Delay Evaluation and Compensation in Ethernet-Networked Control Systems

Boussad Addad, Saïd Amari

Delay Evaluation and Compensation in Ethernet-Networked Control Systems

Boussad Addad, Said Amari
LURPA ENS Cachan, 61 av du Président Wilson, 94235 Cedex France
{surname.name}@lurpa.ens-cachan.fr

Abstract

Research in networked control systems (NCS), deals with both communication networks and control. Usually, these two points of view are treated separately and rarely together in the same context. In this paper, an overall study of a networked control system is presented. First, a formal method to evaluate the response time of Ethernet-based automation architectures using a client server protocol is presented. It is based on a modeling by the use of timed event graphs and Max-plus algebra. An algorithm and analytical formulas to evaluate the response time are obtained. This approach is validated using measurements taken on a real platform. Thereafter, the results are used in the synthesis of a Smith predictor-based delay compensation strategy to improve the performance of the whole NCS.

1. Introduction

Nowadays automation architectures consist of many intelligent devices connected by a local or global communication network. Indeed, many benefits are achieved in developing networked control systems (NCSs). At the beginning, it was essentially to reduce the system wirings and to ease the information exchange between the different components of the system. Later, the trend was the use of the same network technology at all levels in the industrial organizations; management and automation. A solution that supports such vertical integration had to be able to provide high throughputs in upper level as well as small and accurate response time in field level. Ethernet solutions that were initially developed to office networks can be considered as a new generation of fieldbuses. Currently many automation producers and alliances developed their own industrial Ethernet standard [1]. Each solution with a specific protocol is best suited to a particular application. A client server protocol like Modbus TCP/IP, even not adequate for strict real time applications as motion control, is a simple and a reasonable solution for many purposes in industrial control systems. However, with such protocol, no global resource scheduling is available and different delays due to waiting for resource availability or synchronization are caused. So, the evaluation of its temporal performances like the response time is complex and the investigations that deal with this problem are rare. The existing methods are often based on simulation or experimental measurements [2]. Among them, we find exhaustive space exploration method based on stochastic or not model checking [3], [4]. It is not adequate for response time distribution calculus and the computing limits are quickly reached because of the state explosion problem. Another method uses high level colored Petri nets model and cases simulation with CPNTools [5], [6]. A worst case classical method is also possible by ignoring all dependencies in the system and considering only pessimistic situations [7]. In this paper, we present a novel method we developed to asses the response time either by the use of an easy implemented simulation algorithm or analytical formulas for a trivial calculus.

The previous section is somewhat devoted to the networking community that deals with communication networks and protocols. Another community is that which is interested in networked control strategies. Approaching the control problem in NCS can be achieved in two main different ways. The control is synthesized by ignoring the delay whereas a network scheduling is performed so as to minimize this induced delay. The other one is to consider the priori induced delay and look for adequate control strategies to compensate its negative effect on the NCS. During the last years, this topic has been actively studied and many compensation strategies are proposed. Some recent works without being exhaustive, apply control theory like robust control [8], LMI based control [9], fuzzy logic control [10], and Smith predictor-based compensation control [11]. In these papers, it is usually assumed that information about the delay like upper bound, mean, distribution or other features are known in advance. To our knowledge, the studies that consider both the control synthesis and the induced delay evaluation are rare. When it is the case as in [12] for instance, the application protocol due delays are ignored. It is better but not enough in NCS using client server protocol since the delays due to waiting for resources availability or synchronization are considerable and above all, time varying. They have to be taken into account.

This paper tackles the gap that still exists between the different communities. Indeed, we firstly develop a formal method to assess the response time of the whole automation architecture before to move on to the synthesis of a compensation and control strategy using these results.

The rest of this paper is organized as follows: first a brief recall about the timed event graphs (TEGs) and their modeling using the Max-Plus algebra is presented. Then in section 3, they are used for the modeling of the considered automation architecture. An algorithm and analytical formulas for direct calculus of the response times are obtained. In section 4, we check their validity by comparing them with experimental measurements obtained on a patented laboratory platform [13]. After in section 5, the results are used to synthesize a Smith predictor-based compensation strategy so as to improve the NCS performances. Finally, a short conclusion and some prospects are given in the last section.

2. Timed event graphs/Max-Plus algebra

An event graph is an ordinary Petri net where all the places have at most one upstream and one downstream transition. An event graph is timed if the transitions or the places are affected with delays. We note n the number of transitions with at least one place upstream and m the number of source transitions t_u. The only place relying the transitions t_j and t_i is noted p_{ij} and its delay τ_{ij}. In the modeling of our study (section 3), we assign simply a delay τ_i to a place p_i.

To study the dynamic behaviour of a timed event graph, we associate to each transition the date of its firing for k^{th} time. It is noted $u_i(k)$ for a source transition and $\theta_j(k)$ for other transitions.

Example:

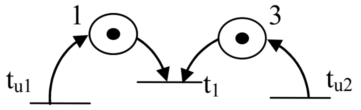

Figure 1. Example of TEG.

The timed graph of the example leads to the equation: $\theta_1(k) = \max(1+u_1(k-1), 3+u_2(k-1))$, that represents the date of firing the transition t_1 for the k^{th} time. It is a linear equation in the Max-Plus algebra. Indeed, a new algebraic structure emerged around two laws. The classical maximum noted indifferently $\oplus$ or "max" with a neutral element $\varepsilon = -\infty$ and the classical addition noted $\otimes$ with a neutral element $e = 0$. More details about this algebra are available in [14]. So, the previous equation can be rewritten using these laws as:

$$\theta_1(k) = (1 \otimes u_1(k-1)) \oplus (3 \otimes u_2(k-1)) \qquad (1)$$

In general, the behaviour of a TEG can be expressed by the following Max-Plus linear equation:

$$\theta(k) = \underset{\varphi \geq 0}{\oplus} (A_\varphi \otimes \theta(k-\varphi) \oplus B_\varphi \otimes u(k-\varphi)), \qquad (2)$$

where the vectors $\theta(k)$ and $u(k)$ components are the firing times of the n and m transitions of the system for the k^{th} time. The matrix A_φ with the element $A_{\varphi,ij}$ represents the delay τ_{ij} associated to the place p_{ij} (with the marking φ) if it exists and the neutral element ε else. Similarly for B_φ, it corresponds to the delays of the places downstream of the source transitions.

In an analogous manner as in usual linear systems, this form can be brought to a state space representation by replacing all the places with markings $\varphi_{ij} > 1$ by φ_{ij} other places with $(\varphi_{ij} - 1)$ intermediate transitions. Hence, we obtain an extended system with a state vector $x(k)$ with $(n+n')$ components where n' the number of added transitions.

The new system is now described by the equation:

$$x(k) = \hat{A}_0 \otimes x(k) \oplus \hat{A}_1 \otimes x(k-1) \oplus \hat{B} \otimes u(k), \qquad (3)$$

can be rewritten in an explicit form:

$$x(k) = A \otimes x(k-1) \oplus B \otimes u(k), \qquad (4)$$

where $A = \hat{A}_0^* \otimes \hat{A}_1$, $B = \hat{A}_0^* \otimes \hat{B}$ and $\hat{A}_0^* = \underset{i \in \mathbb{N}}{\oplus} \hat{A}_0^i$ is the Kleene star of $\hat{A}_0$.

The last formulations (3) and (4) permit to point out that the dynamic behaviour of a timed event graph is determinist, depending only on the source transitions and the initial conditions [15].

3. Modeling and response time evaluation

In this study, the automation architecture works according to Modbus TCP/IP client server protocol. The communication module of the PLC (programmable logic controller) is the client and the remote input output modules (RIOMs) are the servers. The considered PLC comprises two modules, the CPU (central processing unit) that executes the user program and the Ethernet board (ETHb) that sends requests (combined requests: read and write data) to the RIOMs. They operate cyclically but are not synchronized. The CPU accomplishes periodically the tasks: reading inputs, execution of user program and update of the outputs with respect of the cycle period. Regardless of the CPU, the ETHb sends requests to RIOMs and waits for the returned answers. Once all answers arrived, it waits the end of cycle time to begin a new scanning. In this study, no frames loss or time-out have to be taken into account.

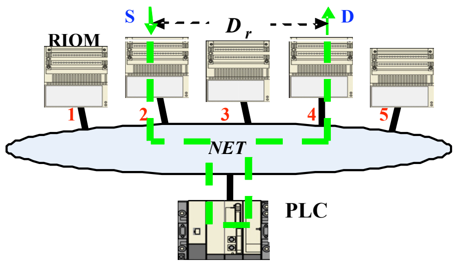

Figure 2. Automation architecture

Whatever is the used protocol in the automation architecture, a major criterion of real time performances evaluation is the response time D_r. It is the delay between the occurrence of an event on the plant and the occurrence of the reaction event issued from the controller, on the controlled plant (Figure 2). Two cases are to be considered in control architectures. It may be, for instance, the reaction delay to the detection of danger. In this case the evaluation of the maximal bound of the reaction time is of top priority. However, if the automation system concerns cyclical control, it is distribution rather than bounds of the response time that is more important to assess.

In our work, we consider the general case regardless of the consequences of the occurring events.

3.1. Architecture modeling with TEGs

According to the previous description of the architecture, we got to the model of Figure 3. It comprises two independent TEGs: one at the left that models the CPU and the other at the right for the rest of the system. This model is very abstract and represents only the application layer protocol. It is over TCP/IP and all what is linked to these protocols can be simply represented by abstract processing steps. We are looking for time performances and only these phases due delays interest us. They can be obtained experimentally and sometimes from the components vendors. So, the details about these delays are out of our study and they are supposed known beforehand.

In Figure 3, we assigned simply a delay τ_i to a place p_i. So, the places p_1, p_2, p_3, and p_4 with delays τ_1, τ_2, τ_3 and τ_4 of the CPU, model respectively the phases of waiting, user program execution during T_{CLC} (reading and writing included), CPU busy and finally CPU idle. So, we easily note the periodical operating of the CPU with cycle period τ_3 noted T_{CPU}. Similarly, τ_{15} is the scanning period T_{SCN} of the ETHb and a token in the place p_{15} means it is busy during at least this period. The requests are sent from the ETHb in an invariant order as shown on Figure 3. The RIOMs are affected with indexes according to the order of their scanning. We associate the index *i* to the RIOM receiving the i^{th} request from the ETHb. Particularly, N_S and N_D are the indexes assigned to respectively the event source (S) and destination (D) of its consequence. On Figure 2 for example, $N_S = 2$ and $N_D = 4$. The model comprises N scanned RIOMs and therefore $1 \le i \le N$. So, the sending of the i^{th} request starts by firing the transition $t_{5(i-1)}$ (except for the first where it is t_4) and finishes by firing t_{5i}, τ_{6i} or T_{EM} being the required time to send a frame. A token in p_{14} means that all the requests are sent and the ETHb is waiting for the answers. The places p_{7i} and p_{12i} model the network (a store and forward switch in our study) delays imposed to the i^{th} sent request and the corresponding returned answer. The separation of these places is made possible since the links are full duplex and colliding is not possible. Once this answer arrives to the input buffer of the ETHb in p_{13i}, it is put in a FIFO queue for a time τ_{13i}. According to the FIFO policy, once its turn arrives, it is processed and copied into the shared memory with the CPU in a time equal to T_{CPY} (practically negligible). The places p_{8i}, p_{9i}, p_{10i}, and p_{11i} represent the i^{th} scanned RIOM. It stays waiting in p_{9i} until the i^{th} request arrives to its input buffer p_{8i}. By firing t_{7i}, the processing starts and goes on for a time τ_{10i} equal to $T_{I/Oi}$. At the end, it puts the answer in its output buffer p_{11i}. On Figure 3, only the first RIOM is represented since all the RIOMs are modeled similarly.

The grey arrows represent the source (data coming from the sensor) and output (data toward the actuator). They are not considered at this stage since the system is not constrained and data are available at the output of the sensor as long as it is functional. Another reason is that at every scanning cycle, an event is generated at the input and we are interested in its corresponding response time. In this way, we are sure to consider all the possible situations in the system and therefore the calculated bounds are formal as we will see later.

To solve the problem of conflicts due to sharing the switch and the ETHb, we introduce variable delays that a frame suffers according to its arrival to these shared resources. This is made possible since all the responses come back before the considered scanning period T_{SCN} elapses. The blue places p_{7i} represent the switch delays affecting the sent requests and the yellow places p_{12i} the delays affecting the returned responses. Finally, the green places p_{13i} represent the delays in the FIFO queue of responses sharing the input buffer of the ETHb. So, this solution is initially graphical with TEGs and subsequently analytical to calculate the variable delays of the colored places. For this purpose, we introduce the function *order* to define the order of arrival of the frames to a shared resource. So, $order(i_f, l)$ gives the order of the sent request relative to the i^{th} RIOM at the l^{th} scanning cycle and $order(i_b, l)$ the order of arrival of the corresponding response. In this paper, we define the FIFO policy in the switch. Thus, at the l^{th} scanning cycle, the variable delays are given by:

$$\begin{cases} \tau_{7i}(l) = \max(T_{SWf}, \theta_{j'j}(l) - \theta_{5i}(l) + T_{SWf}) \\ \tau_{12i}(l) = \max(T_{SWb}, \theta_{j'j}(l) - \theta_{9i}(l) + T_{SWb}) \\ \tau_{13i}(l) = \max(T_{CPY}, \theta_{10j}(l) + \tau_{13j}(l) - \theta_{10i}(l) + T_{CPY}) \end{cases} \quad (5)$$

Where $order(j_x, l) = order(i_y, l) - 1$, $j' = 6$ if $x \equiv f$ and $j' = 10$ if $x \equiv b$, T_{SWf} and T_{SWb} are the intrinsic switch delays imposed respectively to a sent request and a returned response. The system (5) is initialized by considering the first frames to arrive to the switch and the ETHb i.e. $\tau_{71}(l) = T_{SWf}$ and $\tau_{13j}(l) = T_{CPY}$ where $j_b = \min_i(i_b)$.

By applying the method of the section 2 to the model of the architecture with the initial conditions on Figure 3, we got to the Max-Plus equations:

$$\begin{cases} \theta_1(k) = (\theta_2(k-1) \otimes \tau_1) \oplus (\theta_3(k-1) \otimes \tau_4) \\ \theta_2(k) = \theta_1(k) \otimes \tau_2 \\ \theta_3(k) = \theta_1(k) \otimes \tau_3 \end{cases} \quad (6)$$

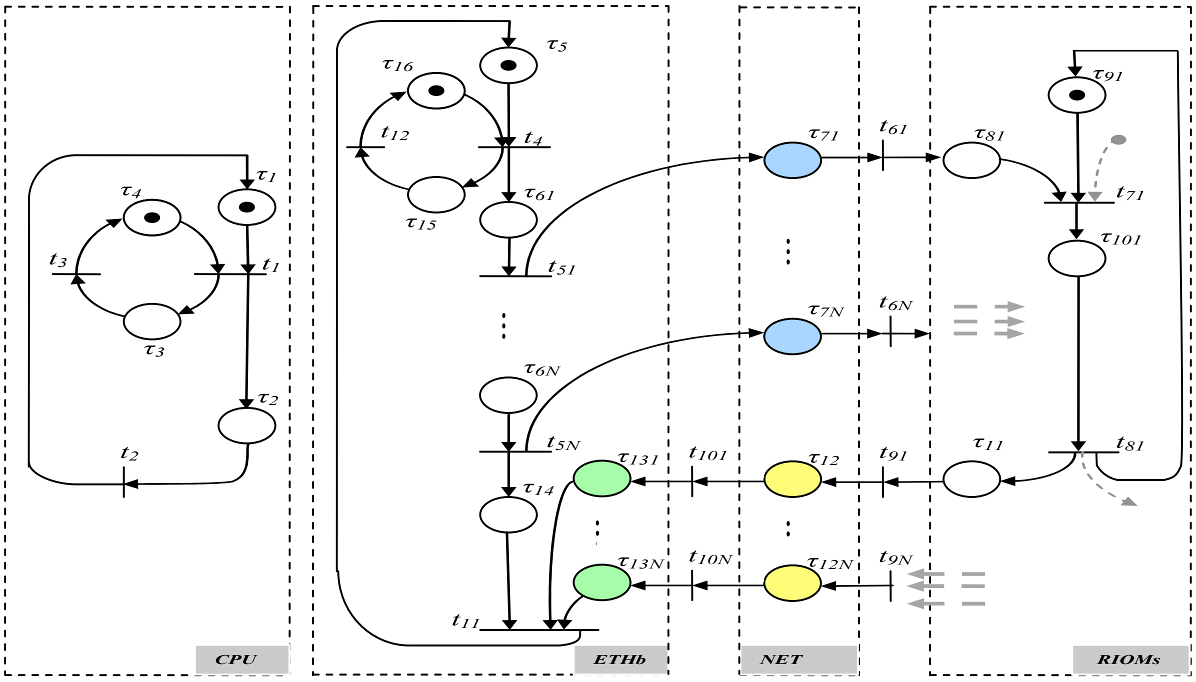

Figure 3. TEGs model of the Automation architecture.

$$\begin{cases} \theta_4(l) = (\theta_{11}(l-1) \otimes \tau_5) \oplus (\theta_{12}(l-1) \otimes \tau_{16}) \\ \theta_{50}(l) = \theta_4(l) \quad // \, \theta_{50} : only \; for \; loop \; initialization \, // \\ for \quad i = 1 \; to \; N \\ \qquad \theta_{5i}(l) = \theta_{5(i-1)}(l) \otimes \tau_{6i} \\ \qquad \theta_{6i}(l) = \theta_{5i}(l) \otimes \tau_{7i}(l) \\ \qquad \theta_{7i}(l) = (\theta_{6i}(l) \otimes \tau_{8i}) \oplus (\theta_{8i}(l-1) \otimes \tau_{9i}) \\ \qquad \theta_{8i}(l) = \theta_{7i}(l) \otimes \tau_{10i}(l) \\ \qquad \theta_{9i}(l) = \theta_{8i}(l) \otimes \tau_{11i} \\ \qquad \theta_{10i}(l) = \theta_{9i}(l) \otimes \tau_{12i}(l) \qquad end \\ \theta_{11}(l) = (\theta_{5N}(l) \otimes \tau_{14}) \oplus \underset{1 \le j \le N}{\oplus} (\theta_{10j}(l) \otimes \tau_{13j}(l)) \\ \theta_{12}(l) = \theta_4(l) \otimes \tau_{15} \end{cases} \qquad (7)$$

The systems (6) and (7) can be written in a similar form as (4) but with time-variant matrix. As we can note, they are assigned different indexes (*k* and *l*). Indeed they are not synchronized, exactly like the CPU and the ETHb. It is the main difficulty of this study.

3.2. Principle of the proposed method

First we solve (6) and (7) to find the dates of firing the transitions as functions of the indexes *k* and *l*. However, the systems are time-variant and their resolution is not obvious. We bypass it by regarding only the beginning of the two modules cycles .i.e. the transitions θ_1 and θ_4 whose dates are trivially obtained under the previous assumptions. The other transition dates are deduced accordingly. So, the resolution of the Max-Plus equations systems leads to:

$$\begin{cases} \theta_1(k) = (k-1) \cdot T_{CPU} \\ \theta_2(k) = (k-1) \cdot T_{CPU} \otimes T_{CLC} \\ \theta_3(k) = k \cdot T_{CPU} \end{cases} \qquad (8)$$

$$\begin{cases} \theta_4(l) = (l-1) \cdot T_{SCN}, \quad \theta_{50}(l) = \theta_4(l); \\ for \; i = 1 \; to \; N \\ \qquad \theta_{5i}(l) = \theta_{5(i-1)}(l) \otimes \tau_{6i} \\ \qquad \theta_{6i}(l) = \theta_{5i}(l) \otimes \tau_{7i}(l) \\ \qquad \theta_{7i}(l) = (\theta_{6i}(l) \otimes \tau_{8i}) \oplus (\theta_{8i}(l-1) \otimes \tau_{9i}) \\ \qquad \theta_{8i}(l) = \theta_{7i}(l) \otimes \tau_{10i}(l) \\ \qquad \theta_{9i}(l) = \theta_{8i}(l) \otimes \tau_{11i} \\ \qquad \theta_{10i}(l) = \theta_{9i}(l) \otimes \tau_{12i}(l) \qquad end \\ \theta_{11}(l) = (\theta_{5N}(l) \otimes \tau_{14}) \oplus \underset{1 \le j \le N}{\oplus} (\theta_{10j}(l) \otimes \tau_{13j}(l)) \\ \theta_{12}(l) = \theta_4(l) \otimes \tau_{15} \end{cases} \qquad (9)$$

The loop is used to avoid repeating the same calculus for the N RIOMs. The variable delays involved in each step are given by the calculus in (5).

The second step of our method is the fusion of the two systems (8) and (9). So, we have to consider the important events that link the CPU and the ETHb. In the model, the transition t_{11} models only the fact that a new cycle will not begin while all the responses are not received. So, an important event to be considered is the end of copy of the response frame coming from the event source (S) to the shared memory with the CPU. For this purpose, we introduce a virtual transition θ_S that represents this important event i.e. $\theta_S(l) = \max(\theta_{10N_S}(l) + \tau_{13N_S}(l), \theta_{5N}(l) \otimes \tau_{14})$.

Thus, in the previous solutions, only the equations that represent the following events interest us:

- Beginning of a new CPU cycle (θ_1).
- End of processing in CPU and output update (θ_2).
- Beginning of a new scanning cycle (θ_4).
- Reception of the answer in the shared memory (θ_S).

Indeed, they are the events that link the CPU and the ETHb. When the answer coming from the event source arrives (θ_S), it is taken into account (with already arrived responses) at the next beginning of the CPU cycle (θ_1). They are read and used in calculus in the CPU. Once the processing finishes, the results are written in a shared memory with the ETHb (θ_2). Finally, they are encapsulated in frames and sent in order to the corresponding RIOMs at the next beginning of the scanning cycle (θ_4).

Let us put $T_r(l) = \theta_S(l) - \theta_4(l)$, the time between the beginning of scanning and the reception of the response from the event source. Then, we obtain the equations:

$$\begin{cases} \theta_1(k) = (k-1) \cdot T_{CPU} \\ \theta_2(k) = (k-1) \cdot T_{CPU} + T_{CLC} \end{cases} \qquad (10)$$

$$\begin{cases} \theta_4(l) = (l-1) \cdot T_{SCN} \\ \theta_S(l) = (l-1) \cdot T_{SCN} + T_r(l) \end{cases} \qquad (11)$$

At the l^{th} scanning cycle, the answer is received at time $\theta_S(l)$ and to be taken into account, it has to wait for m^{th} (but immediate next one with respect to $\theta_S(l)$) beginning of the CPU cycle. The condition to check is: $m = \underset{i \in \mathbb{N} / \theta_1(i) > \theta_S(l)}{Arg \min} \; (\theta_1(i) - \theta_S(l))$ where "$Arg \min$" is the converse function which gives the index that minimizes the positive term: $(\theta_1(i) - \theta_S(l))$.

Hence, we introduce a new variable $\hat{\theta}_1$ that represents this event: $\hat{\theta}_1(l) = \theta_2(m) = \theta_1(m) + T_{CLC}$.

As soon as the results of calculus are obtained, they are written in a shared memory with the ETHb. At the next beginning of scanning cycle, the n^{th} one, the results are encapsulated in the requests frames and sent in order to the RIOMs. In similar way, we add another new variable $\hat{\theta}_2$ that represents this important event with: $\hat{\theta}_2(l) = \theta_4(n)$ and verifying the condition: $n = \underset{j \in \mathbb{N} / \theta_4(j) > \hat{\theta}_1(l)}{Arg\min} (\theta_4(j) - \hat{\theta}_1(l))$. So, the date of arrival of the event consequence to the controlled process, noted $\hat{\theta}_f(l)$, is therefore $\theta_{8N_D}(n)$. Finally, for the p^{th} event generated at a time $\theta_e(p)$ and taken into account at the l^{th} scanning cycle, the associated response time of the architecture is given by:

$$D_r(l) = \hat{\theta}_f(l) - \theta_e(p) \tag{12}$$

This formula enables to have the distribution of the response time if a model of generating the events is given i.e. $\theta_e(p)$ as a function of p or l.

This delay is minimal if the data coming from the sensor are used in processing in the RIOM (S) immediately after they are generated. So the minimal delay relative to the l^{th} scanning cycle is:

$$D_{MIN}(l) = \hat{\theta}_f(l) - \theta_{7N_S}(l) + d_f, \tag{13}$$

d_f being the delay due to data filtering in the sensor.

On the contrary, this delay is maximal if the data arrived immediately after the beginning of the RIOM processing relative to the previous scanning cycle. So, it is given by:

$$D_{MAX}(l) = \hat{\theta}_f(l) - \theta_{7N_S}(l-1) + d_f \tag{14}$$

These delays are always valid and it is the case if the frequency of update of the sensor output is smaller than the frequency of scanning. Else, some events may be erased and not used in any processing. This case is not useful to be considered since it is not really necessary to the response time evaluation.

To sum up, the proposed method is achieved by following the steps: modeling the architecture using TEGs, writing the corresponding Max-Plus equations, resolution of these equations and finally the introduction of the secondary variables to their fusion. Hence, we obtained an algorithm to evaluate the response time of the architecture relative to any occurring event. It is fast and easily implemented using any programming language. The features of the system are introduced as parameters with constant or variable values and even stochastic. So, it is flexible for use for different configurations of the architecture with realistic time characteristics considerations. Simulations of this algorithm are used to check the validity of the formulas obtained next.

3.3. Analytical calculus of response time

The previous algorithm is enough to calculate the bounds of the response time and its distribution. However, it is preferable to have formal analytic formulas giving directly these results, and above all, eases the analysis of the influence of the architecture parameters on its performances. It is the object of this section. We use the results (10), (11) and the principle of the previous algorithm.

Let us put: $T_r(l) = \alpha_l \cdot T_{CPU} + \gamma_l \cdot T_{CPU}$, with

$$\begin{aligned} T_r(l) = \max(N \cdot T_{EM}\,, N_S \cdot T_{EM} + \tau_{7N_S}(l) \\ + \tau_{10N_S}(l) + \tau_{12N_S}(l) + \tau_{13N_S}(l)), \end{aligned} \tag{15}$$

where α_l and γ_l are respectively the entire part and fractional part of the division result of $T_r(l)$ by T_{CPU}.

Let us put also: $T_{CLC} = \beta \cdot T_{CPU}$ $(\beta < 1)$ and $T_{SCN} = r \cdot T_{CPU} = \rho \cdot T_{CPU} + \varepsilon \cdot T_{CPU}$ where ρ and ε are respectively the entire and fractional part of r.

At the l^{th} scanning cycle, we have:

$$\theta_S(l) = (l-1) \cdot r \cdot T_{CPU} + (\alpha_l + \gamma_l) \cdot T_{CPU} \tag{16}$$

Let us take $i \in \mathbb{N}$ where:

$i \le \gamma_l + \varepsilon \cdot (l-1) < (i+1)$ (*)

For $k - 1 = (l-1) \cdot \rho + \alpha_l + 1 + i$ we get:

$$\theta_1(k) = \theta_S(l) + \left[i + 1 - (\gamma_l + \varepsilon \cdot (l-1))\right] \cdot T_{CPU} \tag{17}$$

Since in (*) $i \le \gamma_l + \varepsilon \cdot (l-1) < (i+1)$, then:

$$\hat{\theta}_1(l) = \theta_S(l) + \left[i + 1 + \beta - (\gamma_l + \varepsilon \cdot (l-1))\right] \cdot T_{CPU}. \tag{18}$$

We have also:

$\theta_4(n) = (n-1) \cdot r \cdot T_{CPU}$ and for $n = l+1$ then: $\theta_4(n) = l \cdot r \cdot T_{CPU}$ i.e.

$$\theta_4(n) = \hat{\theta}_1(k) + \left[r - \left[\beta + \alpha_l + i + 1 - \varepsilon \cdot (l-1)\right]\right] \cdot T_{CPU}. \tag{19}$$

From (*), we deduce: $\gamma_l < i + 1 - \varepsilon \cdot (l-1) \le 1 + \gamma_l$

Let us put: $\Gamma_{l,i} = \alpha_l + i + 1 - \varepsilon \cdot (l-1)$ with: $\alpha_l + \gamma_l < \Gamma_{l,i} \le 1 + \alpha_l + \gamma_l$ and note the bounds: $\Gamma_{MIN} = \min\limits_{i \in \mathbb{N}, l \in \mathbb{N}} (\Gamma_{l,i})$ and $\Gamma_{MAX} = \max\limits_{i \in \mathbb{N}, l \in \mathbb{N}} (\Gamma_{l,i})$.

So, (19) becomes: $\theta_4(n) = \hat{\theta}_1(k) + \left[r - (\beta + \Gamma_{l,i})\right] \cdot T_{CPU}$.

Thus, on condition C_d: $r > (\beta + \Gamma_{MAX})$, we can write: $\hat{\theta}_2(l) = \theta_4(n) = \theta_4(l+1)$ and $\hat{\theta}_f(l) = \theta_{8N_D}(l+1)$, which implies:

$$\begin{cases} \hat{\theta}_f(l) = \theta_{8N_D}(l+1) \\ D_{MIN}(l) = \theta_{8N_D}(l+1) - \theta_{7N_S}(l) + d_f \\ D_{MAX}(l) = \theta_{8N_D}(l+1) - \theta_{7N_S}(l-1) + d_f \end{cases} \quad (20)$$

Finally:

$$\begin{cases} D_{MIN}(l) = T_{SCN} + (N_D - N_S)\cdot T_{EM} + \Delta(l,1) \\ D_{MAX}(l) = 2\cdot T_{SCN} + (N_D - N_S)\cdot T_{EM} + \Delta(l-1,2) \\ D_r(p) = \theta_{8N_D}(l+1) - \theta_e(p) \end{cases} \quad (21)$$

where:

$\Delta(l,q) = \tau_{7N_D}(l+q) - \tau_{7N_S}(l) + \tau_{10N_D}(l+q) + d_f$

$\tau_{10N_D}(l+q)$ being the processing time of the RIOM (D) at the $(l+q)^{th}$ scanning cycle.

In practice the condition C_d is often respected and the results (21) are valid since the scanning period is by far longer than the period of the CPU ($T_{SCN} \gg T_{CPU}$).

But if the condition C_d is not respected, the calculus of delays will depend on $\Gamma_{l,i}$. So, the conditions of bounds calculus are global (absolute) and the delay condition relative to the p^{th} generated event is local.

Thus, on the following global and local conditions:

$$Global: \begin{cases} r\cdot q_1 > (\Gamma_{MIN} + \beta) > r\cdot(q_1 - 1) \\ r\cdot q_2 > (\Gamma_{MAX} + \beta) > r\cdot(q_2 - 1) \end{cases},$$

$$Local: \ \{ r\cdot q_3 > (\Gamma_{l,i} + \beta) > r\cdot(q_3 - 1)$$

the generalized response times are:

$$\begin{cases} D_{MIN}(l) = q_1\cdot T_{SCN} + (N_D - N_S)\cdot T_{EM} + \Delta(l,q) \\ D_{MAX}(l) = (q_2+1)\cdot T_{SCN} + (N_D - N_S)\cdot T_{EM} + \Delta(l-1,q+1) \\ D_r(p) = \theta_{8N_D}(l+q_3) - \theta_e(p) \end{cases} \quad (22)$$

We point out that the optimal case (where the maximal bound is minimal) is obtained if ($q_2 = 1$) or the condition C_d: $r > (\beta + \Gamma_{MAX})$ is satisfied. In analysis, two cases can be considered:

- $r \in \mathbb{N}$ $(\varepsilon = 0)$

In this case $\Gamma_{l,i} = \alpha_l + i + 1$ and to satisfy (*), we have to take simply $i = 0$ and therefore the optimality condition becomes C_{d1}: $r > (\beta + \alpha_l + 1)$.

- $r \in \mathbb{Q}^+$ $(\varepsilon \neq 0)$

Let us put $\varepsilon = n_1/n_2$. Then it is enough to take $(l-1) = n_2$ and $i = n_1$ to obtain $\Gamma_{l,i} = 1 + \alpha_l$. This implies $\Gamma_{MAX} \geq 1 + \alpha_l$ and since the optimality condition is: C_{d2}: $r > (\beta + \Gamma_{MAX})$, then we can deduce that the condition C_{d2} is more restrictive than C_{d1}.

It is an important result which suggests to fix the period of scanning as a multiple of the period of the CPU (of course minimize T_{CPU} first), in order to minimize the maximal bound of the response time.

On the other hand, for a given architecture without acyclic traffic and RIOMs with constant processing times (or slightly varying which is case in practice), Δ is practically invariant and we have to calculate it only once. Thus, the formulae of (22) become:

$$\begin{cases} D_{MIN} = q_1\cdot T_{SCN} + (N_D - N_S)\cdot T_{EM} + \Delta \\ D_{MAX} = (q_2+1)\cdot T_{SCN} + (N_D - N_S)\cdot T_{EM} + \Delta \end{cases} \quad (23)$$

The results (22) and (23) are very interesting and to minimize the response time, we should assign a great index to the source and small one to the destination: the order of scanning the RIOMs is important. However, we have to keep in mind that the condition of calculus of the delays depends on $T_r(l)$ or α_l (see (15)) and we should decrease N_S. So, the optimal case is got by increasing N_S while q_2 remains equal to 1.

4. Validation

To check the validity of the model and the results developed previously, we consider the configuration of Figure 4. We compare the results obtained using simulation of the algorithm and the developed formulas with measurements taken on a laboratory platform [13].

We are interested in the delay between an event generated on the input of the RIOM R4 and its consequence on the output of the RIOM R5. The histograms of Figure 5 represent a series of 10,000 experimental measurements and simulations of the algorithm for this configuration.

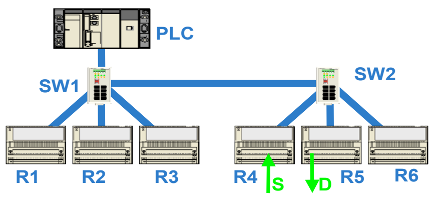

Figure 4. Application architecture.

The CPU period is set up to 5 ms and scanning to 10 ms. However, in practice the architecture presents a jitter of 15% with a maximum value of 10.74 ms and a minimum of 9.24 ms. These bounds of time are used in the formulas (21) to calculate the bounds of response time. The jitter is also considered in the algorithm by imposing a random distribution of scanning cycles with a mean of 10 ms. We obtained the results of Figure 5 and Table 1.

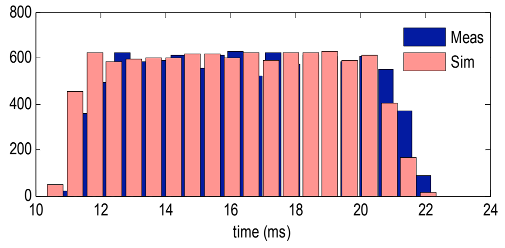

Figure 5. Histograms of measured (blue) and simulated delays (red).

Table 1. Results of delays calculus.

$r=2$, $\gamma_l \approx 0.22$, $\alpha=0$, $\beta=0.6$, $q_1=1$, $q_2=1$	Response delays in ms		
	Min	Max	Mean
Measures	10.65	21.25	16.40
Simulation	10.31	22.49	16.12
Formulas	10.31	22.49	/
Classical	5.91	32.44	/

As expected, the results of simulation of the algorithm and the formulas are exactly the same in all cases. Indeed, they are based on the same principle.

In all cases, we can conclude about the validity of the formulas because the maximal delay is greater than the measured one and smaller than the obtained using the classical method (worst case method). On the other hand, the minimum delay calculated using the formulas or the algorithm is valid compared to the results of the other methods. The gaps of delays, with respect to measurements bounds, are in all cases smaller than 3.27% for analytical formulas or simulations. This gap is only about 1.73% in the calculus of the mean of responses times. A random event generator is used in simulation to obtain realistic distribution of delays (to offset effects of the jitter). Consequently, the shapes of the measurements and simulations histograms are very similar (Figure 5).

5. Application: Smith predictor synthesis

In the study, the plant and the control strategy are ignored so far. However, the final aim is the performance evaluation of the whole NCS. Whatever is the controlled plant, the main feature of the NCSs is the introduction of a delay that menaces the stability of the system if it is over a critical value. The delay is not removable but many solutions are proposed to compensate its negative effect. The Smith predictor is one of the most well known thanks to its simplicity and effectiveness [11]. By the use of the developed results of delays calculus, a Smith predictor is synthesized so as to improve the performance of a classical PI controller in presence of the induced varying delay in the architecture.

5.1. Problem description

In NCS, the network induced delays are often taken into account but those due to the communication protocol and non synchronization, are ignored. By the use of our early results of this study, all delays are considered. Thus, the NCS can be represented by the control loop on Figure 6.

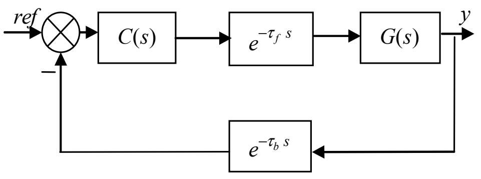

Figure 6. Structure of the NCS.

where $G(s)$ is the transfer function of controlled plant and $C(s)$ of the controller. In this paper, in order to ease understanding, it is synthesized in continuous time and subsequently discretized for implementation (zero order holder is added). *ref* and y are respectively the reference input and the output of the plant. τ_f is the delay from the controller to the plant and τ_b from the plant to the controller. Thus, their sum is the response time calculated earlier i.e. $D_r = \tau_f + \tau_b$. So, the transfer function of the NCS is:

$$F(s) = \frac{G(s)\cdot e^{-\tau_f s}\cdot C(s)}{1+G(s)\cdot C(s)\cdot e^{-D_r s}} \tag{24}$$

From (24), it is clear that the presence of the delay in the denominator will degrade the stability of the NCS.

5.2. Compensation strategy: Smith predictor

The smith predictor can be described as in Figure 7:

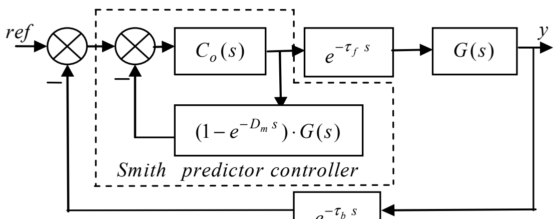

Figure 7. NCS with Smith predictor.

where $C_o(s)$ is a controller of the plant. D_m is the estimated response time of the architecture. So, the transfer function of the whole system is:

$$F(s) = \frac{C_o(s) \cdot e^{-\tau_f s} \cdot G(s)}{1 + C_o(s) \cdot G(s) \cdot (1 + e^{-D_r s} - e^{-D_m s})} \quad (25)$$

If $D_m = D_r$ then:

$$F(s) = \frac{C_o(s) \cdot e^{-\tau_f s} \cdot G(s)}{1 + C_o(s) \cdot G(s)} \quad (26)$$

We see that the controller $C_o(s)$ is to be synthesized as if there is no delay. It is the big interest of the Smith predictor. However, in NCSs it is difficult to assess the response time exactly. So, to get a satisfactory performance, we should have: $D_m > D_r$. Else, the stability of the system is menaced (positive poles). This is possible by setting D_m great enough but not too. Else, the effect of the controller will be very slow and the performances deteriorated. The maximal bound of response time is a compromise between stability and performance. Thus, it is to be well estimated. We are going to see this by an application on concrete system.

5.3. Application and analysis

The goal being to show only the interest of our earlier results concerning the response time evaluation of networked automation architectures, we chose a first order linear system and a classical PI controller given:

$$G(s) = \frac{1}{1+Ts} \to \frac{1+z}{5z-3}, \quad C_o(s) = K\frac{1+T_i s}{s} \to 0.3\frac{5z-3}{z-1}$$

Where the parameters are set to the values:

$K = 60$, $T_i = T = 20$ ms, $T_{sampling} = T_{SCN} = 10$ ms.

The result of simulation of the system under the previous PI controller without any delay is shown on Figure 7. As we see, the behaviour of the system is very satisfactory. This will be our reference performance when we will consider the varying induced delays.

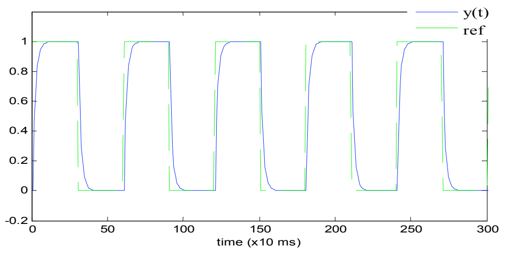

Figure 7. NCS performance: without delay.

However, in NCS, there is always an induced delay. So, we use the practical results of the section 4 by considering an NCS with a jitter of 15%. We obtained the NCS delays on Figure 8. We represented results of only a period of 3s to see clearly and distinguish the performances of the NCS under different conditions.

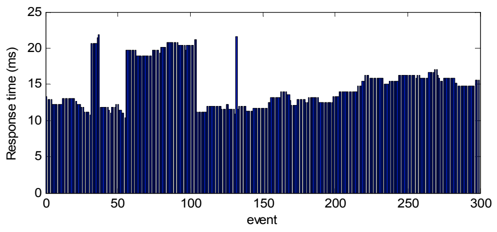

Figure 8. Architecture induced delays.

On Figure 8, it is clear that the changes of the response time are sometimes very abrupt and therefore an online delay estimator would not give perfect results. So, we use an offline estimator (formula of D_r in (21)) giving the response time without considering the jitter. To do this, we consider a sensor that updates its outputs cyclically with the same frequency as scanning but with a lag of τ_0 : $\theta_e(p) = \theta_{7N_S}(l) - \tau_0$.

So, the formula of (21) becomes:

$$D_m = \theta_{8N_D}(l+q) - \theta_{7N_S}(l) + \tau_0 \quad (27)$$

In the particular case of the previous practical considered system (Figure 4), this is written:

$$D_m \approx T_{SCN} + T_{EM} + T_{I/O5} + d_f + \tau_0 \quad (28)$$

The numerical application leads to: $D_m \approx 13$ ms.

The simulation of the NCS with a Smith predictor delay set to $D_m = 13$ ms is shown on Figure 9:

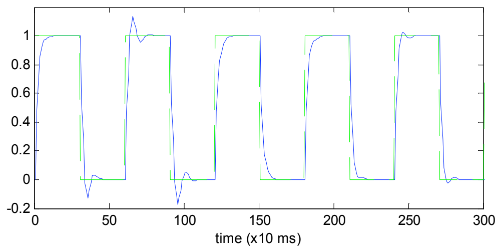

Figure 9. NCS performance: offline delay estimation.

The results may be not satisfactory since we note some overruns of until 20%, not always acceptable in practice. We can notice that they occur exactly when the response time is greater than 13 ms (see Figure 8). They are very striking between the times 300 ms and 1s and this corresponds to reaching the high response times on Figure 8. Indeed, the stability margin of the NCS is reduced when the delay is underestimated.

To avoid this risk of instability, we should use the maximal bound of time we calculated before by the use of the formulas. In Table 1, $D_m = D_{MAX} = 22.49$ ms. The simulation result is shown on Figure 10.a:

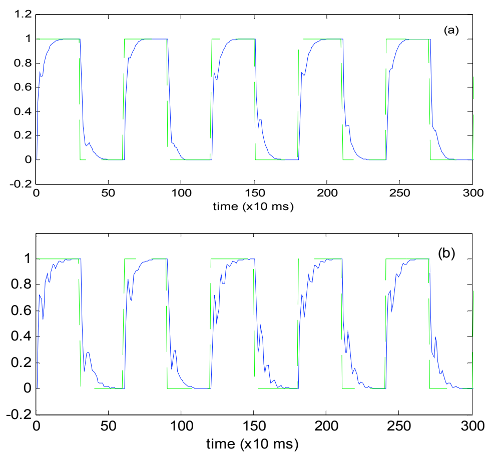

Figure 9. NCS performance: (a) maximal delay estimated using formulas, (b) maximal delay estimated using worst case method.

The results are very satisfactory even the system is slightly slowed than in the case without any delay. In contrary to previous simulation, the behaviour is better when the response time is great. Indeed, the delay is close to the maximal bound and therefore the term of the denominator of the transfer function in (25) is small. Its effect is weakened. However, this maximal bound is to be assessed with enough accuracy. The simulation of the NCS by considering the maximal response time calculated using the classical worst case method is presented on Figure 10.b. We note that the NCS is stable but the performance is very degraded with undesirable oscillations and slow reaction time. Hence, we see the importance of a good evaluation of the response time of the architecture.

6. Conclusion

In this work, we presented an overall study of a NCS taking into account all the induced delays in the architecture. Many important results about the architecture performance evaluation are presented. Indeed, by the use of the developed formulas, it is easy to choose the adequate configuration of the components of the NCS to fulfill the desired requirements. Thereafter, on a concrete plant model, we showed how such results can be used to synthesize a compensation strategy to improve the NCS performances. The Smith predictor was only used for illustration but any advanced strategy can be adopted.
A study of more complex architectures considering acyclic traffic and its influence on the NCS performances is prospected.

7. References

[1] P. Neumann, ”Communication in industrial automation-what is going on?”, *Control Engineering Practice*, doi:10.1016/j.conengprac.2006.10.004, 2006.

[2] B. Denis, S. Ruel, J.-M. Faure, and G. Marsal, “Measuring the impact of vertical integration on response times in Ethernet fieldbuses”, In *Proc. of 12th IEEE Int. Conf. on Emerging Technologies and Factory Automation*, Patras, Greece, 2007.

[3] J. Greifeneder, G. Frey, “Optimizing Quality of Control in Networked Automation Systems using Probabilistic Models”, In *Proc. of 11th IEEE Int. Conf. on ETFA*, Prague, Czech Republic, 2006.

[4] D. Witsch, B. Vogel-Heuser, J.-M Faure, and G. Poulard-Marsal, “Performance analysis of industrial Ethernet networks by means of timed model-checking,” In Proc. of 12th IFAC Symposium on Information Control Problems in Manufacturing, pp. 101–106, 2006.

[5] D.A. Zaitsev, “Switched LAN simulation by colored Petri nets”, *Mathematics and Computers in Simulation*, Vol.65, pp. 245–249, 2004.

[6] G. Marsal et al, “ Evaluation of response time in Ethernet-based automation systems”, In *Proc. of 11th IEEE Int. Conf. on ETFA*, Prague, 2006

[7] K. C. Lee, S. Lee, “Performance evaluation of switched Ethernet for real-time industrial communications”, *Computer standards & interfaces*, Vol.24, pp. 411–423, 2002.

[8] F. Göktas, “Distributed control of systems over communication networks”, *Ph.D. dissertation,* University of Pennsylvania, 2000.

[9] S. Li, Z. Wang, Y. Sun, “Delay-dependent controller design for networked control systems with long time delays: an iterative LMI method”, In *Proc. of the 5th WCICA,* Vol. 2, pp. 1338 – 1342, 2004.

[10] Y.-C. Cao, W.-D. Zhang, “Modified fuzzy PID control for networked control systems with random delays”, In *Proc. of world academy of science engineering and technology*, Vol.12, pp. 313-316, 2006.

[11] P.-H. Bauer, M. Schitiu, C. Lorand, and K. Premaratne, “Total Delay Compensation in LAN Control Systems and Implications for Scheduling”, In *Proc. of the American Control Conference*, pp. 4300-4305, 2001.

[12] J. P. Georges *et al*, “Control compensation based on upper bound delay in networked control systems”, In *17th International Symposium on Mathematical Theory of Networks and Systems*, Kyoto, July, 2006.

[13] B. Denis, O. De-Smet, J.-J. Lesage, J.-M. Roussel, ”Process of performance analysis of systems as finite state automata”, FR. Patent 01 110 933, 2001.

[14] F. Baccelli, G. Cohen, G.-J. Olsder, and J.-P. Quadrat, Synchronization and Linearity: An algebra for Discrete Event Systems, Wiley, 1992.

[15] F. Baccelli, G. Cohen and B. Gaujal, “Recursive equations and basic properties of timed Petri nets”, *Discrete Event Dynamic Systems: Theory and Applications*, 1 (4), 1992.