CAN-Ethernet Architectures for Real-Time Applications

Jean-Luc Scharbarg, Marc Boyer, Christian Fraboul
IRIT - ENSEEIHT
2, rue Camichel
31000 Toulouse - France
Jean-Luc.Scharbarg@enseeiht.fr

Abstract

Embedded systems have specific real-time requirements that led to the development of dedicated communication protocols. Such systems must face increasing communication needs and the evolution of switched Ethernet architecture. But moving from existing dedicated field-busses architectures to new Ethernet based architectures is not always feasible, due to industrial constraints.

In this paper, we compare different solutions for integrating existing data busses (such as CAN, which is an important standard in automotive context) on a global architecture that respects increasing bandwidth requirements. In a first step, we study classical CAN/CAN bridging strategies. In a second step, we propose CAN/Ethernet bridging strategies that respect the real time behavior of CAN End System when communicating through an Ethernet network that can be shared by (non CAN) applications.

1. Introduction

Fieldbusses , e.g. CAN, WorldFIP, Profibus [24] have been developed in the context of real-time applications (distributed computer control systems) that have specific communication requirements such as:

- bounded end-to-end transmission delays in order to guarantee respects of deadlines,
- bounded and small jitter for periodic traffic.

However, the amount of informations that are nowadays exchanged in such systems has been increasing steadily and is now reaching the limits of traditional fieldbusses, especially in terms of bandwidth [6].

Switching from dedicated field-busses to Ethernet is a classical trend in embedded systems [23] due to the wide acceptance of the Ethernet standard and its evolution towards a more predictable switched architecture.

However, successful experience with introduction of a switched Ethernet in avionic systems (AFDX, [11, 10]) is mainly due to the preservation of the applications communication model (periodic schemes) and the respect of the expected real time properties (bounded delay).

The goal of the study presented in this paper is slightly different, as the objective is to build an heterogeneous architecture obtained by interconnecting existing CAN data busses on an Ethernet backbone.

We have build a twofold study: on the one hand, we have developed a prototype with some Linux-PCs with CAN or Ethernet cards inside, on the other hand, we realized a more theoretical study which is shortly presented in this paper.

Section 2 presents CAN and Ethernet technologies and their extensions to cope more efficiently with real-time constraints. In section 3, We study classical pure CAN architectures. In section 4, we consider CAN/Ethernet architectures and propose CAN/Ethernet bridging strategies that respect the real-time behavior of CAN End Systems. Section 5 concludes the paper and presents some ideas for future works.

2. Communication technologies

We present the two communication technologies we intend to use, i.e. the Controller Area Network and Ethernet. We summarize the proposed solutions to make those technologies deterministic, especially in terms of respect of deadlines and boundaries for jitter.

2.1. CAN

The Controller Area Network (CAN) [12] is a serial communication protocol suited for networking sensors, actuators and other nodes in real-time systems. The CAN specification defines several versions of the protocols for the physical and the data link layer. In his paper, we focus on CAN 2.0 A. Several application layer protocols have been proposed.

The CAN addressing system is based on message identifier : a frame does not have a destination nor a source address. Frames are broadcasted on the bus. Stations get the frames they are interested in by a filtering process of the identifiers.

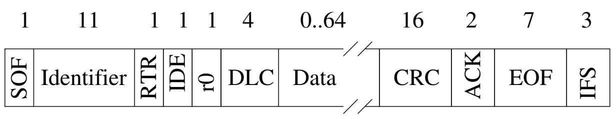

Figure 1. CAN frame (sizes in bits)

The frame format is depicted in figure 1. The detail of each field will not be presented. The relevant fields for the remaining of the paper are the following :

- the identifier field, as mentioned earlier, identifies the data contained in the frame,
- the DLC field gives the length (in bytes) of the data field,
- the data field is the payload of the frame.

Bit-stuffing is used to avoid the transmission of long sequences of bits with identical value. As soon as 5 bits of identical value are transmitted, a complementary bit is automatically inserted. This mechanism is valid for the whole frame, except IFS, EOF, ACK and the last bit of CRC.

The medium access method (MAC) is CSMA/CR : the starting of frame transmissions on the bus are synchronous. When two or more stations start a transmission simultaneously, the one with the smaller frame identifier wins and the others stop their transmission. This is implemented by a collision detection on a bit by bit basis. When a station transmits 1 (recessive bit) and detects 0 (dominant bit), it knows that a frame with a higher priority is being transmitted and, consequently, it immediately stops transmission. This mechanism guaranties strict priority order on identifiers. It implies limitations of the bandwidth and the maximal length of the bus (e.g. 1 Mbs for 40 meters).

2.2. Enhancement of the CAN protocol

Some drawbacks of the CAN native MAC have been identified. First, it is event-triggered : when a station has a frame to transmit, it tries to. It will succeed as soon as no frame with a higher priority is being transmitted. This mechanism can induce large jitter on periodic frames. Second, identifiers are associated with frames statically. This imposes a scheduling algorithm using static priorities, e.g. rate monotonic [17] when periodic traffic is considered. It is well known that higher utilization of the medium is obtained with a scheduling algorithm using dynamic priorities, e.g. Earliest Deadline First [17].

Solutions have been proposed to solve those drawbacks. Most of them add a protocol over CAN native MAC.

Time triggered CAN (TT-CAN) [13] [9] impose a static scheduling on CAN. This scheduling is memorized in a table which is known by all the stations. This scheduling comprises in particular exclusive and arbitration windows. Each exclusive window is dedicated to exactly one frame identifier while an arbitration window is shared. The stations are resynchronized with a trigger message broadcasted periodically by the master station.

Flexible TTCAN (FTT-CAN) [1] aims at introduce flexibility in the static scheduling of TTCAN. The table is memorized by the sole master. The trigger message contains the numbers of the messages that can be transmitted until the next trigger message. Moreover, time is reserved for non synchronous messages. Problems may occur if the master fails. Solutions to this problem have been proposed in [8].

Implementations of EDF scheduling relying on the native MAC have been proposed [26] [5]. They use a part of the identifier to encode the (dynamic) priority of the message, implying a limited number of different messages. In [21], a server-based method is proposed.

This paper will only consider the native CAN MAC. However, it could be of great interest to evaluate the solutions proposed in this section in our context, as will be stated later.

2.3. Ethernet

The Ethernet link layer [7] is designed for computer local networks where high bandwidth and low cost hardware is more important than guaranteed deadlines and/or jitter.

The Ethernet addressing system is based on MAC addresses: each Ethernet entity has a unique MAC address. In each frame, the destination (unicast, broadcast or multicast) and source addresses are inserted. Frames are broadcasted on the physical layer. Entities get the frames there are interested in by a filtering process.

The Ethernet medium access method is CSMA/CD: the time is divided into slots. Emission always begins at slot start. When two entities (or more) start to emit at the same time ($\pm$ the signal propagation delay), a collision is detected by the entities, they immediately stop to emit and go into a *retransmission* state (in order to be sure that all entities have detected the collision, at least 64 bytes are sent). In retransmission state, a sender draws a random integer value n uniformly in $[0, 2^c - 1]$, where c is the number of collisions already observed for this frame (bounded to 10). It waits n slots before to re-emit the frame. The number of retransmissions is bounded to 15.

The Ethernet payload can vary from 46 to 1500 bytes.

The Ethernet traffic was 10Mbs, the most common is now 100Mbs and there also is a 1Gbs solution.

The Ethernet frame format for 10Mbs and 100Mbs is described in figure 2. In an Ethernet frame, there are (at least) 26 bytes of control and 0 up to 1500 bytes of data. Padding guarantees a minimum payload of 46 bytes.

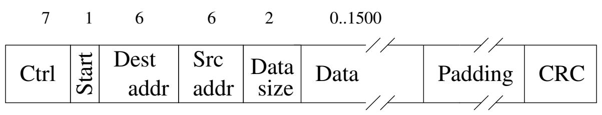

Figure 2. Ethernet frame (sizes in bytes)

2.4. Real-time Ethernet

As stated earlier, the original goal of Ethernet was to maximize the bandwidth utilization and to minimize the mean response time. Consequently, it is not well suited for real-time applications where the main challenge is to guarantee the respect of deadlines (to bound the response time) and to limit jitter. However, there has been many attempts to make Ethernet real-time [22].

A first class of approaches consist in modifying the medium access control to achieve a bounded access time to the bus e.g. [15]. The worst-case transmission delay is frequently orders of magnitude greater than the average transmission time, leading to overscaling of the system. Furthermore, it often implies a modification of the firmware which forbids the use of standard Ethernet cards.

A second class of approaches consists in adding a control layer over Ethernet, in order to bound or even eliminate collisions. Some of those techniques are based on a master/slave architecture, which generate an important overhead, due to master messages. Some others use token-passing, which often induces large jitter and/or overhead. TDMA requires a costly precise clock synchronization [25]. In the virtual time protocol [19] [20], every frame waits for a specific amount of time before being transmitted and, in case a collision occurs, a probabilistic approach is used. Parameters of this technique are hard to optimize and worst-case transmission delays are often much greater than average ones.

A time-triggered approach has been recently proposed. It is an adaptation of FTT-CAN to an Ethernet link and is called FTT-Ethernet [22]. A master transmits periodically a trigger message indicating the frames that should be transmitted until the next trigger message and the instant of transmission. It is a master/slave architecture where the master overhead is reduced. We will propose a solution inspired by FTT-Ethernet in section 4.

Switched Ethernet is a way to bypass the medium access strategy of Ethernet: each station is directly connected to an Ethernet switch with a full duplex link. Then, the medium is always free. Consequently guaranteed performances are strongly connected to policies of the switch. Several approaches have been developed. One generic approach is the Network Calculus ([3, 4]), that have been successfully applied in the AFDX network system for Airbus embedded networks [11, 10]. Another is the Response Time Analysis, [17]. Both approaches have been compared in [14].

As already mentioned, the use of switched Ethernet is out of the scope of this paper.

3. Pure CAN architectures

The real-time network architectures considered in this section are composed of CAN busses. Performance of those architectures will be illustrated with an application comprising the message set listed in table 1. It includes 148 periodic messages. The relative deadline of each message is equal to its period. The values for length and transmission time correspond to a 1 Mbs CAN bus. Let's have a look at the first line. It means there are 13 periodic CAN messages of period 4 ms. Those messages will be called M_1 messages in the following. Each occurrence of an $M1$ message contains 8 bytes of data. The length of an occurrence of the message is 135 bits. It is computed using the following formula :

Type	Nb of mes.	Per. (ms)	Data (bits)	Lg. (bits)	trans. time (ms)	bw Mbs
M_1	13	4	64	135	0.135	0.43
M_2	13	4	48	115	0.115	0.37
M_3	13	4	32	95	0.095	0.31
M_4	13	4	16	75	0.075	0.24
M_5	12	10	64	135	0.135	0.16
M_6	12	10	48	115	0.115	0.14
M_7	12	10	32	95	0.095	0.11
M_8	12	10	16	75	0.075	0.09
M_9	12	15	64	135	0.135	0.11
M_{10}	12	15	48	115	0.115	0.09
M_{11}	12	15	32	95	0.095	0.08
M_{12}	12	15	16	75	0.075	0.06

Table 1. message sets of the application

$$length = 47 + 8 \times DLC + \left\lfloor \frac{34 + 8 \times DLC}{4} \right\rfloor \quad (1)$$

47 is the number of control bits of a CAN frame, including the interframe space. $8 \times DLC$ is the number of data bits of the frame. The remainder of the formula is the maximum number of stuff bits inserted in the frame. So, it is a worst case length. The transmission time is 135 μs. The bandwidth needed by the M_1 CAN messages is 0.43 Mbs.

3.1. One shared CAN bus

This architecture considers a single CAN bus interconnecting all the stations. It is the simplest solution. It is impracticable in the following cases :

- the cumulative bandwidth needed by all the traffics exceeds the available bandwidth of the bus (bandwidth condition),
- the geographical dispatching of the stations is incompatible with the maximal length of the CAN bus, e.g. 40 meters at 1 Mbs (distance condition).

Considering a strictly periodic traffic and allocation of frame identifiers following a rate monotonic policy (the smallest the period of a message, the highest its priority), the bandwidth condition can be evaluated statically. The distance condition is independent of the traffic.

Concerning the application depicted in table 1, the cumulative bandwidth needed by all the traffics is 2.19 Mbs.

It clearly violates the bandwidth condition. Thus, the application cannot be implemented using one shared CAN bus.

3.2. Several CAN busses interconnected by bridge stations

This architecture aims at satisfying the bandwidth condition. It is a classical solution frequently used in embedded systems. An example with 4 CAN busses is depicted on figure 3. Each CAN bus includes two local stations ($S1$ and $S2$ for CAN bus 1) and two bridge stations ($S9$ and $S12$ for CAN bus 1). In this example, each CAN bus is not directly connected to all the other ones. For instance, bus 1 is not directly connected to bus 3.

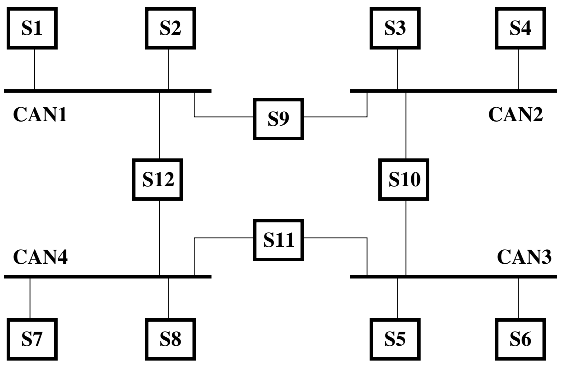

Figure 3. CAN busses interconnected by bridge stations

System operation is very simple. Different kinds of frames have to be considered :

1. frames local to a CAN bus s : they only have to be transmitted over this bus,
2. frames from a local station Sa of a CAN bus s to a local station Sb of a CAN bus d, s and d being directly connected by a bridge Sk : they have to be transmitted by Sa on bus s, received by Sk, transmitted by Sk on bus d and received by Sb,
3. frames from a local station Sa of a CAN bus s to a local station Sb of a CAN bus d, s and d being not directly connected by a bridge : they have to be transmitted by Sa on bus s, received by a bridge connected to bus s, transmitted via busses and bridges to a bridge connected to bus d, and then to station Sb.

We will suppose that the network architecture is build so that the last kind of frame never occurs. Frames of the first type are called local frames, while frames of the second type are called distant frames.

As an example, we map the example of table 1 on the network architecture of figure 3, where the number of local stations per CAN bus is not known (it is two on the figure). The messages are distributed as descibed in table 2 (3 first columns). Priorities are assigned to messages as shown in column 4 of table 2. The higher priority corresponds to the value 1. A rate monotonic policy is applied. For messages with the same period, the local ones have a lower priority. For distant messages with an identical period, priority allocation is made arbitrarly. The same applies for local messages with identical period.

Type	Nb of mes.	Kind	Priority	Per. (ms)	Worst-(case) (ms)
M1	8	Local C1	21-28	4	2.262
M1	5	C1 → C2	1-5	4	2.114
M2	8	Local C2	29-36	4	2.302
M2	5	C2 → C3	6-10	4	2.569
M3	8	Local C3	37-44	4	1.942
M3	5	C3 → C4	11-15	4	2.229
M4	8	Local C4	45-52	4	1.542
M4	5	C4 → C1	16-20	4	2.624
M5	8	Local C1	69-76	10	6.312
M5	4	C1 → C2	53-56	10	6.144
M6	8	Local C2	77-84	10	6.392
M6	4	C2 → C3	57-60	10	6.184
M7	8	Local C3	85-92	10	3.522
M7	4	C3 → C4	61-64	10	5.184
M8	8	Local C4	93-100	10	2.822
M8	4	C4 → C1	65-68	10	5.824
M9	8	Local C1	117-124	15	14.410
M9	4	C1 → C2	101-104	15	14.284
M10	8	Local C2	125-132	15	14.570
M10	4	C2 → C3	105-108	15	11.874
M11	8	Local C3	133-140	15	6.932
M11	4	C3 → C4	109-112	15	9.874
M12	8	Local C4	141-148	15	5.552
M12	4	C4 → C1	113-116	15	11.154

Table 2. Pure CAN architecture

In order to validate the system, it is necessary to guaranty that every frame of every message respects its deadline. One way to do that is to calculate a worst-case end-to-end transmission delay. Let's consider $F_{s,d,m,i}$, the ith distant frame of message M_m from a local station on bus s (source bus) to a local station on bus d (destination bus). Its worst-case end-to-end transmission delay $T_{s,d,m,i}$ is :

$$T_{s,d,m,i} = T^s_{s,d,m,i} + T_B + T^d_{s,d,m,i} \qquad (2)$$

where $T^s_{s,d,m,i}$ (resp. $T^d_{s,d,m,i}$) denotes the worst-case transmission delay of frame $F_{s,d,m,i}$ on CAN bus s (resp. d) and T_B is the maximum overhead induced by the bridge.We consider $T_B = 0.5\ ms$ (it is a reasonable value for an average microcontroller). If the frame $F_{s,d,m,i}$ becomes ready for transmission on bus s at time t_{start}, the worst-case transmission delay $T^s_{s,d,m,i}$ of $F_{s,d,m,i}$ on s is defined following the non-preemptive version of the schedulability test described in [16] [2] [18] :

$$T^s_{s,d,m,i} \quad = \quad min(t)\ with$$

$$t = \Delta_m^s + \Delta_{bloc} + \sum_{M_x \in hp(M_m,s)} \left(\left\lceil \frac{t}{P(x)} \right\rceil \times \Delta_x^s\right) \quad (3)$$

We start from $t = 0$ and iterate until two consecutive iterations produce the same value. Δ_m^s is the transmission time of one frame of message M_m on CAN bus s. Δ_{bloc} is the maximum delay induced by the frame beeing transmitted at time t_{start} (the system is non-preemptive). $\sum_{M_x \in hp(M_m,s)} (\left\lceil \frac{t}{P(x)} \right\rceil \times \Delta_x^s)$ corresponds to the transmission of all the frames with a priority not lower than $F_{s,d,m,i}$ between t_{start} and t. $hp(M_m, s)$ contains all the messages generating frames on bus s with prioriy not lower than $F_{s,d,m,i}$. $P(x)$ is the period of message M_x.

$F_{s,0,m,i}$ denotes a frame of message M_m which is local to bus s. We have

$$T_{s,0,m,i} = T^s_{s,0,m,i} \quad (4)$$

As an illustration, let's consider a frame $F_{4,1,4,i}$ from the lowest priority distant $M4$ message. We have

$$T_{4,1,4,i} = T^4_{4,1,4,i} + T_B + T^1_{4,1,4,i}$$

where

$$\begin{aligned} T^4_{4,1,4,i} &= min(t)\ with \\ t &= \Delta_4^4 + \Delta_{bloc} + \\ &\quad 5 \times (\left\lceil \tfrac{t}{P(3)} \right\rceil \times \Delta_3^4) + \\ &\quad 4 \times (\left\lceil \tfrac{t}{P(4)} \right\rceil \times \Delta_4^4) \\ \Rightarrow\ t &= 0.942\ ms \\ T_B &= 0.5\ ms \\ T^1_{4,1,4,i} &= min(t)\ with \\ t &= \Delta_4^1 + \Delta_{bloc} + \\ &\quad 5 \times (\left\lceil \tfrac{t}{P(1)} \right\rceil \times \Delta_1^1) + \\ &\quad 4 \times (\left\lceil \tfrac{t}{P(4)} \right\rceil \times \Delta_4^1) \\ \Rightarrow\ t &= 1.182\ ms \end{aligned}$$

So, we have $T_{4,1,4,i} = 2.624\ ms$.

Worst-case transmission delays for the example application are given by the last column of table 2. We observe that, for every message, the worst-case delay is smaller than the period. As deadlines equal periods, we can conclude that every frame of every message will meet its deadline.

The architecture presented in this paragraph is a good solution to satisfy the bandwidth solution. However, it is of little efficiency concerning the distance conditon. The architecture proposed in the next section aims at being an answer to this distance condition.

4. Several CAN busses interconnected by bridge stations on Ethernet

An example of such an architecture is depicted on figure 4. It includes four CAN busses and an Ethernet link

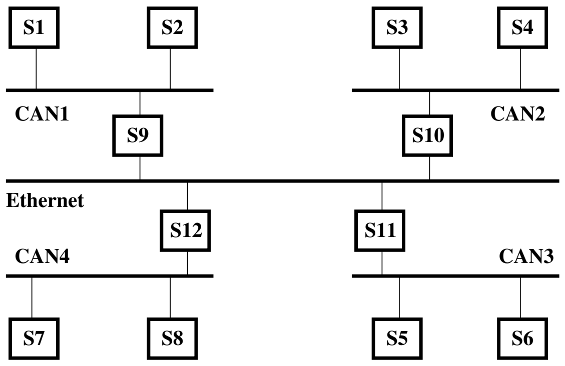

Figure 4. CAN busses interconnected by bridge stations on Ethernet

(no switched Ethernet will be considered in this section). Each CAN bus shares a bridge station with Ethernet (S9, S10, S11 and S12 on figure 4). Two kinds of frames have to be considered :

1. frames local to a CAN bus s : they only have to be transmitted over this bus,
2. frames from a local station Sa of a CAN bus s to a local station Sb of a CAN bus d : they have to be transmitted by Sa on bus s, received by the bridge associated with s, transmitted over Ethernet, received by the bridge associated with d, transmitted by this bridge on bus d and received by Sb.

Again, we consider the application depicted in tables 1 and 2.

The question we have to answer is : what bridging strategy between CAN and Ethernet ? Answering this question, we have to keep in mind that characterisitics of CAN and Ethernet are very different :

- the available bandwidth :1 Mbs or less for CAN, 10 Mbs, 100 Mbs, 1Gbs for Ethernet,
- the addressing system : identifiers associated to data for CAN, MAC addresses of stations for Ethernet,
- the data encapsulted in a frame : between 0 and 8 bytes for CAN, between 46 and 1500 bytes for Ethernet,
- the collision resolution : deterministic and non destructive for CAN, non deterministic and destructive for Ethernet.

The very different addressing systems make an encapsulation bridge the most suitable solution : CAN frames are encapsulated in Ethernet frames. More precisely, Identifier, DLC and Data fields of CAN frames are put in the Data field of Ethernet frames (the other fields of CAN frames can be easily reconstructed). This means that a CAN frame occupies at most 10 bytes of the Data field of an Ethernet frame. Consequently, if one CAN frame is encapsulated in one Ethernet frame, there is at least 36 bytes of padding. This is clearly an important waste of bandwidth.

The worst-case transmission delay of each CAN frame depends on the type of frame. For a distant frame $F_{s,d,m,i}$ from a local station on bus s to a local station on bus d, the worst-case end-to-end transmission delay $T_{s,d,m,i}$ is :

$$T_{s,d,m,i} = T^{s}_{s,d,m,i} + 2 \times T_B + T^{eth}_{s,d,m,i} + T^{d}_{s,d,m,i} \quad (5)$$

$T^{s}_{s,d,m,i}$ and $T^{d}_{s,d,m,i}$ are the transmission delays for the frame $F_{s,d,m,i}$ on bus s and d. Worst-case values are calculated in the same manner as for the previous architecture. T_B is the overhead for one bridge (there are two bridges on the way). $T^{eth}_{s,d,m,i}$ is the transmission delay on the Ethernet link for the CAN frame $F_{s,d,m,i}$ encapsulated in an Ethernet frame.

For a local frame $F_{s,0,m,i}$ of bus s, we have

$$T_{s,0,m,i} = T^{s}_{s,0,m,i} \quad (6)$$

Simulations have been made on the example application of table 1 and 2, using a queueing network modelling and QNAP2. We suppose that each CAN frame is encapsulated in a separate Ethernet frame and transmitted as soon as possible. We consider an Ethernet link at 10 Mbs (a 100 Mbs link is necessary if additional non CAN traffic is considered) and $T_B = 0.05\ ms$ (considering a modern microprocessor). The results show that nearly all distant CAN frames miss their deadline. This is a consequence of the many collisions observed on the Ethernet link. So, this encapsulation strategy is clearly a bad solution for such an application.

The first idea to overcome this problem is to put more than 1 CAN frame in one Ethernet Frame. If, for instance, we put 5 CAN frames of maximum length in one Ethernet frame, it represents 50 bytes of Ethernet Data and no padding is necessary. More formally, we encapsulate N_{ce} CAN frames in an Ethernet frame.

Simulations with QNAP2 have been made on the example application of table 1 and 2, considering an Ethernet link at 10 Mbs and $T_B = 0.05\ ms$. We have considered values of N_{ce} from 2 to 5. For $N_{ce} = 2$, nearly all distant CAN frames miss their deadline. Table 3 gives percentages of CAN frames that miss their deadlines for $N_{ce} = 3$, 4 and 5. For those three values, only a small part of distant CAN frames miss their deadlines. For this application, the optimal value of N_{ce} is 3. In fact, the value N_{ce} influence two characteristics of the system:

- the load of CAN traffic on Ethernet, which decreases as N_{ce} increases,
- the maximal duration a CAN frame has to wait before being encapsulated and transmitted on the Ethernet link, which increases as N_{ce} increases.

Type	Kind	$N_{ce} = 3$	$N_{ce} = 4$	$N_{ce} = 5$
M1	Local C1	0 %	0 %	0 %
M1	C1 $\rightarrow$ C2	6 %	8.3 %	14.8 %
M2	Local C2	0 %	0 %	0 %
M2	C2 $\rightarrow$ C3	7.5 %	10.2 %	13.1 %
M3	Local C3	0 %	0 %	0 %
M3	C3 $\rightarrow$ C4	11.7 %	11.4 %	17.4 %
M4	Local C4	0 %	0 %	0 %
M4	C4 $\rightarrow$ C1	8.3 %	8.2 %	14.3 %
M5	Local C1	0 %	0 %	0 %
M5	C1 $\rightarrow$ C2	0 %	0 %	0 %
M6	Local C2	0 %	0 %	0 %
M6	C2 $\rightarrow$ C3	0.7 %	0 %	0 %
M7	Local C3	0 %	0 %	0 %
M7	C3 $\rightarrow$ C4	1 %	0 %	0 %
M8	Local C4	0 %	0 %	0 %
M8	C4 $\rightarrow$ C1	0 %	0.5 %	0 %
M9	Local C1	0 %	0 %	0 %
M9	C1 $\rightarrow$ C2	0 %	0.4 %	1.9 %
M10	Local C2	0 %	0 %	0 %
M10	C2 $\rightarrow$ C3	0.4 %	0 %	0 %
M11	Local C3	0 %	0 %	0 %
M11	C3 $\rightarrow$ C4	0 %	0 %	0 %
M12	Local C4	0 %	0 %	0 %
M12	C4 $\rightarrow$ C1	0 %	0 %	0 %

Table 3. Missed deadlines with CAN-Ethernet architecture

For our example application, $N_{ce} = 3$ is the best compromise.

A further improvement is to associate a timer $WD_{s,d,m,i}$ with each distant CAN frame $F_{s,d,m,i}$. A bridge transmits an Ethernet frame encapsulating all pending CAN frames as soon as it has N_{ce} pending CAN frames or a pending CAN frame $F_{s,d,m,i}$ has been initiated since a duration of $WD_{s,d,m,i}$. As an example, suppose that $N_{ce} = 2$ and $WD_{s,d,m,i} = 0.5\ ms$ for all distant CAN frames $F_{s,d,m,i}$. A bridge transmits an Ethernet frame as soon as it has two pending CAN frames or one pending CAN frame initiated for more than $0.5\ ms$.

Table 4 shows simulation results for two sets of $WD_{s,d,x,i}$ values. For the first set (column hyp. 1 of the table), we consider

$$WD_{s,d,m,i} = T^{s}_{s,d,m,i} \quad (7)$$

for each distant CAN frame $F_{s,d,m,i}$. For the second set (column hyp. 2 of the table), we consider

$$\begin{aligned} WD_{s,d,m,i} &= T^{s}_{s,d,m,i} + 0.1 \times L_{s,d,m,i} \\ &\quad where \qquad (8) \\ L_{s,d,m,i} &= P(m) - (T^{s}_{s,d,m,i} + 2 \times T_B + T^{d}_{s,d,m,i}) \end{aligned}$$

Type	Kind	hyp. 1	hyp. 2
M1	Local C1	0 %	0 %
M1	C1 → C2	0.3 %	0 %
M2	Local C2	0 %	0 %
M2	C2 → C3	0 %	0.1 %
M3	Local C3	0 %	0 %
M3	C3 → C4	0 %	0.2 %
M4	Local C4	0 %	0 %
M4	C4 → C1	0 %	0 %
M5	Local C1	0 %	0 %
M5	C1 → C2	0 %	0 %
M6	Local C2	0 %	0 %
M6	C2 → C3	0 %	0 %
M7	Local C3	0 %	0 %
M7	C3 → C4	0 %	0 %
M8	Local C4	0 %	0 %
M8	C4 → C1	0 %	0 %
M9	Local C1	0 %	0 %
M9	C1 → C2	0 %	0 %
M10	Local C2	0 %	0 %
M10	C2 → C3	0 %	0 %
M11	Local C3	0 %	0 %
M11	C3 → C4	0 %	0 %
M12	Local C4	0 %	0 %
M12	C4 → C1	0 %	0 %

Table 4. Missed deadlines with CAN-Ethernet architecture

$L_{s,d,m,i}$ is the duration available for the transmission of $F_{s,d,m,i}$, considering bridges overhead and worst-case transmission delays on CAN busses. The second set aims at reducing the load on the Ethernet link, compared with the first set. For the two sets, we fix $N_{ce} = 100$. Results of simulation show that nearly all CAN frames meat their deadline, whatever set of $WD_{s,d,m,i}$ values is used. So, this solution is much better than the previous ones. We can say it is a good solution for soft real-time applications. However, it is not well-suited for hard real-time applications:

- there are still some CAN frames that miss their deadlines,
- there are still collisions on the Ethernet link and no guarantee on worst-case transmission delays can be given.

Futhermore, finding optimal values for the $WD_{s,d,m,i}$ is a tricky problem.

Actually, this solution aims at limiting the load on the Ethernet link, but it does nothing to prevent collisions. The use of a time-triggered technique is one way to eliminate collisions. The implementation of the time-triggered paradigm on Ethernet has already been proposed. The solution we suggest is inspired from FTT-Ethernet [22].

We define a master station Sm on the Ethernet link. The other stations Ss_p are the slaves. As long as Ms has not transmitted a frame on Ethernet, no other station is allowed to transmit. Each time a station Ss_p receives a frame from Ms, it is allowed to transmit a frame after a delay of WSs_p. Sm transmits Ethernet frames periodically with perod PS_m. The sequence of Ethernet frames is depicted on figure 5, considering 3 slave stations. Compared with FTT-Ethernet, our proposal is not

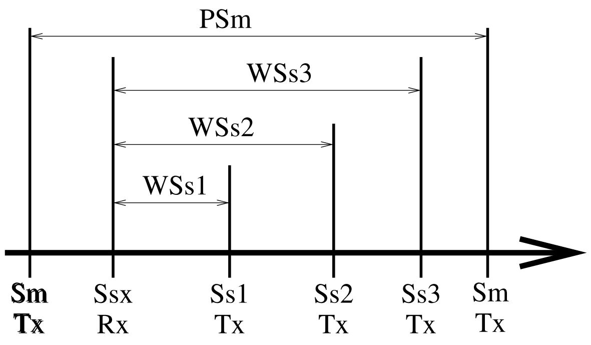

Figure 5. Time-triggered communications on Ethernet link

flexible, since the moment when slave stations are allowed to transmit is computed only from the moment they receive the master frame. There is no overhead induced by the master, since the data frame is used as trigger message.

Simulations have been done considering the architecture of figure 4 and the application of tables 1 and 2. the master station is $Ms = S9$, the slaves stations are $Ss_1 = S10$, $Ss_2 = S11$ and $Ss_3 = S12$. The period associated to the master is $Ps_m = 1\ ms$. The delays of slave stations are $WSs_1 = 0\ ms$, $WSs_2 = 0.2\ ms$ and $WSs_3 = 0.4\ ms$. Results of simulation show no missed deadlines for CAN frames and no collisions on the Ethernet link.

This solution is promising. However, further analysis should be done and a worst-case transmission delay calculation should be developped in order to guarantee that, for a given application, no CAN frame misses its deadline.

5. Conclusion and future works

In this paper, we mainly focused on two types of communication technologies :

- the first one is Controller Area Network (CAN), which is a good example of deterministic real-time communcation system,
- the second one is Ethernet, which is the most popular non real-time communication system.

The aim of the paper was to study the use of Ethernet in conjunction with CAN for communications in a real-time system. Pure CAN architectures have first been studied. They are limited in terms of available bandwidth (CAN maximum bandwidth is 1 Mbs) and area coverage (maximum length of a CAN bus at 1 Mbs is 40 meters). The use of several CAN busses directly interconned by bridges partially solve the bandwidth limitation, but is quite inefficient against the area coverage limitation.

The use of an Ethernet link to interconnect the various CAN busses is a good alternative if a solution can be found to bound the transmission delay on Ethernet. This is very difficult with pure CSMA/CD Ethernet, since we have no control on collisions.

We propose the implementation of a time-triggered solution on CSMA/CD Ethernet and show that it allows to avoid collision. So, transmission delays on Ethernet can be bounded and the respect of CAN traffic deadlines can be guaranteed. This proposal only considers periodic CAN traffic. It could be interesting to apply it in the presence of non periodic CAN traffic.

Another important characteristic of real-time communication is the limitation of the jitter of periodic traffics. In this context, it would probably be judicious to study the use of the time-triggered paradigm on CAN (via TT-CAN or FTT-CAN) in conjunction with the time-triggered Ethernet solution.

Nowadays, switched Ethernet can be used for real-time applications. There are no more collisions on the medium and guaranteed transmission delays are strongly connected to potential congestion problems that may occur in output queues of the switches. We intend to apply the time-triggered paradigm on switched Ethernet considering both the architecture of one switch and the global architecture of the network. The open question is : given their service disciplines, are switches able to give a time-triggered communication schema ?

References

[1] L. Almeida, P. Pedreiras, and J. A. G. Fonseca. The ftt-can protocol : why and how. *IEEE transactions on industrial electronics*, 49(6), dec 2002.

[2] N. Audsley, A. Burns, K. Tindell, M. Richardson, and A. Wellings. Applying a new scheduling theory to static priority preemptive scheduling. *Software engineering journal*, 5(5):284–292, 1993.

[3] R. Cruz. A calculus for network delay, part I. *IEEE Transactions on Information Theory*, 37(1):114–131, January 1991.

[4] R. Cruz. A calculus for network delay, part II. *IEEE Transactions on Information Theory*, 37(1):132–141, January 1991.

[5] M. Di Natale. Scheduling the can bus with earliest deadline techniques. In *Proceedings of the IEEE Real-Time Systems Symposium*, 2000.

[6] D. Dietrich and T. Sauter. Evolution potentials for fieldbus systems. In *IEEE Workshop on Factory Communication systems*, Porto, September 2000.

[7] CSMA/CD access method. IEEE Standard 802.3, IEEE, 2002.

[8] J. A. Fonseca, J. Ferreira, M. Calha, P. Pedreiras, and L. Almeida. Issues on task dispatching and master replication in ftt-can. In *IEEE Africon*, 2002.

[9] T. Führer, B. Müller, W. Dieterle, F. Hartwich, R. Hugel, and M. Walther. Time triggered communication on can. In *International CAN Conference*, 2000.

[10] J. Grieu. *Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systmes avioniques*. PhD thesis, Insitut National Polytechnique de Toulouse (INPT), INPT – Toulouse – France, Juin 2004.

[11] J. Grieu, F. Frances, and C. Fraboul. Preuve de déterminisme d'un réseau embarqué avionique. In *Actes du 10ème Colloque Francophone sur l'Ingenierie des Protocoles*, Paris, 7-10 Octobre 2003.

[12] ISO. *ISO International Standard 11898 - Road vehicles - Interchange of digital information - Controller Area Network for high-speed communication*, nov 1993.

[13] ISO. *ISO International Standard 11898-4 - Road vehicles - Controller Area Network - Part 4 : Time-Triggered Communication*, dec 2000.

[14] A. Koubâa and Y. Q. Song. évaluation et amélioration des bornes du temps de réponse pour des applications temps réel avec ordonnancement á priorité fixe et non-préemptif. In *Actes du 4éme Colloque Francophone sur la Modélisation des Systèmes Réactifs*, 2003.

[15] G. Le Lann and N. Rivierre. Real-time communications over broadcast networks : the csma-dcr and the dod-csma-cd protocols. Report RR1863, INRIA, 1993.

[16] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm : exact characterization and average case behavior. In *Proceedings of Real-Time Systems Symposium*, pages 166–171, dec 1989.

[17] C. Liu and L. J.W. Scheduling algorithms for multiprogramming in hard real-time environment. *Journal of ACM*, 20(1):46–61, 1973.

[18] J. W. S. Liu. *Real-Time systems*. Prentice Hall, 2000.

[19] N. Malcolm and W. Zhao. Hard real-time communications in multiple-access networks. *Real Time systems*, 9:75–107, 1995.

[20] M. Molle and L. Kleinrock. Virtual time csma : why two clocks are better than one. *IEEE Transactions on Communications*, 33(9):919–933, 1985.

[21] T. Nolte, M. Sjödin, and H. Hansson. Server-based scheduling of the can bus. In *IEEE International Conference on Emerging Technologies and Factory Automation*, 2003.

[22] P. Pedreiras, L. Almeida, and G. Paolo. The ftt-ethernet protocol : merging flexibility, timeliness and efficiency. In *Euromicro conference on real-time systems*, 2002.

[23] S. Schoenberg. Etherenet bursts the field-bus war bubble. *Industrial computing online*, April 1998. http://www.sixnetio.com/html_files/web_articles/Ethernet%20Bursts%20the%20Field-Bus%20War%20Bubble.htm.

[24] J.-P. Thomesse. Fieldbusses and interoperability. *Control Engineering Practice*, 7:81–94, 1999.

[25] C. Venkatramani and T. Chiueh. Supporting real-time traffic on ethernet. In *IEEE Real-Time Systems Symposium*, San Juan, dec 1994.

[26] K. M. Zuberi and K. G. Shin. Scheduling messages on controller area network for real-time cim applications. In *Proceedings of Real-Time Technology and Applications symposium*, 1995.