schlager, *Hardware-in-the-Loop Simulation*. Omniscriptum Gmbh & Company Kg., 2008.
[28] M. Nahas, "Impact of Reducing Bit Stuffing Jitter on the Control Performance of a CAN-Based Distributed Furnace System," *Journal of Embedded Systems*, vol. 5, no. 1, pp. 1–6, 2018.
[29] N. A. Stanton, M. Young, and B. McCaulder, "Drive-by-wire: The case of driver workload and reclaiming control with adaptive cruise control," *Safety science*, vol. 27, no. 2–3, pp. 149–159, 1997.

[30] M. Short and M. J. Pont, "Hardware in the loop simulation of embedded automotive control system," in *Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005.*, 2005, pp. 426–431.
[31] M. Short, M. J. Pont, and J. Fang, "Assessment of performance and dependability in embedded control systems: Methodology and case study," *Control Engineering Practice*, vol. 16, no. 11, pp. 1293–1307, Nov. 2008.
[32] M. Nahas, "Developing a Novel Shared-Clock Scheduling Protocol for Highly-Predictable Distributed Real-Time Embedded Systems," *American Journal of Intelligent Systems*, vol. 2, no. 5, pp. 118–128, Dec. 2012.
[33] M. J. Pont, *Patterns for time-triggered embedded systems: building reliable applications with the 8051 family of microcontrollers*. Harlow: Addison-Wesley, 2001.
[34] M. Nahas, "Implementation of highly-predictable time-triggered cooperative scheduler using simple super loop architecture," *International Journal of Electrical & Computer Sciences*, vol. 11, pp. 33–38, 2011.
[35] M. Nahas and A. Maaita, "Choosing Appropriate Programming Language to Implement Software for Real-Time Resource-Constrained Embedded Systems," in *Embedded Systems - Theory and Design Methodology*, K. Tanaka, Ed. InTech, 2012.
[36] S. Johansson, E. Langjord, and S. Pettersson, "Objective evaluation of shunt and shuffle in vehicle powertrains," in *7th International Symposium on Advanced Vehicle Control, Arnhem, The Netherlands*, 2004.