Bandwidth-Efficient Burst Error Tolerance in TDMA-based CAN Networks

Michael Short
Electronics & Control Group,
Teesside University, Middlesbrough, U.K.
m.short@tees.ac.uk

Imran Sheikh and Syed Aley Imran Rizvi
Embedded Systems Laboratory,
University of Leicester, Leicester, U.K.
{si52; sair1}@leicester.ac.uk

Abstract

Many distributed control systems employ TDMA-based communication over CAN in order to meet real-time constraints. Whilst this form of media access control brings several timeliness benefits, studies have also illustrated negative effects on transmission reliability. This paper extends the ‘window transmission’ technique which was recently proposed by the authors to help overcome this problem in TDMA-based networks to include the effects of correlated (burst) errors. This paper employs a simple Markov model to describe burst error behaviors in a CAN network, and the model is used to develop an algorithm for calculating TDMA slot sizes which aim to provide prespecified statistical guarantees of message delivery. Computational results are presented which indicate that the technique can reduce the amount of bandwidth needed for specified reliability levels by a significant factor when compared to the use of message duplicates. The paper is concluded with an empirical study which provides further supportive evidence for the described technique.

1. Introduction

Many networked sensing and control systems employ Time Division Multiple Access (TDMA) communication schemes to increase the predictability of communication and to meet real-time constraints. Although the benefits of TDMA-based systems are numerous and have been well discussed in the literature (e.g. [2][3][4][7][8]), this form of media access control is also known to have several drawbacks, most notably in terms of message transmission reliability [5][14][19]. The principal focus of the paper is upon systems implemented using the Controller Area Network (CAN) protocol [1], although the techniques readily extend to other TDMA-based industrial communication networks.

The CAN protocol was originally intended to allow event-triggered communication between unsynchronized nodes in automotive applications [1][2], and more recently time-triggered communication over CAN has generated a large amount of interest. This is thought to increase the predictability and overall reliability of the network, along with several other benefits (see, for example, [2][3][4][7][8]). A potential drawback with most existing time-triggered CAN implementations lies in the enforcement of single-shot message transmissions [19][20]. Although this effectively bounds worst-case message transmission times, even single bit-errors may directly lead to critical message omissions [3][5]. This can be contrasted with the behaviour of a regular CAN network, in which unbounded re-transmission attempts are allowed. This native approach effectively ensures delivery of critical messages, at the expense of unbounded message transmission times and hence predictability [3][5]. This can be considered as two extremes of behaviour, neither of which is acceptable for most networked real-time applications.

This paper will introduce the notion of the $\lambda$-firm real-time constraint, which intends to model the multi-criteria real-time/reliability guarantees that are required for real-time industrial networks. The paper then extends a simple ‘windowed transmission’ technique previously developed by the authors for the implementation of $\lambda$-firm constraints in TDMA-based transmission schemes to include the effects of burst errors. 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) can then be used to create the frame schedule, using existing frame packing algorithms. The papers employs a simple Markov model to probabilistically describe burst error behaviours in CAN networks, and employs this model in an algorithm for calculating the smallest TDMA slot sizes such that a given set of $\lambda$-firm constraints can be met.

The remainder of this paper is structured as follows. Section 2 reviews previous work in the area. Section 3 introduces the notion of the $\lambda$-firm real-time constraint, and describes the simple modification to existing TDMA-based transmission schemes to accommodate the windowed transmission technique. Section 4 introduces the burst error model, and presents the algorithm to efficiently calculate the required TDMA slot size for a given message. Section 5 describes a simulation-based study carried out to assess some of the statistical properties of the proposed scheme and its potential benefits on network bandwidth, whilst Section 6 describes a practical case study using a modified FPGA-
based CAN controller. This case study was employed to assess the behaviour of the proposed scheme under fault-injection conditions. Both studies give promising initial results. The paper is concluded and areas of future work are described 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 [3] and Rodriguez-Navas et al. [6]. 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. Several software-only synchronization algorithms have been described. 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 of the latter category is 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 complicated software clock synchronization algorithms [3][7].

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 [8]. Again, the protocol provides a maximum accuracy of +/-1 μs, and supports a static TDMA schedule which can provide ‘empty’ slots that allow normal message arbitration for dynamic messages. A full implementation of TT-CAN normally requires dedicated hardware and, at the present time, such hardware has not been widely adopted. 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 [3]. Due to the finite clock error ε which always exists between any two clocks in the distributed system, a small inter-slot spacing \( P = 2ε \) must be employed. 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) are known to generate feasible TDMA schedules (see e.g. [9][10]). It is not uncommon to achieve bus utilizations in excess of 90% using such techniques [3].

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 [3][6][8]. Replica determinism can be enforced - in part - by the use of single-shot transmissions or 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 [3]. 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 [11] - 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, [5] 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\(^{-7}\) in ‘aggressive’ environments [12]. 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 [13]. In more aggressive environments, it has also been reported that around 90% of these errors occur in short correlated bursts, with durations typically between 5 to 20 bit times [5][12][13].

With these points in mind, in a real-time application some form of timeout or upper bound is 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 [3][5][17]. 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 behaviour. Previous works such as [3] and [6] 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 [8]. As this has an undesirable effect on transmission reliability (e.g. [19][20]), for critical message streams, duplication of message instances is the principal means to achieve the desired reliability. For a message requiring \( b \) bits to be transmitted in an
environment with a BER of $\beta$, if $r$ message duplicates are sent then – assuming the error occurrences are memoryless with an underlying Bernoulli distribution - the probability of failure $\lambda$ approximately reduces with $r$ as follows [3]:

$$\lambda = \left(1 - (1 - \beta)^r\right)^\gamma$$  \hspace{1cm} (1)

Similar formulae can be derived under the assumption that fault arrivals possess other underlying distributions; for example Broster et al. derive the following relationship for Poisson fault arrivals [19]:

$$\lambda = \left(1 - e^{-\beta r}\right)^\gamma$$  \hspace{1cm} (2)

However, as was previously shown by the current authors, in most cases sending full message duplicates provides a bandwidth-inefficient solution to the problem [20]. Since bandwidth is relatively scarce in CAN networks, this can be a major issue; in order to rectify this problem, a windowed transmission scheme was also proposed in [20]. Under the assumption that error arrivals are not correlated, the scheme was shown to bring several key benefits to TDMA-based schemes. In Sections 3 and 4, this scheme will be extended to include the case of correlated (burst) errors.

In addition, several schemes have been derived to limit the number of retransmission attempts allowed in native CAN networks. The ‘Timely CAN’ scheme described in [5] proposes a technique to upper-bound the latest time that a particular message can be scheduled for re-transmission under an optimal priority assignment, in order 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. 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 are treated as sporadic events, see e.g. [11][14]. More recently, Aysan et al. propose a technique for scheduling mixed-criticality messages on CAN network [21]. 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 minimise the number of deadline misses under error and error-free scenarios. It is clear from the previous work that has been done in this area that since electromagnetic disturbances cannot be completely eliminated from industrial communication systems unless specialised and expensive (e.g. optical) equipment is employed, then probabilistic guarantees of real-time behaviour need to be sought. A bandwidth-efficient method to achieve these guarantees in TDMA systems in the presence of burst errors is described in the next Section.

3. The CAN Window Transmission Protocol

3.1. $\lambda$-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 behaviour 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. As such, a firm constraint differs significantly from a ‘soft’ real-time constraint, where some benefit is still assumed to be achievable in performing actions after deadlines have elapsed. However, unlike hard constraints, whilst deadline misses are deemed undesirable, they are not deemed to be a serious failure 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 [5][17]. For example, variations of the $(m, k)$-firm CPU tasking model – in which the system is deemed schedulable if at least $m$ from every $k$ deadlines are met – have proved popular in real-time control applications [17].

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 and wireless network applications must be considered very carefully, as it is effectively impossible to provide any crisp, absolute guarantees of timeliness [18]. Thus the notion of the $\lambda$-firm real-time constraint is intended to directly reflect the probabilistic nature of the underlying transmission medium. A message stream with this type of constraint is described by the following four parameters:

$$f_i = (p_i, c_i, d_i, \lambda_i)$$  \hspace{1cm} (3)

Where $p_i$ is the message period, $c_i$ is the message (worst-case) transmission time, and $d_i$ is the message relative deadline. These three parameters are assumed w.l.o.g. to be positive integers in suitable time units, e.g.

---

\(^1\)It should be noted that the original analysis provided in [5] contains an error due to ‘push-through’ blocking; see [11] for further details.
network bit times. To model a \( \lambda \)-firm real-time constraint, each message stream is also allocated an additional parameter, \( \lambda_x \), which is a failure rate for the message stream specified as a probability of unsuccessful transmission per time unit. Note that \( \lambda \) may be derived from a higher-level safety analysis, e.g. an assigned Safety Integrity Level (SIL).

A message set with \( \lambda \)-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); if a message instance is not delivered (omitted), the failure rate for omission is less than \( \lambda \).

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 noise and interference properties of the channel in question. Note that for certain types of message streams, soft real-time constraints may be more suitable; practical message sets may contain a mix of both types of constraints.

3.2. Implementation in TDMA-based Schemes

In a TDMA-based environment, one possible method to increase reliability is to employ duplicate, redundant copies of critical messages. In this paper, we consider an alternative ‘windowed’ transmission scheme which – in many circumstances – can significantly reduce the required amount of bandwidth to be allocated. Suppose we have a critical 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 is assigned a slot immediately following the original. Suppose that a bit-error occurs in both of these slots, as shown in Figure 2 (middle) – in the case of a network such as CAN, this will lead to both messages effectively being dropped, and will result in wasted bandwidth as only single-shot transmission is allowed. However, now consider the situation depicted in Figure 2 (bottom). Both slots are merged in a single window of length \( m \), where \( m \) is specified in network bit times; (re)transmission of message X is allowed from the start of the slot, but only up to the point labelled ‘Upper Bound X’ in the Figure, i.e. \( m_{-c} \) bit times after transmission has first been requested. As can be seen in the Figure, this can have a positive effect on the reliability of the message delivery; even when numerous bit-errors may occur, the probability of successful message delivery can be increased. Also as the slots have effectively been merged, only a single inter-slot spacing is required. In fact, in most cases the required window size can be reduced by a significant amount over sending duplicated single-shot copies; much better use can be made of the available bandwidth.

![Figure 2. Concept of a windowed transmission.](image)

As the window size is fixed, all the desired properties of a time-triggered communication system are retained. Given a set of slot sizes \( m_i \) 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 \( m \) – and hence bandwidth allocated to each message – in order to ensure the message will be delivered with the required statistical guarantee? This question will be addressed in the following Section.

4. Calculating Minimum TDMA Slot Sizes

4.1. General procedure

As was formally shown in a previous work by the authors, for a constant (static) bit error probability, the probability of a successful transmission can be computed in a relatively straightforward manner for increasing values of \( m \) [20]. If the probability of a successful bit transmission is given by \( (1-\beta) \), and the probability of an error is given by \( \beta \) (where \( \beta \) is the BER) the following recursive relationship allows the probability \( p_i \) of successfully transmitting \( b \) bits in an error-free sequence to be calculated for increasing \( j \) (with \( j > b \)):

\[
p_j = p_{j-1} + (1 - p_{j-b-1}) \beta \cdot (1-\beta)^b \tag{4}
\]

With the base cases that for \( j < b, p_j = 0 \) and for \( j = b, p_b = (1-\beta)^b \). In the case of burst errors the probability of a bit error at each step does not remain fixed, but is a function of previous probability values. Nevertheless, if the bit error probabilities can be calculated using some appropriate means and denoted by a set of values \( \beta_j \), equation (4) may be extend to this more general case as follows:
\[ p_j = p_{j-1} + (1 - p_{j-b-1}) \beta_{j-b} \prod_{i=j-b+1}^{j}(1 - \beta_i) \] (5)

With the corresponding base cases that for \( j < b, p_j = 0 \) and \( p_j = (1-\beta_j) \times (1-\beta_j) \times \ldots \times (1-\beta_j) \). In the next Section, a simple Markov model will be presented which may be used to determine the required values of \( \beta \).

### 4.2. Burst Error Modeling

As previously mentioned, much research has shown that many errors in CAN communication links are correlated, and are most likely to occur in isolated transient bursts [12][13][14]. In order to develop a technique for calculating bit error probabilities as described above, an error model that is rich enough to lend itself to analysis is required. The most common way to model bursty behaviour is to introduce memory into the link, and use a simple two-state Markov model of the Gilbert or Gilbert-Elliot types [15], the latter of which is shown in Figure 3.

![Figure 3. Two-state Gilbert-Elliot model of a bursty communication link.](image)

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 \( p_{GG} \) and \( p_{GB} \). The probability of remaining in a given state is then given by \( p_{GG} = 1 - p_{BG} \) and \( p_{GB} = 1 - p_{BG} \). Each state also has associated with it a probability of bit error occurrence (\( BER_g \) and \( BER_B \)). Although this may be useful for exploring the effectiveness of error detection schemes such as CRCs, the probability of full CAN frame transmission occurring whilst the link is in the burst state is effectively negligible² these probabilities can be set to 0 and 1 respectively, significantly simplifying the analysis.

The remaining model parameters \( p_{GB} \) and \( p_{BG} \) can be loosely interpreted as follows: the reciprocal of \( p_{GB} \) defines a mean error gap length \( \mu_{GEG} \), and the reciprocal of \( p_{BG} \) defines a mean error burst length \( \mu_{BER} \) both having a geometric distribution. These parameters may be directly estimated from observed data, or alternatively statistical techniques can be applied to derive their values from larger empirical data sets [15].

²For example, if the mean burst length is 10 bits and \( \beta \) is 0.5 – a realistic assumption for CAN – the probability of successfully sending a 60-bit frame during a burst state is less than \( 10^{-5} \).

Let the state of the Markov model at step \( i \) be encoded as the probability \( \beta_i \) that the link is in the error state. Assuming that each step represents a single bit time, since we have that \( BER_B = 1 \) and \( BER_G = 0 \), \( \beta_i \) also represents the probability that this bit is not successfully transmitted. Applying the normal rules for Markov model state transitions, then the probability that the link will be in an error state at step \( i+1 \) depends only upon the probabilities \( p_{GB} \) and \( p_{BG} \) which can be recursively computed as follows:

\[ \beta_{i+1} = p_{GB} \cdot \beta_i + p_{BG} \cdot (1 - \beta_i) \] (6)

Assuming that the initial state of the link \( \beta_0 \) is known, the bit error probabilities required for the computation of \( p_i \) in (5) can be calculated. In this paper, since there is no activity on the bus prior to the start of any transmission window, it is assumed that \( \beta_0 = 0 \).

### 4.3. Window Sizing Algorithm

The results of the previous two Sections may be combined to create an algorithm to calculate optimal windows sizes for bursty links, which is shown in Figure 4. The algorithm takes as input the message parameters \( b, d \) (expressed in network bit times) and the failure probability \( \lambda \), along with the link error probabilities \( p_{GB} \) and \( p_{BG} \) and calculates the smallest value of window size \( m \) such that the required failure probability is achieved. The notation \( \Pi_{\text{retry}} \) is used to denote the product of the values of \( \beta \) from index \( i=a \) to \( i=b \). It can be observed that only the last \( (b+1) \) values of the probability \( p \) are required to be stored for the computation, and an array of size \( (b+1) \) (which is indexed \( \text{mod}(b+1) \)) can therefore be used to limit the required memory storage of the algorithm. In addition, assuming the \( \beta \) probabilities are stored and indexed in a similar fashion, the computation of \( \Pi_{\text{retry}} \) can be reduced to a single multiplication and division at each step.

```python
01 Proc Optimal_Window(b, d, \lambda, p_{GB}, p_{BG})
02 {
03 \hspace{1em} p_{0} := 0;
04 \hspace{1em} B_{0} := 0;
05 \hspace{1em} \Pi_{0} := 1 - p_{GB};
06 \hspace{1em} \Pi_{1} := 1 - p_{BG};
07 \hspace{1em} \text{FOR} \ m:=1 \ \text{TO} \ b \ \text{DO:}
08 \hspace{2em} B_{m} := \Pi_{0} \times (B_{m-1}) + \Pi_{1} \times (1 - B_{m-1});
09 \hspace{2em} \Pi_{m} := \Pi_{m-1} \times (1 - B_{m});
10 \hspace{1em} \text{END FOR}
11 \hspace{1em} p_{m} := \Pi_{m} \times (1 - B_{m});
12 \hspace{1em} \text{WHILE} \ ((1.0 - p_{m}) > \lambda) \ \text{AND} \ (m < d) \ \text{DO:}
13 \hspace{2em} m := m+1;
14 \hspace{1em} B_{m} := \Pi_{0} \times (B_{m-1}) + \Pi_{1} \times (1 - B_{m-1});
15 \hspace{2em} p_{m} := p_{m-1} \times (1 - p_{m-1}) \times (B_{m} \times \Pi_{0} \times (1 - B_{m})};
16 \hspace{1em} \text{END WHILE}
17 \hspace{1em} \text{RETURN}(m);
18 }
```

![Figure 4: Computing \( m \) for a bursty link.](image)
4.4. Algorithm Analysis

For large $j$, the probability $\beta_j$ converges to a non-zero steady state value corresponding to the resting BER of the link [15]. In conjunction with the analysis developed in [20], the algorithm above is guaranteed to converge for input values $\lambda \in (0, 1)$ and $\beta \in (0, 1)$; however, convergence may take some time for small $\lambda$ and large $\beta$. Given the real-valued representation of these parameters, it is very difficult to provide a (worst-case) run-time bound on the time complexity of the algorithm, when the input is expressed in bits. In a real-time application, however, it can be observed that the message can never meet its deadline (with probability less than $\lambda$) if its window size exceeds the message deadline. As such, the additional termination condition applied at line 12 ($m < d$) bounds such that the algorithm terminates when $m = d$, where $d$ is the relative deadline of the message, expressed in bit times. Note that an appropriate error message can be generated at this point. As with message response time analysis, this effectively bounds the time complexity to be pseudo-polynomial in the task parameters. For $n$ message streams, optimal window sizes may therefore be computed with complexity $O(nd_{\text{ave}})$.

5. Simulation Study

In order to begin to investigate the proposed technique, a small simulation study was carried out. This study was carried out to assess the potential bandwidth savings that may be achieved by employing the windowed technique as the message parameters are varied. Three experiments were carried out. In each experiment, 100,000 message streams were generated with parameters randomly selected (using a uniform distribution) from the following intervals: $p \in [1, 1000]$ ms, $DLC \in [0, 8]$ bytes, $\lambda \in [10^{-5}, 10^{-3}]$ BER. Note that DLC represent the Data Length Code (payload size) of the CAN message. Standard and extended identifiers were randomly employed; a 31-bit worst-case error frame was also added to the length of each message. Three different classes of burst error severity were employed; Benign / Normal with $p_{\text{err}} \in [10^{-5}, 10^{-3}]$, Normal / Aggressive with $p_{\text{err}} \in [10^{-5}, 10^{-5}]$ and Aggressive / Hostile with $p_{\text{err}} \in [10^{-5}, 10^{-7}]$. In each case mean burst error length was drawn randomly from the interval $\mu_{\text{err}} \in [5, 20]$ bits, and the percentage reduction in the number of bits requiring transmission was calculated when employing windowed transmission and message duplicate transmissions.

The average and maximum recorded values for each environment are as shown in Table I. Also shown is the average and maximum recorded window sizes $m$, which also gives an indication of the quick convergence of the slot sizing algorithm, even for low failure rates and high burst error rates. From the Table, it can be seen that the average effectiveness of the proposed technique depends upon the target environment; the worse the expected level of errors, the higher the average reduction in required bandwidth.

<table>
<thead>
<tr>
<th>Environment</th>
<th>Reduction (%) Ave</th>
<th>Window Size (bits) Ave</th>
<th>Reduction (%) Max</th>
<th>Window Size (bits) Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benign / Normal</td>
<td>6.2</td>
<td>279</td>
<td>16.8</td>
<td>792</td>
</tr>
<tr>
<td>Normal / Aggressive</td>
<td>16.8</td>
<td>409</td>
<td>38.5</td>
<td>571</td>
</tr>
<tr>
<td>Aggressive / Hostile</td>
<td>34.1</td>
<td>803</td>
<td>44.6</td>
<td>1863</td>
</tr>
</tbody>
</table>

For hostile environments, an average 34.1% reduction can be achieved; given the comparative scarcity of available bandwidth with CAN, this is a potentially large saving. In normal and benign environments, the average reductions drop to 16.8% and 6.2% respectively; however, the best case reductions remain at around 38%. In each case, the worst-case reductions were 0%, i.e. the technique performed no worse than sending full duplicates. Another interesting extrapolation from the obtained data may be that, for an equal allocation of bandwidth, the windowed technique may be able to increase the reliability of message delivery when compared with sending full duplicates. In the following Section, a case study to test this hypothesis is described.

6. Case Study

6.1. Modified CAN Controller

The case study under consideration in this paper makes use of a modified CAN controller that has previously been developed by the authors. Modifications to the CAN protocol at the silicon level have traditionally been very difficult to implement. The advent of programmable logic devices such as FPGA’s and the maturation of hardware description languages such as VHDL have now changed this situation considerably. Previous work by the authors of the current paper has developed a fully CAN-conformant soft-core protocol controller suitable for FPGA implementation [16]. 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, we have proposed and implemented the following small but powerful 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 $C$ (which, when active, is incremented by 1 with every bit
time on the bus) and two 32-bit match registers (CLB and CUB) were added to each CAN object. In addition, the host CPU sees an ‘effective’ TXRQ bit, but this 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 TXRQ), the counter C is reset to 0; setting of the TXRQ# bit of the CAN object is delayed until C = CLB. Also, when C = CUB, both the TXRQ and TXRQ# bits are automatically re-set in hardware. Since C is incremented at the same rate as the bit-time, this provides for a programmable ‘allowed transmission window’ for a given CAN object which does not require the need for further CPU intervention. The window can be defined relative to any specific point in time, although the message activation time seems most useful. Additionally, the impact of jitter and latency on the host CPU is minimized, as the timer is referenced to the local oscillator. 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 [5] and shared-clock [3][7] protocols. As such this very simple hardware change could easily be incorporated into future generations of CAN controllers.

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 5, with the CAN network 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 II. In each case, the DLC of the message was set to 8, corresponding to a worst-case message length of 155 bits. Media access control was implemented using a static TDMA schedule, repeating every 8 ms. A bandwidth allocation of 300 bits per slot was allotted. In order to prevent any clock drift issues potentially impacting on the results, the node clocks were explicitly synchronized using a separate strobe line from a designated master node to the 3 remaining slaves. A ‘tick’ strobe was generated every millisecond using a timer overflow interrupt.

The data contents of each message send 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 5, an additional node was used to inject faults on to the CAN bus. A pseudo-random fault generation algorithm was employed on the fault injector node, configured to disturb the bus and inject bit errors with a burst error rate of approximately 0.5 x 10^4 and mean length of 20 bits, corresponding to an aggressive environment. Two experiments were conducted for duration of 20 hours each. In the first experiment, standard single-shot transmission was used in each TDMA slot. In the second, the window transmission technique was employed. The corresponding CUB and CLB values to achieve the desired behaviour for experiments 1 and 2 are shown in Table II. For both experiments, in error-free conditions 1875 messages should be sent every second. During the course of the two experiments, the master node was employed to receive all messages successfully sent by each CAN node, and this count was logged in memory. The logged data was then transmitted at regular intervals through a serial interface to a data logging PC. The results obtained are described in the next Section.

| Table II. Static schedule running on the CAN nodes |
|----------------|-----------|-----------|-----------|
| Message | Period (ms) | CLB (µs) | CUB1/2 (µs) |
| A | 1 | 0 | 1/145 |
| B | 2 | 500 | 501/645 |
| C | 4 | 1500 | 1501/1645 |
| D | 8 | 3500 | 3501/3645 |

6.3. Experimental Results and Discussion

In experiment 1, message transmissions were only single-shot during the TDMA slot. Only a single message copy could be send in the allotted slot size of 300 bits. The statistics found were as follows: the average number of messages successfully received every second over the test run was 1739, hence an average drop of 1875-1739 = 136 messages per second. The probability of successful delivery is therefore approximately 0.927. The standard deviation from the Mean was 60 messages. The maximum and minimum values recorded over the duration of the test run were 1848 and 1637 respectively. In experiment 2, message transmission attempts were allowed up to 145 bits following the start of the TDMA slot. The statistics found were as follows: the average number of messages successfully received every second over the test run was 1860, hence an average drop of 1875-1859 = 15 messages per second. The probability of delivery is therefore approximately 0.991. The standard deviation from the Mean was 8.2 messages. The maximum and minimum values recorded over the duration of the test run were 1875 and 1816 respectively. As can be seen from the results, the window technique significantly improved the reliability of message delivery over the course of the experiments. For the employed burst error characteristics and 155 bit messages, the expected probability of message delivery is 0.925, which is close to the value obtained. Using the techniques described in Section 4, for an m = 300 the expected probability of delivery is 0.989, again close to the value obtained.
7. Conclusions and Further Work

This paper has considered the potential reliability drawbacks that may occur when implementing TDMA-based CAN systems, and has introduced the notion of the $\lambda$-firm real-time constraint to provide multi-criteria real-time/reliability guarantees. An algorithm to calculate the smallest TDMA slot sizes such that messages can meet $\lambda$-firm real-time constraints in the presence of burst errors has been described. Simulation and empirical results seem to indicate that the technique can maximise reliability whilst providing timeliness guarantees; however further experimental evidence is needed in this respect. Future work will attempt to obtain such evidence, and extend the concept of $\lambda$-firm real-time constraints into priority driven communications schemes.

References