Flexible Bus Media Redundancy

Valter Filipe Silva
ESTGA-University of Aveiro
vfs@ua.pt

Joaquim Ferreira
EST - Polytechnic Institute of Castelo Branco
jjf@est.ipcb.pt

José Alberto Fonseca
DETI - University of Aveiro
jaf@ua.pt

Abstract

This paper proposes a flexible approach to bus media redundancy in Controller Area Network (CAN) fieldbuses, both to improve the bandwidth by transmitting different traffic in different channels or to promote redundancy by transmitting the same message in more than one channel. Specifically the proposed solution is discussed in the context of Flexible Time-Triggered protocol over CAN (FTT-CAN) and inherits the online scheduling flexibility of FTT-CAN, enabling on-the-fly modifications of the traffic conveyed in the replicated buses. Flexible bus media redundancy is useful to fulfill application requirements in terms of additional bandwidth or to react to bus failures leading the system to a degraded operational mode, without compromising safety. The arguments for and against flexible bus media redundancy in the context of FTT-CAN are also discussed in detail.

1 FTT-CAN With Multiple Buses Basis

FTT-CAN (Flexible Time-Triggered communication protocol on CAN) [1] has been developed with the main purpose of combining a high level of operational flexibility with timeliness guarantees. It uses the dual-phase elementary cycle concept to isolated time and event-triggered communication. The time-triggered traffic is scheduled online in a particular node called a master, facilitating online admission control of requests, thus being managed in a flexible way, under guaranteed timeliness. The protocol relies on a relaxed master-slave medium access control in which the same master message triggers the transmission of messages in several slaves simultaneously (master/multi-slave). Eventual collisions between slave messages are handled by the native distributed arbitration of CAN.

FTT-CAN slots the bus time in consecutive Elementary Cycles (ECs) with fixed duration. All nodes are synchronized at the start of each EC by the reception of a particular message known as an EC Trigger Message (TM), which is sent by the master node. Within each EC the protocol defines two consecutive windows, asynchronous (law in Figure 1 stands for length of asynchronous window) in and synchronous (lsw in Figure 1 stands for length of synchronous window), that correspond to two separate phases (see Figure 1). The first is used to convey event-triggered traffic (AM in Figure 1 stands for Asynchronous Messages) and the second is used to convey time-triggered traffic (SM in Figure 1 stands for Synchronous Messages). Between these two windows there is a guardian time to guarantee the temporal isolation (α in Figure 1). The synchronous window of the n^{th} EC has a duration that is set according to the traffic scheduled for it. The schedule for each EC is conveyed by the respective EC trigger message (see Figure 2). Since this window is placed at the end of the EC, its starting instant is variable and it is also encoded in the respective EC trigger message.

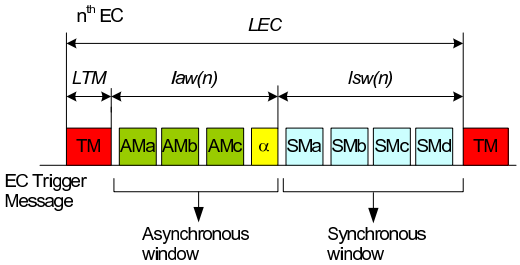

Figure 1. The Elementary Cycle

The communication requirements are held in a database located in the master node [1], the System Requirements Database (SRDB). This database holds several components, one of which is the Synchronous Requirements Table (SRT), that contains the description of the periodic message streams. Based on the SRT, an online scheduler builds the synchronous schedules for each EC. These schedules are then inserted in the data area of the appropriate trigger message (see Figure 2) and broad casted with it. Due to the online nature of the scheduling function, changes performed in the SRT at run time will be reflected in the bus traffic within a bounded delay,

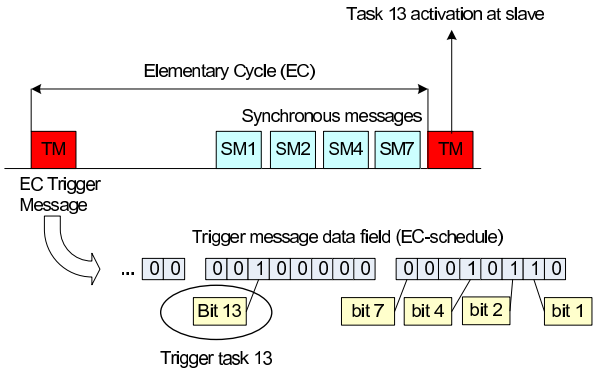

Figure 2. master/multi-slave access control and EC schedule coding scheme

resulting in a flexible behavior.

One recent improvement on the FTT-CAN is the use of the master to control more than one buses in the system [18][19]. Using more than one CAN bus improves both the fault tolerance of the system and the available bandwidth, since messages can be transmitted on different buses. This solution provides additional bandwidth and overcomes the single point of failure of a non replicated CAN bus [18]. In this way, multiple buses can be used either to improve the bandwidth by transmitting different traffic in different channels or to promote redundancy by transmitting the same message in more than one channel. This architecture (see Figure 3) inherits the dispatching flexibility of FTT-CAN, enabling online changes on the traffic conveyed in the channels. This is useful to fulfill application requirements in terms of additional bandwidth or to react to bus failures leading the system to a degraded operational state, without compromising safety.

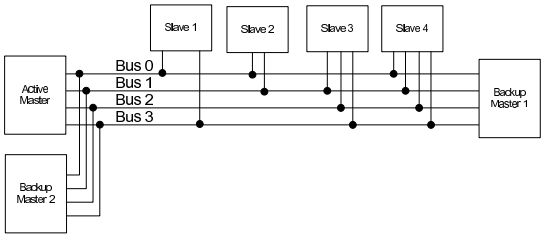

Figure 3. FTT-CAN using multiple buses

Notice that slaves can be connected to just one CAN bus or to a set of buses, depending on the tasks that a specific slave has to perform, on the dependability level and on the bandwidth requirements.

Similarly to the single bus system case, all the buses must convey a synchronized Trigger Message, with the same Elementary Cycle in all of them. That is, the Trigger Messages are issued to all the buses at the same time, dividing the bus time in all buses in the same way. Figure 4 presents an example with two buses.

In the small example of Figure 4 synchronous message 1 is replicated in both buses improving its dependability.

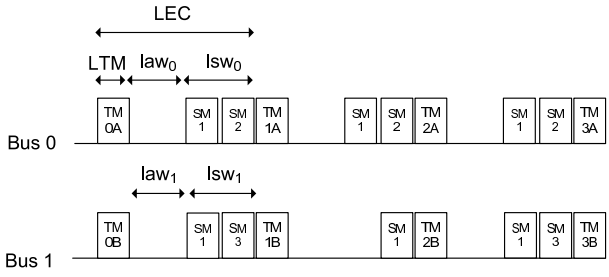

Figure 4. Bus timing with two buses

In contrast, synchronous messages 2 and 3 do not require redundancy and, thus, are not transmitted in both buses. In fact, as it can be seen in Figure 3, the proposed system also includes replicated masters, adopting a leader-follower behavior. The system only has a single active master at each time, being all the others backup masters. In case of an error in the active master, one backup master will become active since the previous active will be stopped (fail silent) . The master nodes are located at the end of the buses and the number of backup masters at one end of the buses equals the number of backup masters at the opposite end. This facilitates the bus error detection since, one Trigger Message omission can be easily detected by the master located in the opposite end of the bus. In this way, if a Trigger Message is omitted the backup master located at the opposite end of the bus will inform the active master of the error. The active master, if not crashed, could then re-schedule the traffic to the non-faulty buses.

2 Pros and Cons of Flexible Bus Media Redundancy

This section presents the arguments for and against flexible bus media redundancy in the context of FTT-CAN. A multiple bus FTT-CAN architecture inherits most of the good properties of FTT-CAN, and adds some others, namely:

- Increased bandwidth
- Increased resilience to bus failures
- Increased flexibility
- Scalability of replicated buses
- Master replication is still feasible

Despite these advantages, there are also some drawbacks and limitations:

- Increased complexity and price of the master node
- Increased complexity of the slave nodes, in some cases
- Inflexibility in terms of spacial location of the masters nodes

- The overall architecture complexity is higher

These aspects discussed in detail in the following sections.

2.1 Increased bandwidth

The widespread use of CAN networks in applications with increasingly higher bandwidth requirements, calls for innovative solutions able to provide some extra bandwidth. For example, in the automotive domain, where the *infotainment* data is currently not conveyed in the CAN network due to bandwidth restrictions. MOST [4] or FlexRay [3] are used instead. Also some CAN based networks use more than one CAN bus, but do not improve the overall available bandwidth, example of such networks are TTCAN [12] networks and also the ones based on the *Columbus Egg* idea [16].

Since the FTT-CAN is based on the CAN protocol, the bandwidth available in the CAN bus cannot be exceeded. The throughput of a CAN bus is affected by the protocol overhead and depends on the bit rate and on the number of stuff bits. The CAN overhead varies from 42% to 88% [14]. The minimum value for overhead is obtained with a frame of 8 data bytes and the higher value (88%) is obtained with a frame of 1 data byte and the maximum number of stuff bits. This means that for a bit rate of 1 Mbps, the maximum throughput available for the data transmission is 420 kBps.

Moreover, the FTT-CAN uses the Trigger Message to control purposes, increasing thus the global overhead. This overheads depends on the bit rate, the Elementary Cycle length (LEC in Figure 4) and the number of data bytes of the Trigger Message (proportional to the number of the synchronous messages in the system). The number of stuff bits of the Trigger Message also play an important role in the overhead of the Trigger Message. The overhead of the Trigger Message varies from 2.7% to 28.4% [14]. Using more than one CAN bus it is possible to partially improve this scenario, since the additional buses can be used to transmit both replicated and non-replicated data. The improvement of the available bandwidth is proportional to the number of replicated buses. In this way, the total available bandwidth in the system could double with two buses, if different message streams are transmitted in each bus.

2.2 Increased resilience to bus failures

CAN already provides some fault tolerance mechanisms. Examples of such mechanisms, at the physical level, are the use of a differential voltage and the network operation with just one wire.

Some mechanism at the data link layer are implement in order to detect and signaling errors. This mechanisms are: Cyclic Redundancy Check (CRC), to account for message corruption; Frame Check, to detect message format violations; Acknowledge errors, to allow a node to detect if it is isolate from the network; Transmission monitoring, to allow distinguishing global errors from errors in the transmitter only; and bit stuffing, to prevent synchronization loss due to several consecutive bits of the same polarity.

In the recent years, several solutions adopting star topologies instead of traditional bus topologies, were proposed. Star topologies use one CAN link per each node, and can isolate (in case of an active star) any faulty segment of the system. However, the use of star topologies goes against one of the initial design requirements of the fieldbuses: reduce the wiring harness [20].

Using a FTT-CAN architecture with more than one bus, means that, at an application level, the data can be transmitted in more than one bus, improving the resilience to bus failures.

2.3 Increased Flexibility

The two advantages presented before can be combined together, i.e., transmitting the same data in different buses can be combined with the transmission of different data in different buses. This results in an important flexibility improvment, since the master node can schedule a specific message transmission to a specific bus or to several buses, depending on the criticality of the message. Notice that the master node holds a global and centralized knowledge of the system state, thus it can easily change a message stream from a bus to another.

In case of an error in one bus, the master can schedule the messages assigned to the faulty bus to other buses. This is done online without any interference in the service provided. This means that the master has one more degree of flexibility to schedule the data in each Elementary Cycle.

Moreover there is also an improvement on the flexibility of the topology of the architecture, as it can be seen in Figure 3, where the slaves can be connected to one bus or a set of buses. This improves the flexibility of the system and also enables the use of legacy slave nodes.

2.4 Scalability of replicated buses

In FlexRay the same message can be transmitted in all available channels, or in just one channel, i.e., FlexRay also uses multiple buses, in this case only two, both to improve the bandwidth and fault tolerance. On the other hand, TTP/C allow replicated channels, but it is only possible to use two channels (channels: bus or star) [9]. This is also true for TTCAN, where the same restriction is imposed. Both in TTP/C and TTCAN, the additional buses cannot be used to improve the bandwidth of the system.

In the FTT-CAN architecture with multiple buses the number of buses is only limited by the number of CAN controllers available at the master nodes. Nowadays is possible to find microcontrollers with 6 CAN controllers [13]. This leads to an important scalability of the proposed architecture.

The flexibility of adding more buses and to use the additional bandwidth in a efficient way, together with the scalability makes FTT-CAN with multiple buses a unique

solution in terms of fieldbuses.

2.5 Master replication is still feasible

The master node is the central point of the FTT-CAN protocol and its main tasks are the scheduling of the synchronous messages and to set all the bus timing. The master node needs some modifications to deal with multiple buses, these changes are described in detail in [20].

The master node is a single point of failure, so a replication protocol for an FTT-CAN architecture with just one bus has already been presented [6]. The replication of the master node is still possible for the case of multiple buses. In case of a master node failure, the active master is replace by a backup master without any interruption of the system operation. This replacement is done online, and slave nodes do not notice any change in the global system. In [17] the replication mechanisms for the multiple bus FTT-CAN architecture are explained in detail. This mechanism can also be used to detect permanent errors on the buses.

2.6 Increased complexity and price of the master node

The master node was changed to accommodate multiple buses [20]. This modification adds extra (low) complexity to the master node architecture. Specifically, two new modules were added to the master: The bus error detection module and the multi-bus handler. It was also necessary to add extra fields to the table where the synchronous messages properties are stored (called SRT - Synchronous Requirement Table) to include the properties related to the allocation of the messages to a bus.

At the implementation level, the increase on the required RAM memory is 23%, since all the message properties are now 13 bytes long. At a first glance this value seems to be high, but it only depends on the number of messages stored in the SRT. For a typical application [10] with 16 synchronous messages, the increase will be 48 bytes only. This value is negligible regarding the available RAM of most microcontrollers. The increasing in the code size is less than 5% when comparing with the master with just one bus.

The master node must have more than one CAN controller, either built-in or external. In the first case the microcontroller will be more complex and thus it's price will increase. In the second case, one needs to add external CAN controllers, that will also increase the price of the master node. Moreover, each bus must have a bus driver, such as [15] or [11], which also will also increase the cost of the node. Notice, however, that more powerful microcontrollers with more features (possible additions CAN controllers) are expect appear on the market, thus, the pricing impact will tend to be lower.

In what concerns the power consumption, it will increase with the complexity. Nodes will have higher computational load [20] and also has more hardware components. This issue is not negligible for battery powered FTT-CAN applications, e.g. [21].

2.7 Increased complexity of the slaves nodes

The slaves nodes used on the FTT-CAN architecture with multiple buses can be the same as the ones applied to FTT-CAN with just one CAN bus. This legacy nodes can only be connected to just one bus. In contrast, if the slaves are connected to more than one bus, there is the need to implement the necessary software and hardware adjustments. This software, only needs to be able to receive and transmit messages through the additional buses, thus slave nodes only need the software drivers for the additional buses and some adjustments to the FTT code.

2.8 Inflexibility in terms of spacial location of the masters nodes

In the FTT-CAN with just one CAN bus, the master node and its replicas can be located in any part of the bus. However in FTT-CAN with more than one CAN bus a master node and its replica must be located at both ends of the buses to provide effective bus error detection [20]. This results in less flexibility in terms of spacial localization of the master nodes.

2.9 The overall architecture complexity is higher

When a system becomes more complex the probability of an error increases due to the increasing number of components (hardware and software) [5]. In FTT-CAN with multiple bus, the number of hardware components is higher and the complexity of the software, measured in lines of code, is also higher. The higher probability of errors is a price to pay to have more flexibility in the system. For this reason, the FTT-CAN with multiple buses incorporates mechanisms to detect permanent errors on the buses. In the literature there are some example of dependability assessment based on modeling [5][8][7]. This dependability analysis can be done using modeling tools such as möbius [2]. We plan to use this tool to assess if the dependability of the FTT-CAN with multiple buses is kept at an acceptable level.

3 Conclusions

This paper presented a multiple bus FTT-CAN architecture and discussed its pros and cons when compared with the single bus FTT-CAN architecture. Flexibility, which is one of the cornerstones of the Flexible Time-Triggered paradigm, is not compromised by the multiple bus FTT-CAN architecture. In fact, it is improved, since additional buses can be used to improve the bandwidth or to transmit the same data in different buses. Other advantages are the number of buses that are only dependent on hardware resources and the master node replication that is still feasible.

However some drawbacks of using multiple buses arise. The complexity of the nodes (master and slaves)

increases, but in a limited way. The architecture also imposes that master nodes should be located at both ends of the buses. Moreover, a dependability analysis using a modeling tool needs to be performed in order to evaluate the new architecture.

Acknowledgments

This work was supported by *Fundação para a Ciência e Tecnologia* under grant PRODEP 2001 - *Formação Avançada de Docentes do Ensino Superior Nº 200.019* and by ARTIST2, NoE on Embedded Systems Design, (EC-IST - IST-004527).

References

[1] L. Almeida, P. Pedreiras, and J. A. Fonseca. The FTT-CAN Protocol: Why and How. *IEEE Transactions on Industrial Electronics*, 49(6):1189–1201, December 2002.

[2] G. Clark, T. Courtney, D. Daly, D. Deavours, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. Webster. The möbius modeling tool. In *PNPM '01: Proceedings of the 9th international Workshop on Petri Nets and Performance Models (PNPM'01)*, pages 241–250, Washington, DC, USA, 2001. IEEE Computer Society.

[3] F. Consortium. FlexRay Communications System - Protocol Specification, v2.0. Technical report, FlexRay Consortium, 2004.

[4] M. Corperation. Media Oriented System Transport, Multimedia and Control Networking Technology, November 2002.

[5] F. Di Giandomenico, S. Porcarelli, D. Viva, A. Bondavalli, and P. Lollini. Model-based Evaluation for Dependability Assessment of CAUTION++ Instances. In *Venue '04 (informal proceedings)*, Athens, Greece, May 27-28 2004.

[6] J. Ferreira, P. Pedreiras, L. Almeida, and J. Fonseca. The FTT-CAN protocol: improving flexibility in safety-critical systems. *IEEE Micro (special issue on Critical Embedded Automotive Networks)*, 22(4):46–55, 2002.

[7] Q. Gan and B. Helvik. Dependability modelling and analysis of networks as taking routing and traffic into account. In *Proceedings of the* 2^{nd} *Conference on Next Generation Internet Design and Engineering*, April 2006.

[8] R. Ghostine, J.-M. Thiriet, and J.-F. Aubry. Dependability evaluation of networked control systems under transmission faults. In 6^{th} *IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, Safeprocess 2006*, Beijing - China, 2006.

[9] G. B. Hermann Kopetz. The Time-Triggered Architecture. In *Proceedings of the IEEE*, volume 91, January 2003.

[10] R. Marau, L. Almeida, J. Fonseca, J. Ferreira, and V. Silva. Assessment of FTT-CAN master replication mechanisms for safety-critical applications. In *Proceedings of the SAE 2006 World Congress & Exhibition*, 2006. Paper Number: 06AE-278.

[11] Microchip. Mcp2551 data sheet, 2003. DS21667D version.

[12] B. Müller, T. Führer, F. Hartwich, R. Hugel, and H. Weiler. Fault tolerant ttcan networks. In *Proceedings of* 8^{th} *International CAN Conference*. CAN in Automation GmbH, Oct 2002.

[13] NEC Electronics Corporation. μPD70F3430 Data Sheet, November 2005.

[14] T. Nolte, H. Hansson, C. Norström, and S. Punnekkat. Using bit-stuffing distributions in can analysis. In S. L. Iain Bate, editor, *Proceedings of the IEEE/IEE Real-Time Embedded Systems Workshop in conjunction with the 22nd IEEE Real-Time Systems Symposium (RTSS'01)*, London, UK, December 2001. Department of Computer Science, University of York.

[15] Philips Semiconductors. PCA82C250 Data Sheet, January 2000.

[16] J. Rufino, P. Veríssimo, and G. Arroz. A Columbus' egg idea for CAN media redundancy. In *Digest of Papers, The 29th International Symposium on Fault-Tolerant Computing Systems*, pages 286–293, Madison, Wisconsin, USA, June 1999. IEEE.

[17] V. Silva, J. Ferreira, and J. Fonseca. Master Replication and Bus Error Detection in FTT-CAN with Multiple Buses. In *Proceedings of the* 12^{th} *IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2007)*, Patras, Greece, 2007.

[18] V. Silva and J. Fonseca. Using FTT-CAN to Combine Redundancy with Increased Bandwidth. In *Proceedings of the 2006 IEEE International Workshop on Factory Communication Systems*, pages 54–62, June 2006.

[19] V. Silva, J. Fonseca, and J. Ferreira. Using FTT-CAN to the Flexible Control of Bus Redundancy and Bandwidth Usage. In *Proceedings of the 11th International CAN Conference iCC 2006*, pages 5.9 – 5.15, Sweden, September 2006.

[20] V. Silva, J. Fonseca, and J. Ferreira. Adapting the FTT-CAN Master for Multiple-bus Operation. In *Proceedings of the* 5^{th} *IEEE International Conference on Industrial Informatics*, Vienna, Austria, July 2007.

[21] V. Silva, R. Marau, L. Almeida, J. Ferreira, M. Calha, P. Pedreiras, and J. Fonseca. Implementing a distributed sensing and actuation system: The CAMBADA robots case study. In *Proceedings of the 10th IEEE Conference on Emerging Technologies and Factory Automation, 2005. ETFA 2005*, volume 2, pages 781–788, September 2005.