Analytic Calculus of Response Time in Networked Automation Systems

Boussad Addad, Saïd Amari, Jean-Jacques Lesage

Analytic Calculus of Response Time in Networked Automation Systems

B. Addad, S. Amari and J-J. Lesage, *Member, IEEE*

***Abstract* —This paper presents a novel approach to evaluate the response time in networked automation systems (NAS) that use a client/server protocol. The developments introduced are derived from modeling the entire architecture in the form of timed event graphs (TEGs), as well as from the resulting state representation in Max-Plus algebra. The various architectural stages are actually modeled in a very abstract pattern, which yields just those TEG models where local delays are sufficient to perform the overall evaluation. In this manner, linear Max-Plus equations are obtained. A thorough analysis of these equations has led to analytical formulas for direct calculus of NAS response time. As a final step, experimental measurements taken on a laboratory facility have been used to verify the validity of the results. In conclusion, the benefit and effectiveness of this novel method have been demonstrated.**

***Note to Practitioners*—In this work, we present an overall study of networked automation systems working according to client/server paradigm. Unlike systems where a global scheduling of the shared resources is available and the delays well handled, in such systems it is not the case and the investigations to evaluate their real-time performances are required. Actually, these systems are very present in industry but the efforts to deal with this issue are rare and often informal, based on simulation of particular cases. In our work, we assess a major criterion of their time performances, the response time. We give a formal evaluation of this feature through an analytic approach. The results we present are generic and fit the experimental observations in different cases.**

***Index Terms*—Networked Automation Systems, Response Time Evaluation, Max-Plus Algebra, Timed Events Graph, Switched Ethernet Networks.**

I. Introduction

THE new trend within industrial organization networks consists of using the same technology at all levels of communication. A network solution supporting such a transparent vertical integration must be flexible and capable of simultaneously providing a high rate of data transfer in the upper levels and fast response times in the lower levels. The industrial Ethernet has established itself as such a new generation of fieldbuses, and many of them are currently meeting the needs of most automation applications. The increase in information transfer speed (Giga Ethernet), along with both the use of fully-duplex networks that prevent frames from colliding and low component costs, offered the major incentives behind the use of Ethernet in industry. Every Ethernet solution in fact features its own set of advantages and disadvantages. Generally speaking, as the solution becomes more compatible with standard Ethernet, its real-time performance achievement actually drops and *vice versa* [1]. These solutions include "Modbus over Ethernet", an application protocol that makes use of the client/server paradigm. It is simple, accessible and open enough to facilitate vertical integration [2]; however, it is not suitable for strict real-time applications like motion control, yet entirely adequate for the majority of industrial automation systems. With such a protocol therefore, resource scheduling is unavailable and considerable delays due to unsynchronized or unavailable resources are caused, complicating the evaluation of message delay encountered in each system component and consequently impeding an evaluation of the entire system response time. A number of research efforts have been undertaken to assess these NAS delays through the use, for example, of widely-accepted network calculus [3], [4], [5], worst case methods [6], [7] and simulation [8], [9]. Like the majority of studies however, these efforts have focused on just the end-to-end delays or network effect and neglect both the controllers (e.g. PLCs or programmable logic controllers) and RIOMs (remote input output modules). As a matter of fact, the PLC modules are not synchronized and RIOMs may be shared across many applications, which leads to delays that must then be incorporated. To the best of our knowledge, studies that consider the entire architecture (both field devices and the network) are still quite rare and often informally based on case simulation or experimental measurements targeting a limited number of systems [10], [11]. The only formal method developed has been based on model-checking [12] aiming to check if a timing property holds or not. The disadvantage of this approach is its failure to provide the response times distribution and its classical state explosion problem, limiting its applicability to relatively simple cases. This method was

Manuscript received February 15, 2009; revised November 24, 2009. This paper was recommended for publication by Associate Editor A. Chacravarty and Editor M. Zhou upon the reviewers' comments.

B. Addad is 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 (phone: +33-147402752, e-mail: said.amari@lurpa.ens-cachan.fr).

J. J. Lesage is with Automated Production Research Laboratory, 61 av. du Président Wilson, 94235 Cachan Cedex, France (phone: +33-147402218; e-mail: Jean.Jacques.lesage@lurpa.ens-cachan.fr).

enhanced afterwards in [13] and used to calculate the distribution of response times. The complexity of NAS is reduced by focusing on the important events but the state explosion problem still exists.

The objective of the present study is therefore twofold: to offer a formal method of evaluation, and to avoid the state explosion problem. Moreover, a novel method will be proposed in order to analytically assess both the response time bounds (i.e. min and max bounds) and the distribution shape. For this purpose, we have employed a special class of Petri nets (Timed Events Graphs or TEGs) to model the system. The behavior of TEGs can indeed be studied using linear equations (within Max-Plus algebra), making it suitable to analytically evaluate the temporal properties.

The current study extends our preliminary work [14]. We relax many hypotheses (variable network delays, variable processing delays …) while considering more complex architectures (many servers).

The remainder of our study has been organized as follows. In Section II, some of the fundamentals regarding TEGs and Max-Plus algebra will be recalled. Next, Section III and Section IV will explore architectural modeling through the use of TEGs. We will begin by studying a system whose controller sends requests to just one server or RIOM in Section III, before considering a more complex architecture involving any number of servers in Section IV. Following resolution of the Max-Plus equations and fusion of the resulting solutions, a calculus algorithm and analytic formulae will be given. Since the delays caused by the network are needed to complete the evaluation, a method for accurately assessing these delays will be developed in Section V. Afterwards, Section VI will be devoted to validating results using real measurements recorded on a patented experimental platform[1]. Lastly, the outlook for future work will be discussed in Section VII as a conclusion to this paper.

II. TIMED EVENT GRAPHS AND MAX-PLUS ALGEBRA

An event graph is an ordinary Petri net with all places displaying at most one upstream and one downstream transition. So, a TEG is a Petri net without the presence of any conflicts or resource sharing. TEG behavior is deterministic and depends solely on the source transitions and initial conditions (TEG marking and tokens availability times) [16].

An event graph is timed if the places or transitions are assigned with delays. In our study, we are only considering timed-place graphs, yet we are still able to transform a timed place into a timed transition and *vice versa* [15]. For the modeling carried out in the sequel, we will only assign delays to places, and each place p_k will be ascribed a delay denoted τ_k.

In studying TEGs, the variable n generally denotes the number of regular transitions t_i with at least one place upstream, while m represents the number of source transitions t_{uj} without upstream places. To study the dynamic behavior of TEG, we have associated the firing date for the k^{th} time of each transition. This term is denoted $u_j(k)$ for a source transition and $\theta_i(k)$ for other transitions.

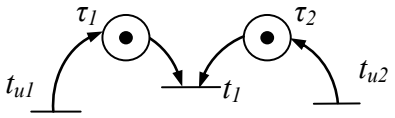

Fig. 1. A timed event graph.

With an initial marking of places as shown in Fig. 1 and given the dates $u_1(k-1)$, $u_2(k-1)$ of firing respectively, of transitions t_{u1} and t_{u2} for the $(k-1)^{th}$ time, then the date $\theta_1(k)$ of firing the transition t_1 for the k^{th} time at maximum speed can be deduced using the following equation ("at maximal speed" means as soon as all upstream place tokens are available).

$$\theta_1(k) = \max(\tau_1 + u_1(k-1), \tau_2 + u_2(k-1)) \quad (1)$$

The above equation is actually a linear equation in Max-Plus algebra. A new algebraic structure has indeed emerged around two laws: the classical maximum denoted in general by "$\oplus$" with identity element $\varepsilon = -\infty$; and the classical addition denoted by "$\otimes$" with identity element $e = 0$.

The previous equation (1) can then be rewritten as:

$$\theta_1(k) = (\tau_1 \otimes u_1(k-1)) \oplus (\tau_2 \otimes u_2(k-1)) \quad (2)$$

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

$$\theta(k) = \bigoplus_{\varphi \geq 0} (A_\varphi \otimes \theta(k-\varphi) \oplus B_\varphi \otimes u(k-\varphi)) \quad (3)$$

where the components of vectors $\theta(k)$ and $u(k)$ are the firing dates for the k^{th} time of the n and m TEG transitions. Matrix A_φ elements belong to $\overline{\mathbb{R}}_{\max} = \mathbb{R} \cup \{-\infty\}$, with element $A_{\varphi,ij}$ representing the delay τ_{ij} associated with place p_{ij} and connecting the transitions t_j and t_i (with the marking φ) should it exist, and with the neutral element ε otherwise. Similarly, for $B_\varphi \in \overline{\mathbb{R}}_{\max}^{n \times m}$, the matrix contains delays of places downstream of the source transitions.

In an analogous manner and as is customary in classical linear systems, this form can be brought to a state representation by replacing all places with markings $\varphi_{ij} > 1$ by φ_{ij} other places (with one token) and by $(\varphi_{ij} - 1)$ intermediate transitions. We thus obtain an extended system with a state vector $x(k) = (\theta(k) \quad \tilde{\theta}(k))^T$, where $\tilde{\theta}$ is the vector of added transitions. TEG can therefore be described by the first-order recursive equation (or standard form [16]):

[1] French patent #01 110 933. This platform is used to perform behavioral identification of discrete event systems and to measure time performances of networked automation systems.

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

This first-order form is quite similar to the state representation of linear systems in classical algebra and proves very useful for studying the time properties of discrete event systems [17], [18]. Further details on this algebra are available in [19].

The form (4) can now be rewritten in an explicit form with both the source transitions and initial conditions clarified:

$$x(k) = A^{k-1} \otimes x(1) \oplus \left[\bigoplus_{i=0}^{k-2} A^i \otimes B \otimes u(k-i) \right] \quad (5)$$

Example II.1: TEG in Fig. 2 represents a manufacturing system with a machine capable of processing two parts at once (two tokens in place p_2). The parts entering the upstream stock (place p_u) become available to the machine τ_u time units after firing the transition t_u. The process lasts τ_1 time units before the finished part exits the machine.

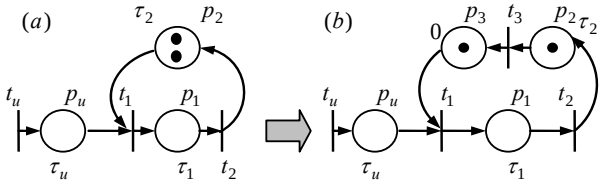

Fig. 2. Timed event graph decomposition.

The behavior of this system depends on the initial marking and the source transition t_u of TEG (Fig. 2a). The dates of firing transitions t_1 and t_2 for the k^{th} time (at maximum speed) are expressed as:

$$\begin{cases} \theta_1(k) = \tau_2 \otimes \theta_2(k-2) \oplus \tau_u \otimes u(k) \\ \theta_2(k) = \tau_1 \otimes \theta_1(k) \end{cases} \quad (6)$$

The equations (6) are linear but not of the form (4). The decomposition process explained previously is needed to get to this first order form. After decomposition (Fig. 2b), the transition t_3 is added to yield just those places with at most one token. The system can then be described by:

$$\begin{cases} \theta_1(k) = e \otimes \theta_3(k-1) \oplus \tau_u \otimes u(k)) \\ \theta_2(k) = \tau_1 \otimes \theta_1(k) \\ \theta_3(k) = \tau_2 \otimes \theta_2(k-1) \end{cases} \quad (7)$$

The standard first-order form (4) is ultimately written with:

$$A = \begin{pmatrix} \varepsilon & \varepsilon & e \\ \varepsilon & \varepsilon & \tau_1 \\ \varepsilon & \tau_2 & \varepsilon \end{pmatrix}, \quad B = \begin{pmatrix} \tau_u \\ \tau_1 + \tau_u \\ \varepsilon \end{pmatrix} \text{ and } x(k) = \begin{pmatrix} \theta_1(k) \\ \theta_2(k) \\ \theta_3(k) \end{pmatrix}$$

The behavior of this system can thus be completely determined if we were to consider the initial conditions (i.e. initial marking and tokens availability). A scenario of the system (all tokens are available initially) at maximum speed has been depicted in Fig. 3.

If the system (Fig. 2b) is not constrained (the upstream stock is never empty and parts remain available at all times), then the system can be expressed by:

$$x(k) = A \otimes x(k-1) \quad (8)$$

Such is the case in our study since data stemming from the plant are available at the sensor output as long as devices do not fail. This observation explains the absence of source transitions within the NAS model of the next section (Fig. 6).

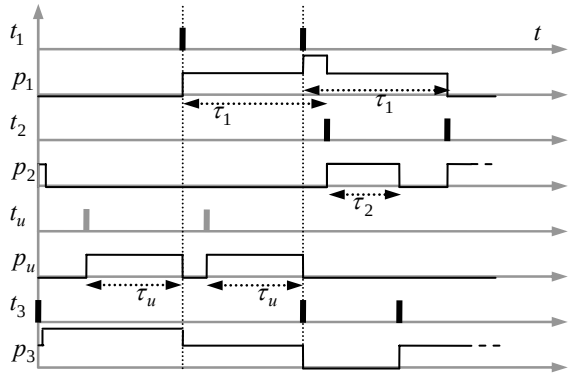

Fig. 3. Chronogram of TEG (Fig. 2b) behavior at maximal speed.

III. Modeling and Response Time Evaluation: Case of a Single RIOM

In our study, we consider NAS based on client/server paradigm. The PLC is the client and the RIOM (remote input output module) is the server. While the PLC sends requests periodically, the server does not send any message autonomously but does only answer the received requests: only synchronous transmissions from the RIOM are then considered.

Fig. 4. Mono-RIOM networked automation system (Case 1).

The considered PLC contains a CPU (central processing unit) module to execute the user program and a network board (NETb) to send requests (combined requests: read and write data) to the RIOM. At each scanning cycle, the NETb sends a message to the RIOM either requesting information on the plant (e.g. is the maximum level of water reached?) or providing the control signal (e.g. close the valve). Neither the CPU nor the NETb are in fact synchronized even though they do belong to the same component, i.e. the PLC; they are both time-driven and operate independently. The CPU periodically accomplishes the tasks of: reading inputs, executing the user program to produce a control signal, and updating outputs. Regardless of the CPU, the NETb sends requests to the RIOM and awaits the replies. Once a reply has been received, the NETb waits further until the period time has elapsed in order to begin a new cycle.

Besides supposing the CPU and the NETb to work periodically without clocks drift, we consider neither frame loss nor timeouts. Such assumptions are often taken for granted in the context of NAS [11], as is the case in our study,

for two main reasons:

- The exchanged data packets are of a short length, and just a few bytes are sufficient to transmit data. With Modbus for instance, the request can reach up to 256 bytes only [2].

- Such automation systems are necessarily divided into many local automation cells so as to achieve traffic isolation, especially from non-real time traffic (long packets, videos, etc.). Only a few packets are in fact exchanged between the various cells and serve to limit congestion or packet drops within real-time domains. Such is true for the case of vertical integration of high-level functions (e.g. supervision) [11].

Regardless of the protocol used in NAS, one major criterion of real-time performance evaluation is *response time*, which reflects the delay between the occurrence of an event in the plant (e.g. the maximum water level has been reached) and the arrival of the consequence event generated by the controller (e.g. close the valve) at the plant (Fig. 4). Two cases need to be distinguished depending on whether this action/reaction loop concerns a simple control event or an event involved in a system safety. In the former case, the designer is merely required to evaluate the time performance of the control architecture (e.g. through knowledge of the response times mean and standard deviation), whereas in the latter case, the maximum response time bound becomes the top priority. In our work, we consider the general case regardless of the consequences of the events occurring in the plant. Both the bounds and distribution of response times will be calculated.

A) Architecture Modeling

According to the client/server protocol described above, we derived the architectural model shown in Fig. 6. This model is highly abstract and only represents the applicative aspect of the architecture protocol, i.e. the top layer in the OSI representation of network systems. So, rather than representing the network in a lower level including the problem of resources sharing that would prevent us from using TEGs, we only consider the delays, even variable, due to that network.

On the model depicted on Fig. 6, we consider just the timed places and simply assign a delay τ_i to place p_i (thus, only the delays τ_i are represented on Fig. 6). The system is assumed to be work conserving and all transitions are fired at maximum speed, as explained in Section II.

Places p_1, p_2, p_3 and p_4 with CPU delays of τ_1, τ_2, τ_3 and τ_4 respectively model the waiting phases to begin a new CPU cycle; user program execution during T_{CLC} (including updates to both reading inputs and outputs); busy CPU; and lastly, idle CPU. Since the CPU calculus always finishes before the CPU period has elapsed, periodic CPU operations can easily be indicated by a cycle period denoted T_{CPU} (equal to τ_3). Similarly, τ_{15} represents the NETb scanning period (denoted T_{SCN}), and a token in place p_{15} indicates a busy status during this period. Transmitting a request therefore starts by firing the transition t_4 and ends by firing t_5. τ_6 or T_{EM} is the time required to transmit the request. A token in p_{14} means the request has been sent and the NETb is waiting for the response. Places p_7 and p_{12} model the network delays imposed upon the transmitted request and the returned response (denoted τ_7 and τ_{12} respectively). The only assumption regarding these delays is the fact that they are bounded. Unlike the majority of studies and our preliminary work [14], in which the network has been represented with constant delays, we now assume that these delays are variable in a given domain, with both a minimum and maximum (finite) bound. At the l^{th} scanning cycle, the network delays experienced by the request and its reply in the network are: $\tau_7(l) \in [\tau_7^{\min}, \tau_7^{\max}]$ and $\tau_{12}(l) \in [\tau_{12}^{\min}, \tau_{12}^{\max}]$, respectively. In assessing the maximum NAS response time bound, only the

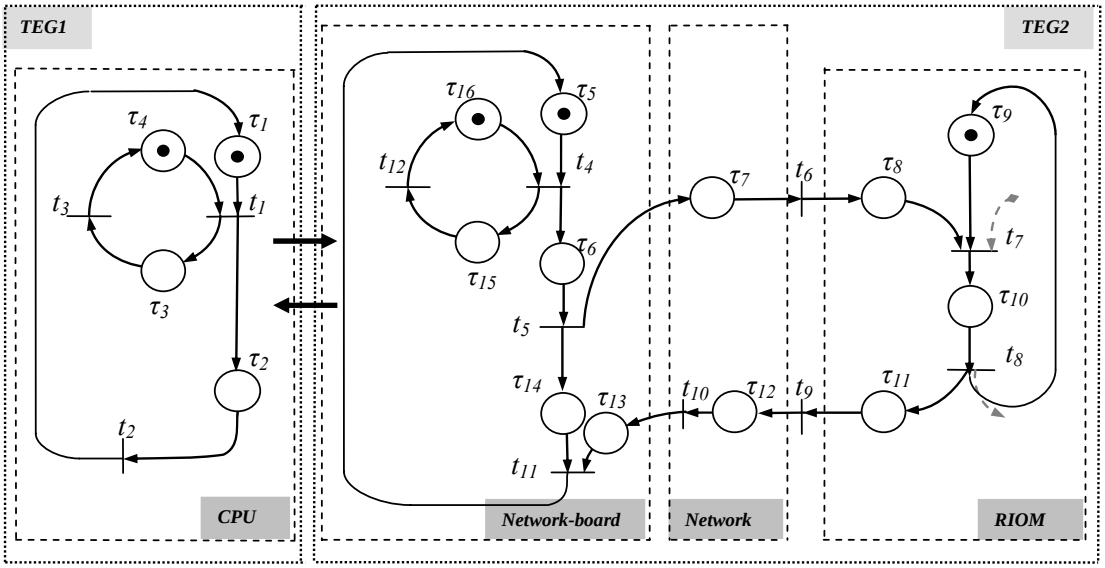

Fig. 6. Mono-RIOM networked automation system modelling using TEGs (Case 1).

upper bounds of these delays are required. This observation will be explained more thoroughly further below when the analytical evaluation is performed.

Once the network has been crossed, the returned response arrives in the NETb input buffer at p_{13} and is copied into the memory shared with CPU in a time τ_{13}, denoted T_{CPY} (much smaller than the other PLC delays since its order of magnitude remains in the microseconds, whereas the others reach the milliseconds). This time period is indeed necessary to copy just those few bytes carrying the application data (without any of the lower layers headers). A place could be added to represent this memory, yet its token would be available at all times and thus exert no impact on the system behavior.

Places p_8, p_9, p_{10}, and p_{11} indicate the phase upon arriving at the RIOM. The RIOM remains in a wait mode at p_9 until a request arrives in its input buffer p_8. By firing t_7, processing begins and continues for a time τ_{10}. At the end of this time, the response is placed in its output buffer p_{11} before being returned into the network. The main aforementioned delays and their description are summed up in Table I:

TABLE I

DELAYS τ_i OR T_x AND THEIR DESCRITPION

τ_i	T_x	Description of delay
τ_2	T_{CLC}	Time to execute the user program by CPU
τ_3	T_{CPU}	CPU period
τ_6	T_{EM}	Time to send a request (emission)
τ_7	/	Delay to cross the network by the sent request
τ_{10}	$T_{I/O}$	Time to process a request by RIOM
τ_{12}	/	Delay to cross the network by the returned answer
τ_{13}	T_{CPY}	Time to copy an answer into shared memory of NETb/CPU
τ_{15}	T_{SCN}	Network board period

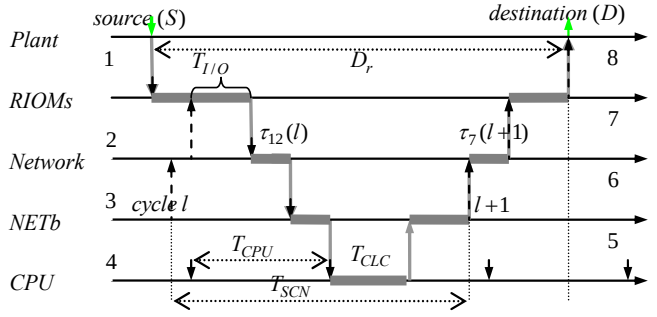

Fig. 5. Experienced delays at the different stages of NAS.

To sum up (Fig. 5), an event occurs at any time (1) in the plant (source S) and waits until a request arrives to RIOM to take it into account. Once the request is received, an answer is processed while considering this event and then returned to PLC (2). This answer crosses the network and gets to NETb (3). At the beginning of a new CPU cycle, this answer is used to execute the user program (4) to perform the reaction event (5) that will be sent to its destination at the next scanning cycle (6). Again, this answer crosses the network to get to RIOM (7) and finally to the plant (8) (destination D).

Remark III. 1: The gray hatched arrows on Fig.6 (in the RIOM model) that intended to represent both the source (data stemming from the sensor) and output (data transmitted towards the actuator) are not being considered since the system has not been constrained. The RIOM is always responding to a request using the latest information provided by the sensor. This situation remains valid as long as the devices are functional, which serves to justify the absence of source transitions first on the model and consequently in the subsequent Max-Plus equations (explanation of Section II with unconstrained systems).

Thus, by applying the method described in Section II to the model shown in Fig. 6 with all tokens available at the beginning, we obtain the Max-Plus equations:

$$\begin{cases} \theta_1(k) = (\theta_2(k-1) \otimes \tau_1) \oplus (\theta_3(k-1) \otimes \tau_4) \\ \theta_2(k) = \theta_1(k) \otimes \tau_2 \\ \theta_3(k) = \theta_1(k) \otimes \tau_3 \end{cases} \quad (9)$$

$$\begin{cases} \theta_4(l) = (\theta_{11}(l-1) \otimes \tau_5) \oplus (\theta_{12}(l-1) \otimes \tau_{16}) \\ \theta_5(l) = \theta_4(l) \otimes \tau_6 \\ \theta_6(l) = \theta_5(l) \otimes \tau_7 \\ \theta_7(l) = (\theta_6(l) \otimes \tau_8) \oplus (\theta_8(l-1) \otimes \tau_9) \\ \theta_8(l) = \theta_7(l) \otimes \tau_{10} \\ \theta_9(l) = \theta_8(l) \otimes \tau_{11} \\ \theta_{10}(l) = \theta_9(l) \otimes \tau_{11} \\ \theta_{11}(l) = (\theta_5(l) \otimes \tau_{14}) \oplus (\theta_{10}(l) \otimes \tau_{13}) \\ \theta_{12}(l) = \theta_4(l) \otimes \tau_{15} \end{cases} \quad (10)$$

Equations systems (9) and (10) are linear in Max-Plus algebra and may be written in the form (4). We have assigned them different indices (*k* and *l*) due to their non-synchronization, just like the CPU and the NETb. This step constitutes an additional difficulty in our study.

B) Principle behind the proposed approach

As a first step, we must solve the systems of equations (9) and (10) in order to determine the transition firing dates as functions of indices *k* and *l*. This resolution step is somewhat complex since the systems are time-variant (the involved delays are variable). Yet, this complexity can be overcome under the aforementioned hypotheses: *i*) the periodic operations of both the CPU and NETb (without clock drift); and *ii*) zero frame loss or component failure. In this case, we are only searching for the two cycles beginnings at $\theta_1(k)$ and $\theta_4(l)$. The other transitions will be deduced accordingly using equations (9) and (10). The following solutions are therefore obtained:

$$\begin{cases} \theta_1(k) = (k-1) \cdot T_{CPU} \\ \theta_2(k) = ((k-1) \cdot T_{CPU}) \otimes T_{CLC} \\ \theta_3(k) = k \cdot T_{CPU} \end{cases} \quad (11)$$

$$\begin{cases} \theta_4(l) = (l-1)\cdot T_{SCN} \\ \theta_5(l) = \theta_4(l) \otimes \tau_6 \\ \theta_6(l) = \theta_5(l) \otimes \tau_7(l) \\ \theta_7(l) = (\theta_6(l) \otimes \tau_8) \oplus (\theta_8(l-1) \otimes \tau_9) \\ \theta_8(l) = \theta_7(l) \otimes \tau_{10} \\ \theta_9(l) = \theta_8(l) \otimes \tau_{11} \\ \theta_{10}(l) = \theta_9(l) \otimes \tau_{12}(l) \\ \theta_{11}(l) = (\theta_5(l) \otimes \tau_{14}) \oplus (\theta_{10}(l) \otimes \tau_{13}) \\ \theta_{12}(l) = \theta_4(l) \otimes \tau_{15} \end{cases} \quad (12)$$

The next step of this method consists of fusing solutions (11) and (12) so as to clarify the link between the CPU and the NETb. Among these solutions, only the equations representing the following events are then of interest (at this stage):

- Beginning of processing in the CPU or reading inputs (θ_1)
- End of processing in the CPU and output update (θ_2)
- Beginning of scanning cycle and transmitting a request (θ_4)
- Reception of a response in the shared memory (θ_{11}).

These are indeed the events representative of communication between the CPU and the NETb. When a response arrives (θ_{11}), it is taken into account at the next CPU cycle beginning (θ_1) and then read and used in CPU calculus. Once processing has been completed, the result is put in the NETb memory (θ_2) before being transmitted to the RIOM at the next scanning cycle beginning (θ_4).

Let's set T_r as the wait time between transmitting a request and receiving the corresponding response (i.e. the round-trip time). The following equations are then derived:

$$\begin{cases} \theta_1(k) = (k-1)\cdot T_{CPU} \\ \theta_2(k) = (k-1)\cdot T_{CPU} \oplus T_{CLC} \end{cases} \quad (13)$$

$$\begin{cases} \theta_4(l) = (l-1)\cdot T_{SCN} \\ \theta_{11}(l) = (l-1)\cdot T_{SCN} \otimes T_r \end{cases} \quad (14)$$

At the l^{th} scanning cycle, the request reply is received at date $\theta_{11}(l)$ and taken into account by the CPU; it must then wait for the m_l^{th} CPU cycle beginning. This m_l^{th} cycle however must be immediately subsequent with respect to $\theta_{11}(l)$. The condition to verify thus becomes:

$m_l = \underset{i\in\mathbb{N}/\theta_1(i)>\theta_{11}(l)}{Arg\min}\ (\theta_1(i)-\theta_{11}(l))$, where "$Arg\min$" is the converse function yielding the index that minimizes the positive term $(\theta_1(i)-\theta_{11}(l))$. Hence, we are introducing a new transition $\hat{\theta}_1$ representing the output update using the l^{th} response, i.e. $\hat{\theta}_1(l) = \theta_2(m_l) = \theta_1(m_l) + T_{CLC}$.

During the next scanning cycle (with respect to $\hat{\theta}_1(l)$), which is the n_l^{th} cycle, the updated result is encapsulated into a request packet and sent to the RIOM. Similarly, another new transition $\hat{\theta}_2$ is added with: $\hat{\theta}_2(l) = \theta_4(n_l)$ and $n_l = \underset{i\in\mathbb{N}/\theta_4(i)>\hat{\theta}_1(l)}{Arg\min}\ (\theta_4(i)-\hat{\theta}_1(l))$. The date of event consequence arrival at the controlled process, is therefore $\theta_8(n_l)$. For the investigated event during the l^{th} scanning cycle generated at a time denoted by $\theta_e(l)$, the associated NAS response time is therefore given by:

$$D_r(l) = \theta_8(n_l) - \theta_e(l) \quad (15)$$

The response time in (15) is minimal provided that the data originating from the sensor are used for processing in the RIOM immediately after being generated, i.e. at the date $\theta_e(l) = \theta_7(l) - (d_{filt} + 0^+)$. The minimum delay relative to the l^{th} scanning cycle therefore equals:

$$D_{MIN}(l) = \theta_8(n_l) - \theta_7(l) + d_{filt} \quad (16)$$

where d_{filt} is the delay due to data filtering in the sensor.

On the other hand, the response time is maximal if the data arrive immediately after the beginning of processing in the RIOM, with respect to the previous scanning cycle, i.e.:

$$D_{MAX}(l) = \theta_8(n_l) - \theta_7(l-1) + d_{filt} \quad (17)$$

As can be seen, the formulas (16) and (17) provide the response time bounds relative to the l^{th} scanning cycle while (15) gives the response time relative to an event generated at time $\theta_e(l)$. Actually, these formulas can be used in calculus provided that the index n_l is calculated. The previous steps to find this index can be achieved using the algorithm depicted on Fig. 7:

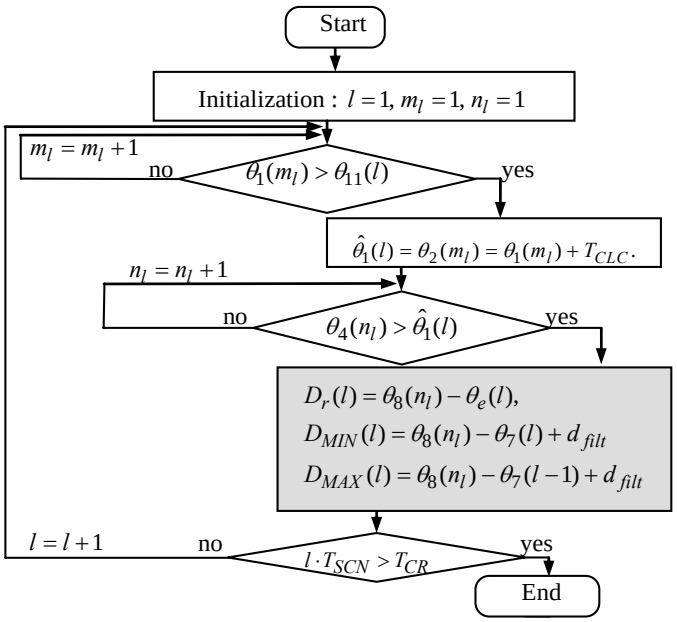

Fig. 7. Response times calculus algorithm (ALGO).

In conclusion, the method development has been achieved by means of the following steps: modeling the architecture through the use of TEGs; writing the corresponding Max-Plus equations; and resolving these equations. After finding the

transitions firing dates (through the previous Max-Plus equations resolution or simply by simulating the system behaviour using the recursive equations (9) and (10)), an algorithm (let's call it ALGO as in Fig. 7) was provided to evaluate the architecture response time relative to any occurring event at any cycle; this algorithm is fast and easily implemented. Besides measurements, this algorithm is used to verify the validity of the formulae (explicit formulas that can be used directly in calculus) derived later in this study.

Lemma

This architecture contains two periodic, yet non-synchronized, processes. In spite of this fact, the entire system remains periodic with a period T_{CR} that verifies: $T_{CR} = k_1 \cdot T_{SCN} = k_2 \cdot T_{CPU}$, where $k_1, k_2 \in \mathbb{N}$ always exist.

Proof: The main PLC parameters of interest are the two module periods T_{CPU} and T_{SCN}, chosen by the system user from among only the multiples of a basis unit, which is generally on the order of a few milliseconds. These parameters can thus be considered as integers under the assumption of zero clock drift. Now let's set: $T_{SCN} = r \cdot T_{CPU}$, with $r = \rho + \omega$, ρ being the integer part of T_{SCN}/T_{CPU} and $\omega \in \mathbb{Q}^+$ its fractional part. Since $\omega = n_1/n_2$ can be written, it is sufficient to take: ($k_1 = n_2$) and ($k_2 = n_2 \cdot \rho + n_1$) in order to reach: $k_1 \cdot (\rho + n_1/n_2) \cdot T_{CPU} = k_2 \cdot T_{CPU}$; hence:

$$T_{CR} = (n_2 \cdot \rho + n_1) \cdot T_{CPU} \tag{18}$$

This period is minimized if n_1 and n_2 are relatively prime numbers.

Hence, we can conclude that the method (or ALGO) is formal and all possible states are scanned if the simulation length of the system covers the critical period T_{CR}. The resultant response time bounds are thus formal as well.

Remark III. 2: the previous method can be used to obtain the shape of the response times distribution as follows: at each scanning cycle l, the different delays $\tau_i(l)$ of TEG are taken randomly from their domain of variation. For example, the network delay $\tau_7(l)$ is randomly chosen from $[\tau_7^{\min}, \tau_7^{\max}]$. Then using the max-plus equation and ALGO, the index n_l is calculated. Finally, by generating randomly an event at date $\theta_e(l)$ such that it is taken into account at cycle l or $\theta_7(l-1) < \theta_e(l) < \theta_7(l)$, the response time relative to this event is calculated using (15). By repeating this operation a large amount of times, histograms giving the response times distribution shape is obtained (see example of Section VI).

C) Analytical calculus of response time

The previous algorithm enables the calculation of both the response time bounds and the distribution shape. It is preferable however to develop analytical formulae that yield these results trivially. Moreover, such formulae will facilitate analyzing the influence of individual architecture parameters on overall performance and serve to answer the "what if" questions. This attribute is very valuable when seeking, for example, an adequate automation system configuration so as to guarantee the stability or safety. Such an analytical evaluation will be the focus of this section.

The results from (13) and (14) along with the algorithm principle will be used for this exercise.

Let's take: $T_r = \alpha \cdot T_{CPU} + \tau_r$, $T_{CLC} = \beta \cdot T_{CPU}$,

where $\beta < 1$, α is the integer part of T_r/T_{CPU} and τ_r the fractional part.

For the calculus complexity to be proven later, let's begin with the case $r \in \mathbb{N}$ and generalize it for $r \in \mathbb{Q}^+$ (for recall, $T_{SCN} = r \cdot T_{CPU}$ with $r = \rho + \omega$, $\rho \in \mathbb{N}$, $\omega < 1$ and $\omega \in \mathbb{Q}^+$).

a) $r \in \mathbb{N}$ ($\omega = 0$)

At the l^{th} scanning cycle, the request response is received at the following date:

$$\theta_{11}(l) = (l-1) \cdot r \cdot T_{CPU} + \alpha \cdot T_{CPU} + \tau_r \tag{19}$$

In order to be taken into account, the response must wait for the next CPU cycle beginning, which entails waiting for the minimum number k (previously denoted m_l) that verifies $\theta_1(k) > \theta_{11}(l)$.

By taking $k - 1 = (l-1) \cdot r + \alpha + 1$, we then obtain:

$$\theta_1(k) = \theta_{11}(l) + T_{CPU} - \tau_r \tag{20}$$

Since $0 < T_{CPU} - \tau_r < T_{CPU}$, $\theta_1(k) > \theta_{11}(l)$ and therefore $m_l = (l-1) \cdot r + \alpha + 2$, which leads to the following result:

$$\hat{\theta}_1(l) = \theta_1(m_l) + T_{CLC} = \left[1 + \alpha + \beta + (l-1) \cdot r\right] \cdot T_{CPU} \tag{21}$$

Since the index m_l has been determined, we must now seek the minimum number n (previously denoted n_l) such that $\theta_4(n) > \hat{\theta}_1(l)$.

The solutions in (14) provides $\theta_4(n) = (n-1) \cdot r \cdot T_{CPU}$; moreover for $n = l + 1$, then $\theta_4(n) = l \cdot r \cdot T_{CPU}$, which can be rewritten as:

$$\theta_4(n) = \hat{\theta}_1(k) + \left[r - (1 + \alpha + \beta)\right] \cdot T_{CPU} \tag{22}$$

In relying on condition C_1: $r > (1 + \alpha + \beta)$, n verifies $\theta_4(n) > \hat{\theta}_1(l)$ and thus $n_l = l + 1$, which justifies writing: $\hat{\theta}_2(l) = \theta_4(n_l) = \theta_4(l+1)$. This development implies:

$$\begin{cases} D_{MIN}(l) = \theta_8(l+1) - \theta_7(l) + d_{filt} \\ D_{MAX}(l) = \theta_8(l+1) - \theta_7(l-1) + d_{filt} \\ D_r(l) = \theta_8(l+1) - \theta_e(l) \end{cases} \tag{23}$$

As a final expression:

$$\begin{cases} D_{MIN}(l) = T_{SCN} + \Delta(l,1) + T_{I/O} \\ D_{MAX}(l) = 2T_{SCN} + \Delta(l-1,2) + T_{I/O} \\ D_r(l) = \theta_8(l+1) - \theta_e(l) \end{cases} \tag{24}$$

where $\Delta(l,q) = \tau_7(l+q) - \tau_7(l)$ is the network jitter and

$T_{I/O} = \tau_8 + \tau_{10} + \tau_{11} + d_{filt}$ the total time spent in the RIOM (including buffering at the input port, data filtering, processing and buffering at the output port).

In practice, condition C_1 is often verified and results (24) are valid given that the scanning period is much longer than the CPU period ($T_{SCN} \gg T_{CPU}$). If such were not the case, then it would be necessary to identify the integer $q \geq 1$ that verifies condition C_q: $r \cdot q > (1+\alpha+\beta) > r \cdot (q-1)$. We thus obtain $n_l = l + q$ and ultimately the general formulae:

$$\begin{cases} D_{MIN}(l) = q \cdot T_{SCN} + \Delta(l,q) + T_{I/O} \\ D_{MAX}(l) = (q+1) \cdot T_{SCN} + \Delta(l-1,q+1) + T_{I/O} \\ D_r(l) = \theta_8(l+q) - \theta_e(l) \end{cases} \quad (25)$$

It can be noted that if condition C_1 is verified, i.e. $r > (1+\alpha+\beta)$, then $q = 1$, which in turn yields exactly the same results as in (24).

Discussion: From the above analysis and formulae, some of the points not trivially expected deserve to be mentioned:

- The network delays are assumed not to be constant (i.e. variable α) as is the user program execution time (variable β). Moreover, according to C_q, the response time reaches its worst case (maximum value) when α and β are maximized. We actually needed to search for the minimum number q such that condition C_q was verified, or $r \cdot q > (1+\alpha+\beta)$. The number q therefore assumes its worst value when parameters α and β are maximized. In order to calculate the upper response time bound using these formulae, only the local worst case CPU calculus and network delays are required.

- The same remarks apply to the effect of delay $T_{I/O}$.

- To obtain fast NAS response times, the ratio $r \in \mathbb{N}$ should be increased by either reducing T_{CPU} or raising T_{SCN}. The optimal configuration is attained by minimizing the period T_{CPU} while maintaining $\beta < 1$ (i.e. the user program must be completely executed before the CPU cycle has elapsed). The reasoning regarding the period T_{SCN} is not so obvious given its direct correlation with the formulae in (25). Nevertheless, certain details about this point are provided in the section discussion below (Case b).

b) $r \in \mathbb{Q}^+$ $(\omega \neq 0)$

Let's set $\tau_r = \gamma \cdot T_{CPU}$ (where obviously $\gamma < 1$).

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

$$\theta_{11}(l) = (l-1) \cdot r \cdot T_{CPU} + (\alpha+\gamma) \cdot T_{CPU} \quad (26)$$

Let's now set $i \in \mathbb{N}$, where: $i \leq \gamma + \omega \cdot (l-1) < (i+1)$ (*)

For $k - 1 = (l-1) \cdot \rho + \alpha + 1 + i$, it follows that:

$$\theta_1(k) = \theta_{11}(l) + \left[i + 1 - (\gamma + \omega \cdot (l-1)) \right] \cdot T_{CPU} \quad (27)$$

Since in (*), $i \leq \gamma + \omega \cdot (l-1) < (i+1)$, then $m_l = (l-1) \cdot \rho + \alpha + 2 + i$ and therefore:

$$\hat{\theta}_1(l) = \theta_{11}(l) + \left[i + 1 + \beta - (\gamma + \omega \cdot (l-1)) \right] \cdot T_{CPU} \quad (28)$$

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

$$\theta_4(n) = \hat{\theta}_1(k) + \left[r - \left[\alpha + \beta + i + 1 - \omega \cdot (l-1) \right] \right] \cdot T_{CPU} \quad (29)$$

From (*), it can be deduced that: $\gamma < i + 1 - \omega \cdot (l-1) \leq 1 + \gamma$.

Let's set: $\Gamma_{l,i} = i + 1 - \omega \cdot (l-1)$, with $\gamma < \Gamma_{l,i} \leq 1 + \gamma$,

$\Gamma_{MIN} = \min_{i \in \mathbb{N}, l \in \mathbb{N}} (\Gamma_{l,i})$ and $\Gamma_{MAX} = \max_{i \in \mathbb{N}, l \in \mathbb{N}} (\Gamma_{l,i})$.

Equation (29) can then be rewritten as:

$$\theta_4(n) = \hat{\theta}_1(k) + \left[r - (\alpha + \beta + \Gamma_{l,i}) \right] \cdot T_{CPU} \quad (30)$$

On condition $\bar{C}: r > (\alpha + \beta + \Gamma_{MAX})$, we have therefore obtained: $\hat{\theta}_2(l) = \theta_4(n_l) = \theta_4(l+1)$, i.e. $n_l = l+1$. As previously noted, this finding has led to the same results as in (24). But should this condition not be respected, the time bounds would depend on $\Gamma_{l,i}$ as well as α and β. Instead of reasoning based on delays relative to a particular scanning cycle, as was previously the case, the discussion here addresses both the global (absolute) bounds and the local delay relative to the l^{th} scanning cycle.

The following global and local conditions are used:

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

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

The response times are then written as:

$$\begin{cases} D_{MIN} = q_1 \cdot T_{SCN} + \Delta(l, q_1) + T_{I/O} \\ D_{MAX} = (q_2 + 1) \cdot T_{SCN} + \Delta(l-1, q_2 + 1) + T_{I/O} \\ D_r(l) = \theta_8(l + q_3) - \theta_e(l) \end{cases} \quad (31)$$

Discussion: In this general case, it is noted that the optimality condition ($q_2 = 1$) or $\bar{C}: r > (\alpha + \beta + \Gamma_{MAX})$ may be more restrictive than in the $r \in \mathbb{N}$ case. For $\omega = n_1/n_2$, it is indeed sufficient to use $(l-1) = n_2$ and $i = n_1$ to obtain $\Gamma_{l,i} = 1$, which implies that $\Gamma_{MAX} \geq 1$ and condition $\bar{C}$ is more restrictive than C_1. This result is an important about the response time calculus. It suggests, in the event the network effect is smaller than T_{CPU} (often the case in such NAS and which then implies $\alpha = 0$), setting the scanning period at twice the CPU period (while naturally first minimizing T_{CPU}, as previously explained). If $T_{SCN} = 2 \cdot T_{CPU}$, we then return to case (a), where $r \in \mathbb{N}$ $(\omega = 0)$ and hence $\Gamma_{MAX} = 1$. Since $\beta < 1$ and $\alpha = 0$, $(\alpha + \beta + \Gamma_{MAX}) < 2$. Setting $T_{SCN} = 2 \cdot T_{CPU}$ will thus serve to satisfy condition C_1 and will surely lead to $q_2 = 1$, which means that the relatively simple results (24) can be used directly for calculus.

Example III.1:

Suppose that the CPU period equals 5 ms, the user program lasts at most 3.5 ms ($\beta = 0.7$) and the network induces a

maximum delay $T_r = 1.24$ ms ($\alpha = 0$ and $\gamma = 0.248$). If the scanning period T_{SCN} were set at 8 ms, then $r = 1.6$ ($\omega = 0.6$) and according to (*), $\Gamma_{MAX} = 1.2$, i.e. $\alpha + \beta + \Gamma_{MAX} = 0 + 0.7 + 1.2 = 1.9$. The minimum integer q satisfying $r \cdot q > 1.9$ is therefore $q = 2$. The corresponding maximum response time will thus equal: $D_{MAX} = 3 \times 8 + \Delta + T_{I/O}$. Should however T_{SCN} equal 10 ms, then $r = 2$ and $\alpha + \beta + \Gamma_{MAX} = 0 + 0.7 + 1 = 1.7$. The minimal number q satisfying $r \cdot q > 1.7$ is this time $q = 1$. The maximum bound is then $D_{MAX} = 2 \times 10 + \Delta + T_{I/O}$, which is less than the previous one. This phenomenon is paradoxical since a faster element may lead to a more significant delay. This consideration can be explained intuitively as follows: when a response is returned during the l^{th} scanning cycle, with a sufficiently long scanning period the consequence derived from the CPU will be transmitted to the actuator during the $(l+1)^{th}$ scanning cycle. However on the other hand if scanning were faster, then the consequence would not be sent during that particular cycle but instead wait until the next one, i.e. $(l+2)^{th}$ or even later, thus explaining the incremental delay with a scanning period T_{SCN}. This phenomenon has also been pointed out in [20].

IV. Modeling and Response Time Evaluation: Case of Multiple RIOMs

The system modeling is different in this case since N RIOMs are to be scanned. Information is obtained from many sensors and multiple control signal destinations are observed as well. The PLC is thus transmitting a burst of requests to the RIOMs and waiting for responses (Fig. 8).

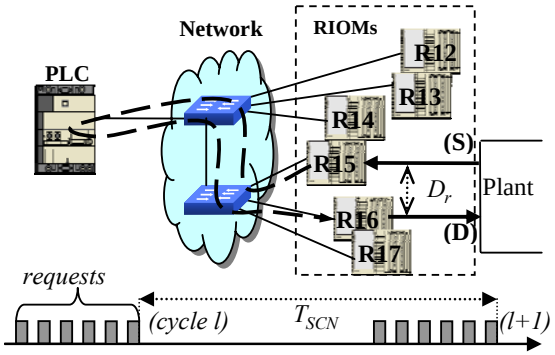

Fig. 8. Multi-RIOMs NAS (Case2).

When a response has been received, it is copied into the memory shared with the CPU. Once all responses have been received, NETb remains in a waiting mode until the scanning period has fully elapsed and then begins another cycle. All received responses are copied as a block (all at once) from the shared memory and then used to update control signals within the same CPU cycle. Only one token is necessary therefore to model CPU operations. CPU model of Case 1 thus remains valid for our second case but the remainder of this system however is much more complex than before.

In this general NAS configuration, requests are sent from NETb in an invariant order (throughout NAS operations), as revealed on the new model in Fig. 9. The RIOMs are assigned indices according to their scanning order. We associate index i to the RIOM receiving the i^{th} request from the NETb. As an example, Fig. 7 shows RIOMs being scanned in the order $\{R_{12}, R_{13}, R_{14}, R_{15}, R_{16}, R_{17}\}$; therefore, for instance, the index assigned to R_{13} is $i = 2$. More specifically, N_S and N_D are the indices assigned respectively to the event source (S) and

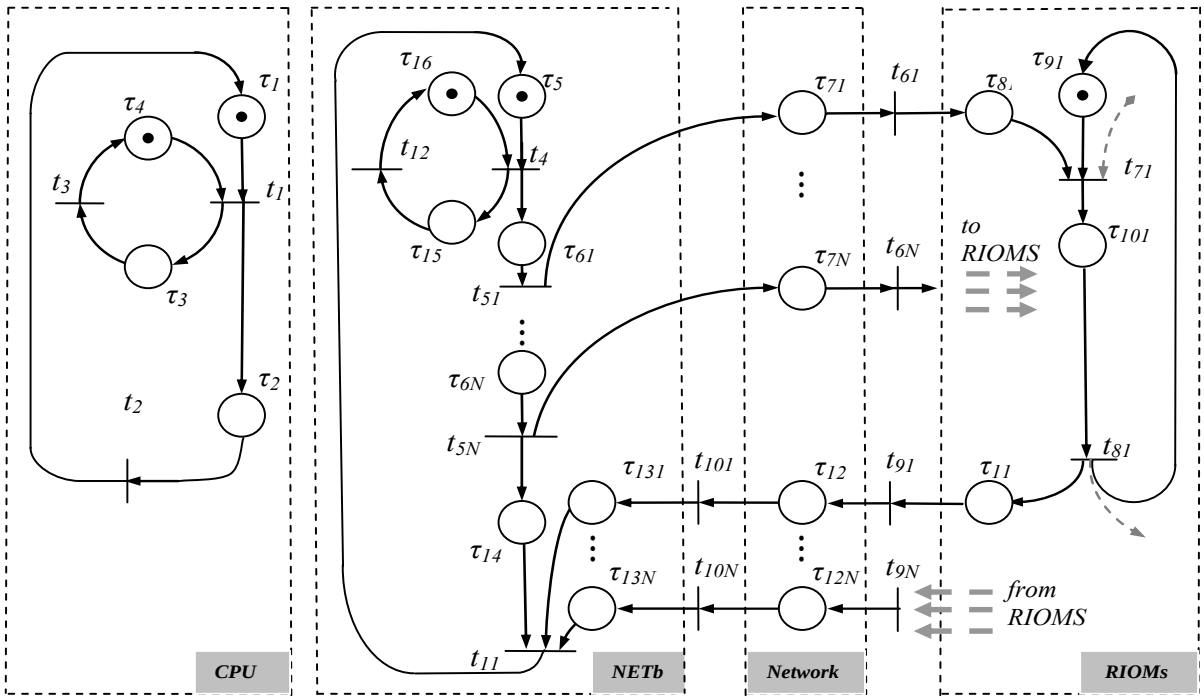

Fig. 9. TEGs based model of the multi-RIOMs NAS (Case2).

consequence destination (D) involved in the targeted NAS loop. In Fig. 7, $N_S = 4$ and $N_D = 5$ (since the source (S) is R_{15} and the destination (D) is R_{16}).

We have introduced the delay experienced by each frame i during the various NAS stages: a delay T_{EM}^{i} to be sent to the network, a delay $T_{I/O}^{i}$ occurring in the RIOM and naturally the delays caused by the network represented by places p_{7i} and p_{12i}. τ_{7i} represents the network delay that affects the i^{th} transmitted request and τ_{12i} the returned response. Furthermore, places p_{13i} represent the FIFO queue delay affecting responses in the NETb input buffer. Once again, these network delays are not presumed to be constant, but rather variable within a bounded domain; they may be due to any kind of network, and their only condition is to be bounded so as to ensure suitability for real-time systems.

All that is necessary to write the equations of the model has been presented, and we now derive the Max-Plus system of equations (N is the number of scanned RIOMs):

$$\begin{cases} \theta_1(k) = (\theta_2(k-1) \otimes \tau_1) \oplus (\theta_3(k-1) \otimes \tau_4) \\ \theta_2(k) = \theta_1(k) \otimes \tau_2 \\ \theta_3(k) = \theta_1(k) \otimes \tau_3 \end{cases} \tag{32}$$

$$\begin{cases} \theta_4(l) = (\theta_{11}(l-1) \otimes \tau_5) \oplus (\theta_{12}(l-1) \otimes \tau_{16}) \\ \textit{for } i = 1 \textit{ to } N \\ \quad \theta_{5i}(l) = \theta_{5(i-1)}(l) \otimes \tau_{6i} \quad // \theta_{50} = \theta_4(l) \\ \quad \theta_{6i}(l) = \theta_{5i}(l) \otimes \tau_{7i}(l) \\ \quad \theta_{7i}(l) = (\theta_{6i}(l) \otimes \tau_{8i}) \oplus (\theta_{8i}(l-1) \otimes \tau_{9i}) \\ \quad \theta_{8i}(l) = \theta_{7i}(l) \otimes \tau_{10i}(l) \\ \quad \theta_{9i}(l) = \theta_{8i}(l) \otimes \tau_{11i} \\ \quad \theta_{10i}(l) = \theta_{9i}(l) \otimes \tau_{12i}(l) \quad \textit{end} \\ \theta_{11}(l) = (\theta_{5N}(l) \otimes \tau_{14}) \oplus \left[\underset{1 \le j \le N}{\oplus} (\theta_{10j}(l) \otimes \tau_{13j}(l)) \right] \\ \theta_{12}(l) = \theta_4(l) \otimes \tau_{15} \end{cases} \tag{33}$$

According to this new model (Fig. 9), the transition t_{11} models only the fact that a new cycle cannot begin while all responses have not been received. The key event to consider is the completion of copying the response emanating from (S) to the memory shared with the CPU. For this purpose, we have introduced a virtual transition t_S fired at date θ_S to represent this important event:

$$\theta_S(l) = (\theta_{10N_S}(l) \otimes \tau_{13N_S}(l)) \oplus (\theta_{5N}(l) \otimes \tau_{14}) \tag{34}$$

In a similar manner and using the same notations as in Case 1, the solutions to be used for the analysis are as follows:

$$\begin{cases} \theta_1(k) = (k-1) \cdot T_{CPU} \\ \theta_2(k) = (k-1) \cdot T_{CPU} \otimes T_{CLC} \end{cases} \tag{35}$$

$$\begin{cases} \theta_4(l) = (l-1) \cdot T_{SCN} \\ \theta_S(l) = (\theta_{10N_S}(l) \otimes \tau_{13N_S}(l)) \oplus (N \cdot T_{EM}) \end{cases} \tag{36}$$

Let's set $T_r = \theta_S(l) - \theta_4(l) = (\alpha + \gamma) \cdot T_{CPU}$.

During the l^{th} scanning cycle, which starts at date $\theta_4(l)$, the response from (S) is received at time $\theta_S(l)$. This reply is used in the calculus of the subsequent CPU cycle, and the control signal is updated and then sent to destination (D) upon the next RIOMs scanning cycle.

By respecting the same analytical principle as in the first case with one RIOM, for an event generated at time $\theta_e(l)$ and taken into account during the l^{th} scanning cycle, we obtain:

$$\begin{cases} D_{MIN}(l) = q \cdot T_{SCN} + \sum_{i=1}^{N_D} T_{EM}^{i} - \sum_{i=1}^{N_S} T_{EM}^{i} + \Delta(l,q) + T_{I/O}^{N_D}(l+q) \\ D_{MAX}(l) = (q+1) \cdot T_{SCN} + \sum_{i=1}^{N_D} T_{EM}^{i} - \sum_{i=1}^{N_S} T_{EM}^{i} + \Delta(l-1,q+1) + T_{I/O}^{N_D}(l+q) \\ D_r(l) = \theta_{8N_D}(l+q) - \theta_e(l) \end{cases} \tag{37}$$

where: $r \cdot q > \Gamma_{l,i} + \alpha + \beta > r \cdot (q-1)$,

$\Delta(l,q) = \tau_{7N_D}(l+q) - \tau_{7N_S}(l)$ is a network jitter, and $T_{I/O}^{N_D}(l+q) = \tau_{8N_D} + \tau_{10N_D}(l+q) + \tau_{11N_D} + d_{filt}$ is the time spent in the RIOM (D) during the $(l+q)^{th}$ scanning cycle.

The results (37) fit the first case with one RIOM, where $N_D = N_S$, since the event source is also the destination.

Discussion: From the results in (37), it can be observed that to minimize response time, a higher index value should be assigned to the event source and a lower value to the destination so as to make the term $(N_D - N_S)$ as highly negative as possible: the RIOMs scanning order is indeed important. This result matches some of the conclusions drawn experimentally in [10], where it is stated that if the PLC were loaded with requests (yet remaining below a threshold), response time bounds would decrease. Loading the PLC and therefore delaying request transmission (to S) or simply increasing the request transmission order to RIOM (S) yields exactly the same phenomenon. Regarding the load threshold, the formulae also match experimental results. We must keep in mind that the delay calculus condition depends on α or $\theta_S(l)$, and N_S should be decreased (see (34)). The optimal case is derived by increasing N_S while maintaining q equal to 1, i.e. remaining under the threshold.

Once again, this phenomenon is paradoxical since delaying the time of request transmission to the event source would lead to faster response times. This scenario can be explained more intuitively by the fact that it would be preferable to delay request transmission by a small amount of time and therefore wait for an event to occur rather than acting too hastily, sending the request too early and potentially missing the event. The event must therefore wait until the next request to be considered, which might necessitate a substantial wait since the scanning period is often relatively long.

V. NETWORK DELAYS EVALUATION IN A SWITCHED ETHERNET CLIENT/SERVER NAS

The analytical method developed above is applicable easily but the delays caused by the network (τ_{7i} and τ_{12i}) are required in our formulae. This section will focus on their evaluation. In the case of a standard full duplex switched Ethernet network, existing methods can be used for this purpose. Unfortunately, in the context of our study, accurate assessments are required. A pessimistic method may cause significant overestimations of the maximum response time as illustrated in the following example.

Example V.1: With $r = 2$, $\beta = 0.6$ and a maximum delay caused by the network leading to $\Gamma_{l,i} + \alpha = 1.3$, the minimum number q verifying the condition C_1 is equal to 1. Using a pessimistic method (with an overestimation of roughly 10%) may lead to $\Gamma_{l,i} + \alpha = 1.43$ and the new integer q that verifies the condition C_1 equals 2. If $T_{SCN} = 10ms$, the maximum response time bound is slightly greater than 20ms, compared to a value of 30ms including the network delay overestimation. This is very pessimistic (approx. 50% overestimation) and unacceptable in the majority of automation systems.

For this reason, new methods had to be investigated in an effort to evaluate network delays with as limited an overestimation as possible, and the present section is devoted to this objective.

In this study, we have considered a client/server automation system (as in Case 2) over a standard full duplex Ethernet switch (Fig. 10). The switch is modeled as in [9] with a central dispatcher forwarding the frames according to FCFS policy (first come first served), at a speed (bits/s) noted C. In considering a "store & forward" switch, a frame is completely received before being forwarded by the dispatcher to the appropriate output port. The data are exchanged with the end stations (PLC or RIOMs) at a bit rate (bits/s) corresponding to physical link capacity, noted C_k (link of port $Port_k$).

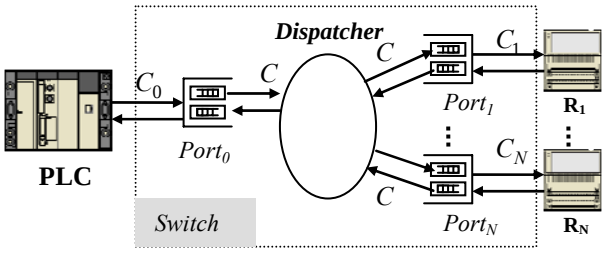

Fig. 10. Client/server automation system over an Ethernet switch.

As aforementioned, the frames entering the switch are forwarded in a FCFS manner without any flow classification or prioritization. At any time, the only criterion therefore for frames differentiation is their order of arrival at the switch. The principle behind the method is to consider a function of *order* for arbitration when many frames at different input ports are waiting to be forwarded. Suppose that the i^{th} frame to enter the switch is completely forwarded to its output port at date $\psi(i)$. So, the $(i+1)^{th}$ frame, supposed of length $L_{(i+1)}$ (overheads included), to be entirely received into the switch at date $\theta_{arrival}(i+1)$, is completely forwarded to its output port at date:

$$\psi(i+1) = max(\theta_{arrival}(i+1), \psi(i)) + L_{(i+1)} / C \quad (38)$$

If the output port is $Port_k$, then the frame exits completely the switch at date:

$$\theta_{exit}(i+1) = \psi(i+1) + L_{(i+1)} / C_k \quad (39)$$

Finally, the network delay that $(i+1)^{th}$ frame suffers from is:

$$\tau_{network}(i+1) = \theta_{exit}(i+1) - \theta_{arrival}(i+1) \quad (40)$$

Note that equation (39) would not be true if there were more than one PLC since this would cause extra delays due to queues of requests at the output ports.

As we can notice, (38) and (39) are recurrent Max-Plus equations that represent the evolution of the system. Moreover, we can see that the dates $\theta_{arrival}(i+1)$ and $\theta_{exit}(i+1)$ correspond to notations used in the modeling sections i.e. θ_{5j} and θ_{6j} (resp. θ_{9j} and θ_{10j}) if the $(i+1)^{th}$ frame is the j^{th} request (resp. j^{th} reply). To express the equations (38) and (39) in a more explicit way, we introduce a function denoted *order*, whereby $order(i_{req})$ is the order of arrival of the i^{th} request at the switch whereas $order(i_{resp})$ is the order of arrival at the switch of the corresponding reply. At the beginning, this function is initialized:

$order(1_{req}) = 1, \cdots, order(k_{req}) = k,$ with $1 \le k \le N$. Indeed, the requests transmission order is known from the beginning and invariant, as previously mentioned in Section IV. Since the responses do not yet exist (at time $t = 0$), then: $order(k_{resp}) = +\infty, 1 \le k \le N$.

Since $order(1_{req}) = 1$ (the first frame to enter the switch is the request, of length L_1, sent to R1), using (38)-(39) we have: $\psi(1) = \theta_{51} + L_1 / C$ and $\theta_{61} = \psi(1) + L_1 / C_1$. The delay sought is thus given as: $\tau_{71} = \theta_{61} - \theta_{51} = L_1 \cdot (1/C + 1/C_1)$. This request is received by R_1 at date θ_{61} and the corresponding response is therefore returned at date $\theta_{91} = \theta_{61} + T^1_{I/O}$, which means that another frame is waiting at an input port, requiring the order function to be updated. The order $order(1_{resp})$ is no longer equal to infinity. The other orders ($order(i_{req})$ with $i \neq 1$) may also be modified if, for example, this response enters before the last request is sent (see example below). In this case, $order(1_{resp}) = N$ and $order(N_{req}) = N + 1$. The order function being updated, the frame whose order equals 2 can be found and the equations (38)-(39) used accordingly. These equations are used for the third frame and so forth until all requests have been sent and their responses received (in all, $2 \cdot N$ frames have to be switched). Meanwhile, delays τ_{7i} or

τ_{12i} (depending if a request or response is selected in the arbitration) are calculated.

Example IV.2 (numerical application): a PLC scans three RIOMs (R_1, R_2 and R_3), with requests of lengths L_1, L_2 and L_3, respectively (overheads included). All of the numerical values used have been chosen arbitrarily and only serve an explanatory purpose:
$C_0 = C_1 = C_2 = C_3 = 10Mbps$, $C = 0.16Gbps$, $L_1 = 80Bytes$, $L_2 = 120Bytes$, $L_3 = 200Bytes$, $T^1_{I/O} = 800\mu s$, $T^2_{I/O} = 600\mu s$, $T^3_{I/O} = 480\mu s$.

The evaluation results using the previous FCFS strategy and equations (38)-(39) have been reported in Table II.

TABLE II
NETWORK DELAYS CALCULUS FOR AN EXAMPLE OF NAS WITH N= 3

in μs	θ_{5i}	θ_{6i}	θ_{9i}	θ_{10i}	ψ	**Order of arrival**		τ_{7i}	τ_{12i}
i=1	150	218	1082	1150	154	1	*Req1*		
					1086	3	*Resp1*	68	68
i=2	500	602	1298	1400	506	2	*Req2*		
					1304	5	*Resp2*	102	102
i=3	1084	1256	1926	2180	1096	4	*Req3*		
					2020	6	*Resp3*	172	170

Discussion: As can be seen on Table II (the shaded column), the order function at the last iteration is no longer in its initialized form. We can in fact note that the response to the first request enters completely into the switch ($\theta_{91} = 1082\mu s$) shortly before the arrival of the third request ($\theta_{53} = 1084$). Therefore, $order(1_{resp}) = 3$ and $order(3_{req}) = 4$, while at the beginning $order(1_{resp}) = +\infty$ and $order(3_{req}) = 3$.

VI. APPLICATION AND VALIDATION

To verify the validity of both the models and results generated previously, we will focus on the two automation architectures described previously (see Fig. 4 and Fig. 8). By using the experimental facilities of our laboratory, we have studied these two architectures. Let's now compare the results obtained with ALGO, the formulae and an experimental measurement of the response time.

The second configuration focuses on the delay between an event generated on the input of remote module R_{15} and its consequence on R_{16} output. The histograms in Fig. 11 show a series of 10,000 experimental measurements and response times obtained using ALGO for this configuration.

The CPU period of the PLC has been set at 5 ms and the scanning period at 10ms. In practice, this architecture presents a jitter of approx. 15% with an average value of 10 ms. This feature has been taken into account in the different calculations. Also, to calculate the response times histograms, about 10,000 events were generated. To avoid any underestimation of the response times due to generating events at discrete dates, a random event generator is used. The k^{th} event is generated at date $\theta_e(k) = k \cdot T_{SCN} + \tau$ with τ randomly chosen from $[0, T_{SCN}]$. The simulation was run many times, yielding the results presented in Fig. 11 and Table III.

TABLE III
RESULTS OF THE FORMULAS, ALGO AND MEASURES

		Response times in ms		
		Minimum	Maximum	Mean
Case 1	Measures	10.40	21.90	16.10
	ALGO	10.06	22.24	15.82
	Formulae	10.06	22.24	/
Case 2	Measures	10.65	22.25	16.40
	ALGO	10.31	22.49	16.07
	Formulae	10.31	22.49	/

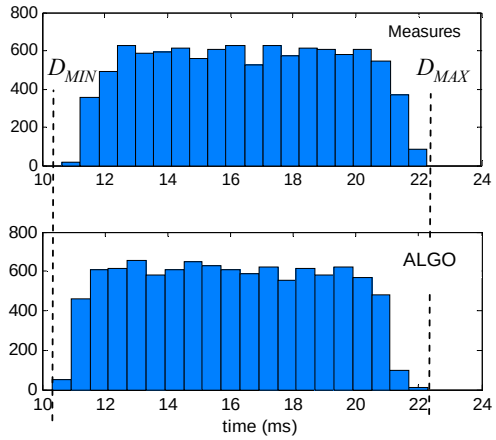

Fig. 11. Histograms of the assessed response times.

Discussion of results: In both cases, a conclusion can be drawn regarding the validity of both the formulae and ALGO since the maximum calculated delays exceed those obtained experimentally (overestimation), and the minimum delays are less than the measured values. As expected, results of ALGO and the formulae are identical in all cases, given that they have been based on the same principle. The discrepancies in delays, with respect to measurements, are less than 3.27% in all cases for either the analytical formulae or ALGO. Both the overestimation and accuracy of the response time assessment have thus been achieved simultaneously. The discrepancies in mean response time calculus are less than 2.01% and the shapes of both the experimental measurement and ALGO histograms are in fact very similar.

On the other hand, if we compare the results of both configurations (Fig. 4 and Fig. 8), a difference of 0.25 ms is detected between the maximum (as well as minimum) bounds. This corresponds exactly to the value of the frame emission time T_{EM}; moreover, since the switches are very fast, the considered configurations turn out to be very similar. The main difference lies in the use of one RIOM as an event source and another as the destination, with a difference of one in the scanning order (R_{15} and R_{16}). This result corroborates the general formulae obtained in Section IV.

VII. Conclusion

In this work, we have presented a new approach to evaluating the response time in networked automation architectures using client/server protocol. All delays experienced during the various system stages have been taken into account. A simple algorithm and analytical formulae for a trivial evaluation of response time have been developed; both the response time bounds and the distribution shape could thus be assessed. Afterwards, a comparison of results with experimental measurements enabled us to verify the validity of this method. Moreover, the analytical results obtained can be easily used *a priori*, during the design phase of an architecture, with application to choosing an adequate configuration for components in order to satisfy the desired time requirements of the plant. These results can also be used *a posteriori* to evaluate the response time of an existing architecture, as displayed in the previous examples. Throughout the study, many of the important results presented have been shown to match experimental observations.

For future studies, it would be worthwhile to consider more general automation architectures along with other protocols, a wide array of control loops with many clients and naturally relax some of our hypotheses like clocks drift of frame loss. The challenge then is to develop an efficient method so as to accurately evaluate the network delays in the presence not only of many loops, but also with acyclic and non-real time traffic, as encountered in modern NAS.

References

[1] P. Neumann, ”Communication in industrial automation - what is going on?”, *Control Engineering Practice*, Vol. 15, issue. 11, pp. 1332-1347, Nov 2007.

[2] MODBUS Messaging on TCP/IP Implementation Guide V1.0b, (2006, October), [Online]. Available:www.modbusida.org/specs

[3] R. L. Cruz, “A calculus for network delay, Part I: network in isolation”, *IEEE Trans. Information Theory*. Vol. 37, Issue. 1, pp. 114-131, 1991.

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

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

[6] X. Fan, M. Jonsson and J. Jonsson, “Guaranteed real-time communication in packet switched networks with FCFS queuing”, *Computer and networks*, Vol. 53, Issue. , pp. 400-417, November 2008.

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

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

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

[10] 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, 2006.

[11] 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.

[12] 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.

[13] Greifeneder and 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.

[14] B. Addad, S. Amari, “Modeling and response time evaluation of Ethernet-based automation systems using Max-plus algebra”, In *Proc. of 4th IEEE CASE*, Washington DC, USA, pp. 418–423, 2008.

[15] T. Murata, “Petri nets Properties analysis and applications” Proceedings of the IEEE Vol. 77 Issue. 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*, l (4), 1992.

[17] J. Mairesse, “Graphical Approach of the Spectral Theory in the (max,+) Algebra”, *IEEE TAC Transactions on Automatic Control*, Vol. 40, Issue. 10, pp. 1783-1789, 1995.

[18] J. Stanczyk and A. Obuchowicz, “The max-plus algebra approach to the prototyping of concurrent processes”, *In Proc. 9th IEEE Int. Conf. Methods and Models in Automation and Robotics*, Vol. 2, pp. 857-862, Miedzyzdroje, Poland, Aug. 2003.

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

[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 National Polytechnic School of Algiers, Algiers, Algeria, in 2007, and the M.S degree in automated systems engineering from Ecole Normale Supérieure de Cachan, Cachan, France, in 2008, and is currently pursuing the Ph.D. degree at 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. 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.