Schedulability Analysis of Server-Based Error-Recovery Mechanisms for Time-Triggered Systems

Luis Marques, Verónica Vasconcelos
Instituto de Telecomunicações
Instituto Politécnico de Coimbra,
DEE, ISEC
Coimbra, Portugal
lmarques@isec.pt, veronica@isec.pt

Paulo Pedreiras
Instituto de Telecomunicações,
DETI, Universidade de Aveiro
Aveiro, Portugal
pbrp@ua.pt

Luís Almeida
Instituto de Telecomunicações,
Fac. Engenharia, Univ. Porto
Porto, Portugal
lda@fe.up.pt

Abstract

Distributed Embedded Systems are subject to transient communication faults that need being detected and mitigated in safety-critical scopes. This paper addresses error recovery in time-triggered systems based on the Controller Area Network (CAN). It extends a recent work that proposed using online traffic scheduling, combined with servers, to implement dynamic message retransmissions. In particular, we provide a schedulability analysis that considers the interference of the error-recovery server in the time-triggered traffic, as well as a methodology to compute the worst-case response time of messages affected by errors. We also present a comparison with related error-recovery methods that confirms the superiority of the proposed method.

1. Introduction

Distributed Embedded Systems (DES) are used in a broad range of application domains and, due to their distributed nature and environment constraints, often depend on dedicated networks to achieve reliability and timeliness, such as Controller Area Network (CAN) [1].

Many DES applications are intrinsically safety-critical, requiring high levels of reliability and integrity, which can be achieved using fault tolerance techniques. These techniques typically fall in one of two classes: temporal or spatial redundancy. This paper focuses on temporal redundancy techniques to handle transient faults in time-triggered communication systems using dynamic traffic scheduling. It builds upon the work in [2] that proposes using a dynamic server, instead of a static allocation of slots, to convey message retransmissions, generating significant bandwidth savings. In particular, we present analytic tools to compute the interference of the server in the time-triggered traffic as well as the worst-case response time of messages affected by errors.

The paper is organized as follows. Section 2 presents an overview of related work. Section 3 presents the system model together with the schedulability analysis for the periodic traffic with errors. Section 4 presents a brief comparison with other proposals. Section 5 concludes the paper and refers to future work.

2. Related work

Highly-reliable systems typically use networks with statically scheduled time-triggered (TT) traffic due to the determinism of its transmission pattern, allowing prompt detection of omitted messages. Conversely, error correction in TT systems implies reserving a priori retransmission slots for message replicas. This imposes an undesired compromise between bandwidth efficiency, timeliness and reliability, since the recovery latency depends on the slots frequency and, due to the stochastic nature of faults, most of the time they are left unused.

Recent work proposes reducing such bandwidth by using optimization techniques in the scheduling process of FlexRay, to minimize the number of extra slots for a reliability goal [3]. The work in [4] was applied to TDMA over CAN and reduced error-recovery bandwidth enlarging slots slightly while allowing retransmissions inside the slot. Still on CAN, a different approach proposed the coexistence of TT slots with lower priority event-triggered (ET) traffic that could reuse the bandwidth left free by the former [5]. Despite interesting, this approach relies on the native retransmission and arbitration mechanisms of CAN, lacking flexibility in traffic management. Finally, the FTT-CAN [6] protocol schedules the TT traffic online according to any desired policy, granting a significant flexibility in managing this traffic and its retransmissions. In this paper we will use the omissions detection mechanism of FTT-CAN proposed in [2], and schedule the retransmissions through a periodic server integrated with the TT traffic.

3. System model and schedulability analysis

The FTT paradigm [6] uses dual-phase transmission cycles, termed Elementary Cycles (EC), where synchronous (TT) and asynchronous messages (ET) are transmitted in disjoint windows. A master node schedules the synchronous messages online, EC by EC, and dispatches them through an EC-schedule disseminated via a broadcast Trigger Message (TM). The master also listens to the bus traffic and detects missing messages.

The system is composed of a set of nodes connected to a bus, each node transmitting n_i messages. We also consider a synchronous message set M comprised of n synchronous messages characterized by a period (T_i), deadline (D_i) and offset (O_i), all expressed as integer number of ECs, plus transmission time (C_i) and priority (Pr_i), as in Equation (1).

$$M = \{m_i(C_i, T_i, D_i \leq T_i, O_i, Pr_i), i = 1..n\} \qquad (1)$$

The proposed mechanism for error recovery uses a Deferrable Server S with period T_S and capacity $C_S = n_{s*}C_{MAX}$, where C_{MAX} is the transmission time of the longest message, parameterized as described in [2].

3.1 Fault model

In this work we consider a distributed embedded system based on a simplex CAN bus in which messages can be corrupted by random bit errors. The arrival rate of the errors, i.e., the Bit-Error Rate (BER), is a function of the environment and it is known [7]. The occurrence of bit errors is assumed to follow a Poisson process with average error rate equal to λ, where the mean depends on the environment. We also consider at most one fault per EC, which is a reasonable assumption since the EC duration is typically small compared to 1/λ. This assumption will be dropped in future works. Since the focus is on transient faults and temporal redundancy techniques, we do not consider bus partitions. Moreover, we also consider a few simplifying assumptions that we expect to drop in future work, namely that all nodes are fail silent, and that all faults are symmetric.

3.2 Assessing traffic schedulability

According to [2] we will use servers to recover from transient errors affecting the TT messages and we focus, here, on the schedulability aspects.

3.2.1 Indirect interference

Error recovery is carried out within servers, which bound the errors impact on the overall synchronous message set schedulability. We call the impact of the servers *indirect interference*. Figure 1 (top and center) shows the case of messages 4, 7 and 8 that are delayed by one EC due to such interference. Analytically, the error-handling server is modeled as an extra synchronous message with maximum priority for swift error recovery.

Given the assumption that only one error occurs per EC, the execution of the server is also constrained to C_{MAX} per EC. This behavior contrasts with the standard execution model associated with servers, in which the requests arrival rate is unconstrained and thus servers may execute continuously during C_S time units.

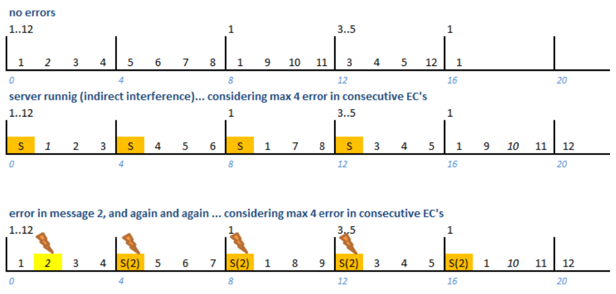
no errors
1..12
1
3..5
1
1 2 3 4 5 6 7 8 1 9 10 11 3 4 5 12 1
0 4 8 12 16 20
server runnig (indirect interference)... considering max 4 error in consecutive EC's
1..12
1
3..5
1
S 1 2 3 S 4 5 6 S 1 7 8 S 3 4 5 1 9 10 11 12
0 4 8 12 16 20
error in message 2, and again and again ... considering max 4 error in consecutive EC's
1..12
1
3..5
1
1 2 3 4 S(2) 5 6 7 S(2) 1 8 9 S(2) 3 4 5 S(2) 1 10 11 12
0 4 8 12 16 20

Figure 1: Indirect interference and Synchronous messages transmission and retransmission affected by errors

Therefore, we model the execution of the error-handling server with a set $\hat{S}$ of n_S maximum-size messages, with a 1 EC relative offset among them, as in Equation (2), where Pr_{MAX} represents the maximum priority.

$$\hat{S} = \{m_i^S(C_{MAX}, T_S, T_S, (i-1), Pr_{MAX}), i = 1..n_S\} \quad (2)$$

We then assemble an extended message set M^E composed by the original set M and the server set $\hat{S}$. With this transformation, the schedulability of the synchronous message set, considering indirect interference, can be assessed by applying the response-time analysis to all messages of M over the extended set M^E. The whole procedure is described in Algorithm 1, where the inflation of transmission times is necessary to adapt classic response time analysis taking into account the inserted idle time [8]. Note that, for the sake of flexibility of scheduling, we consider a synchronous release of the synchronous messages and server m_1^S.

One relevant detail in our model is that the negative impact in schedulability incurred by using a DS due to back-to-back execution [9] does not occur. In fact, a back-to-back execution in which the capacity of the server would be consumed, in a given instance, at the end of its period and, in the following instance, at the beginning, would represent a number of errors within a window of a server period higher than what our model accounts for.

3.2.2 Schedulability of the synchronous messages affected by errors

In addition to indirect interference, messages can also be hit by one or more errors. Figure 1 (bottom) shows such a scenario with an error affecting the transmission of message 2 and afterwards further errors affecting its retransmissions within the error-recovery server.

In the general case, messages can be affected both by indirect error interference (when in the ready queue) and direct error interference (when being transmitted or retransmitted). The former can be accounted for with one additional (virtual) message (m_i^S) per EC, only, while the latter implies an additional latency of one EC per error. Therefore, the worst-case response time [10] happens when the n_S consecutive errors affect a message directly, i.e., when the message transmission and its successive n_s-1 retransmissions are affected by errors.

The worst-case response time (Rwc_i^{Err}) can be trivially computed as in Equation (3) in which Rwc_i is computed without considering the impact of errors and then n_s additional ECs are added to account for the possible retransmissions.

$$Rwc_i^{Err} = Rwc_i + n_s * LEC \qquad (3)$$

Algorithm 1: Indirect Server Interference Computation

```
Inputs: M, S
Output: Schedulable (Boolean)

1.  Compute $C_{MAX}$
2.  Compute $\hat{S}$, $M^E$ (inflate all transmission times)
3.  For each $m_i$ in M
3.1       Compute $Rwc_i$ considering $M^E$;
          the server interference $C_{MAX}$ is added only once
          each EC and at most $n_s$ times
3.2       If $Rwc_i > D_i$ return Schedulable=false
4.  Return Schedulable=true
```

Algorithm 2: Rwc considering errors

```
Inputs: M, S, $n_S$
Output: Schedulable (Boolean)

1. Compute $C_{MAX}$
2. Inflate all transmission times
3. For each $m_i$ in M
4.1       Compute $Rwc_i$ considering M
4.2       $Rwc^{Err}_i$= $Rwc_i$+$n_s$*LEC
4.3       If $Rwc^{Err}_i$ > Di return Schedulable=false
5.  Return Schedulable=true
```

3.3 Analytical example

To illustrate the use of this analysis, we use the message set based on the updated SAE benchmark [10]. Table 1 presents this message set, where ID is the CAN message identifier, DLC is the message payload in bytes, T is the message period, both expressed in number of ECs. The maximum message uses approximately 10% of the channel bandwidth, LEC is 2.5ms and LSW takes 85% of *LEC*. A server S(2500,4) is used to handle retransmissions.

Columns 4 to 8 of Table 1 present the worst-case response time of each message considering different error incidences. A scenario without errors (col. 4) is first considered, to establish a baseline. Two other scenarios, with one and two consecutive errors, are then presented. The first one (columns 5, 6) considers only the indirect interference, while the second one (columns 7, 8) takes into consideration the direct impact of errors.

The results in columns 5 and 6 of Table 1 show that indirect interference has a small impact in the worst-case response time of the messages, as expected, with most of the messages not suffering any penalty. In the first 30 messages, only messages 13, 23-25, 28 and 29 suffer an additional delay of one EC. Messages 30 to 36 suffer a delay of one or two ECs. Note, however, that these latter messages have long deadlines. It can be readily observed that the message set maintains its schedulability. However, the situation changes when considering the direct impact of errors (Table 1, columns 7 and 8). The rescheduling-based recovery mechanism imposes at least a one cycle delay per error, thus the worst-case response time of all messages is delayed by at least one or two ECs. For the single error scenario, all messages still meet the deadline, since they have a slack of at least one EC and thus can accommodate the additional delay. However, when considering two consecutive errors the situation changes. All messages with 2ECs deadline and a few with 3ECs deadline violate the deadline. This corresponds to the same message being hit by an error twice in a row, i.e., transmission and successive retransmission, which is a possible but rather infrequent situation, having a probability of around 3E-9 (aggressive BER, 100 bits message).

Table 1 – Message set and Rwc obtained by Alg. 1 and 2

			Number of Errors in Consecutive EC's				
N	Ti = Di	DLC	0	1	2	1	2
1	20	1	1	1	1	2	3
2	2	2	1	1	1	2	**3X**
3	2	1	1	1	1	2	**3X**
4	2	2	1	1	1	2	**3X**
5	2	1	1	1	1	2	**3X**
6	2	2	1	1	1	2	**3X**
7	2	1	1	1	1	2	**3X**
8	2	1	1	1	1	2	**3X**
9	3	1	1	1	1	2	3
10	3	1	1	1	1	2	3
11	3	1	1	1	1	2	3
12	3	1	1	1	1	2	3
13	3	1	1	2	2	2	3
14	3	4	2	2	2	3	**4X**
15	3	4	2	2	2	3	**4X**
16	3	4	2	2	2	3	**4X**
17	4	1	2	2	2	3	4
18	4	2	2	2	2	3	4
19	4	6	2	2	2	3	4
20	4	2	2	2	2	3	4
21	4	3	2	2	2	3	4
22	4	2	2	2	2	3	4
23	5	2	2	2	3	3	4
24	5	2	2	3	3	3	4
25	5	2	2	3	3	3	4
26	5	2	3	3	3	4	5
27	5	4	3	3	3	4	5
28	5	5	3	3	4	4	5
29	5	3	3	3	4	4	5
30	20	1	3	4	4	4	5
31	40	4	4	4	4	5	6
32	40	1	4	4	4	5	6
33	40	1	4	4	6	5	6
34	400	3	4	6	6	5	6
35	400	1	4	6	6	5	6
36	400	1	6	6	6	7	8

4. Comparison with other proposals

In our previous work [2] we had already shown that the proposed error recovery method could achieve high reliability, better than (1-1E-7), even for BER of the order of 3E-7 (aggressive environment), and using bandwidth for the recovery server as small as 0.06% of the channel bandwidth.

The method presented by [3] is applied to the static segment of a FlexRay system and tries to find the minimum number of retransmissions to meet a communications reliability goal. Applying such approach to the benchmark in Table 1 (excluding the server) we conclude that we need 2 retransmissions for every message in order to get (1-1E-6) reliability, meaning tripling the bandwidth used by the original communication requirements. We conjecture that such high penalty is due to the fact that this mechanism is purely static, so always transmitting duplicates even when there are no errors (most of the time). Concerning the recovery latency, it depends on where the retransmissions are positioned in each message period.

Using a Polling Server (PS) instead of a DS, showed that a recovery rate equal to the DS is only possible with a *Ts* equal to 1 EC and *Cs* equal to C_{MAX} (10% in this benchmark) because the PS does not preserve bandwidth. Thus, comparing to the DS server it uses about 1500 times more bandwidth to obtain the same level of message recovery. In this situation, both servers present a recovery latency of 1 EC.

Another alternative that has the same bandwidth impact as the PS is to reserve C_{MAX} every EC for the retransmission of any message that is affected by an error, using the native retransmission mechanism of CAN. This scheme has been used in FTT-CAN and it allows the message recovery to occur immediately after the error inside the same EC. However, note that the use of the CAN native retransmission might be a constraining factor as opposed to using a server with which the error messages are scheduled by the Master as desired.

5. Conclusions

The design of DES in high reliability systems must take into account communication faults, especially transient ones, which are the most frequent. Temporal redundancy by retransmitting messages allows obtaining the necessary reliability level. Nevertheless, the devised mechanisms must be bandwidth efficient, have a small impact in the global system schedulability and be properly modeled and analyzable.

In a recent work the authors proposed a mechanism to recover from transient faults in time-triggered systems using a Deferrable Server. In this paper, we assessed the impact of the server in the schedulability of the time-triggered messages and derived algorithms to compute the worst-case response time of messages affected by errors. We also carried out a comparison with related error recovery techniques found in the literature, highlighting the minimal use of extra bandwidth incurred by our error recovery mechanism without sacrificing recovery latency.

As future work we will analyze the possibility of applying the proposed mechanisms to other dual-phase elementary cycle protocols, e.g. the FTT-SE, and FlexRay. We will also target relaxing the most restrictive assumptions, such as atomic broadcast and independence of consecutive bit errors.

Acknowledgments

This work was partially supported by the Portuguese Government through FCT - Fundação para a Ciência e a Tecnologia.

6. References

[1] Controller Area Network (CAN) specification - version 2.0., Bosch GmbH, 1991

[2] L. Marques, V. Vasconcelos, P. Pedreiras and L. Almeida, “Error Recovery in Time-Triggered Communication Systems Using Servers”, 8th IEEE International Symposium on Industrial Embedded Systems, 19-21 June 2013, Porto, Portugal.

[3] B. Tanasa, U. Bordoloi, P. Eles and Z. Peng, "Scheduling for fault-tolerant communication on the static segment of FlexRay", Proceedings of 31st IEEE Real-Time Systems Symposium, 2010

[4] M. Short, I. Sheikh and S. Rizvi, “Bandwidth-Efficient Burst Error Tolerance in TDMA-based CAN Networks”, Proc. of the IEEE 16th Conference on Emerging Technologies & Factory Automation (ETFA), Touluse, France, September 2011

[5] J. Kaiser, B. Cristiano and C. Mitidieri. "COSMIC: A real-time event-based middleware for the CAN-bus," Journal of Systems and Software 77.1, 2005

[6] L. Almeida, P. Pedreiras and J.A. Fonseca, “The FTT-CAN protocol: Why and how”, IEEE Transactions on Industrial Electronics, 49 (6), December 2002.

[7] J. Ferreira, A. Oliveira, P. Fonseca and J.A. Fonseca, "An Experiment to Assess Bit Error Rate in CAN", Proc. of 3rd International Workshop of Real-Time Networks, 2004

[8] L. Almeida and J. A. Fonseca, “Adapting Preemptive Scheduling Analysis to cope with Non-Preemption and Inserted Idle-Time”, Proc. Work-in-Progress session of IEEE RTSS 2000, Orlando, USA, November 2000

[9] G. C. Buttazzo, *Hard Real-Time Computing Systems - Predictable Scheduling Algorithms and Applications*, Springer, 2011

[10] N. C. Audsley, A. Burns, M. Richardson, and A. Wellings, “Hard Real-Time Scheduling: The Deadline Monotonic Approach”, IEEE Workshop on Real-Time Operating Systems, 1992.

[11] U. Mohammad and N. Al-Holou, “Development of An Automotive Communication Benchmark”, Canadian Journal on Electrical and Electronics Engineering, Vol. 1, No. 5, August 2010