Received January 25, 2018, accepted March 6, 2018, date of publication April 16, 2018, date of current version May 2, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2826522

Analytical and Experimental Performance Evaluations of CAN-FD Bus

RICARDO DE ANDRADE[1], KLEBER N. HODEL[1], JOÃO FRANCISCO JUSTO[1], ARMANDO M. LAGANÁ[1], MAX MAURO SANTOS[2], (Senior Member, IEEE), AND ZONGHUA GU[3]

[1]Escola Politécnica, Universidade de São Paulo, CP 61548, CEP 05424-97, São Paulo, Brazil
[2]Departamento de Eletronica, Universidade Tecnológica Federal do Paraná, Ponta Grossa, Paraná, Brazil
[3]Department of EECS, University of Missouri, Columbia, MO 65211, USA

Corresponding author: Zonghua Gu (guzo@missouri.edu)

This work was supported in part by the U.S. Department of Commerce under Grant BS123456 and in part by NSFC under Grant 61672454.

ABSTRACT Controller area network (CAN) is a widely-used bus protocol in automotive distributed embedded systems, but its limited communication bandwidth (up to 1 Mbps) and payload size (up to 8 Bytes) limit its applicability in today's increasingly complex automotive electrical/electronic systems. CAN with flexible data rate (CAN-FD) is an improved CAN-based communication protocol, with higher communication bandwidth (up to 8 Mbps for the payload) and increased payload size (up to 64 Bytes). In this paper, we perform analytical and experimental performance comparisons of CAN-FD bus with conventional CAN bus. We consider a message set obtained by reverse-engineering a real CAN -based system, with additional high-priority interference messages for stress-testing the system with different bus loads. We also consider a networked control system based on the message set, and analyze the control system performance measured by step responses under different bus loads. Experimental results validate the performance advantages of CAN-FD over conventional CAN bus.

INDEX TERMS Automotive networks, CAN bus, CAN-FD bus, in-vehicle communication systems.

I. INTRODUCTION

The upcoming automotive functionalities, such as advanced drive assistance systems, will require dependability attributes, such as safety, security, reliability, and availability [1]. In-vehicle communication is a key component to support the distributed automotive architecture, providing an infrastructure of control, monitoring, and diagnostic applications. Faults generate errors on transmitted messages, by corrupting their contents, such that, in order to recover from those situations, fieldbus networks implement several fault-tolerant mechanisms. However, this leads to communication overload due to delivery delays in messages, which impacts the performance of the control system. When messages have real-time requirements, which is common in control systems, these problems can seriously disturb its operation and lead to failure. A proper operation of those automotive control systems is imperative, which compels an evaluation of their vulnerabilities [2].

The CAN bus is a multi-master message broadcast system used in most automobiles, with a maximum signal rate of 1 Mbps. In contrast to traditional networks, such as USB and Ethernet, CAN does not send large data blocks point-to-point under the supervision of a central bus master. In a CAN network, many short messages are broadcasted to the entire network, which provides data consistency in every node of the system [3].

The recent increase on bandwidth requirements for automotive networks has led CAN to its limit. In order to address those requirements, CAN-FD, which has been recently developed, allows higher rates and payloads. This new protocol has controllers that also perform standard CAN communication, representing a new alternative for future automotive systems [4]. The motivations leading to this new protocol include: (i) to meet increased demand for communication bandwidth; (ii) to provide a protocol that fills the gap between the low-speed and cheap CAN bus with maximum bandwidth of 1 Mbps and the high-speed but expensive FlexRay with maximum bandwidth of 10-20 Mbps; (iii) to avoid the expensive porting effort involved in migrating from CAN to FlexRay or Ethernet [4], [10].

The actual and future systems such in automotive and industrial fields contain complex functions that demand for

CAN-based higher layer protocols with high data throughput, CAN (Controller Area Network) technology provides a bandwidth up to 1 Mbps and payload up to 8 Bytes, that for actually is overloaded and cannot address the demand for the next generation of automotive electronic systems. Despite several proposed alternatives, much effort has been focused on the enhancement of CAN, which is born the CAN-FD (Controller Area Network with Flexible Data Rate). Then, CAN-FD is the improved communication technology with a high bandwidth up to 8 Mbps and payload up to 64 Bytes [4].

In addition, the authors of this paper aim to highlight some other functions that have to be fulfilled by the higher layer protocols such flow control, transportation of messages, node addressing, networking via gateways and network management that CAN-FD allows to keep their original properties with a wide range of actuation also. Then, the updating for CAN-FD isn't only limited to increasing of bandwidth and payload.

This investigation has explored the CAN-FD bus, presenting an analytical analysis on the response time of messages that need to satisfy their deadlines, in scenarios with and without errors. A new model has been proposed, which enables to evaluate the network dependability with respect to deadline failures in the presence of transient faults induced by external sources. Additionally, the effects of faults on real-time properties of the network are determined [5].

The remainder of this paper is organized as follows. In Section II, we describe the major characteristics and properties of the CAN-FD such the specification, frame formats, bandwidth and payload and controller structure. In Section III, we describe the analytical model of timing analysis and bus load for CAN-FD. In Section IV, we evaluate the response timing and bus load improvement of adopt the CAN-FD in the application of automotive distributed architecture in the level of control. It can be extended for the diagnostic and end-of-line target. In Section V, we use an industrial case study of a networked control system for motor control, to illustrate how the bus load affects control system performance for both CAN and CAN-FD buses. Finally, conclusions and future work are discussed in Section VI.

II. CONTROLLER AREA NETWORK WITH FLEXIBLE DATA RATE

The CAN-FD protocol [5], [6], which satisfies those requirements, extends the useful data length or payload in frames from 8 to 64 Bytes, along with higher data transmission rates, with speeds up to 8 Mbps. Fig. 1 presents the evolution of CAN specifications, since its first release in 1991. This new bus is inexpensive and retains the good performance and robustness of the original CAN.

CAN-FD specifications have been recently presented [7], [8], and the respective protocol has been submitted as ISO 11898-2 2016 for international standardization. It presents new features that enable higher speeds than the traditional CAN.

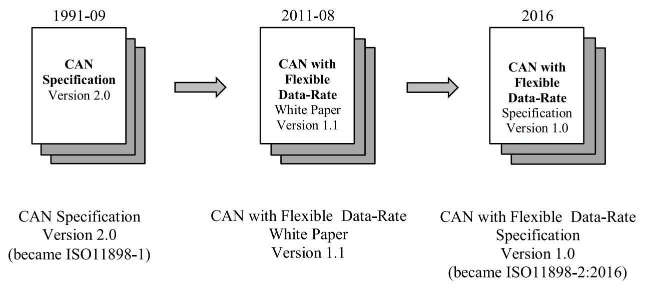
1991-09
CAN Specification Version 2.0
2011-08
CAN with Flexible Data-Rate White Paper Version 1.1
2016
CAN with Flexible Data-Rate Specification Version 1.0
CAN Specification Version 2.0 (became ISO11898-1)
CAN with Flexible Data-Rate White Paper Version 1.1
CAN with Flexible Data-Rate Specification Version 1.0 (became ISO11898-2:2016)

FIGURE 1. From CAN Specification V2.0 to CAN-FD Specification V1.0.

Three new bits are added in the arbitration field: the EDL (Extended Data Length), the BRS (Bit-Rate Switch), and the ESI (Error State Indicator). The EDL field defines the data length with the highest bit at zero if the message size is 0 to 8 Bytes (0000 to 1000) and using the remaining 7 combinations with the highest bit at 1 to encode the lengths 8, 12, 16, 20, 24, 32, 48, and 64 (1000 to 1111). The BRS bit determines whether the message is transmitted with the normal rate (up to 1 Mbps) or with the increased one (up to 8 Mbps). The ESI bit is transmitted dominant by error active nodes, recessive by error passive nodes [1]. Furthermore, CAN in Automation (CiA) recommends using the terms "ISO CAN-FD" and "non-ISO CAN-FD". All products compliant to the upcoming ISO standard 11898-2:2016 should be called "ISO CAN-FD". Products implementing Bosch's original protocol should be named "non-ISO CAN-FD" [5].

The maximal CAN bus speed has been limited due to the In-Frame Response (IFR) mechanism and ACK (Acknowledgement) generation delay in the controller, there is a delay through the transceiver and over wire propagation [7], [9].

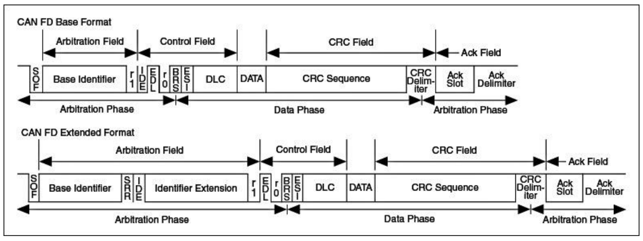

FIGURE 2. The CAN-FD Base and Extended Formats in accordance with ISO 11898-1:2016.

Fig. 2 shows the frame structure of CAN-FD, with the standard and extended frames. The standard frame has arbitration phase with length of 29 bit times (bit stuffing is not considered) and maximal data-rate of 1Mbps. Data phase has 86 bit times (8 data Bytes) and remote frames contain RTR (Remote Transmission Request) bit replaced by reserved bit r1, taking part in CAN arbitration. The extended frame has arbitration phase with length of 49 bit times (bit stuffing is not considered) and maximal data-rate of 1 Mbps. Data phase has 86 bit times and remote frames contain RTR bit replaced by reserved bit r1, taking part in CAN arbitration and a reserved one for protocol expansion [4], [11].

EDL – Extended Data Length

First reserved bit in standard frames

EDL = recessive indicates CAN-FD frame format (new DLC-coding and CRC)

EDL = dominant indicates standard CAN frame format

r1, r0 – reserved bits

Transmitted dominant, reserved for future protocol variants

BRS – Bit Rate Switch

BRS = recessive: switch to alternate bit rate

BRS = dominant: do not switch bit rate

ESI – Error State Indicator

ESI = recessive: transmitting node is error passive

ESI = dominant: transmitting node is error active

As in traditional CAN, message ID encodes message priority (lower ID implies higher priority). Transmission of the message ID forms the arbitration stage, with bandwidth of traditional CAN bus (up to 1Mbps). The payload of a CAN-FD message can be up to 64 Bytes, with increased bandwidth (up to 8 Mbps). Fig. 3 shows that CAN-FD has higher bandwidth, hence lower transmission time in the Data Phase (assumed to be 4Mbps) compared to the CAN bus (assumed to be 1Mbps). However, the larger payload can be exploited only by redesigning (entirely or in part) the message set. Finding a new packing of signals into CAN-FD frames is not a trivial problem, indeed, it is an instance of a variable size bin-packing problem [2].

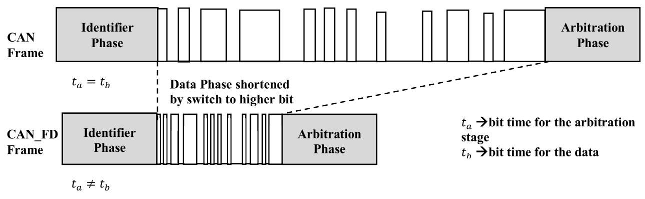
CAN Frame
Identifier Phase
Arbitration Phase
$t_a = t_b$
Data Phase shortened by switch to higher bit
CAN_FD Frame
Identifier Phase
Arbitration Phase
t_a →bit time for the arbitration stage
t_b →bit time for the data
$t_a \neq t_b$

FIGURE 3. Data field phase is shortened by switching from lower to higher bit rate.

A distributed application for automotive systems demands event-triggered and time-triggered communication in serial bus, in which the signal latency is a requirement to maintain good performance. Physical layer is unchanged using the same CAN transceiver. The protocol controller has two sets of timing configuration. BTL (Bit Timing Logic) and BRP (Baud Rate Prescaler) control bit timing switch between two sets and BSP (Bit Stream Processor) controls frame (de) coding defining Arbitration and Data Phase. CAN Message Handling has a shift register as (de) serialized and BSP does not limit data field length. Fig. 4 presents the CAN-FD controller structure.

In a CAN protocol controller, the state machine of Bit Timing Logic (BTL) is evaluated in each bit based on time quantum base to synchronize the position of the Sample-Point to a specific phase with respect to the edges in the monitored bit stream. CAN nodes synchronize on received edges from recessive to dominant in the bus line. The phases

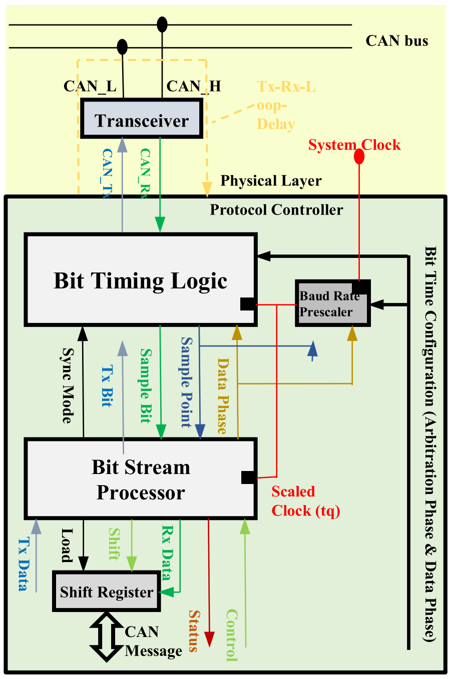

FIGURE 4. The block diagram structure of CAN-FD controller.

of their Sample-Points are shifted with respect to the phase of the transmitter Sample-Point. A node specific phase shift depends on the signal delay time from the transmitter to that specific node [11].

The message delay time between the nodes is based on the net cable length and transceiver delay, which must be considered when more than one node transmits a bit. The configuration of the CAN bit time, especially the Propagation Segment length should be programmed to define where the Sample-Point position cannot be set. Once the arbitration phase of the message is decided, until the end of the CRC Field, only one node transmits dominant bits, while all other nodes synchronize themselves to this single transmitter. Therefore, it is possible to switch to a predefined (shorter) bit time in this part of a CAN frame, here labeled Data-Phase.

III. TIMING ANALYSIS AND BUS LOAD OF CAN-FD

A distributed application for automotive systems or industry automation demands event-trigger and time-trigger communication in serial bus, where the signal latency is a requirement to keep a good performance. Besides that, the applications for the next generation of automotive systems are requiring for the communication bus by higher bandwidth and payload as well.

The Tindell equation [12] can be used to calculate the Worst-Case Transmission Time (WCTT) of a CAN message:

$$C_m = \left(O + 8s_m + \left\lfloor \frac{T_m + 8s_m}{5} \right\rfloor \right) \tau_{bit} \tag{1}$$

We adopt a similar method to calculate the WCTT of a CAN-FD message, C_m:

$$C_m = T_s + T_f \tag{2}$$

where T_s is transmission time of the fixed-size message header, with low transmission bandwidth of traditional CAN (up to 1 Mbps) to have effective bit-wise priority arbitration; T_f is transmission time of the variable-size message payload, with high transmission bandwidth of CAN-FD (up to 8 Mbps).

The CAN-FD protocol defines five different error detection mechanisms: Two of them work at the bit level, and the other three at the message level. They are (i) Bit Monitoring, (ii) Bit Stuffing, (iii) Frame Check, (iv) Acknowledgement Check and (v) Cyclic Redundancy Check. There are two options of *CRC* which should be denoted as $T_{f17}$for *CRC* length of 17 bits or T_{f21} for *CRC* length of 21 bits.

$$T_S = \left[\frac{\left(SOF+ID+r1+IDE+EDL+r0+\frac{BRS}{2}+\frac{CRCdel}{2}\right)*1.2}{t_X}\right] + \frac{ACK+DEL+EOF+IFS}{t_X} \quad (3)$$

where *SOF (Start of Frame)* + *ID* (Identifier) + *r1 (reserved bit 1)* + *IDE* + *EDL(Extended Data Length)* + *r0(reserved bit 0)* + *BRS/2 (Bit Rate Switch)* + *CRCdel/2* (*CRC delimiter*) = 17 bits, 1.2 is the factor of the worst case bit stuffing, which means it is necessary to divide by 5. It is considered BRS and CRCdel divided by 2, because they are exactly in the shift of bit rate transition. The *ACK* (Acknowledge) + *DEL* (Delimiter) + *EOF* (End-of-Frame) + *IFS* (Interframe Spacing) = 12 bits without bit stuffing. The CAN-FD payload size D_f may be $0, 8, 12, 16, 20, 24, 32, 36, 48, 64$ Bytes. t_X is the transmission bandwidth for the message header (up to 1 Mbps). Table 1 shows the meaning and bit width of each field in a CAN-FD message.

The calculation of variable time bits, when the data length is smaller than 16 Bytes, is:

$$T_{f17} = \frac{\left[\left(D_f+\frac{BRS}{2}+ESI+DLC+\frac{CRCdel}{2}\right)*1.2\right]+CRC_{17}+5}{t_Y} \quad (4)$$

where the CRC_{17} bit field is 5 bits from bit stuffing + 17 bits from *CRC* and $D_f > 16$ Bytes of data. t_Y is the transmission bandwidth for the payload (up to 8 Mbps).

The calculation of variable time bits, when the data length is higher than 16 Bytes is:

$$T_{f21} = \frac{\left[\left(D_f+\frac{BRS}{2}+ESI+DLC+\frac{CRCdel}{2}\right)*1.2\right]+CRC_{21}+6}{t_Y} \quad (5)$$

where the CRC_{21} bits field is 6 bits from bit stuffing +21 bits from *CRC*. It must be 21, since the field has a larger space of Bytes, such that it is necessary to increase the field to improve the accuracy of the error detection.

TABLE 1. Bit fields in a CAN-FD message and their lengths.

Field Name	Length (bits)
SOF (Start of Frame)	1
ID (Identifier)	11
r1 (Reserved Bit 1)	1
IDE (Identifier Extension Bit)	1
EDL (Extended Data Length)	1
r0 (reserved Bit 0)	1
BRS (Bit Rate Switch)	1
CRCdel (CRC Delimiter)	1
ACK(Acknowledge)	1
DEL (delimiter)	1
EOF (End of Frame)	7
IFS (Interframe Spacing)	3
DLC (Data Length Code)	4
ESI (Error State Indicator)	1

Having obtained the WCTT of each message C_m, and given the period of each message T_m, the system bus load can be obtained by summing up the bus load of each message:

$$U = \sum_m \frac{C_m}{T_m} \quad (6)$$

In this paper, we assume the transmission bandwidth for CAN bus is **0.5**Mbps; the transmission bandwidth for the CAN-FD bus header is $\mathbf{t_X} = \mathbf{0.5}$Mbp*s*; the transmission bandwidth for the CAN-FD message payload is $\mathbf{t_Y} = \mathbf{4}$Mbps.

IV. CASE STUDY I : AUTOMOTIVE DISTRIBUTED ARCHITECTURE

A comparison between standard CAN and CAN-FD is performed considering a case study, with several data load configurations. The case study used a Volkswagen Polo vehicle in order to read the CAN frames on the bus and to build a CAN-FD simulation with that data. The CAN-FD simulation communication is built to transmit the same content of vehicular messages, but in the CAN-FD frame.

The first step is to reverse engineer the CAN bus message set by using Vector's toolset for measurement on a real CAN-based system, consisting of 5 ECUs, including the Engine Controller, the Comfort Controller, the Instrument Cluster, the Air Conditioning Controller, and one unidentified ECU. We can identify the source of each message by removing certain ECUs from the bus and observing which messages remain on the bus.

Table 2 shows the 19 CAN messages obtained from reverse-engineering. The 1st column contains the CAN frame IDs in hex number notation; the 2nd column contains the ECU that sends that message with the frame ID; the 3rd column contains the Data Length Code (DLC), ranging from 0 to 8 in Bytes; the 4th column contains the message period; the 5th column contains the WCTT of the message obtained with Equation (1); the 6th column contains the message frequency

TABLE 2. CAN message set by reverse engineering.

Frame ID (hex)	Sender ECU	DLC (Bytes)	Period T_m(ms)	WCCT C_m (ms)	Frame /s	Bus load (%)
280	Engine	8	10	0.2532	100	2.53
288	Engine	8	10	0.2532	100	2.53
380	Engine	8	10	0.2532	100	2.53
480	Engine	8	20	0.2532	50	1.27
488	Engine	8	10	0.2532	100	2.53
588	Engine	8	20	0.2532	50	1.27
580	Engine	8	1000	0.2532	1	0.03
320	Cluster	8	25	0.2532	40	1.01
420	Cluster	8	200	0.2532	5	0.13
520	Cluster	8	200	0.2532	5	0.13
5E0	AC	8	20	0.2532	50	1.27
388	Comfort	3	20	0.1572	50	0.79
38A	Comfort	4	20	0.1764	50	0.88
470	Comfort	5	50	0.1956	20	0.39
5D0	Comfort	6	100	0.2148	10	0.21
5D8	Comfort	8	100	0.2532	10	0.25
570	Comfort	4	100	0.1764	10	0.18
50	Unidentified	2	20	0.138	50	0.69
3D0	Unidentified	4	20	0.1764	50	0.88
Total bus load						19.49

(inverse of message period); the last column contains the bus load of each message, obtained with C_m/T_m; the last row contains the total bus load of the message set. Each message has a priority that is inversely related to its message ID, i.e., smaller ID means higher priority, as defined by the bit-wise arbitration mechanism of the CAN bus header.

In order to stress-test the system, we generated additional dummy messages to create interference and increase the bus load. All dummy messages have period 10ms and payload DLC = 8 Bytes. Table 3 shows the five different configurations we created for a CAN bus-based system, with bus loads of ∼20%, ∼30%, ∼50%, ∼70% and ∼100%, , respectively. For example, CAN_Cfg0 consists of the 19 original messages in Table 2 with bus load of 19.49%; CAN_Cfg1 consists of the 19 original messages in Table 2 plus 5 dummy messages with total bus load of 29.62%.

TABLE 3. Message set configurations for CAN bus.

	# Messages	Bus Load(%)
CAN_Cfg0	19	19.49
CAN_Cfg1	19+5=24	29.62
CAN_Cfg2	19+12=31	49.88
CAN_Cfg3	19+20=39	70.13
CAN_Cfg4	19+32=51	98.86

A. TRANSFORMATION OF MESSAGES FROM CAN TO CAN-FD

This section presents the procedure to obtain the simulation results, the development of the case study and the modeling of data for CAN-FD. A methodology is developed to transform a CAN bus network in a CAN-FD one. CAN bus can carry from 0 to 8 data Bytes while CAN-FD from 0 to 64 data Bytes. The premises used for this transformation are to keep the maximum time performance and the smallest number of messages in the net.

TABLE 4. CAN-FD message set corresponding to Table 2.

Prio	Frame IDs Merged	New Frame ID	Period T_m(ms)	DLC (Bytes)	Real Time (ms)	Bus load (%)
1	280+288+380+480+488+588+580	280	10	64	9.92	2.269
2	320+420+520	320	25	24	25.00	0.523
3	5E0	5E0	20	8	19.97	0.456
4	388+38A+470+5D0+5D8+570	388	20	32	20.00	0.750
5	50+3D0	50	20	6	20.00	0.432
Total bus load						4.43

Table 4 shows the equivalent CAN-FD message set created by grouping multiple messages in the CAN message set in Table 2 by sender ECU, e.g., 7 CAN messages with IDs 280, 288, 380, 480, 488, 588 and 580 sent by the engine ECU are transformed into a single CAN-FD message with ID 280 that carries the same payload. The CAN-FD message ID is set to be the smallest CAN message ID (280), and the CAN-FD message period is set to the smallest CAN message period (10ms), among the group of 7 CAN messages. Other CAN-FD messages are created similarly, e.g., the unidentified CAN messages with IDs 50 and 3D0 are grouped into a CAN-FD message with ID 50.

We have to look for messages to combine them from CAN to CAN-FD denoted such piggybacking. Even 1 bit payloads consume a whole message that we have to consider for messages with same or multiple period and same source.

TABLE 5. Message set configurations for CAN-FD bus.

	# Messages	Bus Load(%)
CANFD_Cfg0	5	4.43
CANFD_Cfg1	5+1=6	6.11
CANFD_Cfg2	5+2=7	8.18
CANFD_Cfg3	5+3=8	10.44
CANFD_Cfg4	5+5=10	13.46

Table 5 shows the five different configurations we created for a CAN-FD bus-based system, with bus loads of 4.43%, 6.11%, 8.18%, 10.44% and 13.46%, respectively. Each CAN-FD configuration contains a CAN-FD message set with the same payload as the corresponding CAN bus configuration in Table 3. For example, CANFD_Cfg0 consists of the 5 original messages in Table 4 with bus load of 4.43%; with the same payload as CAN_Cfg0; CANFD_Cfg1 consists of the 5 original messages in Table 4 plus 1 dummy message with total bus load of 6.11%, with the same payload as CAN_Cfg1. We can see that a CAN-FD message set has much lower bus load than an equivalent CAN message set

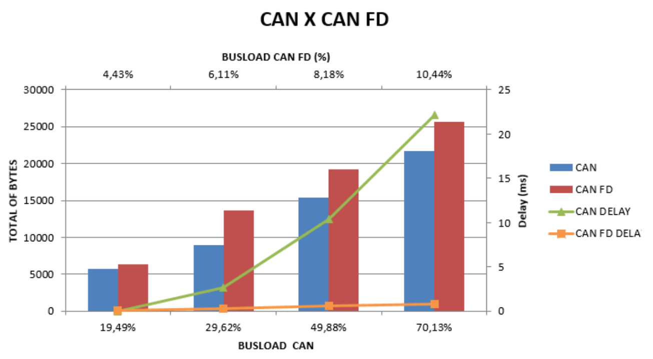

FIGURE 5. Comparison of a real CAN bus and a simulated CAN-FD bus.

thanks to the higher transmission bandwidth for the CAN-FD payload.

B. TIMING ANALYSIS OF MESSAGES

Fig. 5 shows comparisons between a real CAN bus and a simulated CAN-FD bus, showing the total number of transmitted Bytes in one measurement period of 1 second, and the worst-case delay time of the message with the longest delay for 4 CAN or CAN-FD configurations (Cfg0 - Cfg3). The worst-case message delay for the CAN bus ranges from ∼1ms for CAN_Cfg0 with bus load of 19.49%, to ∼20ms for CAN-Cfg3 with bus load of 70.13%. The worst-case message delay for the CAN-FD bus ranges from ∼0.2ms for CANFD_Cfg0 with bus load of 4.43%, to ∼0.8ms for CANFD_Cfg3 with bus load of 10.44%. The results indicate a major limitation of CAN, related to the delay time, when the bus load is high. Such limitations are critical for the current automotive technology, in which vehicles already have many modules and the communication system should have negligible time delays. In the case of CAN-FD, the limits are extended with the possibility to increase the number of messages in the network, in order to address the current requirement of the automotive electronic architectures.

V. CASE STUDY II: NCS – AWARE FOR MOTOR CONTROL

Considering that automotive systems are controlled by sensors and actuators, in most cases using closed loop control, the second case study analyzed the behavior of a closed loop system bus, where the CAN or CAN-FD bus forms part of the loop, as shown in Fig. 6. The system consists of a DC motor (Fig. 7), an actuator, a sensor, and a PID (Proportional, Integral and Derivative) controller. The comparison is carried by measurements of output signals coupled to the control loop sensors. The controller output sends a signal to the driver in CAN or CAN-FD buses, sending the data to an actuator, which is activated, powering the DC motor. The sensor receives the information from the motor and sends it over the network back to the controller, in a closed loop cycle. This closed loop system could eliminate perturbations that may occur in the network. Therefore, considering the

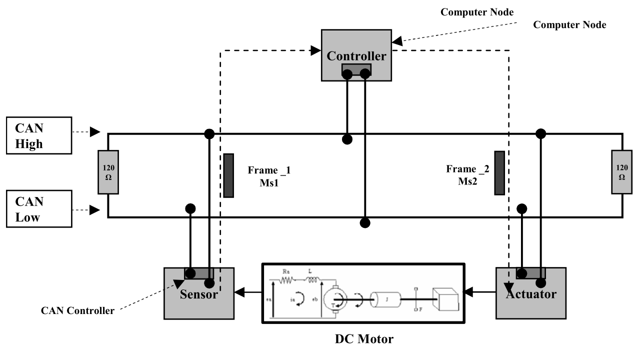

FIGURE 6. Networked control system case study.

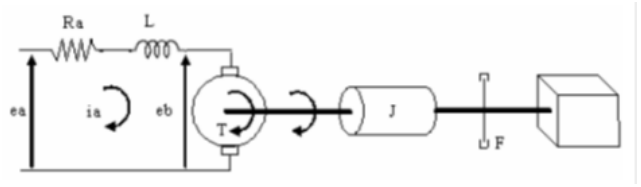

FIGURE 7. Dynamic model of a DC motor [11].

message delays, according to the bus load of each setting, it is possible to identify if the communication network can send all the messages without critical delays.

The equations that describe the dynamic model of the DC motor are given by [4]:

$$e_a = e_b + L_a \frac{di_a}{dt} + R_a i_a \tag{7}$$

$$e_b = K_l \frac{d\theta}{dt} \tag{8}$$

where $e_a\,(E_a), e_b, R_a, L_a$, and i_a are respectively input voltage, output voltage, resistance, inductance, and current of the motor armor. K_l is a constant and electromotive force is the angular position of the motor. The torque T is defined as:

$$T = J\frac{d^2\theta}{dt^2} + F\frac{d\theta}{dt} \tag{9}$$

$$T = K_2 \cdot i_a \tag{10}$$

where J is the moment of inertia, F is constant of viscous friction, and K_2 is the motor torque constant. It is the classical model of DC-motor.

The system transfer function in Laplace domain is:

$$\frac{\theta\,(s)}{E_a\,(s)} = K\frac{k}{s\left[L_aJs^2 + (L_aF + R_aJ)\,s + R_aF + K_2K_1\right]} \tag{11}$$

The transfer function can be simplified by neglecting the armor inductance L_a:

$$\frac{\theta\,(s)}{E_a\,(s)} = \frac{k}{s\,(\tau s + 1)} = G_p\,(s) \tag{12}$$

$$K = \frac{K_2}{(R_aF + K_2K_1)} \tag{13}$$

where:

$$\tau = \frac{R_aJ}{(R_aF + K_2K_1)} \tag{14}$$

The DC motor dynamics can be described as [11]:

$$G_p(s) = \frac{2029.826}{(s+26.29)(s+2.296)} \tag{15}$$

We use a PID controller with the following generic equation:

$$G_C(s) = \frac{\beta K_p\left[s+\left(\frac{K_1}{K_p}\right)\right]}{s} \tag{16}$$

where β is a tuning parameter of K_1 and K_p (the integral and proportional gains), assumed to be $\beta = 1$ here.

We use a heuristic method [13] to choose the sampling period h with value range defined by the condition $0.2 < \omega_b.h < 0.6$. For instance, for a physical process with $\omega_b = 40\ rad/s$, the sampling period may be chosen as $h = 6ms$.

Fig. 8 shows the closed loop system consisting of the physical plant described with Equations (15), and the controller described with Equation (16), with proportional and integral gains set to be $K_p = 0.1701$ and $K_i = 0.3780$.

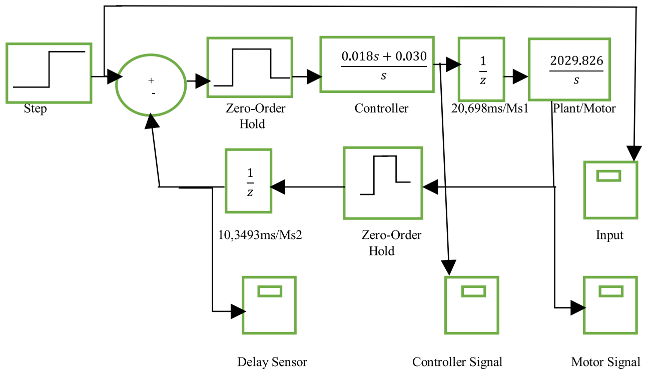

FIGURE 8. Block diagram of motor control system with a PID controller.

Fig. 9 shows the Simulink block diagram of the networked control system, where the physical plant and controller communicating through a bus (CAN or CAN-FD), and A/D, D/A converters.

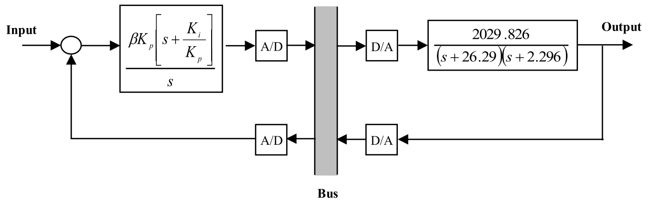

FIGURE 9. Simulink block diagram representing a networked control system connected by a CAN or CAN-FD bus.

In this example, messages are transmitted periodically. The control for the motor drive is connected to the controller and is periodically transmitted by Ms1, according to Fig. 10, and the actuator had a filter to receive this message and start the motor. Some tasks are included in the loop (τ_{1a} to τ_3) with additional time slots.

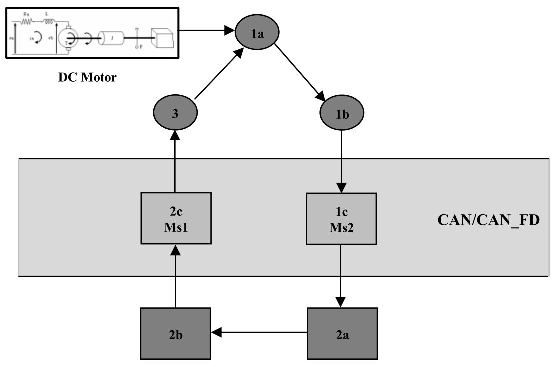

FIGURE 10. Timing diagram of the networked control system.

TABLE 6. Tasks description for networked control systems over CAN and CAN-FD bus.

Time	Task Description
τ_{1b}	Transmit the sensor message (queue time for transmission)
τ_{1c}	Transmit messages through bus communication
τ_{2a}	Receive message of sensor, process and generate message with signal actuator
τ_{2b}	Transmit message of actuator (queue time for transmission)
τ_{2c}	Transmit messages through bus communication
τ_3	Receive messages of controller and actuator

The closed loop of CAN and CAN-FD networks consists of 7 tasks that are messages transmitted between the bus and the motor control, as shown in Table 6. Each task introduces a corresponding delay in the control loop. The total time of the control loop:

$$\sum_{i=1}^{n} \tau_i = \tau_{1a} + \tau_{1b} + \tau_{1c} + \tau_{2a} + \tau_{2b} + \tau_{2c} + \tau_3 \tag{17}$$

We assume delays τ_{1a}, τ_{2a} and τ_3 to be 0, since we want to focus on the timing of the bus, hence:

$$\sum_{i=1}^{n} \tau_i = \tau_{1b} + \tau_{1c} + \tau_{2b} + \tau_{2c} \tag{18}$$

With CAN bus speed of 500 Kbps and message size of 8 Bytes, we have $T_m = 34\ bits, O = 43\ bits, s_m = 64\ bits, \tau_{bit} = 2\mu$ s. Using Equation (1), we can compute the WCTT $\tau_{1c} = \tau_{2c} =$0.25ms.

TABLE 7. Queue delay of messages in CAN bus for networked control system.

ID	CAN_Cfg1	CAN_Cfg2	CAN_Cfg3	CAN_Cfg4
Bus load (%)	29.62	49.88	70.13	98.86
Queue time of Ms1 (ms)	0.445	-0.224	2.975	1000+
Queue time of Ms2 (ms)	-0.0007	0.778	2.538	1000+

Table 7 shows the 4 CAN bus configurations considered here. We focus on two CAN messages, Ms1 (with ID 5E0 and period 20ms) and Ms2 (with ID 620 and period 10ms). The queue time of each message is the subtraction from theoretical message time triggered and real message time from

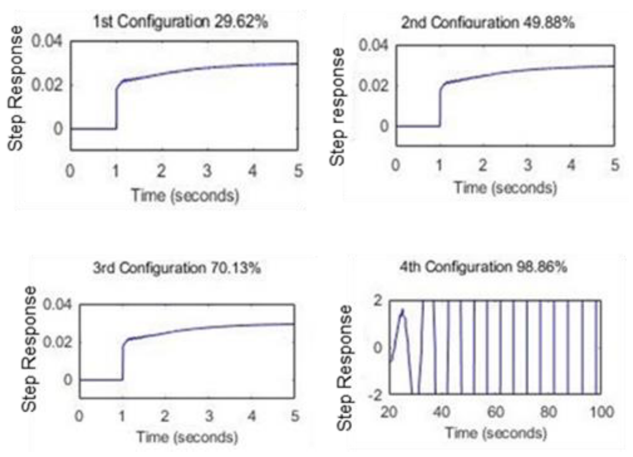

FIGURE 11. Step responses of the networked control system with CAN bus.

measurement. For example, ID 5E0 in CAN_Cfg1 has 20ms theoretical message time triggered and 20.445ms real message time, then the queue time is 0.445ms. A negative value indicates that the message is transmitted instantaneously, and the specific value is just within the error in the simulation. On the other hand, a positive value indicates that the message arrived late. In addition to the time spent in the queue for transmission, the message periods (20ms and 10ms) are considered delays in the control loop, since message period also affects control system performance.

(i) CAN_Cfg1:

Total delay of Ms1 $= \tau_{2b} + \tau_{2c} + 20 = 20.695$ms;

Total delay of Ms2 $= \tau_{1b} + \tau_{1c} + 10 = 10.2493$ms.

(ii) CAN_Cfg2:

Total delay of Ms1 $= \tau_{2b} + \tau_{2c} + 20 = 20.026$ms;

Total delay of Ms2 $= \tau_{1b} + \tau_{1c} + 10 = 11.028$ms;

(iii) CAN_Cfg3:

Total delay of Ms1 $= \tau_{2b} + \tau_{2c} + 20 = 23.225$ms;

Total delay of Ms2 $= \tau_{1b} + \tau_{1c} + 10 = 12.788$ms.

(iv) CAN_Cfg4:

Total delay of Ms1 $= \tau_{2b} + \tau_{2c} + 20 > 1000$ms;

Total delay of Ms2 $= \tau_{1b} + \tau_{1c} + 10 > 1000$ms.

Fig. 11 shows the step responses for 4 different CAN bus configurations, i.e., the plant output in response to a reference signal of step input at time 1s. As expected, the control performance deteriorates with increasing bus load, especially for CAN_Cfg4 with bus load of 98.86%, the plant goes unstable and oscillates instead of tracking the reference signal.

Next, we consider the CAN-FD Bus. We focus on two CAN-FD messages, Ms1 (with ID 5E0 and period 20ms) and Ms2 (with ID 620 and period 10ms). Using Equation (2), we can compute the WCTT for Ms1 $\tau_{1b} = 0.27$ms; WCTT for Ms2 $\tau_{2b} = 0.22$ms. Table 8 shows the time of sending messages in CAN-FD, according to the bus load of each configuration.

(i) CANFD_Cfg1:

Total delay of Ms1 $= \tau_{2b} + \tau_{2c} + 20 = 10.138$ms;

Total delay of Ms2 $= \tau_{1b} + \tau_{1c} + 10 = 20.239$ms.

TABLE 8. Queue delay of messages in CAN-FD bus for networked control system.

ID	CANFD _Cfg1	CANFD _Cfg2	CANFD _Cfg3	CANFD _Cfg4
Bus load (%)	6.11	8.18	10.44	13.46
Queue time of Ms1 (ms)	-0.082	-0.082	-0.082	-0.082
Queue time of Ms2 (ms)	-0.031	-0.031	-0.043	-0.043

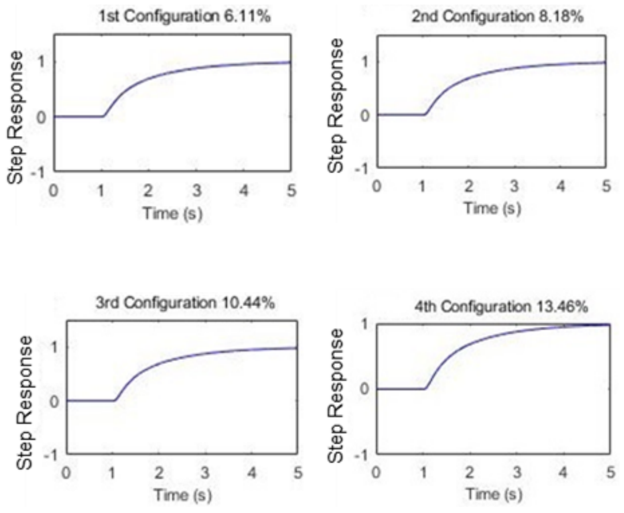

FIGURE 12. Step responses of the networked control system with CAN-FD bus.

(ii) CANFD_Cfg2:

Total delay of Ms1 $= \tau_{2b} + \tau_{2c} + 20 = 10.138$ms;

Total delay of Ms2 $= \tau_{1b} + \tau_{1c} + 10 = 20.239$ms.

(iii) CANFD_Cfg3:

Total delay of Ms1 $= \tau_{2b} + \tau_{2c} + 20 = 10.138$ms;

Total delay of Ms2 $= \tau_{1b} + \tau_{1c} + 10 = 20.227$ms.

(iv) CANFD_Cfg4:

Total delay of Ms1 $= \tau_{2b} + \tau_{2c} + 20 = 10.138$ms;

Total delay of Ms2 $= \tau_{1b} + \tau_{1c} + 10 = 20.227$ms.

Fig. 12 shows the step responses for 4 different CAN-FD bus configurations. Especially for CANFD_Cfg4, the control system is stable, in contrast to the unstable system for CAN_Cfg4.

VI. CONCLUSIONS AND FUTURE WORK

This paper explored CAN-FD and its potential advantages over CAN. CAN presented major limitations and low predictability; for example, when a system is added or changed, the entire network is affected. On the other hand, CAN-FD allowed addition of many messages with many modules, even at a high bus load. However, predictability is compromised beyond a certain number of inserted messages. The tests indicated that the system stopped transmitting messages for a bus load over 98%.

We developed equations to compute the bus load in CAN-FD and the limit to transmit messages without large delays, when a control system is inserted in the bus, allowing to further an understanding on this protocol. Moreover, this information allows to predict precisely if the new bandwidth requirements from current safety and comfort systems in the automotive applications could be covered by the CAN-FD.

Results of simulations with both networks indicated that CAN-FD performed considerably better than CAN.

As part of future work, we plan to consider optimized mapping and scheduling of application tasks to a distributed platform [14], as well as other non-functional properties such as security [15]. For example, the increased payload size of CAN-FD gives more space in the message body for including a Message Authentication Code for defense against message spoofing attacks.

REFERENCES

[1] H. Ueda, R. Kurachi, H. Takada, T. Mizutani, M. Inoue, and S. Horihata, "Security authentication system for in-vehicle network," SEI, Oaks, PA, USA, Tech. Rep. 8, Oct. 2015.

[2] S. Woo, H. J. Jo, I. S. Kim, and D. H. Lee, "A practical security architecture for in-vehicle CAN-FD," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 8, pp. 2248–2261, Aug. 2016.

[3] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, "Trends in automotive communication systems," *Proc. IEEE*, vol. 93, no. 6, pp. 1023–1024, Jun. 2005.

[4] G. Cena, I. C. Bertolotti, T. Hu, and A. Valenzano, "CAN with eXtensible in-frame reply: Protocol definition and prototype implementation," *IEEE Trans. Ind. Informat.*, vol. 13, no. 5, pp. 2436–2446, Oct. 2017.

[5] A. Carvalho and F. Vasques, "A model based on a stochastic petri net approach dependability evaluation of controller area networks," in *Proc. 6th IFAC Int. Conf. Fieldbus Syst. Appl.*, Puebla, Mexico, Nov. 2005, pp. 14–16.

[6] (Dec. 2014). *CAN In Automation (CIA)—Newsletter Magazine*. [Online]. Available: https://can-newsletter.org/engineering/standardization/141209_iso-can-fd-or-non-iso-can-fd

[7] I. Broster, A. Burns, and G. Rodrígues-Navas, "Comparing real-time communication under electromagnetic interference," in *Proc. 16th Euromicro Conf. Real-Time Syst. ECTRS*, 2004, pp. 45–52.

[8] (Mar. 2012). *The CAN-FD Protocol Overcomes CAN Limits*. [Online]. Available: http://can-newsletter.org/catalogue/do/printArticle/id/nr_stand_can-fd_2012-03-08/nr_stand_can-fd_2012-03-08.pdf

[9] R. Pinto, J. Rufino, and C. Almeida, "High availability in controller area networks," in *Proc. Int. Conf. Comput*, 2011, pp. 1–4.

[10] R. Murphy, "A CAN to FlexRay migration framework," M.S. thesis, Dept. Comput., Maths Phys., Waterford Inst. Technol., Waterford, Republic of Ireland, 2009.

[11] *Controller Area Network with Flexible Data-Rate Specification—Version 1.0*, Robert Bosch GmbH, Stuttgart, Germany, Apr. 2012.

[12] K. Tindell, H. Hanssmon, and A. Wellings, "Analysing real-time communications: Controller area network (CAN)," in *Proc. RTSS*, 1994, pp. 259–263.

[13] K. J. Astrom and B. Wittenmark, *Computer-Controlled Systems: Theory and Design*, 3rd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 1997.

[14] X. He, Z. Gu, and Y. Zhu, "Task allocation and optimization of distributed embedded systems with simulated annealing and geometric programming," *Comput. J.* vol. 53, no. 7, pp. 1071–1091, 2010.

[15] Z. Gu, G. Han, H. Zeng, and Q. Zhao, "Security-aware mapping and scheduling with hardware co-processors for FlexRay-based distributed embedded systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 10, pp. 3044–3057, Oct. 2016.

RICARDO DE ANDRADE received the technological degree in automotive electronics from the State Center for Technological Education Paula Souza–Fatec Santo André, from 2007 to 2010, and the M.Sc. degree in electrical engineering from the Universidade de São Paulo from 2012 to 2014, where he is currently the Ph.D. degree in electrical engineering. He has experience in electrical engineering, with emphasis in electrical engineering, involved mainly in the following subjects: automotive electronics, CAN bus, CAN-FD bus, automotive communication, automotive electronics, CAN, and CAN-FD bus vulnerabilities and automotive security system reliability. He is currently a Laboratory Technician in electro and electronics with the Federal University of ABC and also an Assistant Professor with the Brazilian University of Advanced Technology.

KLEBER N. HODEL received the bachelor's degree in electrical engineering from the FEI University Center from 1998 to 2003 and the master's degree in electrical engineering from the Universidade de São Paulo from 2005 to 2008, where he is currently pursuing the Ph.D. degree in electrical engineering. He has been a Manager of the New Technologies for Mercedes Benz Chassis Worldwide since 2004 and has been a Professor with the Fatec Santo André since 2008. His main research areas are in electrical engineering with emphasis in automotive electronics, systems and electronic control, and industrial electronics.

JOÃO FRANCISCO JUSTO received the B.Sc. and M.Sc. degrees in physics from the Universidade de São Paulo, in 1988 and 1991, respectively, and the Ph.D. degree in nuclear engineering from the Massachusetts Institute of Technology in 1997. He was a Visiting Associate Professor with the University of Minnesota from 2007 to 2008. He is currently a Professor with the Polytechnic School, Universidade de São Paulo. He has experience in computational modeling of electronic materials and embedded electronics.

ARMANDO M. LAGANÁ received the bachelor's degree in electrical engineering from the Escola de Engenharia Mauá in 1975 and the Ph.D. degree in electrical engineering from the Polytechnic School, University of São Paulo, in 1994. He has been an Associate Professor with the Department of Electronics Systems Engineering, University of São Paulo, Brazil, since 1986. His main research areas are automotive embedded systems, automotive electronics, and software engineering.

MAX MAURO SANTOS (M'15–SM'17) received the bachelor's degree in electrical engineering from the Instituto Católico de Minas Gerais, Brazil, in 1993, and the M.E. and Ph.D. degrees in electrical engineering from the Universidade Federal de Santa Catarina, Brazil, in 1996 and 2004, respectively. He was a Post-Doctoral Fellow in electrical engineering from the Universidade de Aveiro, Portugal, from 2005 to 2006. He is currently an Assistant Professor with the Electronic Department, Universidade Tecnológica Federal do Paraná, Ponta Grossa, Brazil. His main research areas are embedded systems, automotive systems, and industrial automation.

ZONGHUA GU received the Ph.D. degree in computer science and engineering from the University of Michigan, Ann Arbor, in 2004. He was with the Hong Kong University of Science and Technology and Zhejiang University, China, before joining the University of Missouri in 2018. He is currently an Assistant Professor with the Department of EECS, University of Missouri, USA. His research area is embedded and cyber-physical systems.