Flexible Time-Triggered Communication on a Controller Area Network

L. Almeida, J. A. Fonseca, P. Fonseca

Dep. de Electrónica e Telecomunicações
Universidade de Aveiro
P-3810 Aveiro, Portugal

Abstract

Controller Area Network is normally used in event-triggered communication systems. These are known for not supporting composability with respect to the system temporal behavior. When using a time-triggered communication paradigm, such composability is achieved. However, common time-triggered systems rely on static message scheduling which compromises system flexibility. In this paper the authors study the possibility of using centralized scheduling together with a planning scheduler on a CAN system to achieve flexible time-triggered communication. Some preliminary results seem to validate the proposal but show an important influence of the low processing capability of the microcontroller used (80C592), which strongly limits the overall bus utilization factor.

1. Introduction

Several authors have stressed the growing importance of flexibility in modern real-time systems, particularly those used in industry [1][2][3]. Only flexible real-time systems will be capable of operating in dynamic environments, i.e., those where a complete specification is not possible and where operational requirements may change during system lifetime. However, flexibility in real-time systems must be achieved without jeopardizing the understandability and predictability of the system temporal behavior allowing safe and predictable upgrades with negligible down-time. Two important properties to support such a flexibility are composability (the subsystems' properties remain valid upon integration in the whole system) and scalability (the extensibility of the system is not limited by some predefined upper limit).

In distributed real-time systems, the communication system has a central role in determining either the composability and scalability of a distributed architecture. In particular concerning composability, Kopetz [3] stresses that the event-triggered communication paradigm does not support composability with respect to the temporal behavior while, on the other hand, the time-triggered paradigm does. However, typical time-triggered communication systems use static message scheduling and this strongly limits system flexibility.

In this work-in-progress paper the authors propose the use of a CAN network [4] in an untypical centralized scheduling architecture to achieve flexible time-triggered communication. The CAN native distributed arbitration is still used to reduce the communication overhead. The central scheduler uses a planning scheduler to combine flexibility with temporal predictability and low run-time overhead. The use of the planning scheduler has already been proposed by the authors to improve flexibility in statically scheduled communication systems with centralized arbitration such as the FIP fieldbus [5][6][7]. This paper includes a few experimental results obtained on a home-made CAN system (CANivete) based on low processing power microcontrollers [8].

2. Time-triggered communication in *CAN*

CAN was developed for the automotive industry with the purpose of supporting low-level communication services between modules in distributed car control systems. The CAN specification covers, essentially, aspects related to the two lower layers of typical fieldbus communication systems, i.e., physical layer and data link layer. The initial proposal omits the application layer aspects leaving it to the application designer (note, however, that there are several proposals for the CAN application layer (e.g. [9]) or for complete profiles (e.g. [10]). This allows to build communication systems with different characteristics based on CAN.

The most common application of CAN is in event-triggered communication systems. In this case, messages are transmitted upon the occurrence of events. The communication system has no prior knowledge of the level of contention that occurs when the bus is being accessed because the exact bus access instants of each station are not known in advance (asynchronous nature of events). This leads to lack of support for composability with respect to the temporal behavior, i.e., the temporal properties of each node when considered separately may not remain valid when all nodes are connected together [3].

The problem referred above can be solved using a time-triggered communication system with autonomous control. In this case, the transmission of a message is triggered at predefined instants of time by initiative of the network controller. Global knowledge about the messages to be transmitted is required in order to avoid or limit the pernicious effects of uncontrolled bus access collisions. This solution allows to achieve composability with respect to the temporal behavior by promoting a temporal encapsulation of each node and preventing undesirable overload effects [3].

One possible way to build a time-triggered communication system based on CAN is using the paradigm of the Time-Triggered Protocol (TTP) [11][3] resulting in a TTP-like network. In this case, a TDMA cycle is used, with slots for each node to transmit its messages at the required instants in time. The TDMA slots are identified by a global notional clock maintained by a clock synchronization algorithm. The dispatching of messages is carried out according to static tables that reside in the nodes' memory. The static nature of these tables adversely affects system flexibility. In fact, most changes to these tables require to stop system operation, particularly those that involve changes in the TDMA cycle.

Another possibility is to use a centralized arbitration approach to control the access of each node to the bus. This is the paradigm of the Factory Instrumentation Protocol (FIP) [12]. In FIP-like networks messages are dispatched according to a table that resides in the master node, or arbitrator. This special node sends one control message to trigger the transmission of each data message that is to be sent by other nodes. When applied to CAN, this solution becomes rather inefficient because of its very large overhead. However, it is possible to take advantage of the CAN native distributed arbitration in order to drastically reduce this overhead.

3. Centralized scheduling in *CAN*

In FIP-like networks, the dispatching table that resides in the master's memory is, normally, static. Changes to this table also require to stop system operation. However, the authors have already proposed the use of a planning scheduler to overcome this limitation on system flexibility [5][6][7]. With the planning scheduler, the table is rebuilt every fixed duration period of time called a plan. Changes to the message set can be easily accommodated in the next plan thus improving the level of system flexibility (fig. 1).

Each plan is divided into a fixed number of constant duration elementary cycles (Ec). All time attributes of messages are expressed as integer multiples of the duration of one Ec. Consequently, within each of these cycles, each message can be transmitted only once. The duration of the Ec determines the temporal resolution of the communication system.

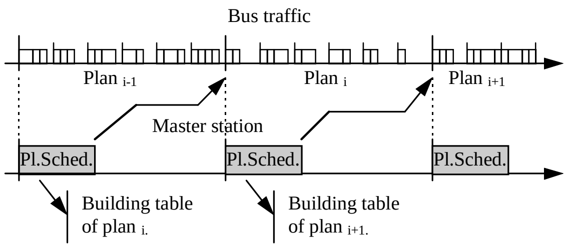

Figure 1. The planning scheduler.

Given this temporal resolution, any transmission delay within the duration of one Ec is irrelevant. Thus, the order by which messages are transmitted within one Ec is also irrelevant. When realizing centralized scheduling in CAN, the master sends a special control message, the master message, in the beginning of each Ec. This message indicates which data messages must be transmitted during that particular Ec. The order by which the data messages are transmitted is left for the CAN native distributed arbitration to sort out (fig. 2). This scheme allows to reduce the overhead of FIP-like centralized arbitration to just one extra message per Ec. It is called centralized scheduling, and not arbitration, because it is the scheduling activity that is centralized while the arbitration is still distributed.

The main advantages of such a communication system are:

- compromise between flexibility and run-time overhead achieved by the planning scheduler;
- implicit synchronization of all nodes using the regular transmission of the master message;
- simplicity of the slave nodes' network interface;
- allows messages' phase control in the scheduling table in order to improve the scheduler's performance and stability of messages periods;
- presents an early detection of missed deadlines and allows the use of mixed scheduling in order to maintain stability under transient overloads.

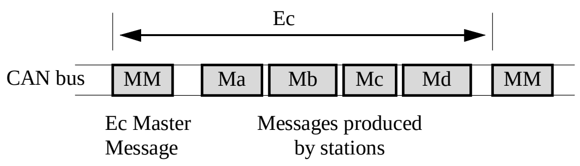

Figure 2. Messages transmitted during one Ec.

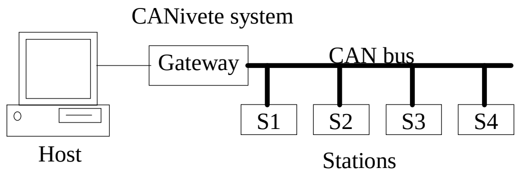

Figure 3. Experimental set-up.

4. Experimental results

An experimental setup was built using a CANivete development system with one board for the master station, two for variables' production stations and one for bus monitoring (fig. 3). The bus transmission rate was 122.9 Kbit/s with the 80C592 controllers clocked at 11.06MHz. The Ec duration was set to 8.9ms and the plan duration to 20 Ec's, i.e. 178ms. The time measurements were carried out using an 8-bit timer of the controller with a resolution of 35µs. The message set used is described in table 1. The data messages were codified in the master message by the position of a single bit. This codification limits the number of different data messages to 64 (8 data bytes) with one master message. However, if necessary, a protocol can be set up to use several consecutive master messages in each Ec to allow more than 64 different data messages.

Some preliminary measurements showed that the processing plus queuing delays of the microcontroller were larger than the transmission delays of the messages on the bus. Figure 4 shows two histograms of the messages' production delays, measured by the monitor station in one Ec, relative to the reception of the respective master message. In the top figure just one station was used to produce all the messages. The processing and queuing delays are serialized with the transmission delays resulting in relatively large values (1.77ms for the first messages). The bottom figure shows the result of using two stations to produce part of the message set each. In this case some of the processing and queuing delays of both stations are superimposed and the messages after the first are produced sooner.

In sake of flexibility, the master station does not know which messages are produced by each station and thus, when scheduling the message set, the maximum delays (as in fig. 4 - top) must be taken into account. This strongly limits the maximum bus utilization factor. For this particular implementation, the maximum utilization would be achieved with three 1-byte messages plus two 2-bytes messages at minimum period (8.9ms) resulting in a value of 31% for the data messages plus 6.8% for the master message.

The regularity of the Ec master message was measured with an oscilloscope, being accurate to 40µs (+/- 10µs). This jitter is due to variations in the master message frame size caused by CAN bit stuffing and thus cannot be improved.

The processing load on the master station caused by the execution of the scheduler and its variation when the plan duration is increased are depicted in figure 5. It can be seen that the planning scheduler demands a decreasing load from the master CPU as the plan duration is increased. However,

Table 1. Message set used in the experiments.

Messages ID	Period (# of Ec's)	DLC (bytes)	Max. n. of bits	Tx delay (µs)
0x101	1	1	63	513
0x102	2	2	73	594
0x103	3	2	73	594
0x104	4	1	63	513
0x105	5	1	63	513
0x106	6	4	92	749
0x107	7	4	92	749
0x108	8	4	92	749
0x109	9	4	92	749

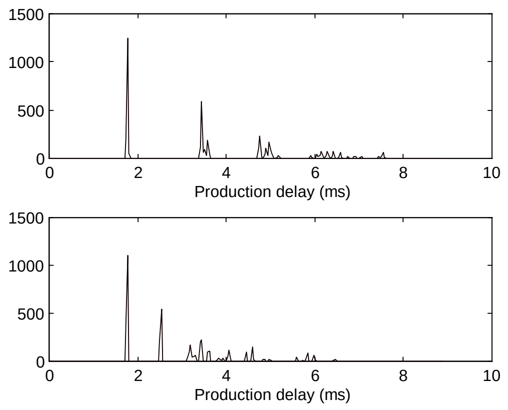

Figure 4. Histograms of data messages' production delays

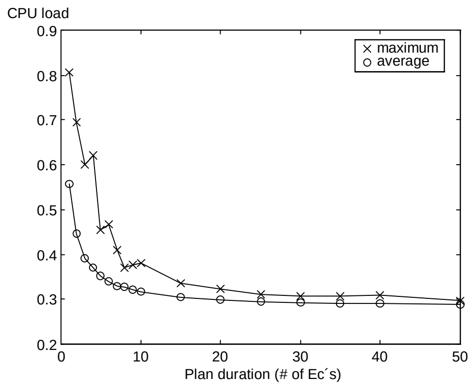

Figure 5. CPU load imposed by the execution of the planning scheduler.

any increase beyond 20 Ec's causes an irrelevant decrease in the CPU load. The dispatcher, which is invoked every Ec, demands an almost fixed load of 11%.

These values are in accordance to those already observed in previous experiments with a FIP-like network [6]. However, it has not yet been determined how well the model presented in [6] [7] fits the results shown in this paper. Also, the variation of the scheduler execution time when new messages are added to the message set is still being evaluated. For this particular implementation, using the message set described in table 1 and a plan duration of 20 Ec's results in a total maximum load at the master CPU of 44% and an average of 41%. This value still leaves some time either to use larger message sets or to execute other tasks as required by the application.

5. Conclusions and work-in-progress

The results obtained seem to validate the initial proposal of using the planning scheduler to realize flexible time-triggered communication in CAN. Using a plan duration of 20 Ec's and an Ec duration of 8.9ms it is possible to introduce changes to the message set every 178ms.

However, the use of low processing power micro-controllers strongly limits the maximum bus utilization factor due to high processing and queuing delays. Increasing the bus transmission rate allows to include more messages in each Ec but the bus utilization factor is reduced even further. A model of the maximum production delay of each message is being developed to assist the scheduler in building plans. Messages can only be scheduled within one Ec if their maximum production delay is shorter than the Ec duration. This will guarantee that the next master message will not be delayed by a late data message in the current Ec.

When the message set is distributed by several producing stations, the messages' production delays become shorter than the maximum value. The unused bus time at the end of each Ec can be seen as a sporadic server to allow the transmission of sporadic messages. The authors are working on a way to allow the coexistence of such sporadic messages without jeopardizing the timing behavior of the time-triggered communication system. This would allow to increase the overall bus utilization factor.

The variation of the scheduler execution time when messages are added to the message set is also an important issue so that the timely execution of the scheduler can be guaranteed. The authors are currently performing some experiments to determine whether the model developed in [6] and [7] can be applied to this situation.

References

[1] Sha, L.. Editorial: Industrial Computing. *IEEE Computer*, **27(1)**, 1994.

[2] Stankovic, J.A. et al.. Strategic Directions in Real-Time and Embedded Systems. *ACM Computing Surveys*, **28(4)**: 751-763, 1996.

[3] Kopetz, H.. *Real-Time Systems Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, 1997.

[4] ISO11519-2. *Road Vehicles - Low speed serial data communication - Part 2: Low speed Controller Area Network*. ISO, Int. Standard Organisation, 1994.

[5] Pasadas R., L. Almeida and J.A. Fonseca. A Proposal to Improve Flexibility in Real-Time Fieldbus Networks. *Proc. of SICICA'97 (Int. Symp. on Intelligent Components and Instruments for Control Applications)*, Annecy, France, 1997.

[6] Almeida, L., R. Pasadas and J.A. Fonseca. Using the Planning Scheduler in Real-Time Fieldbuses: Theoretical Model for Run-Time Overhead. *Proc. of WFCS'97 (IEEE Int. Workshop on Factory Communication Systems)*, Barcelona, Spain, 1997.

[7] Almeida, L. and J.A. Fonseca. The Planning Scheduler: Compromising between Operational Flexibility and Run-Time Overhead. *Proc. of INCOM'98 (IFAC Int. Symp. on Information Control in Manufacturing)*, Nancy/Metz, France, 1998.

[8] Fonseca, P. *et al.*. A Dynamically Reconfigurable CAN system. *Proc. of ICC'98 (Int. CAN Conference).* San Jose, USA, 1998.

[9] CiA/DS201:207. *CAN Application Layer (CAL).* CiA, CAN in Automation International Users Group, 1995.

[10] CENELEC. CLC/TC(SEC)146. *Low voltage switchgear and Controlgear - Part 5: Control Circuit Devices and Switching Elements - Smart Distributed Systems (SDS).* CENELEC, European Committee for Electrotechnical Standardisation, 1997.

[11] Kopetz, H. and G. Grünsteidl. TTP - A Protocol for Fault-Tolerant Real-Time Systems. *IEEE Computer*, **27(1)**, 1994.

[12] Leterrier, P.. *The FIP Protocol.* WorldFip Europe, France, 1992.