A Transmission Window Technique for CAN Networks

Michael Short[1*], Imran Sheikh[2] and Syed Aley Imran Rizvi[3]
[1]*School of Science & Engineering, Teesside University, Borough Road, Middlesbrough, TS1 3BA, UK*
[2]*GE Grid Solutions Ltd., Stafford, Staffordshire, ST17 4LX, UK*
[3]*Embedded Systems Laboratory, University of Leicester, Leicester, LE1 7RH, U.K*
*Corresponding author email: *m.short@tees.ac.uk*

Abstract—The Controller Area Network (CAN) has become a de-facto communication protocol in automation systems over the last three decades. Some CAN networks now employ TDMA-based communication in order to help meet real-time constraints. Whilst this form of media access control brings several timeliness benefits, studies have also illustrated negative effects on transmission reliability; duplicated message instances can help to increase this reliability. In this paper a transmission window technique for CAN is proposed. A bounded amount of re-transmission is allowed for each message within this window, which can in many cases provides increased reliability in the presence of errors or bursts of errors. A probabilistic analysis of transmission windows is presented and used to develop a simple algorithm for calculating the optimal window size to achieve a specified statistical guarantee of message delivery. Stochastic simulations along with computational and empirical results are presented which validate the analysis, and indicate that in many circumstances the technique can potentially reduce the amount of bandwidth needed for specified reliability levels when compared to the use of message duplicates. Suggestions are also made to help increase the reliability of message duplications in error burst environments.

Index Terms—Controller Area Network (CAN), Channel Errors, Fault-Tolerance, Static Scheduling.

1. INTRODUCTION

Over the last three decades, the Controller Area Network (CAN) [1] has become a de-facto communication protocol employed in automotive and automation applications [2][3][4][5]. However during this time, significant increases in the size, scale and complexity of both vehicle electronic and automation systems have also taken place [5]; mature protocols such as CAN are now effectively being pushed to their limits [2][4]. Despite the emergence of communication schemes such as FlexRay© [6] and an increased use of Switched Ethernet technologies, CAN is most likely to co-exist with other protocols for many years to come due to its popularity, relative simplicity and ease of implementation [2][3][4][5]. The principal focus of the current paper is upon systems implemented using the CAN protocol as a primary or secondary communication network protocol.

CAN was originally intended to allow event-triggered communication between unsynchronized nodes in automotive applications [1][2][3][7]. For various reasons, some CAN networks now employ Time Division Multiple Access (TDMA) communication schemes implemented primarily as hardware or software based protocol extensions [2][8][9]. This is seen as an attempt to increase communication predictability, to meet stringent real-time constraints, to simplify channel redundancy mechanisms and to maximize throughput at high network utilizations. In order to provide a balanced discussion, the authors also note that the adoption of TDMA schemes is not the only method proposed to achieve these benefits; for reliability and data throughput enhancement schemes suitable for

CAN in its native event-triggered form, see for example [15][3].

Although these proposed benefits of TDMA-based systems have been well discussed and documented in the literature (e.g. [2][7][8][9][10]), this form of media access control is also known to have several drawbacks, most notably in terms of message *transmission* reliability [12][13][14]. One of the principal causes is a lack of robustness to errors due to the use of single-shot message transmissions in each TDMA slot [12][13]. Although this effectively bounds worst-case message transmission times to the slot duration, single or multiple bit-errors may directly lead to message omissions [2][14]. This can be contrasted with the behavior of a CAN network operating in its native form, in which re-transmissions will be attempted until either the transmission is successful or is either cancelled by an application program or the controller enters a passive or bus-off state [1]. Comparative reductions in delivery reliability of TDMA schemes in the presence of network errors - amounting to many orders of magnitude - have been reported in the literature [12]. Although the native CAN approach increases the chances of delivery of messages, this is at the expense of uncertainty in terms of worst-case message response times [2][14].

For CAN in its native format probabilistic timing analysis has been found to be an appropriate way to reconcile these issues [12][14][32]. For the TDMA case, an obvious way to increase reliability is through message slot duplication [2][9][12]; however as will be argued in this paper, this may not always be the best approach in terms of the available bandwidth. In this paper, the notion of the λ-firm real-time constraint is proposed in order to capture the multi-criteria real-time/reliability guarantees that are often required for industrial networks. The paper then proposes and describes a simple 'transmission window' technique for the implementation of λ-firm constraints in TDMA-based transmission schemes. In the proposed scheme, the effective transmission slots for a given message are extended, allowing a bounded amount of re-transmission to occur following errors. The slot sizes (as opposed to the underlying message sizes) are then used to create the frame schedule, requiring minimal modifications to existing frame packing and schedule building algorithms. The paper presents a probabilistic analysis of transmission windows, and this is then used to derive a relatively simple algorithm to calculate the window size such that a given λ-firm constraint can be met for a given message. As a byproduct of this work, an analysis for message duplicates is also formed and recommendations for the effective use of duplicates in burst error environments are made. The paper then presents a series of stochastic simulations plus computational and empirical studies, the results of which help to validate the analysis and give an indication of under which circumstances transmission windows are likely to lead to higher reliability levels or bandwidth reductions.

The remainder of the article is structured as follows. Section 2 gives a review of related work in the area of reliable message scheduling on CAN networks. Section 3 introduces the notion of the λ-firm real-time constraint and describes the simple modification to existing TDMA-based transmission schemes that has been proposed. Section 4 is concerned with the calculation of the smallest window sizes to achieve the required transmission reliability for channels experiencing random errors and bursts of errors, and also discusses the use of message duplicates. Stochastic simulations are also presented to help validate the analysis. Section 5 describes a computational study carried out to assess some of the statistical properties of the proposed scheme, and its potential benefits or drawbacks when compared with the use of message duplicates. Section 6 describes experimental results obtained from a practical case study employing a modified FPGA-based CAN controller, which were carried out to further validate the analysis of the proposed scheme under fault-injection conditions. The paper is concluded and some areas of future work are briefly outlined in Section 7.

2. BACKGROUND AND PREVIOUS WORK

2.1 Time-Triggered CAN Communication

A number of hardware and software-based protocol extensions and modifications have been proposed to enable time-triggered communications on CAN; comprehensive reviews are provided by Short & Pont [2] and Rodriguez-Navas et al. [16]. The described techniques tend to rely on the use of a global clock which, in turn, supports a Time Division Multiple Access (TDMA) message schedule. Key to achieving clock synchronization is the reliable broadcast of time reference messages from a 'time master' node. These reference messages are then generally used with a hardware- or software-based distributed clock synchronization algorithm. From a hardware perspective, the Time-Triggered CAN (TT-CAN) protocol uses a global clock synchronization method to provide time-triggered operation of CAN at the hardware level [9]. The protocol provides a maximum accuracy of +/- 1 μs and supports a static TDMA schedule, which can also provide 'empty' slots that allow normal message arbitration for dynamic messages. A full implementation of TT-CAN normally requires dedicated hardware which, at the present time, has not been widely adopted. Several software-based synchronization algorithms have been described as valid alternatives. When using such techniques, clock synchronization at an accuracy level of 100 μs is typical; however techniques giving accuracies up to 1 μs are known. An example exhibiting high accuracy would be the family of 'shared-clock' algorithms which – at the expense of a small local CPU overhead - provide time-triggered communications without the need for additional hardware (or even complicated software clock synchronization algorithms) [2][8].

The general goal of all these protocols – whether hardware or software based - is the creation of a collision free (and hence arbitration free) bus access schedule, such as that depicted in Figure 1 [2]. In the Figure, six frames (with worst-case transmission times C_A to C_F) are sent every TDMA cycle. Due to the existence of an unavoidable finite worst-case clock error between any two clocks in a distributed system, a small inter-frame spacing P is employed to widen the effective slot size above the frame transmission time C. A correct choice of P is required to prevent overlap errors due to clock synchronization inaccuracy; in this paper we assume an adequate choice for this parameter has been made. In the general case, designing a message schedule to meet a given set of period requirements is strongly NP-hard, but in practice many fast algorithms (both optimal and heuristic) can be used to pack signals into frames and to generate feasible TDMA schedules (see e.g. [17][18]). It is not uncommon to achieve bus utilizations in excess of 90% using such techniques [2].

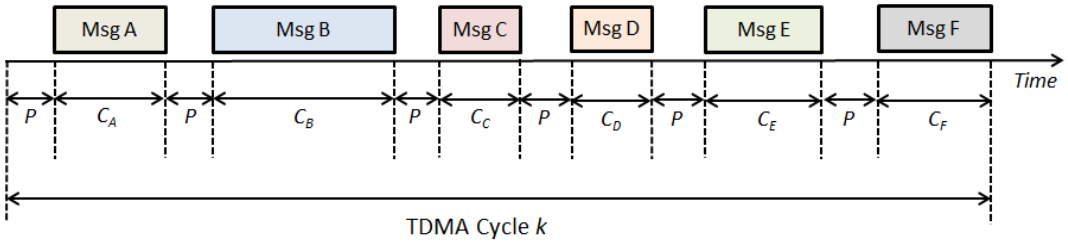

Figure 1. Typical TDMA Structure with inter-frame spacing P.

2.2 Fault-Tolerant CAN Communications

When messages are required to be sent over multiple redundant CAN channels to improve reliability, replica determinism and the notion of global time become of great importance [2][16][9]. This is principally due to potential non-identical message ordering over replicated channels (caused by the effects of clock drift and channel errors), making fault tolerance effectively unmanageable without a global clock [2][16][9]. Replica determinism can be

enforced - in part - by the use of single-shot transmissions or by upper bounding the latest time that a message may commence transmission. If replica determinism can be enforced, then multiple redundant and fault-tolerant CAN networks may be operated in parallel to increase the reliability of the physical layer [2]. When messages are subject to interference such as electromagnetic disturbances, this tends to manifest itself as random bit errors on the network. In response to any detected errors, under the CAN protocol an error frame is generated - which may have a length of up to 31 bits [19] - followed by a re-transmission attempt. In a real-time system this re-transmission can be very problematic due to deadlines being missed in a 'domino-style' effect; see, for example, [14] for further details. Experimental studies would seem to place the Bit Error Rate (BER) for CAN in the region of 10^{-10} in 'benign' environments, increasing to 10^{-6} in 'aggressive' environments [20]. In some extreme cases, BERs as high as 10^{-3} have been reported for vehicles operating in 'hostile' environments, for example when in close proximity to large electromagnetic radiation sources such as radio transmission stations [21]. In more aggressive environments, it has also been suggested that a large proportion (around 90%) of these errors occur in short correlated bursts, with durations typically between 5 to 20 bit times [20][21][31].

With these points in mind, in a real-time application some form of timeout or upper bound is normally required to limit the worst-case transmission time of a given frame. In many hard real-time systems, it is arguably better not to receive an instance of a periodic message at all, than to receive the instance late [2][14][22]. Unfortunately, such timeouts are not provided as a standard CAN feature; at the expense of local CPU overheads, several techniques have been described to enforce this behavior. Previous works such as [2] and [16] have suggested that only single-shot transmissions be attempted in the TDMA round, and it is in fact an enforced requirement in TT-CAN networks [9]. As this has an undesirable effect on transmission reliability (e.g. [12][13]), for critical message streams, duplication of message instances is the principal means to achieve the desired reliability. For a message requiring c bits to be transmitted in an environment with a static BER of β, the error occurrences follow a Bernoulli distribution (and are hence uncorrelated) and the probability of success of an individual message instance $p = (1-\beta)^c$. If r message duplicates are sent then the probability of failure λ reduces with r as follows [2][32]:

$$\lambda = (1-p)^r \tag{1}$$

Similar formulae can be derived under the assumption that the uncorrelated fault arrivals possess other underlying distributions; for example Broster et al. show that for Poisson fault arrivals with intensity β then $p = (1-exp(-\beta c))$ [12]. Relationship (1) may be used to calculate the number of required message duplicates in a TDMA cycle before packing and schedule generation are applied, see e.g. [32]. When correlations between errors (error bursts) are present then equation (1) may not be accurate, even when p represents the probability of success of the first message duplicate, as failures show correlation. As an alternative to the use of message duplicates, a transmission window scheme was proposed in [13]. It was suggested in this preliminary work that the use of transmission windows could potentially reduce the amount of bandwidth required when compared to message duplication, although more detailed investigation was needed. The transmission window scheme will be described in Section 3, and an improved analysis and further investigations form the remaining Sections of this paper. The improved analysis for transmission windows also reveals that additional spacing constraints in a schedule can allow an analog of equation (1) to become applicable for message duplicates in burst error environments. The findings also suggest that although the preliminary findings in [13] hold in many cases there are, in fact, situations in which transmission windows are not as effective as well-spaced message duplicates

Although this windowed scheme is the primary focus of the paper, again to achieve a balanced discussion the authors note that several other schemes and approaches have been presented to achieve similar goals. The

'Timely CAN' scheme described in [14] proposes a simple extension to a Native CAN network, and a technique to upper-bound the latest time that a particular message can be scheduled for re-transmission under an optimal priority assignment is presented. This scheme helps to prevent domino-effect deadline misses; from an estimate of message worst-case response times[1], message transmission times are subtracted to obtain the required 'timeout' parameters for each message. In the worst-case, single bit errors can still lead to omissions; however if 'slack' is placed in the timeout, then a limited amount of re-transmission can be achieved. The amount of slack to employ can be determined by adding an error model to the response time calculations, where error arrivals can be treated as sporadic events with a fixed or stochastic inter-arrival time, see e.g. [19][11]. More recently, Aysan et al. propose a technique for scheduling mixed-criticality messages on a CAN network [23]. Their proposed system allows critical messages to be re-transmitted up to a pre-specified number of times in case of errors, using bandwidth allocated to the non-critical messages. Integer Linear Programming (ILP) techniques are employed to generate schedules and priority assignments that minimize the number of deadline misses under error and error-free scenarios. An alternative approach is adopted by the FTT-CAN protocol, in which elementary cycles (triggered by synchronization messages from a master node) are divided up into a synchronous phase and an asynchronous phase [29]. A TDMA schedule (or alternatively, a priority-based scheduling algorithm) may be implemented for periodic transmissions in the synchronous phase with event-triggering performed in the asynchronous phase. Temporal isolation between the phases can be ensured by the use of guarding timers in nodes. Error recovery for frames lost in the synchronous cycle can be achieved by the use of a server, which has bandwidth reserved in the synchronous phase of the cycle and is scheduled as needed by the master node [33]. Although such a scheme cannot be directly implemented within a purely TDMA framework, a similar approach can nevertheless be employed with TT-CAN where one or more 'free' slots for optional error recovery are embedded in the cyclic schedule [9]. Lost frames can be re-transmitted in these free slots in CAN priority ordering.

It is clear that since electromagnetic disturbances cannot be completely eliminated from industrial communication systems unless specialized and potentially expensive equipment (e.g. optical fiber) is employed, then probabilistic guarantees of real-time behavior should to be sought. A possible method to achieve these guarantees in TDMA systems in the presence of burst errors is described in the next Section.

3. THE CAN TRANSMISSION WINDOW TECHNIQUE

3.1 λ-Firm Real-Time Constraints

The notion of the 'firm' real-time constraint has been in existence for some time, principally in the context of CPU task scheduling. As with a 'hard' real-time constraint, a firm constraint is intended to model real-time behavior in which certain actions should be performed before a pre-specified deadline, and no benefit is gained in delivering the results of the action after the deadline has elapsed. However, unlike hard constraints, whilst deadline misses are deemed undesirable, they are not always deemed to be catastrophic failures in a firm real-time system; the omission of a (bounded) number of actions per unit time is deemed acceptable. Such a formulation has been found to be well-suited to modelling the periodic operations of many sampled data control and signal processing systems, in which occasional missed samples do not have an overly deleterious effect on system performance [14][22][30]. For example, variations of the (*m*, *k*)-firm CPU tasking model – in which the system is deemed schedulable if at least *m* from every *k* consecutive deadlines are met – have proved popular in real-time control applications [22].

In a networked application, due to the presence of noise and other interferences, the successful delivery of a message in any given length of time cannot be guaranteed 100%. As a result, the notion of timeliness in both wired

[1]It should be noted that the original analysis provided in [14] contains an error due to 'push-through' blocking; see [19] for further details.

(and wireless) network applications must be considered very carefully, as it is effectively impossible to provide any crisp, absolute guarantees of timeliness [25][30][31]. The notion of the λ-firm real-time constraint is intended to directly reflect the probabilistic nature of the underlying transmission medium. A message stream τ_i with this type of constraint is described by the following four parameters:

$$\tau_i = (T_i, C_i, D_i, \lambda_i) \tag{2}$$

Where T_i is the message period/inter-arrival, C_i is the message (worst-case) transmission time, and D_i is the message relative deadline. These three parameters are assumed (without loss of generality) to be positive integers in suitable time units, e.g. network bit times. Each message stream is also allocated an additional parameter λ_i, which is a failure rate for the message stream specified as a probability of unsuccessful transmission per delivery attempt. Typically, λ_i may be derived from a higher-level safety analysis, e.g. an assigned Safety Integrity Level (SIL) (e.g. see [24]). A message set with λ-firm constraints that is deemed schedulable can therefore be interpreted as providing a multi-criteria real-time/reliability guarantee of the following nature: if a message instance is successfully delivered, it is delivered on time (i.e. before its relative deadline has elapsed); if a message instance is *not* delivered (omitted), *the probability of omission is less than or equal to* λ_i. Thus, any late messages will be 'dropped' without causing a domino-effect of missed deadlines, with the notion of a schedulable message set providing only a statistical guarantee that messages will be delivered for a given bounded error model. Schedulability analysis will therefore require knowledge of the statistical error properties of the channel in question. For some message streams, soft real-time constraints may be more suitable; practical systems may contain a mix of both soft and firm constraints.

3.2 Transmission Windows in TDMA-based Schemes

In a TDMA-based environment, as discussed one possible method to increase reliability is to employ redundant copies of critical messages. In this paper, the alternative transmission window scheme proposed in [13] is considered. Before introducing the approach, it is worth recalling from the CAN protocol definition two important points related to error detection: (i) the probability of an undetected error is vanishingly small (of the order MER x 4.7 x 10^{-11}, where MER is the message error rate) and (ii) in addition to CRC failures, instantaneous bit errors, bit stuffing errors and form errors may be detected and signaled at any point in a frame transmission by any node (whether transmitter or receiver) [1]. Thus the basic operation of the CAN protocol is intended to support early detection and signaling of many types of errors, and to initiate abandonment of corrupt transmissions at the earliest possible point in time. These points would seem to indicate that a measurable number of CAN messages will be aborted part way through their TMDA slots; for these messages, the remaining time in the slot will be unused (except for the signaling of an error frame). This motivates a search for potentially more efficient approaches to utilizing the available bandwidth.

Although recognizing the probability of undetected errors is higher than was originally specified in the CAN protocol due to interactions between bit stuffing and the CRC [36], and also that inconsistent message omissions can occur, let us proceed under the assumption that there is an (approximately) uniform probability for the detection and signaling of errors along the length of a CAN message. The justification for this is as follows. Due to transmitter bit error monitoring, detected errors affecting any set of nodes which includes the transmitting node (including all global errors) are immediately detected and signaled. Local errors affecting a set of nodes which does not include the transmitting node, but only one or more receivers, typically result in form or bit stuffing violations which are

immediately signaled as errors by the effected receivers. Detailed simulations have indicated that well over 99.9% of detected CAN errors will be signaled by these three primary mechanisms [36]. The remaining proportion of detected local errors will principally manifest as CRC failures, signaled only after the CRC check. Thus, although an assumption of uniform probability for detecting and signaling errors along the length of a CAN message is clearly not 100% accurate, it would be seem to be a 'good enough' approximation for the large majority of cases (note also that commercial grade statistical simulation software for the CAN protocol also makes use of a similar assumption [37]). In Section 6 our experimental findings give some further justification to this assumption.

Suppose there is a message 'X' that requires two duplicate copies to be sent every TDMA cycle. Figure 2 (top) shows such a situation, where for clarity the duplicated copy X_2 is assigned a slot immediately following the original copy X_1. Suppose that a bit-error occurs in both of these slots, as shown in Figure 2 (middle) – in the case of a standard TDMA approach, this will lead to both messages effectively being dropped as only single-shot transmission is allowed. However, now consider the situation depicted in Figure 2 (bottom). Both slots are now effectively merged into a single slot, which becomes a transmission window W_x; (re)transmission of message X is allowed anytime in the slot up until the point $W_x - C_x$. As can be seen in the Figure, this *may* have a positive effect on the reliability of message delivery; even when numerous bit-errors occur, a successful message delivery could still take place in some circumstances. Delivery will occur whenever a (continuous) sequence of error-free bits having a sufficiently long duration is present, and this sequence may start and end at arbitrary points within the window. As the slots have effectively been merged, observe also that only a single inter-slot spacing is required (In the Figure, since two slots have been merged $W_x = P + 2C_x$). However, there are many possible combinations of error arrival patterns (both correlated and uncorrelated), and when using message duplicates the slots do not necessarily have to be located adjacently as shown in Figure 2; each slot may be separated by some minimum length g in addition to the inter-slot spacing. Thus it is not fully clear under what circumstances the window scheme may be beneficial. However, as will be demonstrated in a subsequent Section, for many (but not all) realistic cases covering correlated and uncorrelated errors, window transmissions do seem beneficial.

As the window size is fixed, all the desired properties of a time-triggered communication system are retained. Given a set of slot sizes W_i for each of the messages, the TDMA schedule may be created using existing techniques and frame packing algorithms. However, an important related question arises; namely, for a given message length and burst error characteristic, what is the smallest length of the transmission window W – and hence bandwidth allocated to each message – in order to ensure the message will be delivered with the required statistical guarantees? This question will be addressed in the following Section.

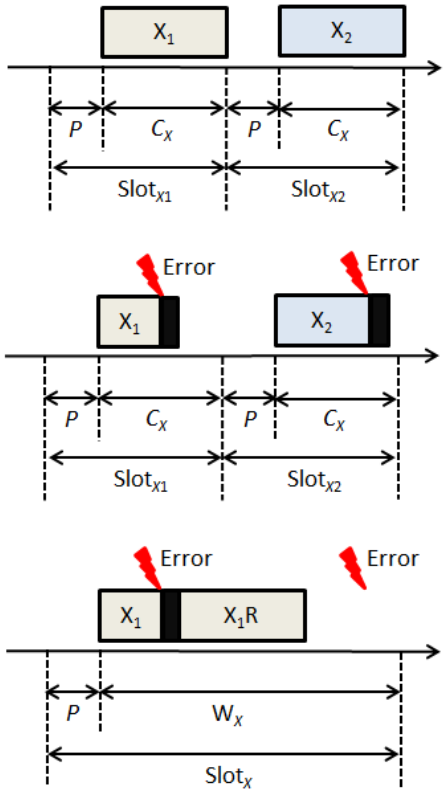

Figure 2. Concept of a transmission window.

4. CALCULATING TDMA WINDOW SIZES

This Section will consider how the probability of a successful transmission can be computed for increasing values of transmission window length for CAN networks experiencing random errors and random bursts of errors. The Section begins with a discussion of the error model that is considered for this purpose.

4.1 Error Modeling

As previously mentioned, research has shown that many errors in CAN communication links are not just single independent events but are likely to occur in isolated transient bursts [20][21][31]. In order to develop a technique to effectively calculate transmission probabilities as a function of window length, an error model that is rich enough to capture this bursty nature yet simple enough to lend itself to tractable analysis is required. The most common way to model bursty bit and packet error behavior is to use a simple two-state Markov model [26][34], such as that shown in Figure 3.

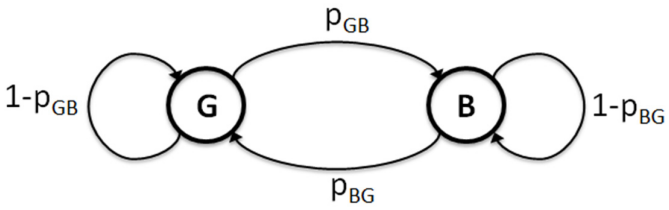

Figure 3. Two-state Markov model of a bursty communication link.

The model has two states *G* and *B*, representing 'Good' and 'Burst' states respectively. Transitions between the two states *G* and *B* have associated with them the probabilities p_{GB} and p_{BG}. The probability of remaining in a given state is then given by $p_{GG} = 1-p_{GB}$ and $p_{BB} = 1-p_{BG}$. Each state may also have associated with it a probability of bit

error occurrence (denoted β_G and β_B respectively). The model parameters p_{GB} and p_{BG} can be interpreted as follows: the reciprocal of p_{GB} defines the expected (mean) gap between error bursts μ_{EG}, and the reciprocal of p_{BG} defines the expected (mean) duration of error bursts μ_{EB}, both variables having a geometric distribution. The expected inter-arrival time of error bursts is then given as (μ_{EG} + μ_{EB}). The optional model parameters β_G and β_B can be interpreted as follows: when the system is in the 'Good' state, the reciprocal of β_G defines the expected (mean) inter-arrival time of errors, and when the system is in the 'Burst' state, the reciprocal of β_B defines of the expected (mean) inter-arrival time of errors, both variables again having a geometric distribution. Typically, it would be the case that β_G << β_B and μ_{EB} << μ_{EG}. Let the state of the link at step k be denoted by $l(k) \in$ {'*G*', '*B*'}), and let the state of the Markov model at step k be encoded as the probability $s(k)$ that the link is in the error state, i.e. $s(k) = P\{l(k) = \text{'}B\text{'}\}$. Applying the normal rules for Markov model state transitions, then $s(k)$ depends only upon the previous state $s(k\text{-}1)$ and the transition probabilities p_{BB} and p_{GB}, and can be recursively computed as follows:

$$s(k) = p_{BB} \cdot s(k-1) + p_{GB} \cdot \left(1 - s(k-1)\right) \tag{3}$$

Assuming that the initial state of the link $s(0)$ is known, the transient and steady-state evolution of the link state can be calculated using (3). The steady-state solution to the Markov chain is obtained by setting $s(k) = s(k\text{-}1) = \pi$ into (3) and solving for π, and we obtain that $\pi = p_{BG}/(1\text{-}p_{BB}\text{-}p_{GB})$. Given any starting state $s(0)$, after a transient period the model converges upon π which represents the fraction of time the link can be expected to be in the 'Bad' state when observed for long periods. π also represents the probability that when the link is observed at some random sample time t we find that $l(t) = \text{'}B\text{'}$. Following the acquisition of explicit knowledge about the link state at step t (such as by making an observation), the state transiently moves back to the steady state π with coefficient $\alpha = (p_{BB}\text{-}p_{GB})$ according to the relationship $s(t+k) = \pi + (s(t)\text{-}\pi)\cdot\alpha^k$.

For simplicity, in this paper it is assumed that β_B and β_G were set to 1 and 0 respectively, under the assumption that the probability of a full CAN frame transmission occurring whilst the link is in the burst state is effectively negligible (for example, if β_B is 0.5 – a realistic assumption for a network such as CAN – the probability of successfully sending a 60-bit frame whilst in a burst state is of the order 10^{-20}). Although other studies have suggested that errors and bursts of errors in CAN networks follow exponential distributions (and can be considered as a Poisson process which is possibly non-homogeneous and/or generalized) [12][14][31], the model above should be at least as descriptive. This is because (i) if multiple errors arrive during the same logical bit-time – as may be predicted with a Poisson process - then only one of these errors will be effectively 'counted' since time is discrete in a CAN network and (ii) the geometric distribution is the discrete equivalent to the continuous exponential distribution. Maximum likelihood estimates of the model parameters p_{GB} and p_{BG} from observed data traces are known and have simple closed-forms [26][34].

4.2 Probability Computations

Consider the transmission of successive bits on a CAN network in a noisy environment where the link errors evolve according to the model described above, with the Good state indicating a successful transmission of a bit and the Bad state indicating an error. We are interested in determining the probability that a sequence of $C > 1$ consecutive bits is successfully transmitted in a window of length $j > 0$. Let us denote this as the probability $P(C, j)$. In our previous work [13], a preliminary method was presented to calculate such probabilities and was based upon iteration of the Markov model using (3) and using the probability that the link is in a Bad state as the probability a bit error will occur. Although the method gave useable results, detailed statistical evaluations revealed high levels

of pessimism in the probability calculations due to the direct use of the probabilities that the link is in a Bad state as the probabilities that bit errors will occur (i.e. the link transition probabilities p_{GB} and p_{BG} were not integrated correctly into the analysis). In addition, the assumption made regarding the initial state of the link at the commencement of the transmission window impacts greatly upon the accuracy of the calculations. In the particular case when the link is initialized with starting state π, the most appropriate choice, π in effect becomes a static bit error rate and the bursty behavior of the errors neglected completely. In the following analysis such problems will be dealt with. In the analysis, it will be useful to define an additional probability $B(C, j)$, which represents the probability that the link is in the Bad state at step j and the transmission of length C bits remains ongoing (i.e. no successful sequence of C consecutive bits has yet been observed at step j). The Theorem below establishes that $P(C, j)$ may be efficiently computed for increasing values of j.

Theorem 1: The probability $P(C, j)$ that at least C consecutive bits have been transmitted without error in a sequence of j bits can be calculated using the following expressions:

$$\forall j, 1 \le j < C: \quad P(C, j) = 0 \tag{4}$$

$$P(C, C) = (1 - \pi) \cdot (p_{GG})^{(C-1)} \tag{5}$$

$$\forall j,\ j > b: \quad P(C, j) = P(C, j-1) + B(C, j-b) \cdot p_{BG} \cdot (p_{GG})^{(C-1)} \tag{6}$$

With $B(C, 1) = \pi$ and $B(C, j)$ updated for $j > 1$ as:

$$B(C, j) = \alpha \cdot B(C, j-1) + p_{GB} \cdot (1 - P(C, j-1)) \tag{7}$$

Proof: When the number of bits transmission attempts is less than C, achieving a consecutive stream of C bits (either with or without error) is impossible and thus $P(C, 1) = P(C, 2) = \ldots = P(C, C\text{-}2) = P(C, C\text{-}1) = 0$ as stated in condition (4). When the number of bits transmitted is exactly C, then the initial link state $s(1)$ must have been observed to be in the 'Good' state when transmitting the first bit (with probability $(1\text{-}\pi)$) and this was followed by a sequence of $(C\text{-}1)$ subsequent observations of 'Good' states, each having probability p_{GG}. This gives rise to expression (5). For all values of $j > C$, $P(C, j)$ is equal to $P(C, j\text{-}1)$ plus the additional probability that a successful sequence has just been observed with the j^{th} bit completing the run of C consecutive successes. As shown in Figure 4 the additional probability that a successful sequence has just been observed with the j^{th} bit ending the sequence is given by the probability that the following three events have been observed: (i) At step $j\text{-}C$, the link has been observed in the Bad state with the transmission still ongoing (i.e. no successful sequence of C consecutive bits had been observed at step j) with probability $B(C, j\text{-}C)$, (ii) at step $j\text{-}C\text{+}1$ the link was observed to transition to the Good state to begin the success sequence with probability p_{BG}, and (iii) a sequence of $(C\text{-}1)$ 'Good' states has been subsequently observed to complete the sequence, each having probability p_{GG}. This gives condition (6).

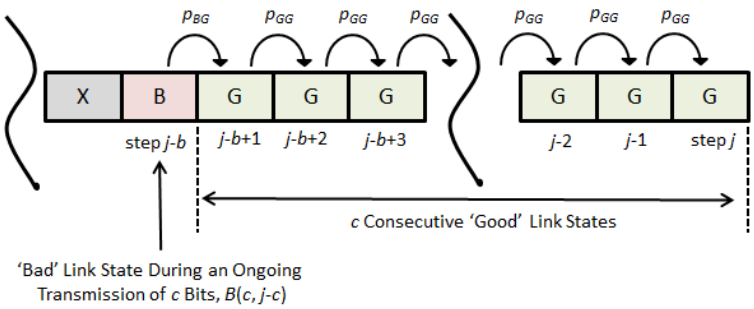

Figure 4. Conditions required for a successful outcome at the j^{th} step, for $j > c$.

It remains to show that the expression for $B(C, j)$ represents the probability that the link is in the Bad state at step j and the transmission remains ongoing. Clearly at step 1, the transmission is ongoing and the probability that the link is observed in the Bad state is given by π, and hence $B(C, 1) = \pi$. For each subsequent step, consider first that if the link was observed in the Bad state and the transmission ongoing at step j-1 with probability $B(C, j\text{-}1)$, then the link will remain in the Bad state with the transmission ongoing with probability $B(C, j\text{-}1)\ p_{BB}$. Secondly, consider that at step j-1 the link was observed in the Good state with the transmission ongoing, i.e. a sequence of at least one - but not more than C-1 - consecutive Good states had been observed with probability $(1.0 - P(C, j\text{-}1) - B(C, j\text{-}1))$. In this latter case the link may transition to the Bad state at step j with probability p_{BB}, in which case the link will also be in the Bad state with the transmission ongoing with probability $(1.0 - P(C, j\text{-}1) - B(C, j\text{-}1)).p_{GB}$. After some simple algebra and using the definition of α, the combined probability of these sequences of events occurring is as stated in the expression for the computation of $B(C, j\text{-}1)$ as given in (7). □

The Theorem above is quite general and may be used for any choice of Markov model parameters p_{BB} and p_{GB}; the special case in which errors do not arrive in bursts but only have a constant bit error rate of β is covered by setting $p_{BB} = p_{GB} = \beta$, after which we may see that $\alpha = 0$ and the computation of (7) simply tracks the probability that no sequence has been observed at step j $(1\text{-}P(C, j))$ multiplied by the bit error rate β, i.e. there is no correlation between errors and a transient link state is not tracked in the computations. The accuracy of the above probability computations has been found to be highly accurate in predicting the results of stochastic simulations; these results will shortly be described.

4.3 Window Sizing Algorithm

The results of the previous sub-section may be utilized to create a recursive algorithm to calculate the required window size to achieve the required transmission probability for a given bursty link. Pseudocode for such an algorithm is as shown below in Figure 5. The algorithm takes as input the message length C, relative deadline D (both expressed in network bit times) along with the required failure probability λ and the Markov model parameters p_{GB} and p_{BG}. The algorithm calculates and returns the smallest value of window size j such that the required failure probability is achieved (note that the algorithm will also terminate when $j > D$, indicating that the required window size exceeds the relative deadline: this should be checked upon termination). This latter termination condition effectively bounds the time complexity to be pseudo-polynomial in the message deadline parameter, giving complexity $O(D)$. Since only the last $(C+1)$ values of the probabilities $B(C, j)$ and the last value of $P(C, j)$ are required to be stored for the computations at step j, and an array of size $C+1$ for the former probability (which is indexed $mod(C+1)$) can be used to limit the required memory storage of the algorithm to $O(C)$.

```
01 Proc Calc_Window(C, D, λ, p_GB, p_BG)
02 {
03 p_GG := (1-p_GB);
04 p_BB := (1-p_BG);
05 π := p_BG /(1-p_BG-p_GB);
06 α := p_BB-p_GB;
07 P(C, C) := (1-π)·(p_GG)^(C-1);
08 FOR j := 1 TO C DO:
09          B(C, j) := π;
10 END FOR
11 WHILE ((1.0 - P(C, j)) >= λ) AND (j <= D)) DO:
12  j := j + 1;
13  B(C, j) := α·B(C, j-1) + p_GB·(1.0−P(C,j-1));
14  P(C, j) := P(C, j-1) + B(c, j-C)·p_BG·(p_GG)^(C-1);
15 END WHILE
16 RETURN(j);
17 }
```

Figure 5: Computing window length (returned as the variable *j*) for a bursty link.

4.4 Statistical Evaluation: Transmission Windows

In order to evaluate the accuracy and predictive abilities of the probability computations detailed in Section 4.2, a series of stochastic simulations was carried out to compare theoretical calculated values for window transmissions to statistical results. This was deemed important since the analysis in our previous work [13] contained some incorrect assumptions which resulted in unacceptable pessimism; hence a statistical evaluation was performed to validate the improved analysis and assumptions made. The simulations were carried out to assess the measured message success rate (i.e. the probability a single instance of a periodic message being delivered) for a simulated static TDMA schedule using transmission windows in the presence of both static bit errors and bursts of bit errors. The schedule consisted of 4 messages with characteristics as shown in Table I. In each case the DLC of the message was set to 8, corresponding to a worst-case message length of 166 bits (including a worst-case error frame). Relative deadlines were taken as equal to periods ($D_i = T_i$) for all messages. The CAN network speed was assumed to be 1 Mbps. The stochastic simulator was written in C++.

Table I. Messages comprising the static schedule

Message	T_i (bits)	C_i (bits)
A	1000	166
B	2000	166
C	4000	166
D	8000	166

Given a calculated probability p for the successful delivery of a message, suppose that some number N of attempted transmissions are made. Since each attempt may either result in success or failure, a count of the number of successes X in these N trials follows a Binomial distribution with parameter p ($X \sim \text{Binomial}(p, N)$). For an experiment with large N, then a Normal approximation to the distribution of X becomes very accurate and upper and lower 95% confidence limits on the *measured* probability of success $\hat{p}$ (Binomial proportion) during the

experiment can be constructed as follows:

$$\hat{p}_L = p - 1.96\sqrt{\frac{p(1-p)}{N}}, \quad \hat{p}_U = p + 1.96\sqrt{\frac{p(1-p)}{N}} \tag{8}$$

Where 1.96 is the 0.975 Quantile of a standard normal distribution. For large N, the confidence interval becomes small, and following a stochastic simulation and experimental determination of $\hat{p}$ if the value lies within the confidence limits given by (8) then there is no significant reason to reject the model as inaccurate. Conversely, should it lie well outside these limits, there is reason to suspect model inaccuracy and further investigation or improvements may be required. In our simulations, we carry out a number of simulations and statistical tests to investigate the validity of the analysis. In the first set of experiments, static bit errors were injected into the simulated network using an error rate of 0.001/bit. All messages were assumed to always take their worst-case lengths. Ten simulations were performed for different lengths of transmission window, with the simulations lasting for one simulated hour each (giving N > 6.7 million sample points to calculate $\hat{p}$ for each simulation). In the second set of experiments, bursts of bit errors were injected into the network using a burst inter-arrival of 20,000 bits and average burst length of 20 bits. All messages were assumed to always take their worst-case lengths. Ten simulations lasting for one simulated hour were again performed for different lengths of transmission window. The results of these experiments are summarized in Table II, which shows the measured probability of message delivery $\hat{p}$ versus the calculated probability of message delivery p in each case.

Table II. Comparison of calculated versus measured statistics for window transmissions

	Static Errors		Burst Errors	
Window Length (bits)	Calculated (p)	Measured ($\hat{p}$)	Calculated (p)	Measured ($\hat{p}$)
166	0.8469759	0.8469619	0.9907838	0.9907590
207	0.8817019	0.8817647	0.9928169	0.9928034
249	0.9172749	0.9172416	0.9948996	0.9948956
290	0.9520009	0.9520137	0.9969328	0.9969208
300	0.9604707	0.9604141	0.9974287	0.9974424
332	0.9875739	0.9875893	0.9990155	0.9990229
373	0.9922996	0.9922613	0.9998522	0.9998603
415	0.9958901	0.9958886	0.9999674	0.9999684
456	0.9981746	0.9981929	0.9999892	0.9999902
498	0.9992644	0.9992772	0.9999968	0.9999972

From the data reported in the Table, a close correspondence between the calculated and measured probability of delivery was observed. The maximum relative absolute error in the number of messages predicted to be delivered per second and the number recorded in the simulation was < 0.003215 %. The obtained values of $\hat{p}$ were all within the 95% confidence limits as calculated by (8), and hence we have no reason to suspect model inaccuracy. For both static and burst errors the growth in probability of successful delivery as the window size is increased can be clearly observed. Although these simulations represent an idealized approximation of a real CAN network and error model, this nevertheless gives a good indication of the accuracy and applicability of the probability computations outlined in Section 4.2. Some results are forthcoming in a later Section which presents a comparison with data obtained for a real CAN network under fault injection.

4.5 Modifications for Message Duplicates

In Section 5 of the paper, a detailed comparison of the transmission window technique against the use of message duplicates (in terms of the bandwidth required to achieve a target transmission reliability) will be performed. As such, some means to determine the effective probability of delivery when using message duplicates in static error and burst error environments is required. For the former, equation (1) may be employed using p as the effective probability a message instance is delivered. For the latter a more careful consideration is required. In our previous work [13], a preliminary method was used to calculate the effective probability of message duplicates is burst error environments. The method was based upon iteration of the Markov model using (3) and taking the probabilities that the link is in a Bad state as the probabilities of bit error occurrence to derive an approximation of failure probability of successive message duplicates. However this method suffered from similar problems to that observed in the window case, and detailed statistical evaluations again revealed high levels of pessimism in the calculations. In this Section, some adaptations of the analysis presented in Section 4.2 are discussed to provide improved calculations and a guideline for the appropriate use of message duplicates in burst error environments is suggested.

First, some recorded statistics are presented to illustrate how the temporal correlation or errors can impact upon message duplicates and render equation (1) invalid. Consider the case in which two 166-bit messages ‘A’ and ‘B’ are transmitted in sequence (i.e. with no gap between transmission slots) in the burst error environment described in the previous Section. Equation (5) may be used to calculate the probability of successful delivery for each of these one-shot transmissions, which in this case is equal to 0.9907838. Using stochastic simulation, the measured probability of success for message A was found to be 0.9907945 and the probability of success for message B was found to be 0.9907940, both in close agreement with the calculated value. Next consider the case in which A and B are in fact duplicated copies of the same message, and we seek the probability that either copy (or both) are successfully delivered. Setting p = 0.9907838 and naively applying equation (1) gives a calculated probability of success as 0.9999151. Using stochastic simulation, however, the measured probability of success for delivering either copy A or copy B or both was found to be 0.9989777; a considerable reduction in the expected success probability. The principal reason for the reduction is that the failures of B are correlated with those of A, and one may hypothesize that the level of correlation depends upon the temporal distance between the two transmission attempts. Some modifications to the analysis presented in Section 4.2 allow this to be confirmed. The first modification is required upon equation (5); for single-shot transmissions, the probability of successfully receiving a message at step j - $P(C, j)$ – should only be updated for values of j corresponding to the scheduled end of a transmission initiated at step (j-C-1). Assuming the first single-shot transmission is initiated at step 1, and subsequent transmissions occur g bits after the end of the last transmission (with $g \geq 0$), then the transmissions can be modeled as periodic with effective period $T = (C + g)$ within a pseudo-window of size j bits. Secondly, it is no longer required that the link is in a bad state one step before the transmission occurs; instead to calculate the increase in probability of successful delivery, we only require that the link is in the good state (and the previous transmission have failed) at step (j-C-1), which occurs with probability (1.0-$P(C, j\text{-}C\text{-}1)$-$B(C, j\text{-}C\text{-}1)$). The recursion captured in the following Corollary captures these conditions.

Corollary 1: The probability $P(C, j)$ that at least one message consisting of C consecutive bits has been transmitted without error in a window of length j alternating between single shot message attempts interlaced with gaps of g bits can be calculated using the following expressions:

$$\forall j, 1 \le j < C: \quad P(C, j) = 0 \tag{9}$$

$$\forall j,\ j \ge C: \quad P(C, j) = P(C, j-1) + \begin{Bmatrix} (1.0 - P(C, j-C+1) - B(C, j-C+1)) \cdot (p_{GG})^{(C-1)} & : \text{If mod}(j-C+1, T) = 1 \\ 0 & : \text{Otherwise} \end{Bmatrix} \tag{10}$$

With $B(C, 1) = \pi$, $B(C, j)$ updated for $j > 1$ as given by (7) and $T = C + g$.

Proof: Follows from Theorem 1 and the discussions above. □

When applying the recursion above it must be remembered that if *k* duplicate copies of a message in addition to the primary copy are allocated in a 'window' of length *j*, then there are also *k* gaps of length *g* bits which do not correspond to allocated bandwidth (and could be used, for example, to transmit a message of a different type). Nevertheless the recursion reveals the effect of temporal separation between successive transmissions of the same message; following the scheduled reception of a message at step *j*, the link state is perturbed by the increase in $P(C, j)$ and enters a transient which steadily subsides to the new steady-state $\pi \cdot (1\text{-}P(C, j))$ with co-efficient α. For static errors, one sees that the new steady-state $\pi \cdot (1\text{-}P(C, j))$ is immediately entered with the implication that the choice of *g* has no impact upon success probability. Thus, one may ask for burst errors, how large does the gap *g* need to be such that equation (1) becomes approximately valid again? An acceptable criterion may be that the initial transient perturbation on the link decays by some factor $\delta > 0$, e.g. 0.001 (decay to 0.1% of initial value). Thus we would be very close to (but not exactly at) the new steady-state $\pi \cdot (1\text{-}P(C, j))$. Since the transient decays as a geometric progression with coefficient α, this criterion is easily met with choice of a gap *g* as follows:

$$g = \lceil \log_{\alpha}(\delta) \rceil = \left\lceil \frac{\ln(\delta)}{\ln(\alpha)} \right\rceil \tag{11}$$

If the first message in the sequence has probability of success $p = (1\text{-}\pi) \cdot (p_{gg})^{(C\text{-}1)}$, a safe and accurate lower-bound to use for the effective independent success probability of each subsequent message duplicate is then $p_d = (1\text{-}\delta)\ p$. This is a lower bound since even if the link state is accurately known at the time the first message commences transmission at $j = 0$, the link state $B(C, j)$ for $j > 0$ will never be driven to 0 with 100% certainty hence any perturbation has magnitude < 1. From this we may upper bound the failure rate with an analog of equation (1) for burst error environments in the case of one primary message followed by *k* duplicates each with gaps satisfying (11):

$$\lambda_f \le (1 - p)(1 - p_d)^k \tag{12}$$

Equations (11) and (12) thus give guidance as to how message duplicates should be effectively spaced in a burst error environment and how the reliability of delivery grows with each successive duplicate copy of the message. By making δ suitable small (e.g. 10^{-9}), equation (1) becomes directly applicable again (for all practical purposes) at the expense of a larger gap requirement. When (11) and (12) are applied to the example of two 166-bit messages 'A' and 'B' for the burst error environment with $\alpha = 0.94995$, choosing $\delta = 0.001$ in equation (11) dictates a gap $g = 135$ bits should be used. The success probability for two copies of the message (i.e. with $k = 1$) is lower bounded by 0.9999060 when using *p* as given by (5) and applying equation (12).

4.6 Statistical Evaluation: Duplicated Transmissions

In order to evaluate the accuracy and predictive abilities of the duplicate message probability computations detailed in Section 4.5, an additional series of stochastic simulations was carried out to compare theoretical values to statistical results. In this case, the measured message success rate $\hat{p}$ (i.e. the probability that any one single instance of a periodic message is successful) for a simulated static TDMA schedule using message duplication in the presence of both static bit errors and bursts of bit errors was obtained and compared with the 95% confidence limits on the calculated value of p. The schedule consisted of 4 messages with characteristics as shown in Table I, with one primary copy send in a first slot, and one duplicate copy sent in a second slot with a gap of g bits in between the end of slot 1 and the start of slot 2. The same assumptions were made of the error environments. Ten simulations were performed for different lengths of transmission gap in the presence of static errors, and ten simulations carried out for bursts of errors. All messages were again assumed to always take their worst-case lengths. The results of these experiments are summarized in Table III, which shows the measured probability of message delivery $\hat{p}$ versus the calculated probability of message delivery p in each case.

Table III. Comparison of calculated versus measured statistics for duplicate transmissions

	Static Errors		Burst Errors	
Gap Length (bits)	Calculated (p)	Measured ($\hat{p}$)	Calculated (p)	Measured ($\hat{p}$)
0	0.9765836	0.9766910	0.9989816	0.9989813
5	0.9765836	0.9766400	0.9991930	0.9992060
10	0.9765836	0.9766502	0.9993565	0.9993679
15	0.9765836	0.9766244	0.9994829	0.9994846
20	0.9765836	0.9766291	0.9995808	0.9995910
25	0.9765836	0.9765478	0.9996565	0.9996561
30	0.9765836	0.9765314	0.9997150	0.9997308
35	0.9765836	0.9766016	0.9997603	0.9997683
40	0.9765836	0.9765336	0.9997954	0.9997976
135	0.9765836	0.9765594	0.9999141	0.9999134
∞	0.9765836	-	0.9999151	-

From the data reported in the Table, a close correspondence between the calculated and measured probability of delivery can again be observed. The maximum relative absolute error in the number of messages predicted to be delivered per second and the number recorded in the simulation was < 0.0109 %. The obtained values of $\hat{p}$ were again within the 95% confidence limits as calculated by (8), and hence we have no reason to suspect model inaccuracy. As can be seen in the burst error results, the probability of successful delivery grows slowly as the transmission gap g is increased. In particular, one observes that for a gap of 135 bits, the probability is close to its maximum Theoretical value achievable using an infinitely large gap. The resulting probability is greater than that obtained when the duplicate probability is lower bounded using (12), although the result is close. This indicates the effectiveness of using equations (11) and (12) in the case of message duplicates. For the static error case, as expected there is no significant change in the success probability as g is increased. Again, although these simulations represent an idealized approximation of a real CAN network and error model, the results nevertheless give a good indication of the accuracy and applicability of the probability computations outlined in Section 4.5.

Before leaving this Section, the reader is pointed to two observations from these results. Firstly, in the presence of static errors the probability of success in Table II for a window size of 332 bits is greater than an equivalent allocation of 332 bits (two messages) in Table III for any choice of the gap g. Secondly, in the presence of burst

errors the probability of success in Table II for a window size of 332 bits is *not* greater than an equivalent allocation of 332 bits in Table III for choices of the gap $g > 0$. In particular, when the gap was chosen according to equation (11) with the value $\delta = 0.001$, message duplicates were significantly more reliable than a transmission window. This is not in agreement with our previous analysis [13] and has been revealed by the more accurate analysis. As such, in the next Section we investigate further the conditions in which transmission windows are beneficial through further computational experiments.

5. COMPUTATIONAL STUDY

In order to begin to investigate the effectiveness (or otherwise) of the proposed transmission window technique, several randomized computational experiments were carried out. The experiments were carried out to statistically assess the potential bandwidth savings (or expense) that may be achieved by employing the windowed technique when compared to using message duplicates as the message and environment parameters were randomly varied within sensible limits. The accuracy of equation (11) was also evaluated (where appropriate) during the course of the experiments. Both single errors and bursts of errors were considered. As discussed in Section 4.5, gap lengths were removed from the results in the case of message duplicates.

5.1 Experiment Configuration

For the first experiment configuration 500,000 message streams were generated with parameters randomly selected from the following intervals: DLC$\in$[0, 8] bytes (uniform distribution), $\lambda \in [10^{-9}, 10^{-4}]$ failures (uniform distribution on log scale). DLC represented the Data Length Code (payload size) of the CAN message. Standard (11 bit) and extended (29 bit) identifiers were randomly employed; from [19], the total number of bits C for a message with an 11-bit identifier was then given by:

$$C = 55 + 31 + 10DLC \tag{12}$$

And for a 29-bit identifier:

$$C = 80 + 31 + 10DLC \tag{13}$$

Both of the above expressions include a 31-bit worst-case error frame [19]. In addition, an inter-slot clearance of 10 bits was assumed to be added to each transmission slot (or window) to allow for clock errors, a not unreasonable assumption. For each message stream, the static BER β of the environment was chosen from the interval $\beta \in [10^{-7}, 10^{-3}]$ (uniform distribution on log scale), covering Aggressive/Hostile through Normal to relatively Benign environments [20]. Equation (1) was employed to calculate the number of duplicates required to meet the target failure rate for message duplicates, and the window sizing algorithm employed for window transmissions.

For the second experiment configuration 500,000 message streams were again generated with parameters randomly selected as before. This time, although the BER β was selected from the same interval, it was treated as the transition probability p_{GB} in a burst error model. In each case the mean burst error length $L = (1/p_{GB})$ was drawn randomly from the interval $L \in [5, 20]$ bits (uniform distribution). Such burst lengths were selected since the experimental findings in [20] found burst lengths with mean duration 5 bits at 1 Mbps; the illustrative examples based upon an experimental CAN implementation in [31] employed mean burst lengths of up to 49 bits. In this set of experiments, the gap for duplicates g were taken to be as small as possible (i.e. the inter-slot spacing P) and equation (11) was not employed. In the third and final experiment configuration, the parameters were as in the

second configuration – however this time the ideal duplicate gap *g* was calculated using equation (11) for each test case, using $\delta = 10^{-9}$. In this final configuration, the required number of duplicates was also calculated analytically from equation (12) and the result compared with the result of the exact recursion. A custom program written in C++ was employed to carry out these computational experiments, which were all ran on a standard personal computer. Results from all three experiments were as shown in Table IV.

Table IV. Comparison between message duplication and window transmission.

	Bandwidth Increase/Decrease (%)			Window Size (bits)		Duplicates Size (bits)	
Configuration	Ave	Max	Min	Ave	Max	Ave	Max
Static Errors	22.4	72.7	0.0	341.3	684	494.2	2412
Burst Errors / Minimal Gaps	32.7	81.3	0.0	345.4	578	582.4	2814
Burst Errors / Ideal Gaps	19.1	79.8	-75.0	345.4	577	500.6	2613

5.2 Results and Discussion

The average and maximum bandwidth reduction and the maximum bandwidth increase for the three experiment configurations was as shown in Table IV. Also shown are the average and maximum required window sizes and the total number of bits need to implement message duplicates. From the data it may observed that for the static error case and the burst error case with minimal gaps for duplicates (configurations one and two), the window transmission scheme poses several benefits in terms of the required bandwidth to meet the target failure rate. On average, the technique saves over 22% in terms of the number of bits allocated to a message transmission for static errors and over 32% for burst errors, and can potentially save over 72% for static errors and 81% for burst errors. The technique performed no worse that message duplicates in any of the tests performed for both these configurations.

Considering now the third case in which ideal gaps between duplicates are employed, the data reveal that the window transmission scheme still poses several benefits in terms of the required bandwidth in the average case, with savings of over 19% potentially rising to over 79%. However, the more considered use of message duplicates results in the average case being reduced by over 13% when compared to using the smallest gaps for duplicates as in configuration two. In addition, it can now be observed that the duplicate technique performed better in some cases, leading to savings over 75% in the best case. In fact, although windowed transmissions were better overall, duplicated messages with ideal gaps performed better in 24.8% of all cases, i.e. almost a quarter of all tests. This is an interesting finding which is not in agreement with our previous results in [13] and is revealed due to the improved analysis presented in Section 4. Another interesting statistic obtained in this configuration was that the average required gap calculated from equation (11) was equal to 238.4 bits, and the largest required gap was equal to 384 bits. The lower bound of equation (12) was also measured in this configuration (using the ideally computed gaps), and found to be worse in only 203 cases out of 500,000 - in other words only 0.04 % of the time. In each case the over-estimation was by never more than one duplicate.

Since message duplication using gaps can be easily designed from equation (12), and the gaps are not of a sufficiently large magnitude to pose especially problematic constraints when generating a TDMA schedule, in the case of burst errors it is suggested to compare the size of the required transmission window with the corresponding allocation in terms of duplicates before any implementation. Analysis of the results indicates that the specific cases in which message duplicates outperform window transmissions were typically those in which comparatively shorter

messages with low failure probability were required in environments with infrequent bursts having longer burst lengths (giving higher values of α). In these cases a small number of well-spaced duplicates typically give the required reliability; however a single continuous transmission window needs to be comparatively longer as the link recovery from transmission failures is slower. Interestingly, our initial analysis has indicated that using duplicated transmission windows separated by ideal gaps can actually leverage the best aspects of both approaches; however such developments are left for future work. In the next Section, attention is shifted from simulation studies to a case study using a prototype implementation of the window transmission scheme in a real CAN network.

6. EXPERIMENTAL STUDY

As discussed in Section 4, although the stochastic simulations that have been performed give a good indication of the accuracy and applicability of the probability computations, they represent an idealized approximation of a true environment and CAN network with limited error detection mechanisms. As discussed, several factors affect the validity of the model and its assumptions, including the assumption that all transmissions require exactly the worst-case number of bits to complete, and that the probability of detecting and signaling errors is uniform along the length of a message. The first assumption is clearly not accurate for almost all practical cases; CAN messages are subject to payload-dependent bit stuffing, and when 11-bit identifiers are employed the best-case stuffing has a frame length 22 bits shorter than the worst-case [35]. Although specialized protocol extensions that completely remove the effects of bit-stuffing are known, these are not without overheads [35]. Even if such techniques are applied, in practice error frames are not present in the vast majority of transmissions (which are completed error-free) and in most cases when they are present will have a duration of around 12 bits, only extending to 31 bits in some special cases [19]. Thus, stochastic variations in transmission times will always be present when considering real network performance; by taking the worst-case assumption, the resulting probability thus represents a lower bound on the achieved probability. The second assumption was discussed previously in Section 3.2, and it was argued that the CAN protocol is designed such that most types of error are detected and signaled at any point in a frame transmission by any node, and uniform probability for detecting and signaling errors along the length of a CAN message was assumed to be ‘good enough’. Since there are differences between the assumptions made and the reality of the situation, there is a need to evaluate the extent to which using transmission window in a real network could realize the potential benefits. The next Section describes a modified CAN controller that has been developed to help implement the window transmission technique.

6.1 Enhanced CAN Controller

Modifications to the CAN protocol at the silicon level have traditionally been very difficult to implement. The case study under consideration in this paper makes use of a modified CAN controller that has previously been developed for this reason, implemented (for research purposes) as a soft-core protocol controller ready for FPGA implementation [27][28]. Although the core has passed CAN-conformance tests, it was developed for research purposes only and is therefore not certified for industrial use. The advantage of such an ‘open’ hardware solution is that various extensions to (or modifications of) the CAN protocol can be implemented and investigated with relative ease. In the context of the current work, the authors have proposed and implemented the following small modification to CAN. In addition to allowing each CAN object in the controller to be operated in ‘standard’ or ‘single-shot’ mode, a third mode of operation – window mode - was introduced. In this mode, a 32-bit hardware counter (which, when active, is incremented by 1 with every bit time on the bus) and two 32-bit match registers (Transmission Start value TS and Transmission End value TE) were added to each CAN object. In addition, the host CPU sees an ‘effective’ Transmit Request (TXRQ) bit, but this bit is in fact a dummy, used only for interface

purposes; a 'hidden' register TXRQ# is employed to control the real transmission logic. When a message transmission is initiated by the host (setting the dummy bit TXRQ), the counter is reset to 0; setting of the TXRQ# bit of the CAN object transmission logic is delayed in the hardware until the counter value is equal to the value programmed in TS. When the counter has subsequently incremented to equal the value programmed in TE, both the TXRQ and TXRQ# bits are automatically re-set in hardware. Since the counter is incremented at the same rate as the network bit-time, this provides a mechanism for a programmable hardware-implemented transmission window for a given CAN object to be defined relative to the point in time at which the TXRQ bit is set. Referring to figure 2 and the discussion in Section 3.2, the worst-case transmission time *C* of the CAN message (including the length of a worst-case error frame) should be subtracted from the value programmed into TE to properly define the end of the transmission window. Additionally, the impact of jitter and latency on the host CPU is minimized, as the timer is referenced to the local CAN oscillator and timing circuitry. The only potential drawback of this solution is that it requires a very small increase in hardware complexity. However, this modification not only allows the effective implementation of the proposed transmission scheme, but would also allow hardware support for the implementation of alternate (similar) protocols such as the Timely CAN [14] and shared-clock [2][8] protocols. As such this very simple hardware change could easily be incorporated into future generations of CAN controllers to increase implementation flexibility.

6.2 Experiment Configuration

A test bench was created employing 4 sensor nodes (implemented using ARM7 development boards) communicating over a shared channel using the modified FPGA CAN controllers as described above. A schematic of the test bench configuration is shown in Figure 6: the CAN network was configured to run at 1 Mbps. Each node in the system was required to send a single periodic message with parameters as shown in Table I. Media access control was implemented using a static TDMA schedule, repeating cyclically every 8 ms. In order to prevent any clock drift issues potentially impacting on the results, the node clocks were explicitly synchronized using a separate dedicated strobe line from the designated master node to the 3 remaining slaves. A 'tick' strobe was generated by a pin on the Master every millisecond via a timer overflow interrupt, generating interrupts on each of the Slaves via an edge-triggered external interrupt pin. The interrupts generated on the Master and Slaves were also used to drive a local task and message scheduler.

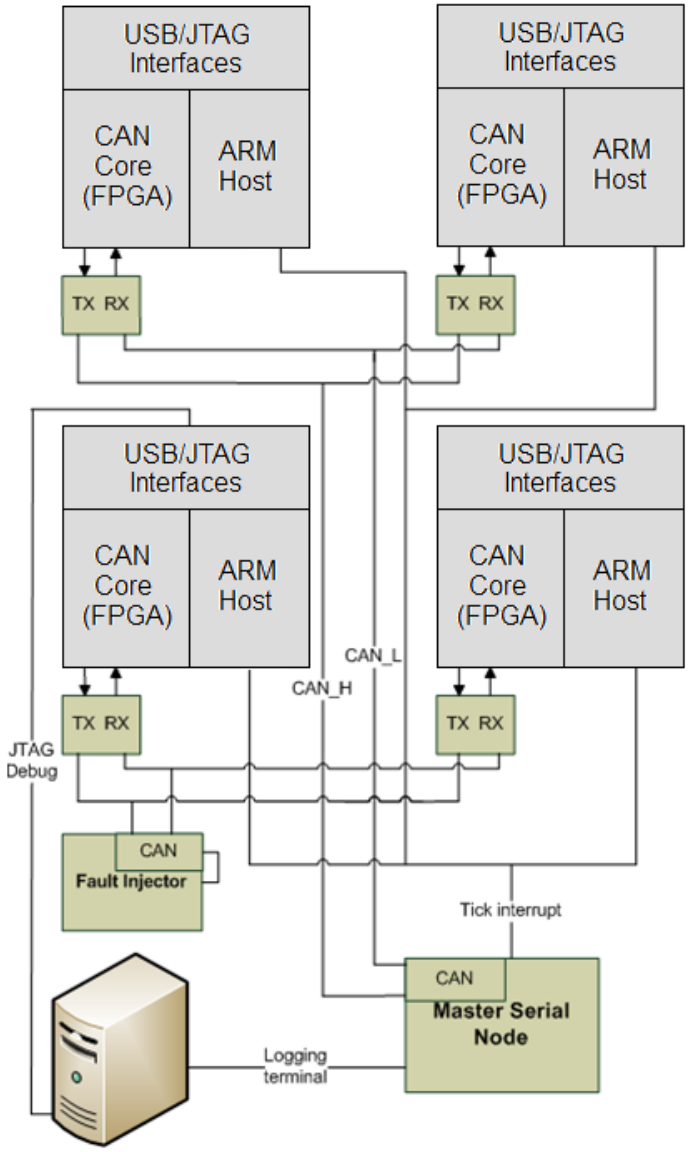

Figure 6. Test bench employed in the experiments.

The data contents of each message sent by each CAN node were generated randomly, but the identifiers were kept constant for identification on the CAN node. As can be seen in Figure 6, an additional node was used to inject bit error sequences onto the CAN bus. A pseudo-random fault generation algorithm was employed on the fault injector node, and was configured to disturb the bus and inject bit errors with a static BER of 0.5×10^{-5}, corresponding to an aggressive environment. Four experiments were conducted for duration of 20 hours each. In the first experiment, one TDMA slot was used to transmit each message using standard single-shot transmission. In the second experiment, two TDMA slots (one primary and one secondary) were used to transmit each message using standard single-shot transmission in each slot. In both of these cases, the start of each slot was separated by 500 μs (bit times) from the previous slot in the schedule. A successful reception was counted when at least one of the messages was delivered. In the third experiment, a 300 bit transmission window was employed; retransmission was allowed at any point up until the 134th bit of the window (TE = TS+134). In the fourth experiment, a 332 bit transmission window was employed; retransmission was allowed at any point up until the 166th bit (midpoint) of the window in this case (TE = TS+166). In both of these cases, the start of each window was separated

by 500 μs (bit times) from the previous windows in the schedule. During the course of the experiments, the master node was employed to log message successes, and the logged data was then transmitted every second through a serial interface to a data logging PC. The same random number generation seed was employed in all experiments. For each case, the measured message success rate $\hat{p}$ (i.e. the probability that any one single instance of the periodic message is successful) was obtained and compared with the calculated value of p. Since the assumption is that the calculated value of p should provide a lower bound on the true value, only a one-sided lower confidence limit was employed, with the 0.975 Quantile (1.960) in equation (8) replaced with a 0.95 Quantile (1.645).

6.3 Experimental Results and Discussion

The results of these experiments are summarized in Table V, which shows the experimentally measured probability of message delivery $\hat{p}$ versus the calculated probability of message delivery p for each case.

Table V. Comparison of calculated versus experimental probabilities

Configuration	Calculated (p)	Measured ($\hat{p}$)
One Single-Shot Message	0.9917341	0.9925217
Two Single-Shot Messages	0.9999317	0.9999437
300 bit Transmission Window	0.9983788	0.9993531
332 bit Transmission Window	0.9999655	0.9999815

From the data reported in the Table, although a correspondence can still be seen between the calculated and measured probability of delivery, it is clear that larger errors are now present. The maximum relative absolute error in the number of messages predicted to be delivered per second and the number recorded in the experiments was around 0.1%, several orders of magnitude larger than the observed differences in the stochastic simulations. Observing the data, however, the obtained values of $\hat{p}$ were all higher than the calculated values and hence satisfy the one-sided lower 95% confidence limit. They would not satisfy corresponding two-sided confidence limits, however. Nevertheless these specific experiment results give no reason to suspect that the calculated values are not useful lower bounds, at least for the specific condition that the error model is representative and its parameters known. Two observations of note appear in the data. The first is that a 300-bit transmission window leads to a higher reliability than a single-shot transmission, as predicted in calculations. This gives some justification for the proposed transmission window approach, since as retransmission was only allowed up until the 134th bit of the window in this configuration (which is less than the worst-case frame size), early detection and signaling of errors must have been taking place along the length of the concerned CAN messages in at least some situations. The second observation is that for an equal bandwidth allocation for duplicates (one single shot message slot of 166 bits plus one duplicate slot of 166 bits) and a transmission window (of total length 332 bits), the window transmission scheme achieved a higher observed (and calculated) probability of delivery. This again gives some justification for the transmission window approach; in that benefits over message duplication may be observed in certain circumstances.

In summary these empirical results partially validate the analysis on a real CAN network, albeit for those situations in which the error model is representative for the target environment and its parameters known. Specifically, they provide partial confirmation that the analytical results presented in Section 4 can represent useful lower bounds on the concerned probabilities. Additionally, the experimental results indicate that the assumption of an (approximately) uniform probability for detecting and signaling errors along the length of a CAN message - although clearly not 100% accurate - has been shown to have merit. Observing the differences between the 300-

bit and 332-bit transmission window, a large increase in reliability for the larger window (which corresponds to the length of two message copies) can be found. Although this could (in principal) be indicative that CRC failures are now better recovered by the network, the jump in success probability is also predicted analytically using a model which takes CRC bit failure probability as being equal to the bit failure probability of any other transmitted bit. Hence the increase between a window of 300 bits and 332 bits can be at least partially attributable to the increased possibility of retransmissions offered by the window technique, and is not just a specific artifact wholly attributable to improved CRC recovery. Nevertheless, we reiterate that although the assumption of uniform bit error detection seems to have practical value, it is not 100% accurate and more detailed modeling and further experimentation are warranted. Such improvements to the analytical model and its empirical validation are planned as part of our future work.

7. CONCLUSIONS

This paper has considered the use of transmission windows as a possible alternative to message duplication when implementing reliable TDMA-based CAN systems, and has introduced the notion of the λ-firm real-time constraint to provide multi-criteria real-time/reliability guarantees. A probabilistic analysis of windowed transmissions and message duplicates has been presented and used to develop techniques for calculating optimal transmission window sizes to achieve specified statistical guarantees of delivery. Our analysis has also suggested a simple formula which calculates appropriate gap sizes for use with message duplicates to de-correlate errors in bursty environments. Stochastic simulations along with computational and empirical results have been presented which have provided validation of the analysis. When compared with message duplicates our findings indicated that for static errors, windowed transmissions were beneficial in all tested cases and that the insertion of gaps between duplicates has no effect on the reliability of message delivery. For burst errors, our findings indicated that windowed transmissions were beneficial in about 75% of tested cases, and that the insertion of appropriate gaps between message duplicates has a significant effect on the reliability of message delivery. Although window transmissions seem beneficial on average, when infrequent burst errors with large expected durations can occur, inserting gaps of length given by our formula between messages duplicates can give a more reliable solution than a transmission window.

ACKNOWLEDGMENT

A preliminary version of the work was presented at 16th International IEEE Conference on Emerging Technologies & Factory Automation (ETFA 2011) [13]. The authors wish to thank the anonymous reviewers for their insightful comments and excellent suggestions which have improved the quality of the paper considerably.

REFERENCES

[1] R. Bosch, “CAN Specification 2.0”, Postfach, Stuttgart, Germany: Robert Bosch GmbH, 1991.

[2] M. Short and M.J. Pont, “Fault-Tolerant Time-Triggered Communication Using CAN”, IEEE Transactions on Industrial Informatics, Vol. 3, No. 2, 2007.

[3] M. Barranco, J. Proenza, & L. Almeida. Boosting the Robustness of Controller Area Networks: CANcentrate and ReCANcentrate. IEEE Computer, Vol. 42, No. 5, pp. 66–73.

[4] S. Kim, E. Lee, M. Choi, H. Jeong & S. Seo. Design Optimization of Vehicle Control Networks. IEEE Transactions on Vehicular Technology, Vol. 60, No. 7, pp. 3002-3016, 2011.

[5] J. A. Cook, I. V. Kolmanovsky, D. McNamara, E. C. Nelson & K. V. Prasad. Control, computing and communications: Technologies for the twenty-first century model T. Proceedings of the IEEE—Special Issue on Automotive Power Electronics and Motor Drives, Vol. 95, No. 2, pp. 334–355, 2007.

[6] FlexRay. [Online]. Available: http://www.flexray-group.com

[7] H. Kopetz, "A Comparison of CAN and TTP", Annual Reviews in Control, Vol. 24, pp. 177–188, 2000.

[8] D. Ayavoo, M.J. Pont, M. Short, and S. Parker, "Two novel shared-clock scheduling algorithms for use with CAN-based distributed systems", Microprocessors and Microsystems, Vol. 31, No. 5, pp. 326-334, 2007.

[9] G. Leen and D. Heffernan, "TT-CAN: a new time-triggered controller area network", Microprocessors and Microsystems, Vol. 26, No. 2, pp. 77-94, 2002.

[10] 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, pp. 1293–1307, 2008.

[11] G.I. Mary, Z.C. Alex and L. Jenkins, "Response Time Analysis of Messages in Controller Area Network: A Review", Journal of Computer Networks and Communications, Volume 2013, Article ID 148015.

[12] I. Broster, A. Burns and G. Rodriguez-Navaz. Comparing real-time communications under electromagnetic interference. In: Proceedings of the 16th Euromicro Conference on Real-time Systems, pp. 45-52, June 2004.

[13] M. Short, I. Sheikh & S.A.I. Rizvi, "Bandwidth-Efficient Burst Error Tolerance in TDMA-based CAN Networks," In: Proceedings of the 16th IEEE International Conference on Emerging Technologies and Factory Automation, Toulouse, France, 5th – 9th September 2011.

[14] I. Broster and A. Burns, "Timely use of the CAN protocol in critical hard real-time systems with faults", In Proceedings of the 13th Euromicro Conference on Real-time Systems, Delft, The Netherlands, June 2001.

[15] R. Miucic, S.M. Mahmud & Z. Popovic. An Enhanced Data Reduction Algorithm for Event Triggered Networks. IEEE Transactions on Vehicular Technology, Vol. 58, No. 6, pp. 2663-2678, 2009.

[16] G. Rodriguez-Navas, S. Roca, and J. Proenza, "Orthogonal, fault-tolerant and high-precision clock synchronization for the Controller Area Network", IEEE Transactions on Industrial Informatics, Vol. 4, No. 2, pp. 92–101, May 2008.

[17] R. Saket and N. Navet, "Frame Packing Algorithms for Automotive Applications", Journal of Embedded Computing, Vol. 2, No. 1, pp 93-102, 2006.

[18] K. Schmidt and E. Schmidt, 'Systematic Message Schedule Construction for Time-Triggered CAN', IEEE Transactions on Vehicular Technology, Vol. 56, No. 6, pp. 3431-3441, 2007.

[19] R. Davis, A. Burns, R. Brill, and J. Lukkien, "Controller Area Network (CAN) schedulability analysis: refuted, revisited and revised", Real-Time Systems, Vol. 35, No. 3, pp. 239–272, 2007.

[20] J. Ferreira, A. Oliveira, P. Fonseca and J.A. Fonseca, "An Experiment to Assess Bit Error Rate in CAN", In Proceedings of the 3rd International Workshop on Real-Time Networks, June 2004.

[21] B. Gaujal and N. Navet, "Fault Confinement Mechanisms on CAN: Analysis and Improvements", IEEE Transactions on Vehicular Technology, Vol. 54, No. 3, pp. 1103-1113, 2005.

[22] D. Liu, X.S. Hu, M.D. Lemmon and Q. Ling, "Firm Real-Time System Scheduling Based on a Novel QoS Constraint", IEEE Transactions on Computers, Vol. 55, No. 3, pp. 1-14, 2006.

[23] H. Aysan, A. Thekkilakattil, R. Dobrin and S. Punnekkat. Efficient Fault Tolerant Scheduling on Controller Area Network. In: Proceedings of the 2010 IEEE Conference on Emerging Technologies and Factory Automation, pp. 1-8, 2010.

[24] D. Smith & K. Simpson. Safety Critical Systems Handbook - A Straightforward Guide to Functional Safety, IEC 61508 (2010 Edition) and Related Standards (3rd Edition), Butterworth-Heinemann, Oxford, UK, 2011.

[25] R.S. Oliver and G. Fohler, “Timeliness in Wireless Sensor Networks: Common Misconceptions”, In: Proceedings of the 9th International Workshop on Real-Time Networks, Brussels, Belgium, July 2010.

[26] E.N. Gilbert, “Capacity of a Burst-Noise Channel”, Bell Systems Technical Journal, Vol. 39, pp. 1253–1265, 1960.

[27] I. Sheikh & M. Short. Conformance Testing of Soft-Core CAN Controllers: A Low-Cost and Practical Approach, In: J.A. Cetto et al. (Eds), Informatics in Control, Automation and Robotics: Lecture Notes in Electrical Engineering, Vol. 85, Part 2, pp. 129-141, Springer-Verlag, Berlin, ISBN: 978-3-642-19729-1, 2011.

[28] I. Sheikh. Improving the performance and reliability of systems which employ the ‘Controller Area Network’ protocol through low-level changes to the controller implementation. PhD Thesis, University of Leicester, UK, April 2011.

[29] L. Almeida, P. Pedreiras and J.A.G. Fonseca, “The FTT-CAN Protocol: Why and How,” IEEE Transactions On Industrial Electronics, Vol. 49, No. 6, pp.1189-1201, 2002.

[30] M. Short & J. Proenza, “Towards efficient probabilistic scheduling guarantees for real-time systems subject to random errors and random bursts of errors,” Proc. of the 25th Euromicro Conference on Real-Time Systems (ECRTS 13), pp. 259-268, Paris, France, July 2013.

[31] N. Navet, Y.-Q. Song, and F. Simonot, “Worst-case deadline failure probability in real-time applications distributed over controller area network,” Journal of Systems Architecture, Vol. 46, no. 7, pp. 607–617, 2000.

[32] B. Tansa, U. Bordoloi, P. Eles and Z. Peng, “Scheduling for fault-tolerant communication on the static segment FlexRay,” Proceedings of the 31st IEEE Real-Time Systems Symposium.

[33] L. Marques, V., Vasconcelos, P. Pedreiras and L. Almeida, “Error recovery in time-triggered communication systems using servers,” 8th International Symposium on Industrial Embedded Systems (SIES), 2013.

[34] M. Yajnik, S. Moon, J. Kurose and D. Towsley, “Measurement and Modelling of the Temporal Dependence in Packet Loss”, Proceedings of INFOCOM 1999, Vol. 1, pp. 345-352, 1999.

[35] 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.

[36] E. Tran, “Multi-bit error vulnerabilities in the Controller Area Network protocol”, Research report series, Carnegie Mellon University: Institute for Complex Engineered Systems, 1999.

[37] Real-Time at Work (RTaW), “RTaW-SIM User Manual v1.4.7”, Technical Report, p.183 (Transmission Error Model), 18th June 2014.