Client-Server Networked Automation Systems Reactivity: Deterministic and Probabilistic Analysis

Boussad Addad, Saïd Amari, Jean-Jacques Lesage

Client-Server Networked Automation Systems Reactivity: Deterministic and Probabilistic Analysis

Boussad Addad, Said Amari and Jean-Jacques. Lesage, *Member, IEEE*

***Abstract*—In this paper, we present an analytic approach to evaluate the reactivity of client-server networked automation systems (NAS). Both deterministic and probabilistic analyses are provided while modeling the NAS using Timed Event Graphs (TEG). Since many results with regard to the deterministic approach have already been published, we recall only its main steps that prove useful while exposing the probabilistic method. Thereby, we provide the density of probability distribution of the response time (or reactivity) using the probability densities of the local delays, experienced at the different stages of the NAS. Furthermore, a case study is presented to compare the results of the study to measures taken from a real platform.**

***Note to Practitioners*—As a matter of fact, client-server networked automation systems are largely used in industry and therefore the efforts to deal with their performances evaluation are necessary. In the current work, we propose an analytic approach to evaluate their reactivity. Analytic formulae are provided to calculate directly and deterministically the bounds of response time along with others to assess its probability density distribution. Moreover, the results of these formulae turn out to be complying with a lot of experimental measurements carried out under different circumstances.**

***Index Terms*—Client-Server Networked Automation System, Response Time, Probabilistic Analysis, Max-Plus Algebra, Timed Event Graph.**

I. Introduction

THE huge breakthroughs carried out in the field of communication technologies by increasing the performances of the devices and decreasing their prices, were compelling incentives towards replacing the traditional directly wired systems by networks in industry [1]. However, a networked system means that messages from different stations share the same resources and therefore undergo delays while waiting for their availability. In networked automation systems (NAS), these delays have a tremendous effect on the reactivity of control. The *reactivity* of NAS or the *response time* is the delay between the date of generation of an event by a sensor and the date of arrival of its consequence, issued by a controller, to an actuator. A number of investigations have been undertaken to assess delays caused by networks but they are in their majority limited to the delays experienced while crossing the communication network and ignore the field devices effects [2], [3], [4], [5], [6], [7]. Among the investigations that consider the whole NAS, we find methods based on models simulation [8], [9], [10], others based on model checking [11], [12] and obviously experimental methods [13]. Depending on the NAS, either the bounds of the response time or its distribution need to be assessed. Indeed, in some systems (e.g. critical systems) it is mandatory that the response time be always under a critical value whereas in others (e.g. only quality aspects are in concern), it is accepted that the response time go past a limit value but the probability of such an event must remain under an acceptable value. Among the cited works, only model-checking provides both guaranteed bounds [11] and distribution [12] but unfortunately suffers from the state explosion problem. In [14], we presented an analytic approach in an effort to avoid such a problem but our investigation was limited to NAS with only one controller (we considered several remote I/O modules nevertheless). Thus, the current study is a generalization that displays two main new contributions (*i*) the NAS may involve multiple controllers (*ii*) analytic probabilistic formulae are provided to calculate the density of probability of the response using the local delays probability densities (in [14], we calculated the distribution using simulation, not analytic formulae as the case now).

The remainder of our study has been organized as follows. In Section II.*A*, an overview of an explanatory example of a client-server NAS is provided. Then in Section II.*B*, some fundamentals about timed event graphs (TEG) and their linear (max,+) representation are recalled. Hence, these tools are used to model a generic client-server NAS, involving *M* clients (controllers) and *N* servers (remote I/O). Next, Section III is dedicated to a deterministic evaluation of the response time, especially its maximal and minimal bound whereas Section I.V to a probabilistic one. A formula for direct calculus of the probability density of response time is provided. Afterwards, Section V is devoted to a case study to compare the results of the exposed methods to experimental measures. Finally, some considerations for future work are discussed in Section VI as a conclusion to this paper.

B. Addad and J. J. Lesage are with Automated Production Research Laboratory LURPA, ENS-Cachan, 61 av. du Président Wilson, 94235 Cachan Cedex, France (phone: +33-147402762; e-mail: boussad.addad@lurpa.ens-cachan.fr).

S. Amari is with Automated Production Research Laboratory and with Univérsité-ParisXIII, 61 av. du Président Wilson, 94235 Cachan Cedex, France (e-mail: said.amari@lurpa.ens-cachan.fr)

II. CLIENT-SERVER NAS FUNCTIONING AND MODELING

A) *Client-server NAS functioning*

As aforementioned, we consider NAS working according to client-server paradigm. The PLCs (programmable logic controller), that play the role of clients, poll periodically the RIOMs (remote I/O module), which are the servers, to request information to sensors or provide orders to actuators. An explanatory example of automation loop of such a system is depicted in Fig. 1. This example is used only for explanation purpose and the results of the study are valid in the general case. It consists of filling bottles automatically as follows: PLC1 sends periodically requests to RIOM R4 to ask for information: *has the detection level X_d been reached?* and requests to RIOM R5 to give order: either close the valve or open the valve, depending on the information collected during the previous scanning cycle from the sensor. The aim of this NAS loop is to close the valve after detection level X_d is reached. The *response time* D_r of this NAS (thick arrow in Fig. 1), can be defined as the delay between the date of reaching level X_d by the liquid and the date of closing the valve. Actually, when position X_d is reached, this information is sent to PLC1 only after the arrival of a request. Once this is done, the response carrying this info crosses the communication network and gets to the input buffer of the network board (NETb) of the PLC (an interface card that enables communication with the network) before being written in the CPU (Central Processing Unit) cache memory. The CPU also works periodically and therefore takes this new info into account only at the beginning of a new cycle and performs the corresponding order. This order is written in the cache shared with the NETb and sent to its destination (R5) at the beginning of the next scanning cycle. This order crosses the network and gets to R5 and causes the closure of the valve to stop filling a bottle (Fig. 2).

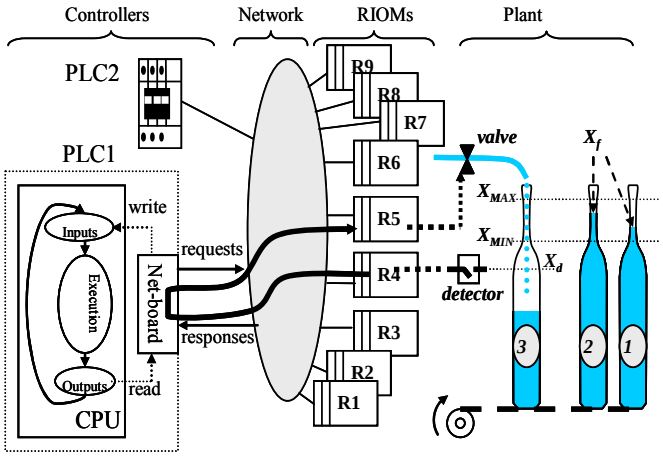

Fig. 1. Client-server NAS: automated bottles-filling system.

Since the response time is never null and even always greater than a minimum delay, the bottle filling continues after position X_d is reached and the final liquid level X_f is somewhere above X_d (see Fig. 1). A problem rises therefore since the NAS has to satisfy two conditions at the same time: on one hand, we have to satisfy the customer's requirement $X_f > X_{MIN}$ and on the other hand $X_f < X_{MAX}$. Indeed, a too full bottle is discarded since putting a cork on it is impossible. So, to be economically viable, the probability to discard a bottle must be smaller than a predefined threshold. So, the probability that the response time be above the delay corresponding to limit X_{MAX} must be under this threshold either. Hence, a probabilistic study is necessary to get the density distribution of the response time. The problem can also be formulated in a stricter way (e.g. safety considerations) by forbidding overflows (e.g. dangerous liquid) and therefore the response time must be under a maximum bound whatever the conditions are. So, either a deterministic (bounds evaluation) or probabilistic (distribution evaluation) of the NAS reactivity is needed.

As it can be seen in Fig. 2, the response time is the sum of many local delays that an event/reaction message experiences at the different stages of the NAS, from the date of generating the event at the sensor to the date of arrival of the reaction to the actuator. Note by the way that these delays are not constant if it is not mentioned otherwise. The following notations designate these timing features (see Fig. 2):

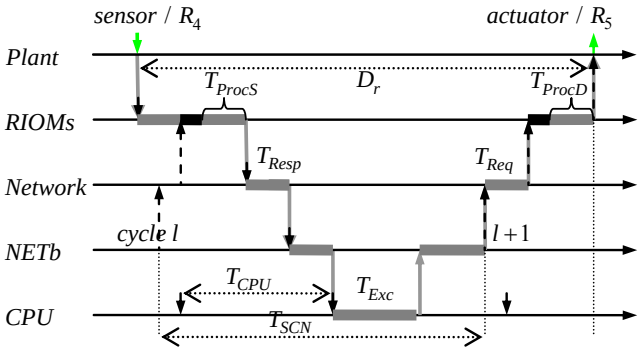

Fig. 2. Response time decomposition.

- T_{SCN} : the scanning period of the NETb
- T_{CPU} : the functioning period of the CPU
- T_{Exc} : the time to execute the user program in the CPU and perform the control signal (orders to RIOMs)
- T_{Q0} : the time spent by a request in the output buffer of the NETb before being sent to the network
- T_{Req} : the necessary time, to a request, to cross the network and get to its destination RIOM
- T_{Q1} : the time spent by a request in the input buffer of the destination RIOM before being taken into account by the RIOM and being processed to return a reply
- T_{Proc} : the necessary time to process a request in a RIOM and return the corresponding response (index S or D is added to refer to source or destination)
- T_{filt} : the time to filter the data issued by the sensor
- T_{Q2} : the time spent by a response in the output buffer of a RIOM before quitting it and entering the network
- T_{Resp} : the necessary time, to a response, to cross the network and get to the input buffer of the NETb

- T_{Q3} : the time spent by a response in the input buffer of the NETb before being taken into account by the NETb and copied into the cache memory of the CPU

***- Hypotheses about NAS functioning*:**

i) The CPU and NETb can be set to function either cyclically or periodically. They are periodic in our study, without clock drift. So, their periods noted respectively T_{SCN} and T_{CPU} are constant. Also, they can be considered as integers since they are set by the user as multipliers of a basis unit. For instance, in our experimental platform, the period T_{SCN} can be set to 10ms, 15ms, 20ms, etc. Besides that, we suppose in this paper that ratio T_{SCN} / T_{CPU} is integer too. Indeed, by performing the transformation exposed in the next section, we get to the same model used in [14] and therefore can consider the general case, which is already studied in [14], in a similar way.

ii) Neither frame loss nor components failure is considered. In the context of this study indeed, we observed the system of Fig. 1 during more than 5 million cycles and no loss or failure has been noticed.

iii) A server does only answer a received request and never send messages autonomously (no asynchronous transmission).

iv) The time to write/read the useful data (without communication layers headers) in cache memory is neglected.

v) All the requested answers are received from the scanned RIOMs before the scanning period elapses (the network due delays are indeed often by far smaller than the scanning period)

vi) According to practical observations, T_{Proc} and T_{filt} can be considered constant (less than 0.08% jitter [14]).

B) Timed Event Graphs and their (Max,+) representation

An event Graph is an ordinary Petri net whose places have at most one upstream transition and one downstream transition [15]. To study the dynamic behavior of a Timed Event Graph (TEG), many mathematical tools can be used, depending on the objective of the study: (Max,+) Algebra with daters [16], (min,+) Algebra with counters [17], a combination of different operators with place-marking and transition-state [18], ... Since daters are more relevant to our work, we consider the first one. So, we associate to each transition t_i its firing date (dater) for the k^{th} time noted $\theta_i(k)$. So, the TEG evolution at maximal speed (maximal speed means that a transition fires as soon as the tokens of the upstream places are available) can be represented using linear (Max,+) equations. Notice that we do consider only P-timed graphs in this study i.e. delays are ascribed only to places. A delay in a place means that a token entering this place is available to fire a downstream transition only after this delay. We explain step by step all these notions using the following simple explanatory example.

Example II.A.1: TEG in Fig. 3 represents a manufacturing system with a machine, represented by place p_1, an upstream stock (place p_u) and two pallets carrying parts. A token in place p_1 means that a part is being processed by the machine. A part entering the stock, by firing transition t_u, becomes available to the machine one time unit later. The process lasts 3 time units before the finished part exits the machine. The machine can then start processing another part 2 time units later i.e. after the comeback of the pallet.

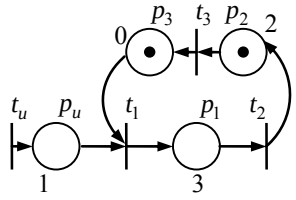

Fig. 3. Example of Timed Event Graph.

The behavior of this system depends obviously on the initial marking of the places and the source transition t_u firing dates. By using the marking of Fig. 3 and supposing that the tokens are initially available, the dates of firing transitions t_1, t_2 and t_3 for the k^{th} time (at maximum speed) are expressed as:

$$\begin{cases} \theta_1(k) = \max[0 + \theta_3(k-1), 1 + u(k)] \\ \theta_2(k) = 3 + \theta_1(k) \\ \theta_3(k) = 2 + \theta_2(k-1) \end{cases} \quad (1)$$

These equations are actually linear in (Max,+) algebra whose operators are the classical addition noted $\otimes$ with identity element 0 noted e and the classical maximum noted $\oplus$ with identity element $-\infty$ noted ε. These equations can be written using these operators as:

$$\begin{cases} \theta_1(k) = e \otimes \theta_3(k-1) \oplus 1 \otimes u(k) \\ \theta_2(k) = 3 \otimes \theta_1(k) = 3 \otimes \theta_3(k-1) \oplus 4 \otimes u(k) \\ \theta_3(k) = 2 \otimes \theta_2(k-1) \end{cases} \quad (2)$$

Moreover, they can be rewritten in a standard state representation as follows:

$$\Theta(k) = A \otimes \Theta(k-1) \oplus B \otimes u(k) \quad (3)$$

where: $A = \begin{pmatrix} \varepsilon & \varepsilon & e \\ \varepsilon & \varepsilon & 3 \\ \varepsilon & 2 & \varepsilon \end{pmatrix}$, $B = \begin{pmatrix} 1 \\ 4 \\ \varepsilon \end{pmatrix}$ and $\Theta(k) = \begin{pmatrix} \theta_1(k) \\ \theta_2(k) \\ \theta_3(k) \end{pmatrix}$

This state representation is very similar to the standard representation of the classical linear systems. It is very useful and used alike for many problems resolution like performance evaluation or control synthesis. For more details about (Max,+) algebra and its applications, we invite the reader to see references [16], [17].

C) NAS Modeling using TEG and (Max,+) Algebra

The TEG of Fig. 4 represents a generic model of Client-Server NAS, independently from the automated plant, with *M* clients (PLCs) polling *N* remote servers (RIOMs). It depicts thoroughly the communication between client PLC*i* and server RIOM*j*. Note that indices $i \in \{1,...,M\}$ and $j \in \{1,...,N\}$ should be added to the different delays of the model but they are omitted purposely for convenience reasons. Since only one automation loop is under consideration at a time (one client and two servers, the source and the destination), we simply omit the index of the PLC, add index *S* to the delays

experienced by the request sent to RIOM$_S$ (event source) and add index D when the request is sent to RIOM$_D$ (event destination). For instance, T_{ReqD} is the necessary time to cross the network by the request sent to RIOM$_D$. Note again that the different delays are not constant and may vary from a cycle to another. So, for instance, the previous delay T_{ReqD} experienced at the l^{th} cycle may be noted $T_{ReqD}\ (l)$ but index l is also omitted for the same reasons.

In the model, we added two delays T_{W1} and T_{W2} to represent two constraints with regard to the NETb functioning; T_{W1} represents the fact that the NETb cannot take into account any received response before getting all the requests sent (the requests are of higher priority) whereas delay T_{W2} means that the NETb has to wait until the reception of all the responses from all the scanned servers before starting another scanning cycle (from hypothesis *v*).

In [14], we used a model constituted exclusively of TEG to get the advantage of the resulting (Max,+) linear equations and perform an analytic study. The model of Fig. 4 however does not verify this condition (it is not a TEG). Indeed, the part representing RIOM*j* contains two places (*Q1* and *Q2*) with more than one input/output transition. These places represent respectively the input and the output buffers (or queues) of RIOM*j*. So, when the request from PLC*i* arrives to the input buffer of RIOM*j*, it must wait until all the waiting requests are processed to be finally taken into account. In terms of Petri nets, the tokens entering place *Q1* contribute to the firing of transition t_6 in a FIFO manner without overtaking. So, the token from client PLC*i* has to wait in place *Q1* during a time noted T_{Q1}. Since our model supports time-varying delays, a linearization is possible. Indeed, instead of considering this non-TEG model, we can remove the transitions from the other clients and simply assign this variable delay T_{Q1} to place *Q1*. Hence, the model becomes a TEG while its delays T_{Q1} and T_{Q2} are variable. The second step of transformation of this model is to fuse delays T_{Req} and T_{Q1}. So, one can equivalently consider that T_{Q1} is null while delay T_{Req} is the total delay experienced by a request from the date of its generation by PLC*i* to the date of beginning of its processing by RIOM*j* (network delay + waiting delay in the queue). A similar fusion can be applied with respect to delays T_{Resp} and T_{Q2}. Finally, the model becomes exactly the same as the one used with a mono-PLC NAS. So, one can always get a TEG-based model of every NAS whatever is the number of its PLCs and the number of its RIOMs. Hence, we can apply exactly the same principle as done in [14], to get to the analytic formulas of response time. So, we are going to recall only the main steps, needed later in the probabilistic study, to obtain theses formulae.

So, the model of Fig. 4 can be represented by:

$$\begin{cases} \theta_1(k) = (\theta_1(k-1) \otimes T_{CPU}) \oplus (\theta_2(k-1) \otimes 0) \\ \theta_2(k) = \theta_1(k) \otimes T_{Exc} \end{cases} \tag{4}$$

$$\begin{cases} \theta_3(l) = (\theta_3(l-1) \otimes T_{SCN}) \oplus \theta_{11}(l-1) \otimes 0 \\ \theta_4(l) = \theta_3(l) \otimes T_{Q0} \\ \theta_5(l) = \theta_4(l) \otimes T_{Req} \\ \theta_6(l) = (\theta_5(l) \otimes T_{Q1}) \oplus (\theta_7(l-1) \otimes 0) \\ \theta_7(l) = \theta_7(l) \otimes T_{Proc} \\ \theta_8(l) = \theta_7(l) \otimes T_{Q2} \\ \theta_9(l) = \theta_8(l) \otimes T_{Resp} \\ \theta_{10}(l) = (\theta_4(l) \otimes T_{W1}) \oplus (\theta_9(l) \otimes T_{Q3}) \\ \theta_{11}(l) = \theta_{10}(l) \otimes T_{W2} \end{cases} \tag{5}$$

Equations systems (4) and (5) are time-variant linear equations in (Max,+) algebra.

Remark II.C.1: in a Client-server NAS, a PLC may work as

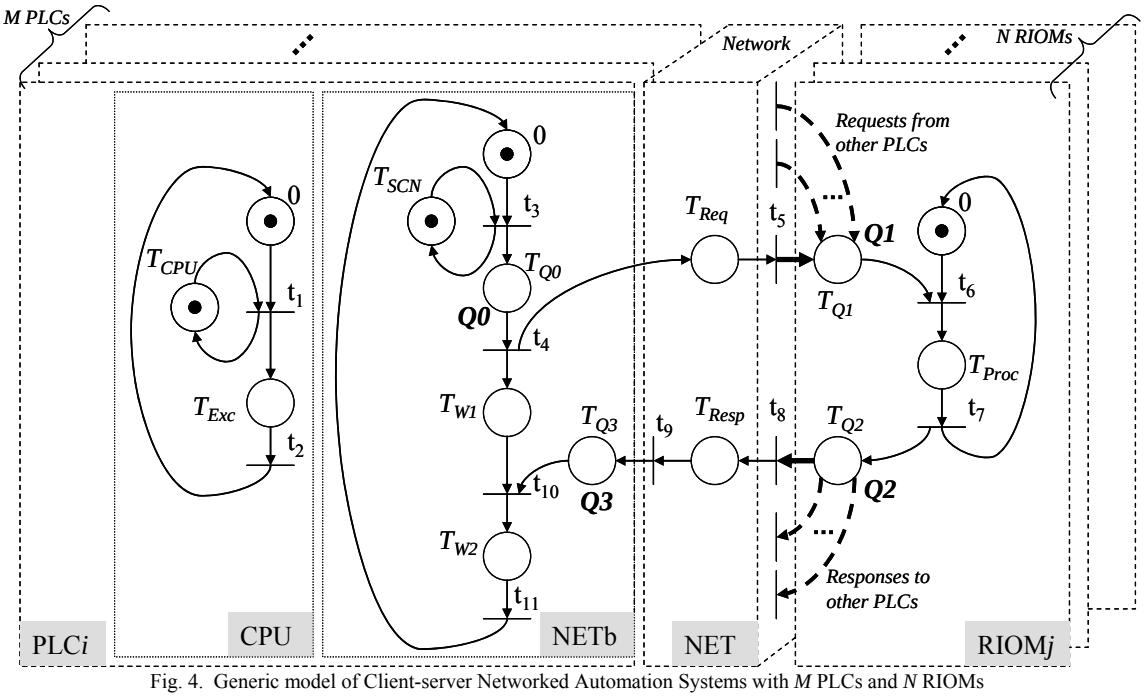

Fig. 4. Generic model of Client-server Networked Automation Systems with *M* PLCs and *N* RIOMs

a client (as explained previously) but also as a server of another client (e.g. a supervisor). In such a case, we can also consider the input/output buffers of the PLC, represented on Fig. 4 by places *Q3* and *Q0*. We firstly ascribe these places variable delays T_{Q0} and T_{Q3} which we subsequently fuse with delays T_{Req} and T_{Resp}, exactly as explained previously with the RIOM to get finally to the mono-PLC model.

III. Deterministic evaluation of response time bounds

As a first step, the systems of equations (4) and (5) have to be solved in order to determine the transition firing dates as functions of indices *k* and *l*. The following solutions are obtained under the hypotheses of Section II.A (especially the periodicity, without clocks drift, of the CPU and the NETb):

$$\begin{cases} \theta_1(k) = (k-1)\cdot T_{CPU} \\ \theta_2(k) = ((k-1)\cdot T_{CPU}) \otimes T_{Exc} \end{cases} \quad (6)$$

$$\begin{cases} \theta_3(l) = ((l-1)\cdot T_{SCN}) \\ \theta_4(l) = \theta_3(l) \otimes T_{Q0} \\ \theta_5(l) = \theta_4(l) \otimes T_{Req} \\ \theta_6(l) = \theta_5(l) \otimes T_{Q1} \\ \theta_7(l) = \theta_7(l) \otimes T_{Proc} \\ \theta_8(l) = \theta_7(l) \otimes T_{Q2} \\ \theta_9(l) = \theta_8(l) \otimes T_{Resp} \\ \theta_{10}(l) = (\theta_4(l) \otimes T_{W1}) \oplus (\theta_9(l) \otimes T_{Q3}) \\ \theta_{11}(l) = \theta_{10}(l) \otimes T_{W2} \end{cases} \quad (7)$$

The next step consists of fusing solutions (6) and (7) so as to express the link between the CPU and the NETb. Among these solutions, only the equations representing the following events are then of interest (at this step):

- Beginning reading inputs by the CPU (θ_1)
- End of program execution by CPU and output update (θ_2)
- Beginning of scanning cycle and transmitting a request (θ_3)
- Reception of a response in the cache memory (θ_{10}).

These are indeed the events that represent the communication between the CPU and the NETb. When a response arrives into the cache of the CPU (θ_{10}), it is taken into account at the next CPU cycle beginning (θ_1) and then read and used in the CPU execution program. Once processing has been completed, the result is written in the NETb cache memory (θ_2) before being transmitted to the RIOM at the next scanning cycle beginning (θ_3).

Let us set T_{RTT} as the round-trip time i.e. the wait time between the beginning of transmitting requests and receiving the response from $RIOM_S$ (the event source):

$$T_{RTT} = T_{Q0S} + T_{ReqS} + T_{ProcS} + T_{RespS} \quad (8)$$

The following equations are then derived:

$$\begin{cases} \theta_1(k) = (k-1)\cdot T_{CPU} \\ \theta_2(k) = (k-1)\cdot T_{CPU} \oplus T_{Exc} \end{cases} \quad (9)$$

$$\begin{cases} \theta_3(l) = (l-1)\cdot T_{SCN} \\ \theta_{10S}(l) = (l-1)\cdot T_{SCN} \otimes (T_{RTT} \oplus T_{Q0S} \otimes T_{W1}) \end{cases} \quad (10)$$

Let us set: $T_{SCN} = r \cdot T_{CPU}$, with $r \in \mathbb{N}$ (hypothesis *i*) and: $T_R = T_{RTT} \oplus T_{Q0} \otimes T_{W1} = (\alpha + \gamma)\cdot T_{CPU}$, $T_{CLC} = \beta \cdot T_{CPU}$, where $\beta < 1$, α is the integer part of T_R / T_{CPU} and γ its fractional part.

At the l^{th} scanning cycle, the response from $RIOM_S$ is received at date $\theta_{10S}(l)$. To be taken into account by the CPU, it must however wait for the m_l^{th} CPU cycle beginning that is immediately subsequent to $\theta_{10}(l)$. In other words, m_l is the smallest integer k so that: $\theta_1(k) > \theta_{10S}(l)$. By taking: $k-1 = (l-1)\cdot r + \alpha + 1$, we have:

$$\theta_1(k) = \theta_{10S}(l) + (1-\gamma)\cdot T_{CPU} \quad (11)$$

$$\theta_1(k-1) = \theta_{10S}(l) - \gamma \cdot T_{CPU} \quad (12)$$

Since $0 < 1-\gamma < 1$, then (11) implies that $\theta_1(k) > \theta_{10S}(l)$ and since (12) shows $\theta_1(k-1) < \theta_{10S}(l)$, then the sought number k is $m_l = (l-1)\cdot r + \alpha + 2$. So, the received response is taken into account by firing transition t_1 at date $\theta_1(m_l)$. It follows that the corresponding consequence is processed in the CPU at date:

$$\theta_2(m_l) = \theta_1(m_l) + T_{Exc} = \left[1 + \alpha + \beta + (l-1)\cdot r\right]\cdot T_{CPU} \quad (13)$$

Once again, this processed consequence is taken into account by the NETb, to be sent to its destination $RIOM_D$, only at the beginning of the immediate next scanning cycle i.e. next firing of transition t_3. So, we have to look for another number, let us note it q_l, so that q_l be the minimal number verifying $\theta_3(l+q_l) > \theta_2(m_l)$ (obviously $q_l \geq 1$).

The equations in (7) provide:

$$\theta_3(l+q_l) = (l+q_l-1)\cdot r \cdot T_{CPU} \quad (14)$$

So, by combining (13) and (14), number q_l must verify:

$$(l+q_l-1)\cdot r > 1 + \alpha + \beta + (l-1)\cdot r$$

One can check easily that this inequality is equivalent to:

$$q_l > (1+\alpha+\beta)/r \quad (15)$$

Thus, q_l being the minimal integer that verifies this condition, the consequence is sent to its destination at date $\theta_3(l+q_l)$ and therefore gets to it at date $\theta_{7D}(l+q_l)$. Finally, with an event generated in the plant at date $\theta_{6S}(l-1) + \tau - T_{filt}$ (the event is generated with delay $\tau - T_{filt}$ after the arrival of the previous request), the response time with regard to the l^{th} cycle is:

$$D_r(l) = \theta_{7D}(l+q_l) - (\theta_{6S}(l-1) + \tau - T_{filt}) \quad (16)$$

Also, from (7) we can obtain:

$$\theta_{7D}(l+q_l) = (l-1+q_l)\cdot T_{SCN} + T_{Q0D} + T_{ReqD} + T_{ProcD}$$

$$\theta_{6S}(l-1) = (l-2)\cdot T_{SCN} + T_{Q0S} + T_{ReqS}$$

Finally, by replacing these expressions in (16), we get to:

$$D_r(l) = (q_l+1)\cdot T_{SCN} + T_{Q0D} - T_{Q0S} + T_{ProcD} + T_{filt} + \Delta \quad (17)$$

where $\Delta = T_{ReqD} - T_{ReqS} - \tau$.

The response time in (17) is minimal provided that the data originating from the detector are used for request processing in $RIOM_S$ immediately after being generated, i.e. at date $\theta_{6S}(l) - (T_{filt} + 0^+)$. The minimum delay relative to the l^{th} scanning cycle therefore equals:

$$D_{MIN}(l) = q_l \cdot T_{SCN} + T_{Q0D} - T_{Q0S} + T_{ProcD} + T_{filt} + T_{ReqD} - T_{ReqS} \quad (18)$$

On the other hand, the response time is maximal if the data are generated a long time before the arrival of the request. The worst case is when the data are generated a bit after the arrival of the previous request (of the $(l\text{-}1)^{th}$ cycle) i.e. at date $\theta_{6S}(l-1) - (T_{filt} - 0^+)$. So, the maximal bound is:

$$D_{MAX}(l) = \theta_{7D}(l + q_l) - \theta_{6S}(l-1) + T_{filt} \quad (19)$$

This leads finally to:

$$D_{MAX}(l) = (q_l + 1) \cdot T_{SCN} + T_{Q0D} - T_{Q0S} + T_{ProcD} + T_{filt} + T_{ReqD} - T_{ReqS} \quad (20)$$

Actually, (18) and (20) give only local bounds of the response time i.e. relative to the l^{th} cycle. The absolute bounds are calculated as:

$$\bar{D}_{MAX} = (q_{\max} + 1) \cdot T_{SCN} + T_{Q0D} - T_{Q0S} + T_{ProcD} + T_{filt} + \Delta_{\max} \quad (21)$$

$$\bar{D}_{MIN} = (q_{\min} + 1) \cdot T_{SCN} + T_{Q0D} - T_{Q0S} + T_{ProcD} + T_{filt} + \Delta_{\min} \quad (22)$$

where $q_{\max} = \max_{l \in \mathbb{N}} \{q_l\}$, $q_{\min} = \min_{l \in \mathbb{N}} \{q_l\}$,

$\Delta_{\max} = \max_{l \in \mathbb{N}} \{T_{ReqD} - T_{ReqS}\}$ and $\Delta_{\min} = \min_{l \in \mathbb{N}} \{T_{ReqD} - T_{ReqS}\}$.

Remark III.1: since delays T_{ReqD} and T_{ReqS} in $\Delta_{\max}$ are experienced by requests sent during different cycles, they are independent from each other (consequence of hypothesis *v*). So, $\Delta_{\max}$ can be calculated as the difference between the maximal value of T_{ReqD} and the minimal value of T_{ReqS}. The probability of such a coincidence may be negligible but we have to consider it to guarantee the maximal bound overestimation. The case study of Section V will illustrate this consideration.

IV. PROBABILISTIC EVALUATION OF RESPONSE TIME

In the previous section, we exposed a method to analyze the reactivity of the NAS and we provided formulae of the bounds of response time. As aforementioned in Section II.*A* with NAS example of Fig. 1, the distribution of the response time may also be needed to check if a non-desired event occurs with an enough low probability. So, a probabilistic evaluation is required and this is the objective of this section.

The analysis of Section III is based on a time-variant system. So, the probabilistic study of the current section will be based on the previously obtained results.

Formula (17) expresses the response time relative to the l^{th} cycle as:

$$D_r(l) = (q_l + 1) \cdot T_{SCN} + T_{Q0D} - T_{Q0S} + T_{ProcD} + T_{filt} + \Delta \quad (23)$$

where $\Delta = T_{ReqD} - T_{ReqS} - \tau$.

Since T_{ProcD} and T_{filt} are constant (hypothesis *vi*) and so is T_{Q0} (the PLC is only a client), the only significant random variables in (23) are q_l and Δ. So, they are the key variables to consider when looking for $P(D_r \geq D_{\lim})$ i.e. the probability that D_r is over a given limit $D_{\lim}$. Obviously, $D_{\lim}$ must be in the neighborhood of D_{MAX} to make sense. Otherwise, we know beforehand that the probability $P(D_r \geq D_{\lim})$ is enough high to not fulfill the requirements of the NAS. So, we suppose that $D_{\lim}$ is close to $\bar{D}_{MAX}$ by setting:

$$D_{\lim} = (q_{\max} + 1) \cdot T_{SCN} + T_{Q0D} - T_{Q0S} + T_{ProcD} + T_{filt} + \Delta_{\lim} \quad (24)$$

where $q_{\max}$ is same as in (21) and obviously $\Delta_{\lim}$ is by far smaller than T_{SCN} (consequence of *hypothesis v*).

Lemma:

$$D_r \geq D_{\lim} \quad \Leftrightarrow \quad (q_l = q_{\max}) \quad AND \quad (\Delta \geq \Delta_{\lim})$$

Proof:

1) The proof of: $(q_l = q_{\max}) \quad AND \quad (\Delta \geq \Delta_{\lim}) \Rightarrow D_r \geq D_{\lim}$ is straightforward. We will prove the opposite implication.

2) $D_r \geq D_{\lim}$ implies that:

$$\begin{aligned}&(q_l + 1) \cdot T_{SCN} + T_{Q0D} - T_{Q0S} + T_{ProcD} + T_{filt} + \Delta \geq \\ &(q_{\max} + 1) \cdot T_{SCN} + T_{Q0D} - T_{Q0S} + T_{ProcD} + T_{filt} + \Delta_{\lim}\end{aligned} \quad (25)$$

It follows that:

$$(q_l + 1) \cdot T_{SCN} \geq (q_{\max} + 1) \cdot T_{SCN} + \Delta_{\lim} - \Delta \,. \quad (26)$$

Since $\Delta \ll T_{SCN}$, then: $-\Delta \gg -T_{SCN}$ and finally: $\Delta_{\lim} - \Delta \gg -T_{SCN}$.

By combining this with (26), we get to:

$(q_l + 1) \cdot T_{SCN} > (q_{\max} + 1) \cdot T_{SCN} - T_{SCN}$

This leads to: $(q_l + 1) > q_{\max}$.

Since we know by definition that $q_l \leq q_{\max}$, then we have necessarily $q_l = q_{\max}$.

By replacing q_l with $q_{\max}$ in (26), we get finally to:

$\Delta \geq \Delta_{\lim}$. ■

Corollary:

$$P(D_r \geq D_{\lim}) = P(q_l = q_{\max}) \cdot P(\Delta \geq \Delta_{\lim})$$

Proof: The proof of this corollary is straightforward using the previous lemma and by taking into account the fact that the events $(q_l = q_{\max})$ and $(\Delta \geq \Delta_{\lim})$ are independent for almost the same reasons explained in *Remark III.1* (the delays involved in these two parameters are experienced during different scanning cycles).

At this step, we can calculate $P(D_r \geq D_{\lim})$ provided that we get $P(q_l = q_{\max})$ and $P(\Delta \geq \Delta_{\lim})$. Let us calculate them using solely the known probability densities.

1) $P(q_l = q_{\max})$?

We know from (15) that: $(q_{\max} - 1) \leq (1 + \alpha + \beta)/r < q_{\max}$, which is rewritten as: $(q_{\max} - 1) \cdot r - 1 \leq (\alpha + \beta) < q_{\max} \cdot r - 1$.

Since $\beta < 1$ and $\alpha, r \in \mathbb{N}$, then: $\alpha = (q_{\max} - 1) \cdot r - 1$. Recall that α is the integer part of T_R / T_{CPU}.

So, $\alpha = (q_{\max} - 1) \cdot r - 1$ is equivalent to:

$$(q_{\max} - 1) \cdot r - 1 \le T_R / T_{CPU} < (q_{\max} - 1) \cdot r \quad (27)$$

This can also be rewritten as:

$$((q_{\max} - 1) \cdot r - 1 \cdot T_{CPU}) \le T_R < (q_{\max} - 1) \cdot r \cdot T_{CPU} \quad (28)$$

Let us set: $a = ((q_{\max} - 1) \cdot r - 1) \cdot T_{CPU}$, $b = (q_{\max} - 1) \cdot r \cdot T_{CPU}$.

So, (28) is written as: $a \le T_R < b$

Since T_R is originally expressed as: $T_R = T_{RTT} \oplus (T_{Q0S} \otimes T_{W1})$, we can equivalently write:

$$a \le T_{RTT} \oplus (T_{Q0S} \otimes T_{W1}) < b \quad (29)$$

We can check easily that this is equivalent to:

$$\begin{cases} a \le T_{RTT} < b & if \quad (T_{Q0S} \otimes T_{W1}) \le a \\ T_{RTT} < b & if \quad a < (T_{Q0S} \otimes T_{W1}) < b \end{cases}$$

Let $f_{RTT}(t)$ be the density of probability of variable T_{RTT}. Finally, we can express the sought probability as:

$$P(q_l = q_{\max}) = \begin{cases} \int_a^b f_{RTT}(t) \cdot dt & if \quad (T_{Q0S} \otimes T_{W1}) \le a \\ \int_0^b f_{RTT}(t) \cdot dt & if \quad a < (T_{Q0S} \otimes T_{W1}) < b \end{cases} \quad (30)$$

where $T_{Q0S} \otimes T_{W1}$ is constant and well known (it is simply the necessary time to send all the requests from the NETb).

Hence, the only we need to calculate (30) is the probability density $f_{RTT}(t)$ of the round trip time. It is either given (the round trip time is a key measure in Client-server protocol) or calculated using sum (7) and the convolution (explained below in subsection 2) of the probability densities of the elementary delays that compose it.

Remark IV.1: Result (30) is used only if we have $q_{\max} \ge 2$. In case $q_{\max} = 1$, we have always $q_l = 1$ (since we have also $q_l \ge 1$) and therefore: $P(q_l = q_{\max}) = 1$. So, $P(D_r \ge D_{\lim}) = P(\Delta \ge \Delta_{\lim})$. In such a case, we also get $D_r = Constant + \Delta$ and therefore the density of probability distribution of the response time D_r is the same as the density of probability of Δ but shifted with a constant:

$$f_{Dr}(t) = f_{\Delta}(t - Constant) \quad (31)$$

with: $Constant = (q_{\max} + 1) \cdot T_{SCN} + T_{Q0D} - T_{Q0S} + T_{ProcD} + T_{filt}$.

2) $P(\Delta \ge \Delta_{\lim})$?

The calculus of probability $P(\Delta \ge \Delta_{\lim})$ is much easier. We know that the density of probability of a sum $z = x + y$ of two independent random variables x, y whose densities are respectively f_x, f_y, is the convolution given by:

$$f_z(t) = (f_x * f_y)(t) = \int_{-\infty}^{+\infty} f_x(t - w) \cdot f_y(w) \cdot dw \quad (32)$$

From (23), we have: $\Delta = T_{ReqD} - T_{ReqS} - \tau$.

The delays T_{ReqD} and T_{ReqS} are independent (*Remark III .1*) and the lag τ is solely dependent on the controlled plant. Indeed, the date of occurrence of an event in the plant is definitely independent from the NAS. The NAS does only react to events from sensors. So, the three delays composing Δ are independent from each other.

By setting: $x = T_{ReqD}$, $y = -T_{ReqS}$ and $z = -\tau$, we then get to:

$$f_{\Delta}(t) = (f_x * f_y * f_z)(t) = \int_{-\infty}^{+\infty} f_z(t - w) \cdot \int_{-\infty}^{+\infty} f_x(w - u) \cdot f_y(u) du\, dw \quad (33)$$

The sought probability $P(\Delta \ge \Delta_{\lim})$ is therefore given by:

$$P(\Delta \ge \Delta_{\lim}) = \int_{\Delta_{\lim}}^{+\infty} f_{\Delta}(t)\, dt \quad (34)$$

Finally, the probability $P(D_r \ge D_{\lim})$ is calculated simply using the previous corollary and results (30) and (34).

V. CASE STUDY

To check the validity of the different results exposed previously, we consider the automation architecture described in Section II.*A* (Fig. 1). Three approaches are considered to perform evaluation of the response time distribution whereas formulae (22), (21) are used to assess respectively the minimal and maximal bound:

i) Experimental (Fig. 4(a)): about 10,200 measures have been taken from a dedicated platform (such a platform is described in detail in [13]). The two main parameters we can tune are the periods T_{SCN} and T_{CPU}. We carried out a lot of measurements by varying them. For length limitation reasons, we will expose only one case with $T_{SCN} = 30ms$ and $T_{CPU} = 5ms$.

ii) Simulations (Fig. 4(b)): in addition to the constant parameters $T_{SCN} = 30ms$, $T_{CPU} = 5ms$, $T_{Exc} \approx 3ms$, $T_{Proc} = 0.7ms$, $T_{filt} = 0.06ms$, we first generated 10,200-length random vectors (according to a given distribution nevertheless, given below while explaining the analytic approach) to represent the different local variable delays of the model on Fig. 4. Then, we simulated the behaviour of the model using equations (6) and (7). Finally, we deduced the delay corresponding to each cycle using (17). At the end of simulation, we represented the obtained response times in the form of histograms (distribution shape of Fig. 4(b)).

iii) Analytic (Fig. 4(c)): this method corresponds simply to the analysis exposed in Section IV. We use (30) and (33) to calculate analytically the distribution of the response time. For this purpose, we consider the following hypotheses:

- The lag τ is uniformly distributed over domain $[0, T_{SCN}]$:

$$f_{\tau}(t) = \begin{cases} 1/T_{SCN} & if \ t \in [0, T_{SCN}] \\ 0 & otherwise \end{cases}$$

This type of distribution is motivated by the fact that the experimentation was carried out with an event generator using this uniform pattern. Obviously, another density may be used and it all depends on the expertise and the knowledge one has

about the occurrence of the events from the automated plant.

- Delays T_{ReqD}, T_{ReqS} and T_{Resp} are supposed Gaussian with distributions $f_{(\mu 1,\sigma 1)}$, $f_{(\mu 2,\sigma 2)}$ and $f_{(\mu 3,\sigma 3)}$ respectively (according to experimental observations) given as: $f_{(\mu,\sigma)}(t) = 1/(\sigma\sqrt{2\pi}) \cdot \exp[-(t-\mu)^2/(2\sigma^2)]$

where μ is the mean of the delay and σ its standard deviation.

It can be shown (see Appendix A) using (34) and these distributions that the final expression of $f_\Delta(t)$ is given by:

$$f_\Delta(t) = 1/T_{SCN} \cdot \int_t^{t+T_{SCN}} f_{(\bar{\mu},\bar{\sigma})}(w) \cdot dw \qquad (35)$$

where: $\bar{\mu} = \mu 1 - \mu 2$ and $\bar{\sigma}^2 = \sigma 1^2 + \sigma 2^2$.

Also, (35) can be rewritten as:

$$f_\Delta(t) = 1/(2T_{SCN}) \cdot [erf\left(\frac{t+T_{SCN}-\bar{\mu}}{\bar{\sigma}\sqrt{2}}\right) - erf\left(\frac{t-\bar{\mu}}{\bar{\sigma}\sqrt{2}}\right)] \qquad (36)$$

where $erf(t) = 2/\sqrt{\pi} \cdot \int_0^t \exp(-w^2) \cdot dw$ is the so called *Gauss error function.* This transformation is motivated by the fact that function $f_\Delta(t)$ is not analytic since $f_{(\mu,\sigma)}(w)$ cannot be integrated analytically. So, it is calculated only numerically using $erf(t)$ function which is implemented in most mathematics software. The result of calculation of (36) using Matlab gives the curve drawn on Fig. 4(c). It is noted $f(t)$ and represents the density of probability of the response time calculated under conditions of *Remark IV. 1*.

- We also used formulae (21) and (22) to calculate the bounds and obtained: $\bar{D}_{MAX} = 62.51\,ms$ and $\bar{D}_{MIN} = 29.31\,ms$ with $\Delta_{Max} = -\Delta_{Min} \approx 1.5\,ms$ being calculated as explained in *Remark III.1.* The minimum and maximum values of the network delays are determined using a (Max,+) algebra based method [19], suitable for modeling and analysis of systems involving shared resources as is the case in our NAS.

Discussion of results:

The maximal bound $\bar{D}_{MAX} = 62.51\,ms$ obtained using formula (21) is higher than all the values obtained using the three considered methods. The probability to reach such a value is negligible but the best satisfaction is that the experimental bound of $61.80\,ms$ be overestimated with only about 1%. The same remarks can be made with regard to the minimal bound of response time.

We can point out in this example the risk of using simulation to calculate the bounds. Indeed, we can notice that a rare event can be swept (the case of the minimal bound $29.90\,ms$) even if its probability is very low but this is not guaranteed at all. This is what can be noticed with regard to the simulated maximal bound $61.79\,ms$ which smaller than the experimental maximal bound. So, the real maximal bound is not swept by simulation. Nevertheless, the simulated distribution of the response time is satisfactory.

To remedy to simulation drawbacks, we can use the results related to the density of probability calculus. The curve of Fig. 4(c) shows (visually) indeed that the maximal bound is around $62.00\,ms$ and the minimal bound is around $30.00\,ms$. We can also point out other advantages; the formula to obtain the curve is easily used (without thousands of simulations) and the shape of the curve complies with the experimental results.

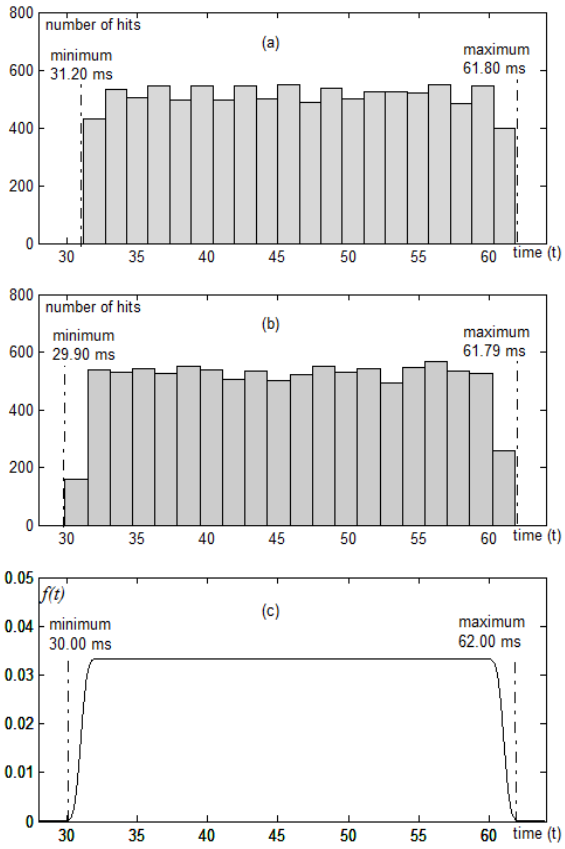

Fig. 4. Results of evaluation: (a) experimental, (b) simulation, (c) analytic.

VI. Conclusion

In this paper, we have presented a generalization of our former work regarding evaluation of response time in Client-server networked automation systems. Moreover, we presented a deterministic and probabilistic analysis, depending on whether strict bounds or only a distribution is needed. In both cases, we provided formulae to calculate directly and easily the bounds and the density of probability distribution of the response time.

For future work, it would be worthwhile to consider more general automation architectures, with other protocols like Producer-consumer and compare the results to the observations provided for instance in [20].

Appendix A

Let us prove expressions (35) and (36). We have:

$$f_x(t) = f_{(\mu 1,\sigma 1)}(t) = 1/(\sigma 1\sqrt{2\pi}) \cdot \exp[-(t-\mu 1)^2/(2\sigma 1^2)]$$

$$f_y(t) = f_{(-\mu 2,\sigma 2)}(t) = 1/(\sigma 2\sqrt{2\pi}) \cdot \exp[-(t+\mu 2)^2/(2\sigma 2^2)]$$

$$f_z(t) = f_{-\tau}(t) = \begin{cases} 1/T_{SCN} & if\ t \in [-T_{SCN}, 0] \\ 0 & otherwise \end{cases}$$

From (33) we have: $f_\Delta(t) = (f_x * f_y * f_z)(t)$. We know that the convolution of two Gaussians is also a Gaussian: $f_{(\mu 1,\sigma 1)} * f_{(-\mu 2,\sigma 2)}(t) = f_{(\bar{\mu},\bar{\sigma})}(t)$ with: $\bar{\mu} = \mu 1 - \mu 2$ and $\bar{\sigma}^2 = \sigma 1^2 + \sigma 2^2$. Hence, we get:

$$f_\Delta(t) = (f_{(\bar{\mu},\bar{\sigma})} * f_z)(t) = \int_{-\infty}^{+\infty} f_z(t-w) \cdot f_{(\bar{\mu},\bar{\sigma})}(w) \cdot dw \qquad (1)$$

The term $f_z(t-w)$ is non zero only if $-T_{SCN} \le t - w \le 0$ or equivalently: $t \le w \le t + T_{SCN}$. If this condition is respected, then $f_z(t-w) = 1/T_{SCN}$ and consequently, (1) becomes:

$$f_\Delta(t) = \int_{t}^{t+T_{SCN}} 1/T_{SCN} \cdot f_{(\bar{\mu},\bar{\sigma})}(w) \cdot dw \qquad \blacksquare$$

This can also be written as:

$$f_\Delta(t) = 1/T_{SCN} \cdot [\int_{0}^{t+T_{SCN}} \cdot f_{(\bar{\mu},\bar{\sigma})}(w) \cdot dw - \int_{0}^{t} \cdot f_{(\bar{\mu},\bar{\sigma})}(w) \cdot dw] \qquad (2)$$

We also know that: $\int_{0}^{t} \cdot f_{(\bar{\mu},\bar{\sigma})}(w) \cdot dw = \frac{1}{2}[1 + erf(\frac{t-\bar{\mu}}{\sigma\sqrt{2}})]$

It follows that (1) is written as:

$$f_\Delta(t) = \frac{1}{2T_{SCN}} \cdot [erf\left(\frac{t+T_{SCN}-\bar{\mu}}{\bar{\sigma}\sqrt{2}}\right) - erf\left(\frac{t-\bar{\mu}}{\bar{\sigma}\sqrt{2}}\right)]. \qquad \blacksquare$$

REFERENCES

[1] P. Neumann, "Communication in industrial automation - what is going on?", *Control Engineering Practice*, 15(11), pp. 1332-1347, 2007.

[2] R. L. Cruz, "A calculus for network delay, Part I: network in isolation", *IEEE Trans. Information Theory*. 37(1), pp. 114-131, 1991.

[3] J. Y. Le Boudec and P. Thiran, "Network calculus: a theory of deterministic queuing systems for internet", Springer Verlag, 2004.

[4] J. P. Georges, E. Rondeau, and T. Divoux, "Confronting the performances of switched Ethernet network with industrial constraints by using Network Calculus", *Communication Systems,* 18(9), pp. 877-903, 2005.

[5] X. Fan, M. Jonsson and J. Jonsson, "Guaranteed real-time communication in packet switched networks with FCFS queuing", *Computer and networks*, 53(3), pp. 400-417, 2008.

[6] K. C. Lee and S. Lee, "Performance evaluation of switched Ethernet for real-time industrial communications", *Computer standards & interfaces*, Vol. 24(5) ,pp. 411–423, 2002.

[7] J. Jasperneite and P. Neumann: Switched Ethernet for Factory Communication. *Proc. of 8th IEEE Int. Conf. Emerging Technologies and Factory Automation, ETFA* , Antibes, France, pp. 205 – 212, 2001.

[8] D.A. Zaitsev, "Switched LAN simulation by colored Petri nets", *Mathematics and Computers in Simulation*, 65(3), pp. 245–249, 2004.

[9] G. Marsal, B. Denis, J. M. Faure, G. Frey, "Evaluation of Response Time in Ethernet-based Automation Systems, *11th IEEE International Conference on Emerging Technologies and Factory Automation*, *ETFA06*, Prague, Czech Republic, pp. 380-387, 2006.

[10] 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.

[11] 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, pp. 372-379, 2006.

[12] J. Greifeneder and G. Frey: Probabilistic Timed Automata for Modeling Networked Automation Systems. *Preprints of the 1st IFAC Workshop on Dependable Control of Discrete Systems DCDS*, pp. 143-148, Cachan, France, June 2007.

[13] 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, pp. 332-339, 2007.

[14] B. Addad, S. Amari and J. J. Lesage, "Analytic calculus of response time in networked automation systems", *IEEE Trans. Automation Science and Engineering*, 7(4), pp. 858-869, 2010.

[15] T. Murata, "Petri nets Properties analysis and applications", *Proceedings of the IEEE*, 77(4), pp. 541 – 580, 1989.

[16] 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.

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

[18] D. A. Zaitsev, A. I. Sleptsov, "State equations and equivalent transformations of timed Petri nets", *Cybernetics and Systems Analysis*, 33(5), pp. 659-672, 1997.

[19] B. Addad, S. Amari and J. J. Lesage, "Linear Time-Varying (Max,+) Representation of Conflicting Timed Event Graphs", *10th Int. Workshop on Discrete Event Systems,* Berlin, Germany, pp.310-315, 2010.

[20] J. Greifeneder and G. Frey, "Reactivity Analysis of different Networked Automation System Architectures", *Proc. 13th IEEE Int. Conf. Emerging Technologies and Factory Automation (ETFA)*, Hamburg, Germany, pp. 1031-1038, 2008.

Boussad Addad received the Engineer degree in control systems from the National Polytechnic School of Algiers, Algiers, Algeria, in 2007, and the M.S degree in automated systems engineering from the Ecole Normale Supérieure de Cachan, Cachan, France, in 2008. He is currently working towards the Ph.D. degree at the Ecole Normale Supérieure de Cachan, Cachan.

His research interests include modeling and analysis of discrete event systems along with time performance evaluation of networked automation systems.

Saïd Amari received the Ph.D. degree from the Institute of Research in Communications and Cybernetic of Nantes, Nantes, France, in 2005.

He is currently an Associate Professor at the University of Paris XIII. He carries out research at the Automated Production Research Laboratory from the École Normale Supérieure de Cachan. His main research interests are performance evaluation and control of Discrete Event Systems with Petri nets and Dioid algebra.

Jean-Jacques Lesage (M'07) received the Ph.D. degree in 1989, and the "Habilitation à Diriger des Recherches" in 1994 from the Université Nancy 1 Henri Poincaré, Nancy, France, in 1989.

He is currently a Professor of Automatic Control at the École Normale Supérieure de Cachan, Cachan, France. His research topics are formal methods and models of Discrete Event Systems (DES), both for modeling synthesis and analysis. The common objective of his works is to increase the dependability of the DES control.