Towards Efficient Transient Fault Handling in Time-Triggered Systems

L. Marques[1], V. Vasconcelos[1], P. Pedreiras[2,], L. Almeida[3], V. Silva[4]

[1]Instituto Superior de Engenharia de Coimbra/IPC
{lmarques, veronica}@isec.pt
[2]DETI-Universidade Aveiro UA/IT
pbrp@ua.pt
[3]Fac. Eng.Universidade do Porto/IT
lda@feup.pt
[4]ESTGA-Universidae Aveiro
vfs@ua.pt

Abstract. *Transient communication faults in distributed control systems (DCS) are unavoidable but must be handled adequately in order to enforce correct system behaviour. A typical way of handling transient faults is temporal redundancy by means of retransmissions. However, DCS are frequently designed with time-triggered architectures, being scheduled offline and not coping efficiently with retransmissions as these require the pre-allocation of bandwidth that, in the absence of errors, is wasted. In this paper we propose using the Flexible Time-Triggered paradigm to reconcile the Time-Triggered model with on-line scheduling of retransmissions when needed, only, leading to an efficient bandwidth usage. This is confirmed with preliminary experimental results obtained on an FTT-CAN network.*

Keywords: transient communication faults, Flexible-Time Triggered, CAN, on-line scheduling

1 Introduction

Distributed Control Systems (DCS) have been widely used for several decades in many application fields, from industrial machinery to vehicles, robots, industrial facilities or medical equipment. In DCS the nodes must cooperate in order to control the system (control plant), which is carried out over an underlying communication system, or simply network, that supports the exchange of the necessary messages.

As in any real system, the presence of faults is unavoidable, particularly communication faults, and we can grossly classify them as permanent or transient. The former are persistent, e.g. due to physical damage, and manifest until that unit is replaced or repaired. Transient faults (and also intermittent faults) manifest themselves for short intervals of time and can be due to electromagnetic interference, radiation, temperature variations, etc. In many DCS domains, e.g., automotive systems, transient faults are prevalent over permanent ones with a ratio that can reach 100:1 [1] which brings the problem of dealing with transient faults to the front line.

A typical technique to deal with transient communication faults is to use temporal redundancy by means of retransmissions. Ideally, these take place when errors occur, only, but the random nature of faults conflicts with static communication schedules that are typical, for example, in time-triggered systems, e.g. TTP/C [2], TT-CAN [3], FlexRay [4] and Ethernet PowerLink [5], that in turn are the usual choice for safety-critical systems [6]. This conflict is handled allocating extra bandwidth at design time for eventual retransmissions, which leads to a waste of bandwidth whenever errors do not occur, even with optimised schedule designs [7]. Alternatively, we explore, in this paper, the unique features of the Flexible Time-Triggered communication paradigm [8] that combine the time-triggered model with on-line traffic scheduling to handle communication faults in a reactive way. This allows re-scheduling the traffic to include retransmissions when necessary, only, with important gains in bandwidth efficiency. Moreover, this solution also grants a high level of flexibility in terms of the scheduling policies that can be used, which can be virtually any, from dual priority mechanisms to server-based approaches.

In this paper we use the FTT-CAN protocol, which is an implementation of FTT on Controller Area Network. This is, probably, still the most used network technology in DCS. Nevertheless, our work should be applicable to other FTT implementations, too. We then propose three alternatives for error detection and recovery and we conclude the paper with a preliminary experimental validation in a laboratory prototype.

2 Brief Introduction to FTT-CAN

The FTT-CAN protocol [8] is an instantiation of the FTT paradigm on CAN exhibiting two main features, centralized on-line scheduling and Master/Multi-Slave transmission control on top of CAN. It uses an Elementary Cycle (EC) structure split in two phases, one for the asynchronous (event-triggered) and the other to the synchronous (time-triggered) traffic, denoted respectively Asynchronous Window (AW) and Synchronous Window (SW), as illustrated in Figure 1. The synchronous messages are scheduled by the Master node according to any chosen scheduling policy. This node also possesses a mechanism for on-line admission control, guaranteeing the timeliness of all admitted messages. The traffic schedules for each EC generated by the Master are broadcast to all nodes in the beginning of each cycle with a special message called Trigger Message (TM). Other works addressing dependability aspects of FTT-CAN included a master-replication technique [9] and channel replication [10].

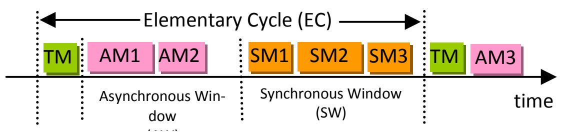

Fig. 1. Elementary Cycle in FTT-CAN

3 Methods to Recovery from Transient Synchronous Message Faults

The fault model adopted in this work considers that channel (bus) faults are symmetric, meaning that the corrupted messages are detected by all nodes in the same way and are of transient or intermittent type. Message corruption is mainly due to EMI and depends on the assumed bit error rate (BER). We also assume that at most one fault occurs per Elementary Cycle, which is considered a pessimistic value [9]. We do not consider bus partitions and the nodes are of fail silent type, i.e., they transmit correctly or not at all. More elaborate fault models are left for future work.

In these conditions, synchronous message omissions can be detected at the end of each cycle comparing the EC schedule in the respective TM with the list of synchronous messages successfully transmitted in that cycle. We then consider three different error recovery methods leading to different compromises in terms of complexity, latency and flexibility.

3.1 Method 1 - Master Rescheduling

In this case, message omissions trigger retransmission requests to the Master that reschedules such communications according to an adequate scheduling policy. Thus, retransmissions take place under the strict control of the Master. This approach requires adding a synchronous messages omissions detector to the Master node that is invoked every EC as shown in the EC management algorithm below.

Algorithm – Updated EC Management

1. Build schedule for EC(n) using the defined scheduling policy and assemble the TM(n).
2. Send the TM(n).
3. Listen to synchronous messages reception along the EC(n) and update a Vector of Received Messages (VRM) until the end of the cycle. Set the corresponding bit in the VRM each time a message is received.
4. Compare TM and VRM and reactivate the missing messages for scheduling in the following ECs under control of the scheduler.
5. Return to 1.

Comment 1: EC(n) and TM(n) are, respectively, the n^{th} Elementary Cycle and the Trigger Message for this Cycle.
Comment 2: the TM(n) payload defines bit-by-bit the messages that must be sent, where each bit position corresponds to a CAN ID field; if the bit is set (logical"1") the CAN message with corresponding ID should be sent in this EC.
Comment 3: TM and VRM have the same format.

This approach has the advantage of keeping the retransmissions under full control of the scheduler that can implement some smart techniques, such as increasing or decreasing priority, use servers to enforce temporal isolation among the messages, etc. Moreover, the rescheduling is completely transparent to the nodes that are simply polled another time. However, the error recovery latency is at least two cycles since missing messages are detected at the end of the EC in which the omissions occurred, the rescheduling request is considered by the scheduler in the following cycle and the retransmission in the next one in the best case, two cycles after the actual omission.

3.2 Method 2 - Autonomous Asynchronous Retransmission by the Sender

A second error recovery mechanism consists in using automatic retransmission by the sender node, in the asynchronous window (AW) of the protocol. In this case, the omission detection is done by the sender node which resubmits the omitted message in the asynchronous window of the next cycle. The error recovery latency depends on the message CAN priority, since it will contend for the bus with the remaining asynchronous messages. Considering that in FTT-CAN periodic messages have higher priority than asynchronous ones, that at most one omission occurs per cycle (according to the fault model) and that the asynchronous window has an adequate minimum length, the retransmission will take place in the EC immediately after the failure. This method keeps the recovery inside each node, transparently to the protocol operation. On the other hand there is a partial loss of temporal isolation between asynchronous and synchronous traffic, due to the extra interference of the latter over the former, and the retransmissions are bound to the CAN arbitration.

3.3 Method 3 - Active Slave Redundancy

Finally, a third recovery mechanism consists in using a slotted version of the protocol as described in [11] together with backup nodes for active redundancy. In this case, each critical message is sent by two nodes in the same slot. The backup node aborts the message transmission if the primary node successfully transmits. If an omission occurs, the backup will succeed in transmitting with a rather short latency, masking the error inside its own slot. This is the same mechanism used for master replacement in FTT-CAN [9]. This technique is based on spatial redundancy rather than temporal redundancy, with the inherent higher costs, beyond the requirement for a specific protocol implementation.

4 Preliminary Results and Conclusions

In this paper we presented methods to deal with message omissions in time-triggered systems. We used as a base technology the FTT-CAN protocol which has the distinctive feature of combining a time-triggered model with on-line traffic scheduling. Therefore, we were able to implement a novel and more efficient omissions recovery mechanism based on messages dynamic rescheduling than typical approaches based on statically pre-allocated bandwidth. With our mechanism, bandwidth is only consumed when errors indeed occur. Then we also analysed qualitatively two other recovery methods that present diverse tradeoffs in terms of complexity, cost and latency.

A prototype implementation of method 1 with two nodes plus the Master (3 PIC18F2680 microcontrollers), a CAN bus operating at 125Kbps, and two synchronous messages, each with a period of 5 ECs, allowed validating the concept. One of the messages was purposely omitted once every 3 instances. The system was kept working for 6 consecutive hours without missing any omission and always recovering in two ECs, as expected. We have also done, with the same set of messages, 10 runs of 3000 ECs with a EC duration of 200ms and 20 ms, where we observe again a correct behaviour, meaning that all the omitted messages were correctly rescheduled.

We are currently working towards a better characterisation and eventual implementation of the three methods. Moreover, we will also develop a membership service to cope with node crashes. In the future we will improve the fault model to consider more complex scenarios and we will address different network technologies.

References

[1] Peti, P., Kopetz, H. Obermaisser, R., Suri, S.: From a Federated to an Integrated Architecture for Dependable Embedded Systems, Technical Report 22, Technische Universistat Wien (2004)

[2] Kopetz, H. and Bauer, G.: The Time Triggered Architecture. In: Proceedings of the IEEE, Vol. 91, Issue1 (2003)

[3] Leen, G., Heffernan, D.: TTCAN: a new time-triggered controller area network. In: Microprocessors and Microsystems, Vol. 26, Issue 2, March, Pages 77-94, (2002)

[4] The FlexRay Communications System Specification, Ver 2.1, www.flexray.org

[5] Felser, M.; Sauter, T.: "Standardization of Industrial Ethernet – the Next Battlefield?". In: IEEE International Workshop on Factory Communication Systems (2004)

[6] Kopetz, H.: Real-Time Systems: Design Principles for Distributed Embedded Applications (Real-Time Systems Series), Springer, 2nd Edition (2011)

[7] Tanasa, B., Bordoloi, U., Eles, P., Peng, Z.: Scheduling for Fault-Tolerant Communication on the Static Segment of FlexRay. In: The 31st IEEE Real-Time Systems Symposium (2010)

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

[9] Ferreira, J.: Fault-Tolerance in Flexible Real-Time Communication Systems, PhD Thesis, University of Aveiro (2005)

[10] Silva, V.: Flexible Redundancy and Bandwidth Management in Fieldbuses, PhD Thesis, University of Aveiro (2010)

[11] Ataide, F.; Pereira, C.; Lages, W.; Assis, C.: "On the design of an embedded FTT-CAN platform with improvement of its inherent jitter". In: IEEE Conference On Industrial Informatics (INDIN 2006), Singapore (2006).