iCC 1995

2nd international CAN Conference

in London (United Kingdom)

Sponsored by

Motorola Semiconductor
National Semiconductor
Philips Semiconductors

Organized by

CAN in Automation (CiA)
international users and manufacturers group
Am Weichselgarten 26
D-91058 Erlangen
Phone +49-9131-69086-0
Fax +49-9131-69086-79
Email:headquarters@can-cia.de
URL: http://www.can-cia.de

A STUDY ON THE INACCESSIBILITY CHARACTERISTICS OF THE CONTROLLER AREA NETWORK

José Rufino Paulo Veríssimo
ruf@inesc.pt paulov@inesc.pt
Technical University of Lisboa
IST - INESC[*]

ABSTRACT

The Controller Area Network (CAN) is a communication bus for message transaction in small-scale distributed environments. Continuity of service and bounded and known message delivery latency are requirements of a number of applications, which are not perfectly fulfilled by existing networks, CAN included. One key issue with this regard, is that networks are subject to failures, namely partitions. However, most of non-critical applications can live with temporary glitches in network operation, provided these temporary partitions are time-bounded. We call these periods of inaccessibility. Should one call for hard real-time behaviour, a worst-case figure for these non-negligible periods should be derived and added to the worst-case transmission delay expected in the absence of faults. This paper does an exhaustive study of CAN inaccessibility characteristics, presenting figures for intervals in CAN operation when the network does not provide service, although not being failed.

1 INTRODUCTION

In reliable real-time systems, the fundamental requirement of communications is that there be a bounded and known message delivery latency, in the presence of disturbing factors such as overload or faults. For applications where this requirement is very strict (e.g. life-critical ones), specialised space-redundant architectures like point-to-point graphs [1] or multiple local area networks [2,3] are clearly the solution. These architectures are however costly and complex. When that requirement is not so strict, one can found a number of design alternatives for implementing cost-effective network infra-structures, through the use of existing technology [4].

The Controller Area Network (CAN) is a communication bus for message transaction in small-scale distributed environments. Although originally designed for automotive applications, CAN has rapidly gathered a growing attention in the control and automation arena, where field-bus[1] settings play an important role. In these areas, continuity of service and determinism in transmission delay are requirements of a number of applications, which are not perfectly fulfilled by existing networks, CAN included. Despite their limitations, CAN is today a very important design component. So, it is worthwhile investigating if the real-time requirement can be reliably met by such a network. To achieve reliable real-time communication, three fundamental conditions must be validated:

1. bounded delay from request to transmission of a frame[2], given the worst case load conditions;
2. message[3] delivery despite the occurrence of omission failures;
3. control of partitions.

All these points have been comprehensively addressed in [5], although that analysis was centred in local area networks. Specifically regarding CAN, point 1 has been addressed in recent studies [6,7]. On the other hand, it is necessary to study the patterns for omission failures (e.g. number of consecutive omission failures) and for

[*] Instituto de Engenharia de Sistemas e Computadores, R. Alves Redol, 9 - Apartado 10105 - 1017 Lisboa - Portugal, Tel.+351-1-3100000. This work was supported in part by JNICT, STRIDE and FEDER within the framework of the CoDiCom Project.

[1] Real-time instrumentation networks, mainly designed for sensing and actuating.

[2] Physical network information packet.

partitions. Uncontrolled omissions and partitions are a source of asynchrony and inconsistency. This is unacceptable for most systems, let alone real-time ones.

A study of the behaviour of a Controller Area Network with regard to partitions is the central issue of this paper. This study contributes to a better understanding of CAN operation having in mind its relevance for control and automation, where real-time, reliability and accessibility are a must.

2 CONTROLLING PARTITIONS

A network is partitioned when there are subsets of nodes which cannot communicate with each other[4]. In this sense, a network displays a number of causes for partition, not all of them of physical nature (e.g. bus contention). Some have means of recovering from some of these situations, and can/should be enhanced to recover from the others, if reliable real-time operation is desired. However, the recovery process takes time, so in the meantime the network is partitioned. A solution to the problem of controlling partitions was presented in [5]. It is based in the following idea: if one knows for how long a network is partitioned, and if those periods are acceptably short, real-time operation of the system is possible. Let us call them periods of *inaccessibility*, to differentiate from classical partitions. The definition of inaccessibility in [8] is summarised here:

> *Certain kinds of components may temporarily refrain from providing service, without that having to be necessarily considered a failure. That state is called* ***inaccessibility****. It can be made known to the users of the component; limits are specified (duration, rate); violation of those limits implies permanent failure of the component.*

This is not hard to implement, as shown in [5]. To achieve it one must first assure that all conditions leading to partition are recovered from. Then, one needs to show that all the inaccessibility periods are time-bounded and determine the upper bound. This figure, added to the worst-case transmission delay in the absence of faults, give us a consolidated transmission delay bound.

3 THE CONTROLLER AREA NETWORK

The Controller Area Network is a broadcast bus with a multi-master architecture. The transmission medium is usually a twisted pair cable. The network maximum length depends of data rate. Typical bounds are: 40m @ 1 Mbps; 1000m @ 50 kbps [9]. Bit transmission takes two possible representations: *recessive*, which only appears on the bus when all the nodes send recessive bits; *dominant*, which needs only to be sent by one node to appear on the bus. This means that a dominant bit sent by one node can overwrite recessive bits sent by other nodes. This feature is exploited for bus arbitration. A given bit-stream is transmitted using the NRZ[5] code. Data transfers are subject to a bit stuffing technique that prevents more than five consecutive bits of identical polarity to be transmitted, through automatic insertion of a complementary bit.

Accordingly with CAN terminology, data to be transferred is encapsulated within "communication objects": a unique identifier is assigned to each communication object. The Controller Area Network [10] is a *carrier sense multi-access with collision resolve* network: nodes delay transmissions if the bus-line is in use; when a bus idle condition is detected, any node may start transmitting; bus access conflicts are resolved trough comparison of communication objects identifiers and work as follows:

- while transmitting a communication object identifier, each node monitors the serial bus-line;
- if the transmitted bit is recessive and a dominant bit is monitored, the node gives up from transmitting and starts to receive incoming data;
- the node transmitting the object with the highest identifier will go through and gets the bus.

That means: arbitration is *non-destructive*, since transmission of the highest identifier object undergoes without delay; bus access is prioritised, allowing transmission of more urgent data to takeover less urgent one. Automatic retransmission of a communication object is provided after a loss in an arbitration process.
For our purposes, two important parameters characterise the basic operation of any CAN network:

[4] The subsets may have a single element. When the network is completely down, *all* partitions have a single element, since each node can communicate with no one.

- **Data Rate** - The rate of data signalling, on the bus. It gives a meaning to the *bit* time (t_{bit}).
- **Intermission Field Time** - t_{IFS} - The minimum bus idle period that mandatory precedes any data or remote frame transmission.

On the other hand, only four types of protocol data units are used by CAN. Their functions are described below:

- **Data Frame** - used for dissemination of communication objects.
- **Remote Frame** - to explicitly request the dissemination of a communication object. For the same identifier, the data frame takes precedence over the remote transmission request.
- **Error Frame** - for error signalling. It is made up from an *error flag*, followed by an *error delimiter*.
- **Overload Frame** - used to extend the interframe spacing, upon detection of overload conditions. It is made up from an *overload flag* plus an *overload delimiter*.

Frame	Symbol	Duration (_s)	
		min.	max.
Data frame	t_{data}	44.0	127.0
Remote frame	t_{rdata}	44.0	44.0
Error frame	t_{error}	14.0	20.0
Overload frame	t_{oload}	14.0	20.0

Table 1 - Duration of CAN Frames (1 Mbps)

The integrity of data and remote frames is checked by a cyclic redundancy code (CRC)15-bit sequence particularly well suited to check the integrity of frames with a total bit length less than 127, thus providing a good error detection coverage [10]. CRC checking, as well as bit-stuff encoding, is only performed on data and remote frames. However, this do not include the fixed form sequence that ends both these frames: the 1-bit *CRC delimiter*; the 1-bit *acknowledge slot* plus a 1-bit *acknowledge delimiter*; the seven-bit *end-of-frame* delimiter. The following discussion assumes familiarity with CAN operation.

4 ACCESSIBILITY CONSTRAINTS

The study of CAN inaccessibility is presented next. The scenarios described are the inaccessibility periods foreseen in the standard specification. The figures presented illustrate the intervals in CAN operation when the network does not provide service, although not being failed. We start with very simple situations that then evolve to less restrictive - and thus more realistic - operating conditions/fault assumptions. For most of the cases, we explicitly derive best and worst case figures, that we will signal with superscripts ^{bc} and ^{wc}, respectively.

BIT ERRORS

Let us start our analysis of CAN inaccessibility by considering that bus operation is disturbed only once, through the corruption of a single bit within a bit-stream, for instance, due to electro-magnetic interference. Globally, both transmitting and receiving nodes cooperate in the detection of deviations from normal bus operation.

Transmitter-based error detection aims to supply each sending node with the mechanisms required to detect single-bit modifications, in its own transmitted bit-stream. Such error detection technique lays down on bus state monitoring, performed on a bit-by-bit basis, while transmitting. A recessive level issued on the bus, by a given transmitter, can only be overwritten by a dominant level, without that be considered an error, in the following two circumstances:

- inside the *arbitration field*, where such an event will be interpreted, by the recessive bit sender, as a loss in an arbitration process;
- during the *acknowledge slot*, meaning that the previously transmitted data-stream has been heard without errors, at least by one node in the system.

In any other case, should the monitored level differ from the one transmitted, it will be considered an error. Upon its detection, the error will be signalled on the bus, by starting the transmission of an error frame at the next bit slot. Bit error detection, as performed by a transmitting node, can take place as soon as the first bit of a data frame is being transmitted. Therefore, the best-case error detection latency equals bit-time, which added to the best time required to signal the error gives the corresponding network inaccessibility duration:

$$t^{bc}_{ina \quad berr} = t_{bit} + t^{bc}_{error} + t_{IFS} \qquad (1)$$

On the other hand, corruption of the transmitted bit-stream cannot occur latter than transmission of *end-of-frame* delimiter last bit. The worst-case inaccessibility time, is obtained when considering the transmission of a data frame with maximum size. Its value is expressed through equation:

$$t^{wc}_{ina \quad berr} = t^{wc}_{data} + t^{wc}_{error} + t_{IFS} \qquad (2)$$

This transmitter-oriented error detection scheme contributes for the reduction of the average network inaccessibility time, since is designed to detect errors immediately upon their occurrence. It provides an effective answer for the treatment of errors that manifest their effects in sets of nodes that also include the sender. However, it is helpless in the detection of errors affecting only sets of receiving nodes. Thus, the CAN access method includes several receiver-oriented error detection mechanisms, to be discussed next.

BIT STUFF ERRORS

As mentioned in section 3, where we have provided an overview of CAN access method protocol, transmission of data and remote frames bit-streams is performed, until the end of their 15-bit *CRC sequence*, by resorting to a bit-stuffing coding scheme. Thus, frame reception entities must monitor such bit-stream, providing bit-stuffing decoding and error detection functions. Whenever a receiving node monitors l_{stuff} consecutive bits of identical level, it automatically deletes the next received (stuff) bit. Under error free operation, the deleted bit presents a polarity opposite to the preceding ones; should this condition be violated, an error will be signalled on the bus, by starting the transmission of an error frame. The earliest moment, within a given bit-stream reception, where a stuff error can be detected is while sampling the bit following the first l_{stuff} bits of each frame. Therefore, the best-case duration of an inaccessibility period, due to a stuff error, is given by equation:

$$t^{bc}_{ina \quad stuff} = (l_{stuff} + 1).\, t_{bit} + t^{bc}_{error} + t_{IFS} \qquad (3)$$

On the other hand, a stuff error cannot occur after reception of the 15-bit *CRC sequence*. The worst-case inaccessibility duration, for this particular scenario, must consider the transmission of a maximum size data frame. Having both these aspects in mind:

$$t^{wc}_{ina \quad stuff} = t^{wc}_{data} - t_{EFS} + t^{wc}_{error} + t_{IFS} \qquad (4)$$

where t_{EFS}, accounts the duration of the fixed form sequence - not subject to bit-stuffing coding - that ends every data or remote frame.

Stuff errors can be due to some kind of bit corruption, occurring in a way that a sequence containing more than the allowed l_{stuff} consecutive bits of identical polarity, is generated. This type of errors are always detected by bit-stuffing monitoring. However, the same kind of interference may falsify one or more bits in the sequence that has originally led to stuff bit insertion. In this case, bit-stuffing monitoring only reacts to the presence of the fault if the non-removal of the stuffed bit leads to a later stuff error. Otherwise, the error will be detected only when the end of frame sequence is received, either by CRC checking or through detection of a frame format violation.

CRC ERRORS

The CAN protocol uses a 15-bit *CRC sequence* plus a 1-bit *CRC delimiter*, transmitted with each data and remote frames, to check correctness of information transfer. After checking frame correctness, a receiving CAN node should take into account that:

- if the frame has been heard without errors, this condition should be signalled to the transmitter, by changing the bus level from recessive to dominant, during the *acknowledge slot*[6].
- otherwise, the receiving node should take no action during the *acknowledge slot*. The detected CRC error is signalled on the bus through the transmission of an error frame, but this action only starts at the bit following the *acknowledge delimiter*.

The corresponding inaccessibility time is generically given by expression:

$$t_{ina\;\;crc} = t_{data} - t_{EOF} + t_{error} + t_{IFS} \qquad (5)$$

where t_{EOF} represents the duration of the *end-of-frame* delimiter. The best and worst-case bounds of equation (5) are obtained considering, respectively, the transaction of the shortest and of the longest data frames, as well as error frame variability.

FORM ERRORS

All the frames used by the CAN protocol obey to a few pre-defined formats. For data or remote frames, the origin of the frame ending sequence is established taking into account the length specified in the *control field* [10]. The fixed form sequence that ends both these frames includes specific elements, transmitted on the bus with a recessive level and whose value should not be changed by any other node in the system. These elements are:

- the *CRC delimiter*,
- the *acknowledge delimiter*,
- the *end-of-frame* field, transmitted as a sequence of seven consecutive recessive bits.

Likewise, error and overload frames ending delimiters also obey to a fixed form rule: a sequence of eight consecutive recessive bits should follow the last dominant bit of error and overload flags. Violation of those frames fixed form usually only occur as a consequence of multiple errors, happening within a given time period. So, their study will be postponed to less restrictive scenarios, where such behaviour is allowed. Under a single-fault assumption, one needs to consider only the violation of data and remote frames fixed form ending sequence. In the best-case, such errors can be detected while receiving the 1-bit *CRC delimiter* of a minimum size data frame. The corresponding inaccessibility period is given by:

$$t^{bc}_{ina\;\;form} = t^{bc}_{data} - (t_{EFS} - t_{bit}) + t^{bc}_{error} + t_{IFS} \qquad (6)$$

Conversely, a form error may occur, at the latest, while receiving the last bit of the *end-of-frame* delimiter. Evaluation of the worst-case inaccessibility time, for this particular scenario, must considerer such an error, disturbing the transaction of a maximum size data frame. With that understood:

$$t^{wc}_{ina\;\;form} = t^{wc}_{data} + t^{wc}_{error} + t_{IFS} \qquad (7)$$

ACKNOWLEDGE ERROR

All CAN nodes receiving a data or remote frame without CRC errors should signal this condition to the sender, by transmitting a dominant bit level on the bus, during the *acknowledge slot*. This action should be performed regardless of acceptance filter test result, i.e. signalling is due either when the frame is about to be discarded, on account of a reject result, or when it is to be actually accepted and delivered to network user-level entities. Whenever a transmitter does not monitor a dominant level on the bus during the *acknowledge slot*, it interprets that as an error, and the transmission of an error frame is started at the next bit. So, the duration of the corresponding network inaccessibility period is generically given by equation:

$$t_{ina\;\;ack} = t_{data} - (t_{EFS} - 2.t_{bit}) + t_{error} + t_{IFS} \qquad (8)$$

[6] Ahead, we will discuss what happens when all the nodes in the system detect a CRC error, thus binding the *acknowledge slot* value to its

As in a previous scenario, the difference between best and worst-case bounds depends only on the variability of data and error frame duration. In the presence of multiple bit disturbances, it may happen the sender monitors a falsified dominant level, during the *acknowledge slot*, instead the correct recessive level, binded on the bus as a consequence of former CRC errors, detected by all the nodes in the system. The transmitting node lacks to detect such error, but bus operation disturbance will be signalled anyway, as a CRC error, by receiving nodes not acknowledging frame reception.

OVERLOAD ERRORS

For correct operation, the CAN protocol requires data and remote frames to be separated from each other and from any other frame by a minimum amount of three recessive bits, known in CAN terminology as *intermission field*. Through the transmission of an overload frame, a receiving node may enforce data and remote frames to be lengthen, as consequence of any of the following abnormal operating conditions:

- a dominant bit is detected within the *intermission field.* Transmission of the overload frame starts one bit after detection of such CAN protocol violation.
- the receiver circuitry is not ready to receive the next frame, thus requires the inter-frame delay to be extended. Transmission of the overload frame may start at the first bit of the expected *intermission field*.

At most, two overload frames may be consecutively transmitted, in the sequence of each data or remote frame reception. Taking into account the facts laid above, we may derive the following inaccessibility bounds:

$$t^{bc}_{ina\quad oload} = t^{bc}_{oload} \qquad (9)$$

which assumes that transmission of a single overload frame is started at the first bit of the expected *intermission field*, and:

$$t^{wc}_{ina\quad oload} = 2.(t^{wc}_{oload} + t_{IFS}) \qquad (10)$$

where we considerer that the disturbance causing overload frame transmission is only detected in the third bit of the *intermission field*.

OVERLOAD FRAME FORM ERRORS

As mentioned above, the CAN protocol signals the absence of a minimum bus idle period, preceding the transmission of data and remote frames, through the issuing of an overload frame. The CAN receiver detecting the overload condition starts, at the next bit slot, the transmission of an *overload flag*, consisting of six dominant bits; all other nodes will react to this action and, as a consequence, also initiate the transmission of an *overload flag*. After transmitting its own *overload flag* each CAN node monitors the bus-line until a transition from a dominant to a recessive bit level is detected. Under normal circumstances[7], such event means that the first of an eight bit *overload delimiter* has just been issued. All CAN nodes will then transmit the remaining seven recessive bits of the fixed form *overload delimiter*. Should a CAN node detect any deviation from the aforementioned overload frame fixed form, it will be considered an error and accordingly with the CAN protocol, the transmission of an error frame is started at the next bit slot. The best-case inaccessibility bound for this particular scenario is obtained under the assumption that overload frame fixed form violation occurs at the first transmitted bit. Therefore:

$$t^{bc}_{ina\quad oform} = t_{bit} + t^{bc}_{error} \qquad (11)$$

On the other hand, for the evaluation of this scenario worst-case inaccessibility bound, we considerer the transmission of two consecutive overload frames, with the fixed form violation occurring only at the last bit of the *overload delimiter* transmitted in second place. The corresponding inaccessibility time is then given by:

$$t^{wc}_{ina\quad oform} = t^{wc}_{ina\quad oload} + t^{wc}_{error} \qquad (12)$$

INCONSISTENT OVERLOAD ERROR

In this scenario we considerer that only a subset[8] of all the nodes in a given CAN network detect, either correctly or erroneously, a dominant level in the third bit of the *intermission field*. The set of nodes detecting such event initiate then the transmission of an overload frame, but this action will be perceived in a different manner by the set of nodes that did not monitored the *intermission field* violation, because the first of the six dominant bits that made up the *overload flag* will be interpreted as a *start-of-frame* delimiter, by such nodes. Under the assumption that the entire *overload flag* is correctly received by all network nodes, the sixth dominant bit will violate bit stuffing rule and cause an error condition signalling. In these circumstances, the inaccessibility time for this scenario equals the one given by equation (3):

$$t_{ina\ \ iload}^{bc} = (l_{stuff} + 1).t_{bit} + t_{error}^{bc} + t_{IFS} \qquad (13)$$

In a less favourable fault scenario, some bits within the overload frame may also get corrupted. Under such circumstances, there are no guarantees that bit-stuffing monitoring is able to detect the disturbance in bus operation, since a sequence containing more than the five consecutive bits of identical polarity, may never be received by the relevant CAN nodes. Nevertheless, as pointed out before in this paper, the error will still be detected either by CRC checking or through detection of a frame format violation. For the evaluation of the worst-case inaccessibility time for this scenario, we take the largest of those error detection bounds and additionally assume that the inconsistency in intermission field monitoring only occurs after the successful transmission of one overload frame. Therefore, we obtain:

$$t_{ina\ \ iload}^{wc} = t_{oload}^{wc} + t_{data}^{wc} + t_{error}^{wc} + 2.t_{IFS} \qquad (14)$$

MULTIPLE CONSECUTIVE ERRORS

As previously mentioned, CAN nodes signal errors through the transmission of an error frame: as soon as a node detects an error condition it starts transmitting an *error flag,* consisting of six consecutive dominant bits; all other nodes will react to this action and, in consequence, also initiate the transmission of an *error flag*. After transmitting its own *error flag* each CAN node monitors the bus-line until a transition from a dominant to a recessive bit level is detected. In the absence of further errors, this means that the first of an eight bit *error delimiter* has just been issued. All CAN nodes will then transmit the remaining seven recessive bits of the fixed form *error delimiter*.

Errors in a communication system may have many origins: mechanical defects in a cable or connector, electro-magnetic interference, loss of synchrony in a receiver circuitry, transmitter fault, etc. For most of the scenarios we have analysed so far, disturbances in CAN bus operation have been restricted to single errors. Let us now to withdraw this restriction, by assuming the following attribute for network operation:

An - *In a known and bounded interval, T_{rd}, at most n transmissions may be affected by errors.*

This assumption is realistic, based on the observation that errors although rare often occur in bursts. Moreover, for the limited duration of T_{rd}, it is reasonable to make a single fault assumption and therefore the error burst derives from a common cause. The **bounded error degree**[9] assumption introduced by **An** is of crucial importance to define an upper-bound for inaccessibility duration, when in the presence of multiple errors: it states that transmission of no more than n error frames is required to recover the network. Furthermore, let us assume that all the n errors are consecutive in the network[10]. In this case, error signalling will also be subject to disturbances: any node detecting a deviation from the error frame fixed form, starts the transmission of a new error frame. Deriving the best-case inaccessibility bound, under these conditions, assumes that errors, in a forerunner data frame and in the subsequent *error flag*, are both detected at the first transmitted bit; a second error frame is correctly received by all CAN nodes. In consequence:

$$t_{ina\ \ mcerr}^{bc} = 2.t_{bit} + t_{error}^{bc} + t_{IFS} \qquad (15)$$

[8] This subset may be restricted to a single node.

[9] We call *error degree* to the number of consecutive errors produced by a component [8].

[10] Meaning they have origin in the medium or are due to some receiver failure; consecutive transmitter failures may display a different semantic

Conversely, for the worst-case scenario, we considerer that the first error occurs at the end of a maximum length data frame transmission and that the following $(n-1)$ errors are only detected at the end of a maximum duration error signalling period. The recovery actions end with the successful transmission of the n^{th} error frame. The time elapsed in these operations is given by:

$$t_{ina\ \ mcerr}^{wc} = t_{data}^{wc} + n.t_{error}^{wc} + t_{IFS} \qquad (16)$$

MULTIPLE SUCCESSIVE ERRORS

Let us now assume that errors occur in a way that only data or remote frame transmissions are affected; the corresponding error signalling, always succeeds. This scenario, referred in a previous note, is realistic enough to be considered: a failure in a transmitter, may lead to this behaviour. The corresponding worst-case inaccessibility bound directly results from application of **An**, taking into account the worst-case duration of data and error frames:

$$t_{ina\ \ mserr}^{wc} = n.(t_{data}^{wc} + t_{error}^{wc} + t_{IFS}) \qquad (17)$$

ERROR CONFINEMENT MECHANISMS

The error confinement mechanisms provided by the CAN protocol are based in two different error counters, that at each node, record transmit and receive errors, as they occur. These error counters have a non-proportional method of counting: usually, an error causes a counter increase larger than the decrease resulting from a successful data or remote frame transfer. The error count level is thus a measure of the relative frequency of bus operation disturbances and is used to restrict the influence of a given CAN node in bus activity, accordingly with the following classification:

Error-Active - the normal operating state. Such a node is able to transmit and receive frames and in both operations fully participates in error detection/signalling activities.

Error-Passive - the node is still able to transmit and receive frames, but displays a restricted influence in bus activity: after transmitting a data or remote frame the node requires an extra eight bit bus idle period following the *intermission field*, before it can start a new transmission[11]; error signalling is performed through transmission of a *passive error flag*, made up of six recessive bits, instead the normal *active error flag*. A node enters the error-passive state when one of its error counters is higher than 127; it will return to the error-active state whenever both error counters become equal or lower than 127.

Bus-Off - the node may not perform transmit or receive operations. This state is entered when the *Transmit-Error-Count* is higher than 255. A node only leaves this state, to become error-active, after a reset has been performed and a pre-defined *wait-time* has elapsed [10].

The error confinement mechanisms included in the CAN protocol aim to localise errors and minimise their influence on normal bus operation. As a matter of fact, the rules used in error counting have been defined in order that nodes closer to the error-locus will experience, with a very high probability, the highest error count increase. Details on *Transmit* and *Receive-Error-Count* update procedures can be found in [10]. Nodes with error count levels above the 127 error points will be put in the error-passive "quarantine" state, where their influence on bus operation is minimised: the suspend transmission delay grants to error-active nodes precedence in data and remote frames transmission; error signalling is not sensed by other nodes, unless the error-passive node is the frame sender. If the "quarantining" node is the failed one, it will continue to experience, in average, an error count level increase and then either continues in the error-passive state or enters the bus-off state. As a consequence of the failed node "removal" from the error-active state, all the correct nodes will observe, in average, a decrease in its error count levels, as network operation goes on. A lasting error-free bus activity will eventually bring the "quarantining" nodes, that remain correct, back into the normal error-active state. On the other hand, the behaviour of CAN nodes is not so severely affected by temporary faults: these short-term disturbances usually do not produce an error count increase able to remove the nodes from the error-active state.

In order to complete our analysis of CAN inaccessibility, we proceed with the study of two examples, where the occurrence of permanent failures is assumed.

[11] An action known in CAN terminology as *Suspend Transmission*. If during this period another node starts transmitting, the suspended CAN

Failed Transmitter

Let us analyse in first place, a scenario where the transmitter of a given error-active CAN node fails in a way that errors only affect the data and remote frames it sends. Such selective error pattern may result from a malfunction in the node CRC generator. Accordingly with the aforementioned error confinement policies, the errors produced by the failed transmitter may only systematically manifest their effects in bus activity until this node enters the error-passive state. The **An** assumption holds and therefore the worst-case period the network can be down due to the failed transmitter, is obtained directly from equation (17), after evaluating the corresponding *error degree* bound:

$$t^{wc}_{ina \quad txfail} = \left\lceil \frac{err_{a\_thd}}{\Delta_{txerr}} \right\rceil . (t^{wc}_{data} + t^{wc}_{error} + t_{IFS}) \qquad (18)$$

where $\lceil \, \rceil$ represents the *ceiling* function[12]; err_{a_thd} is the error-active count threshold and Δ_{txerr} accounts the *Transmit-Error-Count* increase produced by each transmit error.

Failed Receiver

When an error-active CAN receiver is affected by a similar failure, i.e. a malfunction in the CRC checker, errors are also successive in the network because, once again, only data and remote frames transfer is affected; error signalling succeeds to be performed without errors. The corresponding *error degree* bound is now given by:

$$n_{rxfail} = \left\lceil \frac{err_{a\_thd}}{\Delta_{rxer1} + \Delta_{rxer2}} \right\rceil \qquad (19)$$

In this expression, the terms under the fraction represent two different contributions for *Receive-Error-Count* increase. They derive from the application of more than one error counting rule to the same data transfer, as defined in [10]. The time required to recover the network is obtained from equation (17), using expression (19) as *error degree* bound:

$$t^{wc}_{ina \quad rxfail} = n_{rxfail} . (t^{wc}_{data} + t^{wc}_{error} + t_{IFS}) \qquad (20)$$

Analytic Results

For consolidation of our study regarding accessibility constraints we now evaluate the inaccessibility time bounds, for a given CAN network, for example, a small CAN field-bus, in an industrial environment, for real-time sensing and actuating. The data rate is 1 Mbps and we assume a moderate value ($n = 3$) for the *error degree* bound due to medium errors. The results of the evaluation, for each one of the studied scenarios, are summarised in Table 2. The figures obtained, even those due to some component failure, are acceptably short. Compare, for instance, with a 5 Mbps ISO 8802/4 Token-Bus with a length of $500m$ and a total of 32 stations, where the worst-case inaccessibility figures are rather high[13]: $136ms$ for a multiple join scenario [11]. This study is interesting on the grounds that it provides a better understanding of CAN performance in the presence of failures, providing guidance on how to justifiably achieve better performability and thus, better respect of bounded delay requirements.

5 Conclusions

To achieve reliable real-time operation on a given network, a bounded delay requirement must be met. Usually this analysis is performed by computing worst-case access/transmission delays, for normal network operation. However, achieving the bounded delay requirement means, amongst other factors, ensuring continuity of service. Networks, CAN included, are subject to failures, namely partitions: if these are not controlled, the above mentioned requirement is not met.

Data Rate - 1 Mbps

[12] The *ceiling* function $\lceil x \rceil$ is defined as the smallest integer not smaller than x.

Scenario	$t_{ina}(\mu s)$	
	min.	max.
Bit Errors	18.0	150.0
Stuff Errors	23.0	140.0
CRC Errors	54.0	143.0
Form Errors	52.0	150.0
Acknowledge Errors	53.0	142.0
Overload Errors	14.0	46.0
Overload Form Errors	15.0	66.0
Inconsistent Overload Errors	23.0	173.0
Medium Consecutive Errors($n = 3$)	19.0	190.0
Medium Successive Errors ($n = 3$)	-	450.0
Transmitter Failure	-	2400.0
Receiver Failure	-	2250.0

Table 2 - CAN Inaccessibility Times

A network displays a number of causes for partition, not all of them of physical nature. Recovering from these situations takes time, so in the meanwhile the network is partitioned. Most non-critical applications can live with temporary glitches in network operation, provided these temporary partitions are time-bounded. We have dubbed this periods of inaccessibility, to differentiate from classical partitions. A worst-case figure for these non-negligible periods should be derived and added to the worst-case transmission delay expected in the absence of faults. This is a serious problem often disregarded by designers when expecting hard-real time behaviour from a system. A set of methodologies and algorithms to reliable enforce real-time operation on non-replicated local area networks was presented in [5]. The application of these techniques to CAN is currently under study and will be reported in a future work.

This paper does an exhaustive study of the inaccessibility characteristics of the Controller Area Network. To the authors' best knowledge, no previous study addressed the problem of temporary CAN partitioning in this comprehensive way. The figures derived are thus corrective terms for computing transmission delays in various situations.

References

[1] L. Lamport, R. Shostak and M. Pease. The Byzantine Generals Problem. *ACM Transactions on Prog. Lang. and Systems*, 4(3), July 1982.

[2] Flaviu Cristian. Synchronous atomic broadcast for redundant broadcast channels. Technical report, IBM Almaden Research Center, 1989.

[3] Hermann Kopetz, Andreas Damm, Christian Koza, Marco Mulazzani, Wolfgang Schwabl, Christoph Senft, and Ralph Zainlinger. Distributed Fault-Tolerant Real-Time Systems: The MARS Approach. *IEEE Micro*, February 1989.

[4] D.Powell, editor. *Delta-4 - A Generic Architecture for Dependable Distributed Computing*. ESPRIT Research Reports. Springer Verlag, November 1991.

[5] P. Veríssimo, J. Rufino, and L.Rodrigues. Enforcing real-time behaviour of LAN-based protocols. In *Proceedings of the 10th IFAC Workshop on Distributed Computer Control Systems*, Semmering, Austria, September 1991. IFAC.

[6] K. Tindell and A. Burns. Guaranteeing message latencies on Controller Area Network. In *Proceedings of the First International CAN Conference*, Mainz, Germany, September 1994. CiA.

[7] K. Tindell, A. Burns, and A. Wellings. Calculating Controller Area Network (CAN) message response times. In *Proceedings of the IFAC Workshop on Distributed Computer Control Systems*, Toledo, Spain, September 1994. IFAC.

[8] P. Veríssimo and José A. Marques. Reliable broadcast for fault-tolerance on local computer networks. In *Proceedings of the Ninth Symposium on Reliable Distributed Systems*, Huntsville, Alabama-USA, October 1990. IEEE.

[9] *ISO DIS 11898 - Road vehicles - Interchange of digital information - Controller Area Network (CAN) for high-speed communication* 1992.

[10] *ISO DIS 11519 Part1*, 1994.

[11] J. Rufino and P.Veríssimo. A study on the inaccessibility characteristics of ISO 8802/4 Token-Bus LANs. In *Proceedings of the IEEE INFOCOM'92 Conference on Computer Communications*, Florence, Italy, May 1992. IEEE.