Traversal time for weakly synchronized CAN bus

Hugo Daigmorte, Marc Boyer

Traversal time for weakly synchronized CAN bus

Hugo Daigmorte
ONERA, France
2, avenue E. Belin
31055 Toulouse Cedex
hugo.daigmorte@onera.fr

Marc Boyer
ONERA, France
2, avenue E. Belin
31055 Toulouse Cedex
marc.boyer@onera.fr

ABSTRACT

Scheduling frames with offsets has been shown in the literature to be very beneficial for reducing response times in real-time networks because it allows the workload to be better spread over time and thus to reduce peaks of load. Maintaining a global synchronization amongst the stations induces substantial overhead and complexity in networks not providing a global time service such as CAN. Indeed, on CAN, no global clock is implemented in practice and each station possesses its own local clock. Without a global clock, the de-synchronization between the streams of frames created by offsets remains local to each station. The first contribution of this work is to show that important gains with respect to the communication latencies, around 40% in our experiments, can be achieved if we implement bounded clock desynchronization, also refered to as bounded phases, between the stations. The second contribution of this work is to provide a set of network-calculus based timing analyses to handle systems with bounded phases and compare their performances.

CCS Concepts

•General and reference → Performance; •Mathematics of computing → *Numerical analysis;* **•Computer systems organization →** *Real-time systems;*

Keywords

worst-case traversal times, network calculus, CAN bus

1. INTRODUCTION

Controller Area Network (CAN), a serial communication bus network, was initially developed for automotive applications. Due to the many advantages of CAN, including its high reliability and cost effectiveness, it has found application in other industries. In particular the standard ARINC 825, was developed to standardize the use of CAN in aerospace domain [2].

RTNS '16, October 19 - 21, 2016, Brest, France

ISBN 978-1-4503-4787-7/16/10...$15.00

DOI: http://dx.doi.org/10.1145/2997465.2997477

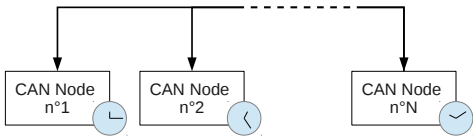

Figure 1: Example of CAN topology

It is well known that the use of offsets reduces response time [17] and increases the bus usage [5, 19], because offsets allow the workload to be spread over time and thus to reduce peak load and avoid contention and so to reduce the worst-case response times (WCRT) and to permit a better bandwidth utilization [5]. However using offsets requires a clock. In distributed systems there are two main solutions: each node having a *local clock* or all nodes sharing a *global clock*. In both, each message is sent at an offset with regard to a clock. In case of global clock each node has (up to some precision) the same clock value, and no contention occurs, neither between flows from the same node, nor from different nodes, for instance in TTCAN [11] or TTP [8]. However synchronized local clocks require synchronization mechanisms, and the clock precision must be smaller than the sending time of one frame. In case of local clocks, the scheduling remains local, synchronization mechanisms between stations are no longer a concern. Using local clocks avoids the contentions between flows from the same network, and reduces the contentions between flows from different nodes, as will be illustrated in Section 3.

Inspired by the "bandwidth management" recommendation of [2], we wondered what could be the gain of some intermediate solution: a system with a global clock but a weak precision, that can also be seen as a system with local clocks, where the phases between the clocks are bounded (see Figure 1).

Nevertheless, there was no method, up to our knowledge, to compute response time of such a system. Then, we have adapted network calculus to model an exact synchronization between flows from the same node, and bounded synchronization between flows from different nodes.

Section 2 will present an overview of the related work, in the area of offsets, synchronized networks and the associated analyze methods. Section 3 will give more details on the approach: the hypotheses, the expected benefits and the kind of systems it can model. Section 4 will recall some network calculus background and present some new properties, required in this context. Finally, numerical results on an example will be given in Section 5.

2. RELATED WORK

The timing analysis of CAN has been the object for various studies in the past. The worst case response times were first provided in [18] and then revisited in [3] but without considering offsets. They gave the exact response time for sporadic messages. In [7, 14] it has been shown that application of Network Calculus can bound the worst case response times, this bound is the exact response time on a large set of tests but there is no guaranty, these results can be applied to non-periodic flows.

Response times on CAN with offsets and local clocks have also been studied first with approximate but lower-complexity forms of analyse in [17] and then an effective worst-case response time analysis in the non-preemptive case with offsets has been given in [19]. Multi-hop systems with local clocks and offsets have also been studied using network calculus in the case of AFDX [12].

In case of global clock, each node has the same clock value, and no contention occurs, for instance: TTCAN [11], TTP [8] or TTEthernet [15]. These systems require the ability to build a global communication schedule and to synchronize local clocks.

Generating a communication schedule is NP-complete and involves non-trivial optimization algorithms, see for instance [16], but this schedule gives the response time of frames by avoiding contentions.

There are several solutions to synchronize clocks throughout a computer network. Some standard protocols exists for example the Precision Time Protocol (PTP), defined in [10]. Achieving clock synchronization can also be done with dedicated protocols, implemented in software (TTCAN level-1 [11, 6]) or requiring specific hardware support (TTEthernet [15], TTP [8], TTCAN level-2 [11, 6]...).

3. BOUNDED PHASES BETWEEN STATIONS

3.1 Model assumptions

In this paper we propose to study a network with local clocks coarsely synchronized.

We assume a CAN bus, with a set of nodes $\mathcal{N}$ and a set of periodic flow $\mathcal{F}$. Each node $n \in \mathcal{N}$ has a local clock: if t is the absolute time, $c_n(t)$ is the node clock value at this instant. For each flow $f \in \mathcal{F}$ is characterized by a sender and a tuple $(O_f, T_f, lmax_f)$ where O_f is its offset, T_f its period and $lmax_f$ its maximal frame size. The k-th instance of a flow f is generated each time the local clock $c_n(t)$ is equal to $kT_f + O_f$.

Moreover, some bound is known on the difference between local clocks from different nodes, $c_n(t) - c_{n'}(t)$, called *phase*.

If all the phases are null, then we are in the case of global clock, if they are completely unknown, then we are in the case of local clocks.

For sake of simplicity, we assumed that there is no jitter at frame sending, and we considered only periodic flow, no sporadic. These are not limitations of the analysis method, as will be explained in the conclusion.

3.2 Expected benefits

Bounded phases are a trade-off between global clock and local clock. In this section, a small example (two nodes and three flows) is presented, illustrating the benefits of the three approaches. We consider a system of two nodes. The flows

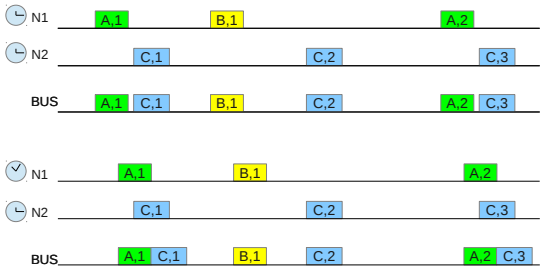

Figure 2: Impact of phasing between two nodes.

A and B are sent by the node $N1$ and the flow C by the node $N2$. The periods are such that only two occurrences of message A, only one of B and three of C are in the time window of Figure 2.

Using a global clock allows to avoid contentions and then provides small network delays. It requires building a global frame schedule such that there is no contention: such a schedule is given in Figure 2 at the top. A time slot is dedicated to each message, and no contention occurs between the flows X, Y, Z.

In case of local clocks there is no control on phases between the stations. Nevertheless offsets with respect to a local clock create some shaping and reduces contentions. These effects can be illustrated in Figure 2 at the bottom. Contentions between flows A and B from node 1 cannot happen. However contentions between flows from different node can happen: both A and B can be delayed by C. Nevertheless, local clocks create some shaping and reduces contentions between nodes: C can be delayed by at most A or B but never both of them.

Considering a network with local clocks coarsely synchronized, *i.e.* where the phases between nodes is not perfectly known but bounded, some contentions can be avoided. Like in the case of local clocks, no contention will occur between the flows A and B. But if the phase between $N1$ and $N2$ is small enough, no contention can occur between flow B and C. This shows it is possible to benefit from some of the advantages of a global clock with fewer constraints on the synchronization between nodes.

3.3 Targeted systems

Several synchronization protocols have been defined, but their implementation on a COTS can be costly, and the use of dedicated hardware is not always possible.

But even having a global reference clock can be insufficient to synchronize the bus usage: the payload of the frames is produced by some tasks, and sending a frame wrt to a clock requires that the data is produced before the frame sending time, and then gives some constraints on the task scheduling itself.

The bandwidth management requirement on ARINC 825 [2] is illustrated by the use of a cyclic scheduling where a major frame (MAF) is divided as a sequence of minor frames (MIF) and possibly some idle time at end of the MAF.

Due to hardware and software latency, it may be hard to start all MAF at exactly the same time, but it could be possible to have a bounded phase between them. In the standard examples, the MAF is 50ms long, and a MIF is 10ms long.

4. COMPUTING AN UPPER BOUND ON NETWORK DELAYS

The network calculus theory will be used to compute upper bound on the CAN bus with bounded phases. We assume that the system respects the nominal CAN behavior, and behaves as a non-preemptive static priority scheduler, where each node is always inserting its frames according to their priorities. Then, from network calculus point of view, the response time of such a system can be computed adapting existing results on non-preemptive static priority scheduler.

This section presents how it has been adapted to handle a system with offsets and bounded phases. Section 4.1 presents some reminds on network calculus. Section 4.2 gives a new expression to upper bound delays: as it will be shown in the experiments, this bound gives good results as long as the phases are small. The core of the contribution is the computation of arrival curves for aggregate flows, taking into account the offsets in a node, and the phases between nodes: this is presented in Section 4.3.

4.1 Network Calculus reminds

4.1.1 Mathematical background: (min;+) dioid

Network calculus is a theory to get deterministic upper bounds in networks. Network calculus mainly handles non decreasing functions, null before 0 : $\mathcal{F}$. It is mathematically based on the (min,+) dioid and beyond classical operations like addition or minimum, network calculus relies on two basics operators the convolution and the deconvolution.

Notation 1. The ceiling function is denoted $\lceil \cdot \rceil$, and we also introduce $[x]^+ = \max(x, 0)$.

Definition 1. (Min-plus convolution and deconvolution) The min-plus convolution $*$ and deconvolution $\oslash$ of two functions f and g are defined by

$$(f * g)(t) = \inf_{0 \le s \le t} \{f(t-s) + g(s)\}$$
$$(f \oslash g)(t) = \sup_{0 \le u} \{f(t+u) - g(u)\}$$

Property 1. ([9, Th. 3.1.12]) For any $f, f', g, g' \in \mathcal{F}$:

$$g \ge g' \Rightarrow f \oslash g \le f \oslash g' \quad (1)$$
$$f \ge f' \Rightarrow f \oslash g \ge f' \oslash g \quad (2)$$
$$(f \oslash g) \oslash g' = f \oslash (g * g') \quad (3)$$

Property 2. Let $f, g, h \in F$ three non decreasing functions, then $(f + g) * h \le (f * h) + g$.

Proof.

$$\begin{aligned}((f+g)*h)(t) &= \inf_{0\le s\le t}\{f(s)+g(s)+h(t-s)\}\\ &\le \inf_{0\le s\le t}\{f(s)+g(t)+h(t-s)\}\\ &\le (f*h)(t)+g(t)\end{aligned}$$

□

Definition 2. Let $d, P, h \in \mathbb{R}$ be some parameters. Then the functions latency δ_d and stair $\nu_{P,h}$ are defined by:

$$\delta_d(t) = \begin{cases} 0 & \text{if } t \le d \\ \infty & \text{otherwise.} \end{cases} \qquad \nu_{P,h}(t) = \left[h \left\lceil \frac{t}{P} \right\rceil\right]^+$$

Note that $(f * \delta_d)(t) = f([t-d]^+)$, $(f \oslash \delta_d)(t) = f(t+d)$.

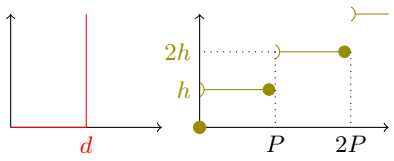

Figure 3: Latency and stair functions

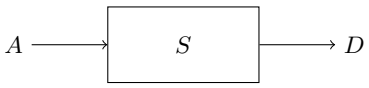

Figure 4: Server with arrival and departure flows

4.1.2 Network calculus: reality modeling

In network calculus, input and output flows of data are modeled by cumulative functions $A \in \mathcal{F}$: $A(t)$ represents the amount of data produced by the flow to time t.

The servers are just relations between some input and output flows ($S \in \mathcal{F} \times \mathcal{F}$). Then $(A, D) \in S$, means that a server S receives an arrival/input flow, $A(t)$, and delivers the data after some delay, it is the departure/output flow, $D(t)$. We have relation $D \le A$, meaning that data goes out after being entered. Figure 5 shows input and output functions for a single server queue.

The main network calculus performance measures are backlog and delay (see Figure 5).

Definition 3. (Backlog period) An interval I is a backlogged period iff the backlog (number of bits that are inside the system) is not null in this interval, *i.e.*

$$\forall t \in I, D(t) - A(t) < 0$$

Definition 4. Delay at a time t is the delay that a bit entered at time t will wait until going out. It is the horizontal deviation ($hDev$) (see Figure 5).

$$hDev(A, D, t) = \inf\{d \in \mathbb{R}^+ | A(t) \le D(t+d)\}$$
$$hDev(A, D) = \sup_{t \in \mathbb{R}^+} \{hDev(A, D, t)\}$$

However the exact input/output data flows are in general unknown at design time, or too complex, and the calculus of these bounds cannot be obtained.

4.1.3 Network calculus: contract modeling

The evolution of input/output data flows can be determined considering traffic contract on the traffics and the services in the network. For this purpose, network calculus provides the concepts of arrival curve and service curve.

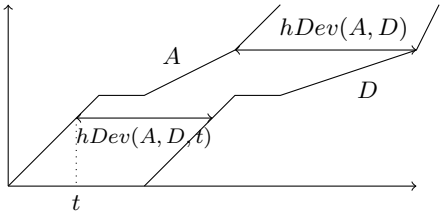

Figure 5: Backlog and delay

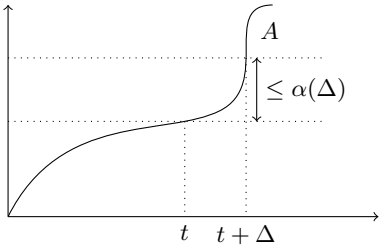

Figure 6: Arrival curve

Definition 5. (Arrival curve) Let $A \in \mathcal{F}$ be a flow, and $\alpha \in \mathcal{F}$ be a function. Then, α is said to be an arrival curve for flow A, iff

$$\forall (t,d) \in \mathbb{R}_+^2,\ A(t+d) - A(t) \leq \alpha(d) \tag{4}$$

The eq. (4) is equivalent to $A \leq A * \alpha$ [9, Lem. 1.2.3].

The contract on flow can be seen either from an engineering point of view (bound on any interval, see Figures 6) or from a mathematician point of view (convolution-based relation).

Property 3. Let A be a flow and α an arrival curve of A.

- If $\alpha' \geq \alpha$ then α' is an arrival curve of A.
- $A \oslash A$ is the "best" arrival curve for A, i.e. $A \oslash A$ is an arrival curve and $A \oslash A \leq \alpha$

It is important to notice that for a flow A, an infinite number of arrival curves exists and depending on the one which will be used to compute bounds on worst case traversal time the result may suffer a lack of accuracy. And so it is essential to have, if possible, the "best" arrival curve possible.

Property 4. Let A be a flow and α an arrival curve of A. If

$$\forall d \in \mathbb{R}_+, \exists\ t \in \mathbb{R}_+ \text{ such as } \alpha(d) = A(t+d) - A(t)$$

then α is the best arrival curve for A.

Proof. Suppose $d \in \mathbb{R}_+$ such a $\alpha(d) > A \oslash A(d)$ then there is $t \in \mathbb{R}^*$ such as $\alpha(d) = A(t+d) - A(t) > A \oslash A(d)$ which is contradictory. So $\alpha = A \oslash A$. □

For the service definition, the two points of view exist, but lead to slightly different definitions: the strict minimal service curve and the min-plus minimal service curve.

Definition 6. (Strict minimal service) A server S offers a strict service β iff for all input/output A,D and for all backlogged period (s,t]

$$D(t) - D(s) \geq \beta(t-s)$$

Definition 7. (Min-plus minimal service curve) A server S offers a min-plus minimal service curve β iff for all input/output A,D

$$D \geq A * \beta$$

These two notions of service curve are not equivalent but are both of interest and are related: a server offering a minimal strict service β also offers the minimal min-plus service β.

Let us now present the main network calculus results which allows, considering contracts, to compute bounds on delay.

Theorem 1. *(Delay bound) Let S be a server offering a min-plus minimal service curve β. If the input flow A has an arrival curve α, then, the delay can be bounded by*

$$hDev(A,D) \leq hDev(\alpha,\beta) \quad hDev(A,D) \leq (\beta - \alpha)^{-1}(0+)$$

where f^{-1} is the pseudo-inverse of the function f.

The bound $hDev(\alpha,\beta)$ is the most commonly used bound, and $(\beta-\alpha)^{-1}(0+)$ an upper bound on the busy period, in general greater than the other bound: cf. [9] at Section 1.6.2, 3.1 and 6.

Theorem 2. *(Non-preemptive static priority, [1]) Let S be a server offering a strict minimal service β, shared by three flows, A, A_H, A_L, A_H having a higher priority than A, and A_L a lower. Then, if α_H is an arrival curve for A_H and $L_L^{\max}$ is an upper bound on the frame size of A_L, the flow A receives a residual min-plus service of curve $[\beta - \alpha_H - L_L^{\max}]^+$.*

Corollary 1. *Let S be a server offering an aggregate minimal strict service of curve β to a set of flows $A_1, \ldots, A_n$, with a non preemptive static priority scheduling policy (taking the natural order as priority order, A_1 having the highest priority, A_n the lowest). For any A_i, if $\alpha_{1..i-1}$ (resp. $\alpha_{1..i}$) is an arrival curve for the aggregate flow $\sum_{k=1}^{i-1} A_k$ (resp. $\sum_{k=1}^{i} A_k$), then two upper bounds on the delay of the flow i are:*

$$hDev(\alpha_i, \beta - \alpha_{1..i-1} - l_{NP}^{\max}) \tag{5a}$$

$$(\beta - \alpha_{1..i} - l_{NP}^{\max})^{-1}(0+) \tag{5b}$$

where $l_{NP}^{\max}$ represents the maximal size of a frame of low priority.

The corollary is just an application of Theorem 1 and 2. One contribution of this article is the computation of an accurate $\alpha_{1..i}$ function for systems with phases.

4.2 A new bound on delay

The common bounds used in network calculus involve the arrival curves of the flows. But when the data flows are well known, when the system is deterministic, or quite deterministic, considering the real flow can lead to better results.

The Theorem 3 gives a new expression of the delay, based on the real arrival curves, A_i. In the model considered, the exact behaviour is unknown, but it is possible to bound the difference between the ideal systems and the real system. Then, the corollary 2 generalises it by considering bounds on phases, and Property 5 generalises it to variable frame sizes.

Theorem 3. *Let S be a non-preemptive SP n-server taking the natural order as priority order, offering an aggregate minimal strict service of curve β. Then an upper bound on the delay of the flow i is:*

$$hDev\left(A_i, (\sum_{j\leq i} A_j) * [\beta - l_{NP}^{\max}]^+ - \sum_{j<i} A_j\right) \tag{6}$$

Proof.

$$\begin{aligned} D_i(t) &= \sum_{j\leq i} D_j - \sum_{j<i} D_j \\ &\geq \sum_{j\leq i} D_j - \sum_{j<i} A_j \\ &\geq (\sum_{j\leq i} A_j) * [\beta - l_{NP}^{\max}]^+ - \sum_{j<i} A_j \end{aligned} \tag{7}$$

Moreover, given f, g, g', three flows. If $g \geq g'$ then

$$hDev(f, g) \leq hDev(f, g')$$

And so we can bound $hDev(A_i, D_i)$. □

Theorem 3, contrary to property 1, bounds the traversal time without using arrival curve and therefore avoids an approximation but it requires to know the exact behavior of the flows.

If phases are unknown but can be bounded, it can be adapted to:

Corollary 2.

$$hDev\left(A_i, (\sum_{j\leq i} A_j \oslash \delta_{\phi(i,j)}) * [\beta - l_{NP}^{\max}]^+ - \sum_{j<i} A_j * \delta_{\phi(i,j)}\right) \tag{8}$$

Where $\phi(i,j)$ is the phases between node which sends A_i and node which sends A_j.

Proof. We will use the clock of the node which sends A_i as reference. Considering a flow A_j we can bound it by: $A_j \oslash \delta_{\phi(i,j)} \leq A_j \leq A_j * \delta_{\phi(i,j)}$ So we can deduce:

$$\begin{aligned} D_i &\geq (\sum_{j\leq i} A_j) * [\beta - l_{NP}^{\max}]^+ - \sum_{j<i} A_j \\ &\geq (\sum_{j\leq i} A_j \oslash \delta_{\phi(i,j)}) * [\beta - l_{NP}^{\max}]^+ - \sum_{j<i} A_j * \delta_{\phi(i,j)} \end{aligned}$$

And then bound $hDev(A_i, D_i)$ in the case of phases. □

Each periodic flow is also characterized by its maximal size of frame, but the real size can vary. These variations can be due to different payload or to bit stuffing (bit stuffing is the insertion of non information bits into data and it can increase by 25% at most the packet length).

Property 5. Let $\overline{A} \geq A$, Theorem 3 can be extended to:

$$hDev(A_i, D_i) \leq hDev(\overline{A_i}, (\sum_{j\leq i} \overline{A_j}) * [\beta - l_{NP}^{\max}]^+ - \sum_{j<i} \overline{A_j}) \tag{9}$$

Proof. First we will only consider that A_i is known but that A_j can only be bounded if $j < i$.

$$(A_i + \sum_{j<i} \overline{A_j}) * [\beta - l_{NP}^{\max}]^+ - \sum_{j<i} \overline{A_j} \tag{10}$$

$$= (A_i + \sum_{j<i} (\overline{A_j} - A_j + A_j)) * [\beta - l_{NP}^{\max}]^+ - \sum_{j<i} \overline{A_j} \tag{11}$$

and using property 2,

$$\leq (\sum_{j\leq i} A_j) * [\beta - l_{NP}^{\max}]^+ + \sum_{j<i} (\overline{A_j} - A_j) - \sum_{j<i} \overline{A_j} \tag{12}$$

$$\leq (\sum_{j\leq i} A_j) * [\beta - l_{NP}^{\max}]^+ - \sum_{j<i} A_j \tag{13}$$

$$\leq D_i$$

Then, D_i is smaller than the expression of eq. (10), leading to

$$hDev(\overline{A_i}, \overline{D_i}) \leq hDev(\overline{A_i}, (\overline{A_i} + \sum_{j<i} \overline{A_j}) * [\beta - l_{NP}^{\max}]^+ - \sum_{j<i} \overline{A_j})$$

Let us now decompose the i-th flow: $\overline{A_i} = A_i + \widetilde{A_i}$ and $\overline{D_i} = D_i + \widetilde{D_i}$, A_i and $\widetilde{A_i}$ have the same priority. For flows of same priority the server apply a FIFO policy. Using the result of [9]:

$$hDev(A_i, D_i) \leq hDev(\overline{A_i}, \overline{D_i})$$

We can deduce:

$$hDev(A_i, D_i) \leq hDev(\overline{A_i}, (\sum_{j\leq i} \overline{A_j}) * [\beta - l_{NP}^{\max}]^+ - \sum_{j<i} \overline{A_j})$$

□

4.3 Accurate arrival curve for flows with phases

Notation 2. Let $A_1, .., A_n$ be a set of flows. In the following, $\alpha_{1..i}$ will denote an arrival curve for the aggregate flow $\sum_{j=1}^{i} A_j$.

Notice that $\sum_{j=1}^{i} \alpha_j$ is one particular arrival curve of the aggregate flow, but the core of this paper will be to find better arrival curve, modelling offsets and phases between flows.

To do so, we first simplify the problem by considering a common global hyper-period, *i.e.* decomposing each flow as a sum of sub-flow with the same hyper-period and specific offsets.

Lemma 1. *(Common period) Let A and B two periodic flows, such as $A = \nu_{T_A,h_A} * \delta_{O_A}$ and $B = \nu_{T_B,h_B} * \delta_{O_B}$ with $T_A, T_B \in \mathbb{N}$. Then we can define T_{AB} and $(n_A, n_B) \in \mathbb{N}$ such as $T_{AB} = n_A T_A = n_B T_B$. Now A (resp. B) can be express as: $A = \sum_{i=1}^{n_A} \nu_{T,h_A} * \delta_{O_A+(i-1)T_A} = \sum_{i=1}^{n_A} A_i$. And so we can define our functions as the sum of T_{AB}-periodic functions.*

Obviously Lemma 1 can be extend to more than two flows.

4.3.1 Flow with known offset difference

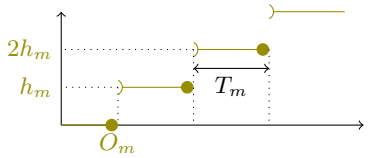

Figure 7: Example of flow M.

We have seen with Lemma 1 that we can consider an unique common period for different flows so we will consider that all the messages are sent periodically every T seconds. First let us suppose that the synchronization is perfect, for each message m its offset O_m is known (we can consider that $O_m < T$), meaning that the message m will be sent on predefined instants: the first time at time O_m and then every T. We will also consider that at this point we are working on the maximal frame size. So the message m can be associated with the flow M (see Figure 7):

$$M = \nu_{T,h_m} * \delta_{O_m}$$

Definition 8. Let $O_1, \ldots, O_n \in [0, T[$ such as $0 \leq O_1 \leq \ldots \leq O_n < T$ and $O_{n+1} = O_1 + T$ then we define for all $t \in [0, T[$: $next(t) = \min_{1\leq i\leq n+1} \{i \mid O_i \geq t\}$. $next(t)$ is the next flow that will send a message after time t.

Property 6. Let $A_1, \ldots, A_n$ a set of periodic flows, such as $A_i = \nu_{T,h_i} * \delta_{O_i}$ with $\forall (i,j) \in \{1,n\}$, $i \leq j \Rightarrow 0 \leq O_i \leq O_j < T$. We define O_{n+1} such as $O_{n+1} = O_1 + T$.

$$\forall j \leq n, \forall t \in [0,T[, A_j(t) = A_j(O_{next(t)}) \tag{14}$$

PROOF. Let $t \in [0,T[$,

$$\begin{cases} j < next(t) \Rightarrow O_j + T \geq O_{next} \geq t > O_j \\ j \geq next(t) \Rightarrow t \leq O_{next(t)} \leq O_j \end{cases}$$
$$\Rightarrow \begin{cases} A_j(t) = A_j(O_j + T) = A(O_{next(t)}) \\ A_j(t) = A_j(O_j) = A(O_{next(t)}) \end{cases}$$

□

THEOREM 4. *(Periodic flow with offset) Let $A_1, \ldots, A_n$ a set of periodic flows, $A = \sum_{i=1}^{n} A_i$ the aggregate flow, such as $A_i = \nu_{T,h_i} * \delta_{O_i}$ with $\forall (i,j) \in \{1,n\}$, $i \leq j \Rightarrow 0 \leq O_i \leq O_j < T$. We define O_{n+1} such as $O_{n+1} = O_1 + T$. Then α is an arrival curve of A with:*

$$\alpha(d) = \max_{1 \leq i \leq n} (A(O_i + d) - A(O_i)) \tag{15}$$

And α is the best arrival curve of A.

PROOF. Let $t \in \mathbb{R}^*$ such as $t = kT + t_0$ with $0 \leq t_0 < T$ and $k \in \mathbb{N}$ and $d \in \mathbb{R}^*$, then

$$\begin{aligned} A(t+d) - A(t) &= A(t_0 + d) - A(t_0) \\ &= A(t_0 + d) - A(O_{next(t_0)}) \\ &\leq A(O_{next(t_0)} + d) - A(O_{next(t_0)}) \\ &\leq \max_{1 \leq i \leq n} (A(O_i + d) - A(O_i)) \end{aligned}$$

So $\max_i(A(O_i + d) - A(O_i))$ is an arrival curve of flow A. Using property 4 we can deduce that it is the best arrival curve. □

Property 7. Let $A_1, \ldots, A_n$ a set of periodic flows, such as $A_i = \nu_{T,h_i} * \delta_{O_i}$ with $\forall (i,j) \in \{1,n\}$, $i \leq j \Rightarrow 0 \leq O_i \leq O_j < T$.

$$\max_i(A(O_i + d) - A(O_i))$$
$$= \max_i \left(\sum_{j=1}^{n} \nu_{T,h_j} * \delta_{O_j - O_i + \left\lceil \frac{O_i - O_j}{T} \right\rceil T}(d) \right) \tag{16}$$

PROOF. Let $i \leq n$, $A(O_i + d) - A(O_i) = \sum_{j=1}^{n} A_j(O_i + d) - A_j(O_i)$

- $j < i : A_j(O_i + d) - A_j(O_i) = h_j \left\lceil \frac{d + (O_i - O_j)}{T} \right\rceil - h_j$
  $= \nu_{T,h_j} * \delta_{O_j - O_i}(d) - h_j = \nu_{T,h_j} * \delta_{O_j - O_i + \left\lceil \frac{O_i - O_j}{T} \right\rceil T}(d)$
  Because $0 \leq O_j \leq O_i < T \Rightarrow \left\lceil \frac{O_i - O_j}{T} \right\rceil T = T$
- $j \geq i : A_j(O_i + d) - A_j(O_i) = h_j \left\lceil \frac{d + (O_i - O_j)}{T} \right\rceil$
  $= \nu_{T,h_j} * \delta_{O_j - O_i}(d) = \nu_{T,h_j} * \delta_{O_j - O_i + \left\lceil \frac{O_i - O_j}{T} \right\rceil T}(d)$
  Because $0 \leq O_i \leq O_j < T \Rightarrow \left\lceil \frac{O_i - O_j}{T} \right\rceil T = 0$

We can deduce that:
$\sum_{j=1}^{n} A_j(O_i + d) - A_j(O_i) = \sum_{j=1}^{n} \nu_{T,h_j} * \delta_{O_j - O_i + \left\lceil \frac{O_i - O_j}{T} \right\rceil T}(d)$ □

This last property gives us the arrival curve for a set a flows considering that the synchronization is perfect.

4.3.2 *Flows with bounded offset difference*

We will now consider that the synchronization is not perfect, there is a phase between each calculator's clock and the global time. However this phase can be bounded, we can consider the offset difference between all the messages.

THEOREM 5. *(Periodic flow with bounded offset difference) Let $A_1, \ldots, A_n$ a set of periodic flows, $A = \sum_{i=1}^{n} A_i$ the aggregate flow, such as $A_i = \nu_{T,h_i} * \delta_{O_i}$ with $\forall (i,j) \in \{1,n\}$,*

$$-T < \Phi_{\underline{ij}} \leq O_i - O_j \leq \Phi_{\overline{ij}} < T \tag{17}$$

Then α is an arrival curve of A with:

$$\alpha = \max_i \left(\sum_{j=1}^{n} \nu_{T,h_i} * \delta_{\max(0, -\Phi_{\overline{ij}} + \left\lceil \frac{\Phi_{\underline{ij}}}{T} \right\rceil T)} \right) \tag{18}$$

PROOF. Theorem 4, have to be adapted since O_i and O_j are unknown. By definition of $\Phi_{\overline{ij}}, \Phi_{\underline{ij}}$, $O_j - O_i + \left\lceil \frac{O_i - O_j}{T} \right\rceil T \geq -\Phi_{\overline{ij}} + \left\lceil \frac{\Phi_{\underline{ij}}}{T} \right\rceil T$. Moreover, the ceiling function ensures $\left\lceil \frac{x}{T} \right\rceil \geq \frac{x}{T}$, to $-x + \left\lceil \frac{x}{T} \right\rceil T \geq 0$, leading to

$$O_j - O_i + \left\lceil \frac{O_i - O_j}{T} \right\rceil T \geq \max(0, -\Phi_{\overline{ij}} + \left\lceil \frac{\Phi_{\underline{ij}}}{T} \right\rceil T)$$

Then we deduce that:

$$\delta_{O_j - O_i + \left\lceil \frac{O_i - O_j}{T} \right\rceil T} \leq \delta_{\max(0, -\Phi_{\overline{ij}} + \left\lceil \frac{\Phi_{\underline{ij}}}{T} \right\rceil T)}$$

And so to conclude:

$$\alpha(d) = \max_i \left(\sum_{j=1}^{n} \nu_{T,h_j} * \delta_{O_j - O_i + \left\lceil \frac{O_i - O_j}{T} \right\rceil T}(d) \right)$$
$$\leq \max_i \left(\sum_{j=1}^{n} \nu_{T,h_i} * \delta_{\max(0, -\Phi_{\overline{ij}} + \left\lceil \frac{\Phi_{\underline{ij}}}{T} \right\rceil T)}(d) \right)$$

We have found a function bigger than an arrival curve so it is also an arrival curve. □

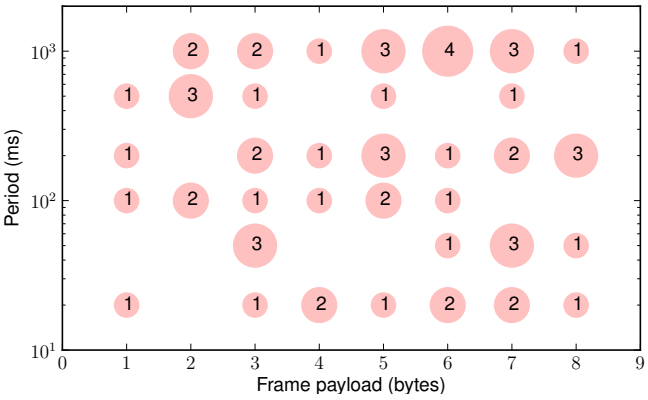

Figure 8: Distribution of the size of the data payloads and periods for the tested configuration.

5. EXPERIMENTAL RESULTS

We evaluate in this section the actual gain that can be expected using frame offsets in a network where local clocks

have bounded phases. We consider a CAN network where the phases between nodes successively amounts to 0ms, 1ms, 5ms and finally 10ms, and compare the bounds on the frame communication latencies obtained with different timing analyses.

5.1 Configuration under study

The CAN network under study runs at 250kbit/s and is made of 10 nodes. The periods of the frames are uniformly chosen from the set $\{20, 50, 100, 200, 500, 1000\}$ms and the data payloads sizes are drawn at random according to a uniform distribution between 1 and 8 bytes. Figure 8 shows for the set of frames the number of flows per period and payload, as indicated by the size of circle and the number inside. The total load on the network is 35%, and there is no difference between stations, each of them generating similar loads. The frame offset assignment that is used in this study is the SOPA algorithm available in the RTaW-Pegase software from the company RTaW, whose results have been used as a benchmark to validate the analysis developed in this paper and the gain obtained with bounded phases. The SOPA algorithm has been chosen because our experience is that it consistently outperforms the few other offset algorithms available [5, 4], and thus provides us with an estimate of the best possible gains that can be achieved in practice with offsets.

5.2 Methodology and timing analyses

Hereafter we evaluate the impact of the phases between the nodes by comparing bounded phases, ideal global clock, and no time synchronization at all between nodes, like usually on CAN. This will be performed by gradually increasing the upper bound of the phases between the nodes. To gain a better understanding, we consider that the bound on the phases is identical for all nodes and symmetrical. For example, a phase of 1ms means that the clock of a station can be in advance, or late, by at most 1ms compared to any other station in the network. In other words from the point of view of a given node, the other nodes can send their frames over a 2ms window that is centered on the expected date.

There are several ways to bound the communication latencies in systems with bounded phases, all of which making use of results that have been introduced before:

- Method 1 (**M.1**): Use of residual service (5a), and calculations of arrival curves with phases (Theorem 5).
- Method 2 (**M.2**): Bound by size of busy window (5b), and again calculations of arrival curves with phases (Theorem 5).
- Method 3 (**M.3**): New method to bound delay without using arrival curves, relying on Property 5.

We then include on the comparison results obtained with local and global clocks. For local clocks:

- Method 4 (**M.4**): Use of the algorithm published in [19] that is available in RTaW-Pegase. This algorithm is exact (except for an overestimation of the blocking factor of CAN frame) and takes into account frame offsets but consider that the phases between nodes are unknown.

For global clocks:

- Method 5 (**M.5**): From the network point of view, a single node or several with perfectly synchronized clocks leads to the exact same workload submitted to the network. The same algorithm as in method 4 is thus used, but this time with a single node on the bus.

And in order to be able to evaluate the gain brought by offsets:

- Method 6 (**M.6**): We compute the exact response time, presented in [3], that does not consider offsets.
- Method 7 (**M.7**): Network calculus without offsets ($\alpha_{1..i-1} = \sum_{k=1}^{i-1} \alpha_k$) with (5a) is used.

5.3 Experimental results

For each experiment, the results obtained with the different methods that are compared are reported in a figure and a table. The table contains the average and maximal delay bounds obtained with the different methods with the flows grouped by priority ranges.

Global clock (null phases).

Let us start by the global clock configuration, *i.e.* null phases between stations, method 5 in Figure 9. This configuration allows to evaluate the extent to which bounded phases degrade the performances in terms of communication latencies. Table 1 summarizes the results with the different methods.

When phases are null, (**M.3**) is as good as (**M.5**).

The results with (**M.1**)(**M.2**) are slightly greater than with (**M.5**) but still close: the difference between the average delay bound with (**M.2**) and (**M.5**) is less than 0.5ms (cf. Table 1), which is the transmission time of a frame with a payload of 8 bytes on a 250kbit/s bus.

This experiment also shows the gain with global clock (methods **M.1,2,3,5**) with regard to local clocks (**M.4**). As expected from other results in the literature [5], the latter is also much better than without any synchronization at all, *i.e.* methods (**M.6**) and (**M.7**). Without offsets, the latency increases almost linearly. The discontinuity at 20ms can be simply explain: it is the shortest transmission period, and from this point on frames can be delayed by more than one frame instance by higher priority flow.

Lastly, we note that network calculus without offset (**M.7**) leads to the exact same response times as with the analysis in [3] (**M.6**). In the following we will refer to these methods as (**M.6,7**).

Bounded phases of maximum 1ms.

In the next experiment the system configuration stays the same, the only difference is that we suppose that now our phases are unknown but bounded by 1ms. The results for this configuration are shown in Figure 10. This choice of phases bound has been done because TTCAN level-1 have a precision in the order of 1 ms [13]. In these setup, the analysis for global clock (**M.5**) cannot be used anymore. First consider (**M.3**): for high priority messages (0-35), it is still the method which provides the minimal bounds, however for flows of lower priorities it returns a bound of variable accuracy. Sometimes this is the best we have, but sometimes it is even worst than when offsets are not used (**M.6,7**). In the tables, in order to avoid extreme values which have

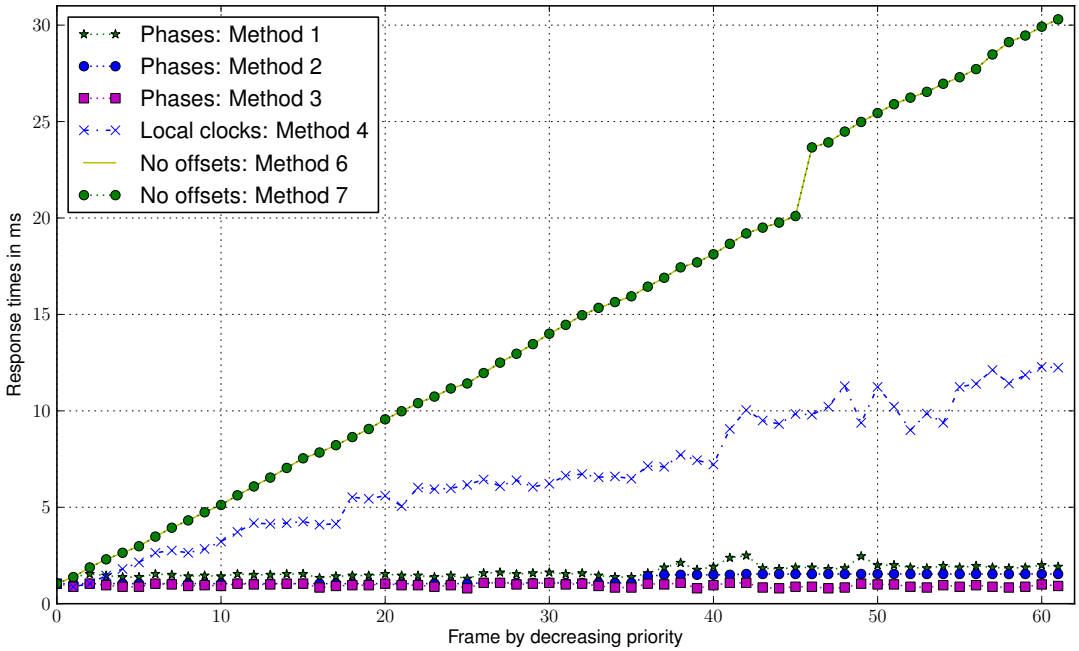

Figure 9: Delay bounds with null phases.

no interest, we consider the minimum between (**M.3**) and (**M.6,7**).

Then let us consider (**M.1**) and (**M.2**): this phase of 1ms increases the delay bound by an average of 0.5ms compared to a global clock. It should be noted that these bounds are still well below those of (**M.4**), which is the exact offset analysis without accounting for inter-node synchronization.

Bounded phases of maximum 5ms.

In the next experiment we increase significantly our bound on phases to 5ms (see Figure 11 for the results obtained with this configuration). First let us consider (**M.3**): the previous observation is confirmed, for very high priority frames (0-12), this is still the method which provides the most accurate bound, however for lower priority frames, the bounds returned are unusable.

Then consider (**M.1**) and (**M.2**): compared to the bounds given by (**M.4**), accouting for bounded phases permits to reduce the bound by almost 40%, which is very significant for many real-time distributed applications.

Bounded phases of maximum 10ms.

In the last experiment presented in this paper the scheduling still remains the same, but now phases are bounded by 10ms (see Figure 12 for the results obtained with this configuration). We remind that phases bounded by more or less 10ms implies a transmission window of length 20ms. The smallest period of emission of our network is 20ms, meaning that offsets have no more influence on flows with the smallest period.

It is confirmed that (**M.3**) should not be used for too large phases, even if it still gives the best result for 8 highest priority messages.

This experiment shows that even if (**M.2**) very often return better results than (**M.1**), it is not always the case. Moreover, if these methods most of the time give better

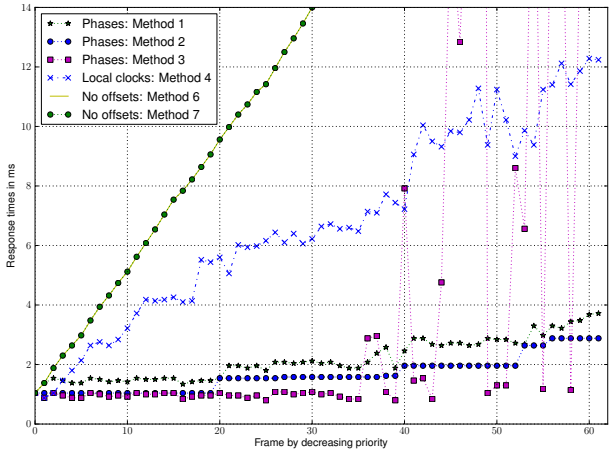

Figure 10: Delay with phases bounded by 1ms.

results than considering fully unsynchronized nodes. However, especially for the lower priority flows, sometimes the bounds calculated using these functions is larger than the one of (**M.4**).

6. CONCLUSIONS AND DISCUSSION

It has been largely evidenced in the litterature that offsets, by reducing contentions, are an efficient way to improve the response times of tasks and messages in periodic real-time systems. In distributed systems one distinguish two main approaches: either the global clock paradigm, leading to a globally time-triggered system, and requiring synchronization mechanisms with a precision much smaller than the frame sending time, or resorting to local clocks only and thus without the requirement of inter-node synchronization.

In this paper, we investigate a trade-off: local clocks with

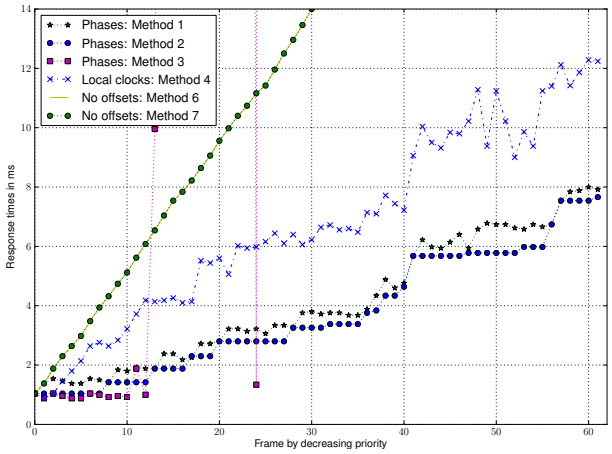

Figure 11: Delay bounds with phases $\leq$ 5ms.

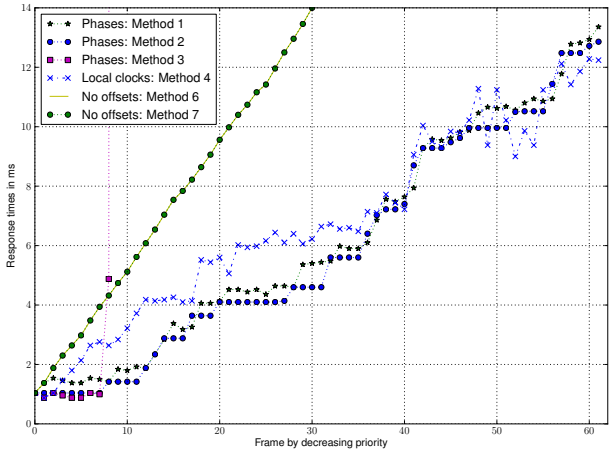

Figure 12: Delay bounds with phases $\leq$ 10ms.

Table 1: Delay bounds when phases are null.

Priorities	**2-16**	**17-31**	**32-46**	**47-61**	**2-61**
Average delay bounds (in ms)					
Phases: **M.1**	1.47	1.49	1.82	1.94	1.68
Phases: **M.2**	1.04	1.05	1.40	1.54	1.25
Phases: **M.3**	0.97	0.98	0.94	0.91	0.95
Local clocks: **M.4**	3.00	5.84	8.04	10.88	6.94
Global clock: **M.5**	0.97	0.98	0.94	0.91	0.95
No offsets: **M.6,7**	4.80	11.23	17.96	27.11	15.27
Maximum delay bounds (in ms)					
Phases: **M.1**	1.54	1.62	2.50	2.46	2.50
Phases: **M.2**	1.04	1.08	1.54	1.54	1.54
Phases: **M.3**	1.04	1.08	1.08	1.04	1.08
Local clocks: **M.4**	4.26	6.64	10.04	12.28	12.28
Global clock: **M.5**	1.04	1.08	1.08	1.04	1.08
No offsets: **M.6,7**	7.84	14.46	23.66	30.30	30.30

Table 2: Delay bounds with phases $\leq$ 1ms.

Priorities	**2-16**	**17-31**	**32-46**	**47-61**	**2-61**
Average delay bounds (in ms)					
Phases: **M.1**	1.47	1.85	2.38	3.09	2.20
Phases: **M.2**	1.04	1.45	1.76	2.46	1.68
Phases: min(**M.3**,**M.7**)	0.97	0.98	3.86	13.88	4.93
Local clocks: **M.4**	3.00	5.84	8.04	10.88	6.94
No offsets: **M.6,7**	4.80	11.23	17.96	27.11	15.28
Maximum delay bounds (in ms)					
Phases: **M.1**	1.54	2.12	2.88	3.72	3.72
Phases: **M.2**	1.04	1.58	1.96	2.88	2.88
Phases: min(**M.3**,**M.7**)	1.04	1.08	17.3	30.3	30.3
Local clocks: **M.4**	4.26	6.64	10.04	12.28	12.28
No offsets: **M.6,7**	7.84	14.46	23.66	30.30	30.30

Table 3: Delay bounds with phases $\leq$ 5ms.

Priorities	**2-16**	**17-31**	**32-46**	**47-61**	**2-61**
Average delay bounds (in ms)					
Phases: **M.1**	1,77	3,17	4,91	7,02	4,22
Phases: **M.2**	1,39	2,82	4,57	6,48	3,82
Phases: min(**M.3**,**M.7**)	2,70	10,58	17,96	27,12	14,59
Local clocks: **M.4**	3,01	5,85	8,04	10,88	6,94
No offsets: **M.6,7**	4,80	11,23	17,96	27,12	15,28
Maximum delay bounds (in ms)					
Phases: **M.1**	2.38	3.80	6.40	8.00	8.00
Phases: **M.2**	1.88	3.26	5.68	7.66	7.66
Phases: min(**M.3**,**M.7**)	7.84	14.46	23.66	30.30	30.30
Local clocks: **M.4**	4.26	6.64	10.04	12.28	12.28
No offsets: **M.6,7**	7.84	14.46	23.66	30.30	30.30

bounded phases between nodes and apply it to CAN networks. Since, to the best of our knowledge, no results were readily available in the litterature for estimating the worst-case traversal times in such systems, the first contribution of this work is an extension of an existing network calculus analysis to handle such systems.

Three methods to compute such bound have actually been proposed. On the case study, it appears that they are incomparable, *i.e.* there is no method that always outperforms the others, even if it appears that method (**M.3**) returns more accurate results when the phases are small (1ms in our example), and that method (**M.2**) give the best results, in average.

For the sake of understandability, some simplifications have been done: the traffic associated to the synchronization mechanism has not been considered, no event-triggered traffic is considered, and frame transmissions do not suffer jitters. These are not limitations of the method, but methodological choices. Adding other kind of data flow can be done by using the Theorem 2 on non-preemptive static priority. Considering a data flow of priority i, if it exists some sporadic flow of higher priority frames with arrival curve α^s, we have to replace β by $\beta - \alpha^s$ in the equations used to compute the delay on the flows with phases. And the same can be done for the flow dedicated to clock synchronization. Adding a jitter d to an arrival curve α is just a deconvolution by δ_d. Then, modeling such a jitter in each node can be done by simply adding a $\oslash\delta_d$ in eq. (18).

The experiments also have produced some insights in the beneficial impact of bounded phases. Comparing bounded phases to the global clock solution, it appears that a phase of 1ms leads to limited additional latencies, smaller than 3ms for the 250kbit/s network used in the experiments. And comparing phases to the local clock, a phase of 5ms already

provides an average gain of around 40%. These are very promising results.

In the future, we would like first to enhance the analysis method itself: currently, some information is lost when a flow is decomposed as a sum of sub-flows, in order to have a common period of all sub-flows. We would like also to compare the analytic results to simulation results. But the main work is about further evaluating the benefits of using bounded phases: how does the gain change with respect to the bus load or with respect to the period distribution? Could offset assignment algorithm tailored to systems with phases lead to substantial improvements?

Thanks

The authors would like to thank Nicolas Navet, for its great help on a preliminary version of this work.

This work has been partially funded by French agency DGA, under project IREHDO 2.

7. REFERENCES

[1] A. Bouillard, L. Jouhet, and E. Thierry. Service curves in Network Calculus: dos and don'ts. Research Report RR-7094, INRIA, 2009.

[2] A. E. E. Committee et al. Arinc specification 825-2: General standardization of CAN (Controller Area Network) bus protocol for airborne use. *Annapolis, Maryland*, 2011.

[3] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien. Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35(3):239–272, 2007.

[4] M. Grenier, J. Goossens, and N. Navet. Near-Optimal Fixed Priority Preemptive Scheduling of Offset Free Systems. In *14th International Conference on Real-Time and Networks Systems (RTNS'06)*, pages 35–42, Poitiers/France, May 2006.

[5] M. Grenier, L. Havet, and N. Navet. Pushing the limits of CAN-scheduling frames with offsets provides a major performance boost. In *4th European Congress on Embedded Real Time Software (ERTS 2008)*, 2008.

[6] F. Hartwich, B. Müller, T. Führer, R. Hugel, et al. Timing in the TTCAN network. In *Proceedings of the 8th International CAN Conference (iCC'02)*, 2002.

[7] U. Klehmet, T. Herpel, K.-S. Hielscher, and R. German. Delay bounds for CAN communication in automotive applications. In *Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB), 2008 14th GI/ITG Conference-*, pages 1–15. VDE, 2008.

[8] H. Kopetz and G. Grunsteidl. TTP-a protocol for fault-tolerant real-time systems. *Computer*, 27(1):14–23, 1994.

[9] J.-Y. Le Boudec and P. Thiran. *Network calculus: a theory of deterministic queuing systems for the internet*, volume 2050. LNCS, Springer, 2001.

[10] K. Lee, J. C. Eidson, H. Weibel, and D. Mohl. IEEE 1588-standard for a precision clock synchronization protocol for networked measurement and control systems. In *Conference on IEEE*, volume 1588, page 2, 2005.

[11] G. Leen and D. Heffernan. TTCAN: a new time-triggered controller area network. *Microprocessors and Microsystems*, 26(2):77–94, 2002.

[12] X. Li, J.-L. Scharbarg, and C. Fraboul. Improving end-to-end delay upper bounds on an AFDX network by integrating offsets in worst-case analysis. In *IEEE Conf. on Emerging Technologies and Factory Automation (ETFA 2010)*, pages 1–8. IEEE, 2010.

[13] G. Rodrıguez-Navas and J. Proenza. Clock synchronization in CAN distributed embedded systems. *RTN 2004*, page 35, 2004.

[14] W. M. Sofack and M. Boyer. Non preemptive static priority with network calculus: Enhancement. In *Proc. of the Workshop on Network Calculus (WoNeCa 2012)*, Kaiserslautern, Germany, March 2012.

[15] W. Steiner, G. Bauer, B. Hall, M. Paulitsch, and S. Varadarajan. TTEthernet dataflow concept. In *8th IEEE Int. Symposium on Network Computing and Applications (NCA 2009)*, pages 319–322. IEEE, 2009.

[16] D. Tamas-Selicean, P. Pop, and W. Steiner. Synthesis of communication schedules for TTEthernet-based mixed-criticality systems. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 473–482. ACM, 2012.

[17] K. Tindell. Adding time-offsets to schedulability analysis. Technical report, University of York, England, 1994.

[18] K. Tindell, H. Hansson, and A. J. Wellings. Analysing Real-Time Communications: Controller Area Network (CAN). In *Real-Time Systems Symposium, 1994., Proceedings.*, pages 259–263. IEEE, 1994.

[19] P. M. Yomsi, D. Bertrand, N. Navet, and R. I. Davis. Controller Area Network (CAN): Response time analysis with offsets. In *9th IEEE Int. Workshop on Factory Communication Systems (WFCS 2012)*, pages 43–52. IEEE, 2012.