A Virtual-Queuing-Based Algorithm for Delay Evaluation in Networked Control Systems

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

***Abstract*—In this paper, an approach to evaluate time performances of networked control systems (NCSs) is presented. Switched Ethernet with a client/server protocol is considered for communication. As a result, the reactivity of these NCSs is not only affected by the network delays but also by the field-device due delays. Obviously, all of them have to be taken into account to evaluate efficiently the NCS reactivity. Thus, by including some preliminary results, we propose an overall study in an effort to achieve this aim. First, we model the traffic of the NCS and provide formal proofs to characterize the scenarios leading to the worst delays. Subsequently, we develop a virtual-queuing-based algorithm (VQA) to search these scenarios and deduce the corresponding delays. Through a practical case study, it is shown how VQA fits experimental observations both quantitatively and qualitatively.**

***Index Terms*—Client/server protocol, delay evaluation, networked control systems (NCSs), switched Ethernet, virtual queue (VQ).**

I. Introduction

NOWADAYS, the industrial organizations are more and more sophisticated, consisting of many intelligent devices connected through communication networks. The widespread of networked control systems (NCSs) was, in essence, motivated by reducing the volume of the wiring systems which are technically and economically a real issue. The bigger they are, the more difficult is their maintenance and the bigger is the bill of the consumed power (particularly with embedded systems in vehicles). Aside from NCS benefits, however, some challenges have emerged too. The network due delays, which are often not constant or even random, affect dramatically the control loops. To deal with this tricky problem, many solutions have been developed according to the context and the application of the NCS. Approaching this problem can be achieved through two different ways. The control is synthesized by ignoring the delays while a network scheduling is performed so as to minimize the delays. Many communication protocols were developed for this purpose [1]–[4]. The other approach is to first assess the network-induced delays and subsequently look for an adequate control strategy to compensate the negative effect of the delays. During the last years, this topic has been actively investigated, and many advanced control strategies have been proposed [5]–[8]. In the majority of these studies, however, only the network due delays have been considered [9]–[11], i.e., the network delays affecting a packet when traveling from a sensor to a controller and from a controller to an actuator. With client/server protocols like Modbus Transmission Control Protocol (TCP) [12], which is one of the most widespread protocols in industry, this assumption is actually no longer viable. Indeed, two different issues are to be investigated at the same time: the network delays on one hand and the field-device due delays on the other hand. Moreover, the latter delays are often by far larger than the network delays. As a matter of fact, no complete investigation has been carried out so as to evaluate the real-time capabilities of such NCS. The rare related works are often a simulation of some scenarios [13]–[15] without being exhaustive. In doing so, the critical cases that correspond to the worst delays are not guaranteed to be found. When it is the case using exhaustive methods like model checking [16], [17], only relatively simple architectures are targeted because of the classical state explosion problem of this method. Moreover, a trivial method that consists of adding up the worst local delays is possible, but it is so pessimistic (50% overestimation) that it is not practically viable [18]. Thus, the aim of this paper is to avoid the state explosion while considering enough large NCS. In addition, a formal proof of an accurate and exhaustive exploration is provided while taking into account both the network and field-device delays.

The remainder of this paper is organized as follows. An overview of client/server NCS and the context of the study are discussed in Section II. Then, in Section III, an analysis about the traffic interference is addressed, and the worst scenarios are identified. Thereafter, a virtual-queuing-based algorithm (VQA), to calculate the delays corresponding to these scenarios, is proposed in Section IV. Afterward in Section V, a practical case study is considered for validation. Finally, some concluding remarks are given in Section VI.

II. Background

As aforementioned, we consider an NCS that works according to the client/server protocol. It is constituted of programmable logic controllers (PLCs), remote input–output modules (RIOMs), PCs to perform high-level functions, and, finally, a switched-Ethernet network for communication (Fig. 1). The PLC comprises two modules, namely, the central processing unit (CPU) that executes the user program to perform the control signal and an Ethernet board for communication.

Manuscript received February 23, 2010; revised September 22, 2010; accepted November 18, 2010. Date of publication December 10, 2010; date of current version August 12, 2011.

The authors are with the Automated Production Research Laboratory (LURPA), Ecole Normale Supérieure de Cachan, 94235 Cachan, France (e-mail: boussad.addad@lurpa.ens-cachan.fr; said.amari@lurpa.ens-cachan.fr; Jean.Jacques.lesage@lurpa.ens-cachan.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2010.2098358

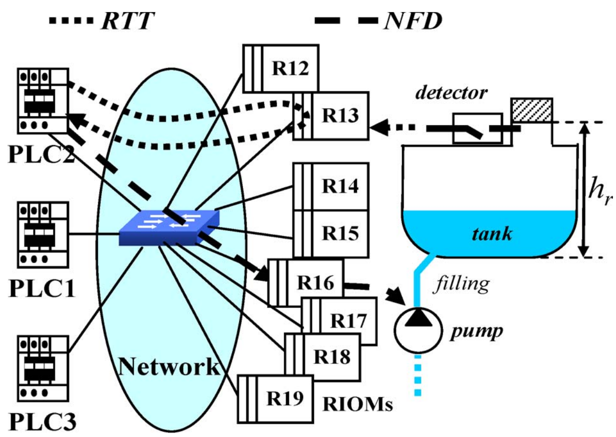

Fig. 1. Example of client/server NCS.

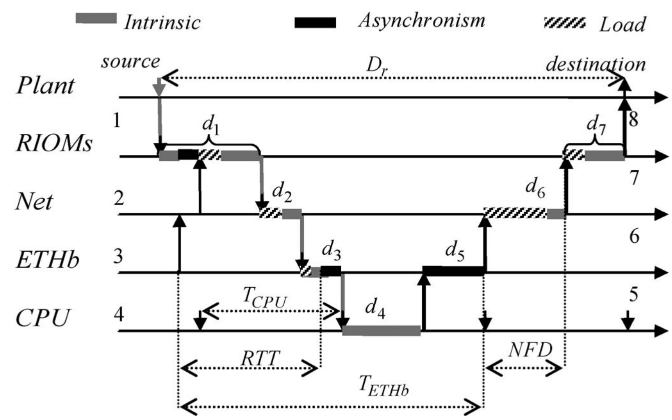

Fig. 2. Response time decomposition into three types of delays: Intrinsic delays, delays due to nonsynchronization, and, finally, delays due to load (waiting for resource availability).

To explain the functioning of such NCS, an example of automated plant is shown in Fig. 1. It consists of filling a tank and maintaining the level of water around a height h_r by operating a pump as long as this level is not reached. This is achieved as follows: PLC2 (client) periodically sends requests to RIOMs R13 and R16 (servers). By using the information got from a level detector (connected to R13), a decision is performed by PLC2 and sent to the pump trigger (R16).

The aim is to maintain the level around h_r but, above all, to *surely* avoid overflows while filling the tank. This depends on the maximal bound of the response time of the NCS and the tank itself. The *response time* noted D_r is defined as the delay between the date of reaching level h_r and the date of stopping the pump. Thus, the maximal bound of D_r must be smaller than the necessary time to fill the volume of the tank from the detector level to the top edge (hatched in Fig. 1). Hence, we notice the importance of this delay and its role in the NCS safety. Moreover, its impact on the stability and the quality of control was shown in [18]. The evaluation of maximal bound of the response time is therefore of top priority. This is the main aim of our study.

As a matter of fact, the evaluation of the response time of such a system is tricky. This is mainly due to the nonsynchronization of the components and the absence of a global scheduling of the shared resources in the NCS.

In Fig. 2, the response time can be seen as the sum of three types of delays at every stage of the NCS

$$D_r = \sum d_k. \tag{1}$$

1) Intrinsic delays: They are the inevitable delays due to the resource capacities [e.g., the delay to execute the user program and perform the control signal (d_4)].
2) Nonsynchronization: They are the delays due to the nonsynchronization of the different components. For example, a new value in the detector output is taken into account only after the arrival of a request from PLC2 (black part of d_1).
3) Load: delays due to waiting for the shared-resource availability, e.g., a request arriving to a RIOM is processed only after all the waiting requests are hatching in d_1.

According to the results exposed in [19], D_r depends on two major time features: the round-trip time (RTT) and the network forwarding delay (NFD). The obtained formula of the maximal response time was given as

$$D_{\mathrm{MAX}} \approx (q+1)\cdot T_{\mathrm{ETHb}} + \Delta_{\mathrm{NFD}} + d_7 \tag{2}$$

where q is the minimal integer that verifies inequality: $q > (RTT + T_{\mathrm{CPU}} + d_4)/T_{\mathrm{ETHb}}$. All the parameters in (2) are shown in Fig. 2; the RTT is the delay between the date of sending a request from PLC2 and the date of arrival of the returned answer from R13 (Figs. 1 and 2). The NFD is the necessary delay, to forward across the network the control signal issued from PLC2 to the destination R16 (Figs. 1 and 2). Thus, Δ_{NFD} is the jitter (the difference between the maximal and minimal values) of the delay NFD. Timing features T_{ETHb}, T_{CPU}, d_4, and d_7 are the period of the Ethernet board, the period of the CPU of PLC2, the necessary time to execute the user program in the CPU, and, finally, the necessary time to process the request in the destination RIOM R16, respectively. Formula (2) gives an upper bound of the response time, but the exact value of the maximal response time is a bit more complicated [19].

Obviously, one can tackle the problem differently by considering an upper bound as the addition of the theoretical maximum values of the local delays d_k

$$D_{\mathrm{MAX}} = \sum \max\{d_k\}. \tag{3}$$

Unfortunately, in such complex systems, these different delays are dependent of each other. When the response time is maximal, it does not mean that *all* these local delays reach their maximal values at the same time. An example in [18] showed that using (3) led to an overestimation of the real maximum bound with about 50%. As a result of this very pessimistic evaluation, the quality of control in the NCS was degraded to guarantee stability [18]. Using (2), however, led to a satisfactory result with an overestimation of only 1%, resulting in a much better quality of control. On top of that, a thorough analysis in [19] showed that the maximum bound D_{MAX} is reached when both the NFD and the RTT are maximal. Hence, we propose an approach to assess the maximal bounds of the RTT and the NFD so as to eventually use them in (2) to evaluate efficiently D_{MAX}.

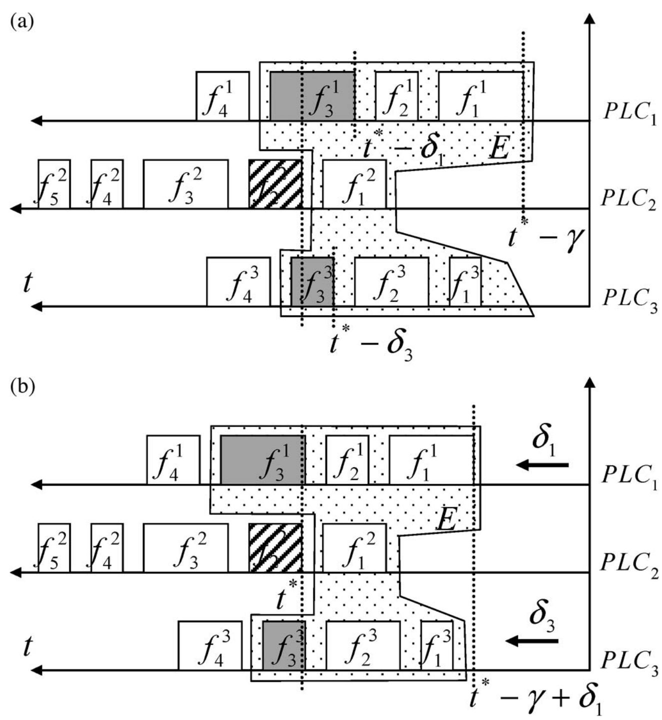

Fig. 3. Analysis of the frame interference. (a) Before shifting frames. (b) After shifting frames.

III. Traffic Modeling and Analysis

With client/server paradigm, the RIOMs are scanned periodically by the controllers. For instance, in Figs. 1–3, PLC2 sends five requests f_1^2, f_2^2, f_3^2, f_4^2, and f_5^2 to RIOMs R_{12}, R_{13}, R_{14}, R_{15}, and R_{16}, respectively. To assess the RTT relative to the control loop of Fig. 1, we have to find the delay that the frame f_2^2 (hatched in Fig. 3) experiences when crossing the network. Since the first-in–first-out (FIFO) scheduling is considered in the switch of the network, when the frame f_2^2 enters the switch at the date denoted by t^*, it experiences a delay that depends solely on the set of frames that entered the switch before it. Let us denote this set by E, as in Fig. 3.

This set corresponds also to the quantity of work (backlog) waiting to be processed at date t^*. Let us denote this backlog by $w(t^*)$. The challenge is then to find the scenario (set E) that corresponds to the worst delay that frame f_2^2 undergoes. The next lemma does not give precisely the worst scenario but provides an important propriety to identify it.

Lemma: Let f_2^2 be the frame under consideration and its arrival date into the switch be t^*. Let E be the set of frames arriving before f_2^2 and the frames from the parallel controllers entering immediately before f_2^2 be f_3^1 and f_3^3 (grayed in Fig. 3). Thus, we can formulate the following lemma.

If the set E corresponds to the worst scenario, then the worst delay is experienced if and only if the frames f_3^1 and f_3^3 entered the switch immediately before f_2^2. In other words, the arrival date of f_3^1 and f_3^3 is $t^* - \varepsilon$, where $\varepsilon > 0$ tends to zero.

Proof (by Contradiction): Suppose that the worst case delay noted Δ is reached when the frames f_3^1 and f_3^3 enter at date $t^* - \delta_1$ and $t^* - \delta_3$, respectively, as in Fig. 3(a) and the backlog at t^* is $w(t^*)$. Let us now shift the groups of requests from PLC1 and PLC3 with the lags δ_1 and δ_3 (this is possible since the PLCs generate their requests independently from each other) while keeping the frames f_3^1 and f_3^3 enter the switch before f_2^2, as in Fig. 3(b). Thus, the set E of frames preceding f_2^2 remains the same, but the quantity of work is no longer $w(t^*)$ but $w'(t^*)$ and the corresponding delay is Δ'. Thus, the frames from PLC1 enter δ_1 later than they do in the first case (case before shift), and the frames from PLC3 enter δ_3 later. Since the set E remains the same, then the backlog $w'(t^*)$ is necessarily greater than $w(t^*)$, and therefore, the delay Δ' is greater than Δ. Indeed, when the same set E begins to be processed at a later date, the processing is finished later too. Obviously, stating that Δ' is greater than Δ is absurd since from the beginning Δ is supposed to be the worst delay. ■

The previous lemma provides us with important information to identify the worst case scenario, but the set E that corresponds to it is still unknown. All we know is that the worst delay is reached when f_2^2 enters a bit after two frames (f_i^1, f_j^3) from the parallel controllers PLC1 and PLC3. Therefore, we can check every combination (f_i^1, f_j^3) for $1 \leq i \leq 4$ and $1 \leq j \leq 4$ and then simulate the system behavior according to the lemma.

Such a method would not be viable in general since it is not easy to identify discrete dates in systems that display uncertainties and complex configurations. Thus, instead of looking for these discrete dates, we rather search for the set E. As we saw before, however, the knowledge of this set is not sufficient since cases (a) and (b) correspond to the same set E but not the same delay. Fortunately, the next theorem will prove that, if the set E remains the same but we know how much the parallel frames are shifted with (at least the maximum shift), then the discrepancy of the corresponding delay from the worst one is known as well.

Theorem: Let the start-sending dates from PLC-1 and PLC3 be $(t^* - \tau_1)$ and $(t^* - \tau_3)$, respectively. In other words, $(t^* - \tau_1)$ is the arrival date of f_1^1, and $(t^* - \tau_3)$ is the arrival date of f_1^3. Looking for the worst scenario is obviously equivalent to looking for the corresponding lags τ_1 and τ_3.

If a step δ_e, smaller than the minimal interarrival time, is used to vary the lags τ_1 and τ_3 in a combinatorial scheme, then we have the following.

1) *An exhaustive exploration is achieved, and the set E that corresponds to the worst case is surely swept.*
2) *The accuracy of the assessed worst delay Δ is worth δ_e. In other words, $(\Delta_{\text{worst}} - \Delta) \leq \delta e$.*

Proof:

1) The first result is straightforward. Indeed, the date t^* varies with a step δ_e smaller than any interarrival period of two consecutive frames from PLC1 (f_i^1 and f_{i+1}^1). As a consequence, when varying the lags, t^* is surely between the dates of arrival of two consecutive frames f_i^1 and f_{i+1}^1. In the case of Fig. 3, when τ_1 varies with a step δ_e, we are sure to obtain t^* between the arrival dates of f_3^1 and f_4^1. Thus, the subset of E from PLC1 is found. The subset from PLC3 is obtained in exactly the same way.
2) The set E is found, but we obtain a configuration as in Fig. 3(a), not the worst case of Fig. 3(b). Indeed, we

obtain the arrival dates of f_3^1 and f_3^3 to be $t^* - \delta_1$ and $t^* - \delta_3$, where $\delta_1 \leq \delta_e$ and $\delta_3 \leq \delta_e$. The worst drift from case (b) is when $\delta_1 = \delta_3 = \delta_e$. This means that the frames from PLC1 and PLC3 enter the switch δ_e units of time earlier. Consequently, the frame f_2^2 will be forwarded at best δ_e earlier. Thus, the frame f_2^2 experiences a delay Δ with $\Delta \geq \Delta_{worst} - \delta e$. This can be written equivalently as $(\Delta_{worst} - \Delta) \leq \delta e$. ■

Remarks III.1:

1) Since an overestimation of the worst delay is sought, it is enough to add δ_e to the assessed delay to obtain $\Delta + \delta e \geq \Delta_{\mathrm{worst}}$ [consequence of result *2*)].
2) Once the set E corresponding to the worst case is found, we can tighten the step δ_e so as to improve the precision of the worst delay assessment (see the case study of Section V).

The issue now is to determine how the system behaves for a given set of lags τ_i. Then, by performing an exhaustive exploration, according to the previous theorem, the maximal bound of a delay is deduced. To the best of our knowledge, no suitable discrete event simulator is available to achieve this purpose. The features of client/server protocol, along with the presence of the RIOMs, are the main obstacles to that. In an effort to solve this problem, an investigation has already been carried out in [15]. Unfortunately, the proposed method is not an exhaustive one, as aforementioned in the introduction. Thus, the bounds of the delays are not guaranteed to be found. Basically, this was the main incentive behind the introduction of VQA, although it can be used in other contexts other than the client/server NCS [20].

IV. VQA

The algorithm we are going to present is based on some concepts we need to explain: virtual queuing and free access date (FAD).

A. Definitions

1) Virtual Queuing—Virtual FIFO queue (VFQ): A virtual queue (VQ) is a concept we associate to a real queue (input port in a switch). At a given time, a VQ not only contains all the frames that have been already received by the real one but also all the frames that will be received in the future, provided that the dates of their arrivals be known. More formally, it is a set $\langle f, \theta \rangle$, where $f = \{A_{\mathrm{SCR}}, A_{\mathrm{DST}}, L\}$ is a frame of length L, issued from a device (e.g., PLC) with address A_{SCR} and going to the destination address A_{DST}. Component θ is the date of arrival of the frame to the switch. At a given date t, some of the waiting frames have already quitted the VQ (black in Fig. 4), and others are still waiting (gray in Fig. 4). Thus, to know at each time the frame that is waiting to be forwarded first, we associate to each VQ a counter as in Fig. 4. Since there are multiple ports in a switch, we associate to each port a VQ. Therefore, we note $\langle f_q^p, \theta_q^p \rangle$ as the qth frame of the VQ relative to the pth port. Its counter is noted n_p.

The VQ is said to be FIFO or VFQ if $\theta^p_{(q+1)} \geq \theta^p_q \; \forall p, q \geq 1$, i.e., the frames are ordered according to their arrivals into the real queue. The first that arrived is the first to be forwarded. When a frame is forwarded, the counter is increased so as to point the potential following frame to depart in the future. That way, we know all the pending frames in all the VFQs at any time.

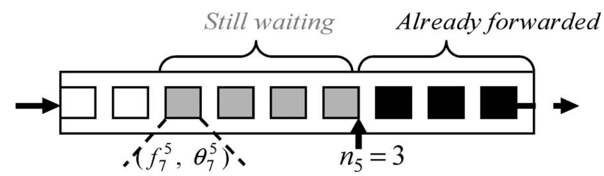

Fig. 4. VQ number 5 (input port 5).

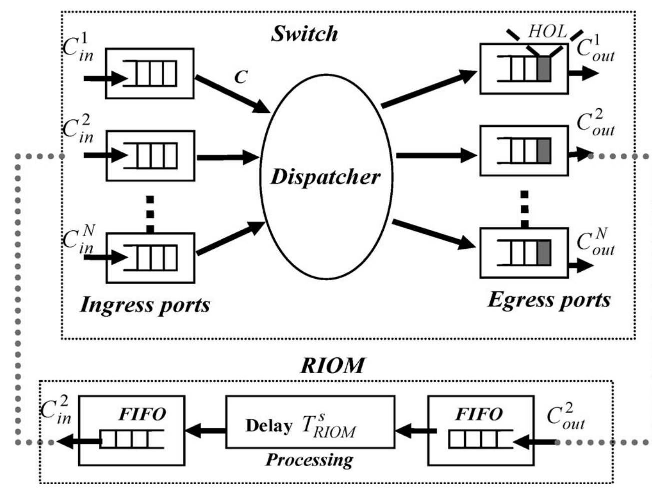

Fig. 5. Device modeling using queues and shared resources.

2) FAD: As aforementioned, no global scheduling of the resources is available in the NCS. Thus, considerable and variable delays due to waiting for access to the shared resources are experienced. The date at which access is possible to a shared resource is called "FAD." Thus, we associate a FAD to each shared resource of the NCS. Naturally, the shared resources depend on the models used in the study. In our case (Fig. 5), we use the same switch model considered in [2]. It involves a central dispatcher that forwards the frames according to FIFO scheduling, at a speed (in bits per second) noted C. In considering a "store-and-forward" switch, a frame is completely received before being forwarded by the dispatcher to the appropriate egress port. The data are exchanged with the end stations (PLC or RIOMs) at a bit rate (in bits per second) corresponding to physical link capacity, noted C^k_{in} or C^k_{out} for port k (Fig. 5). When a RIOM receives a request, at the bit rate of the switch port C^k_{out}, it buffers it in a FIFO queue until all the waiting frames are processed. Then, it processes it during a time T^s_{RIOM} before returning an answer at a bit rate C^s_{in}.

According to the models in Fig. 5, we can easily identify the following shared resources and their FAD:

1) F^u_{sw}: FAD to the dispatcher of the switch number u (the dispatcher can forward only one frame at a time);
2) F^s_{HOL}: FAD to the head of line (HOL) of the output port s (leaving an output port is possible only after all the frames in the head of the queue are retransmitted out of the switch);

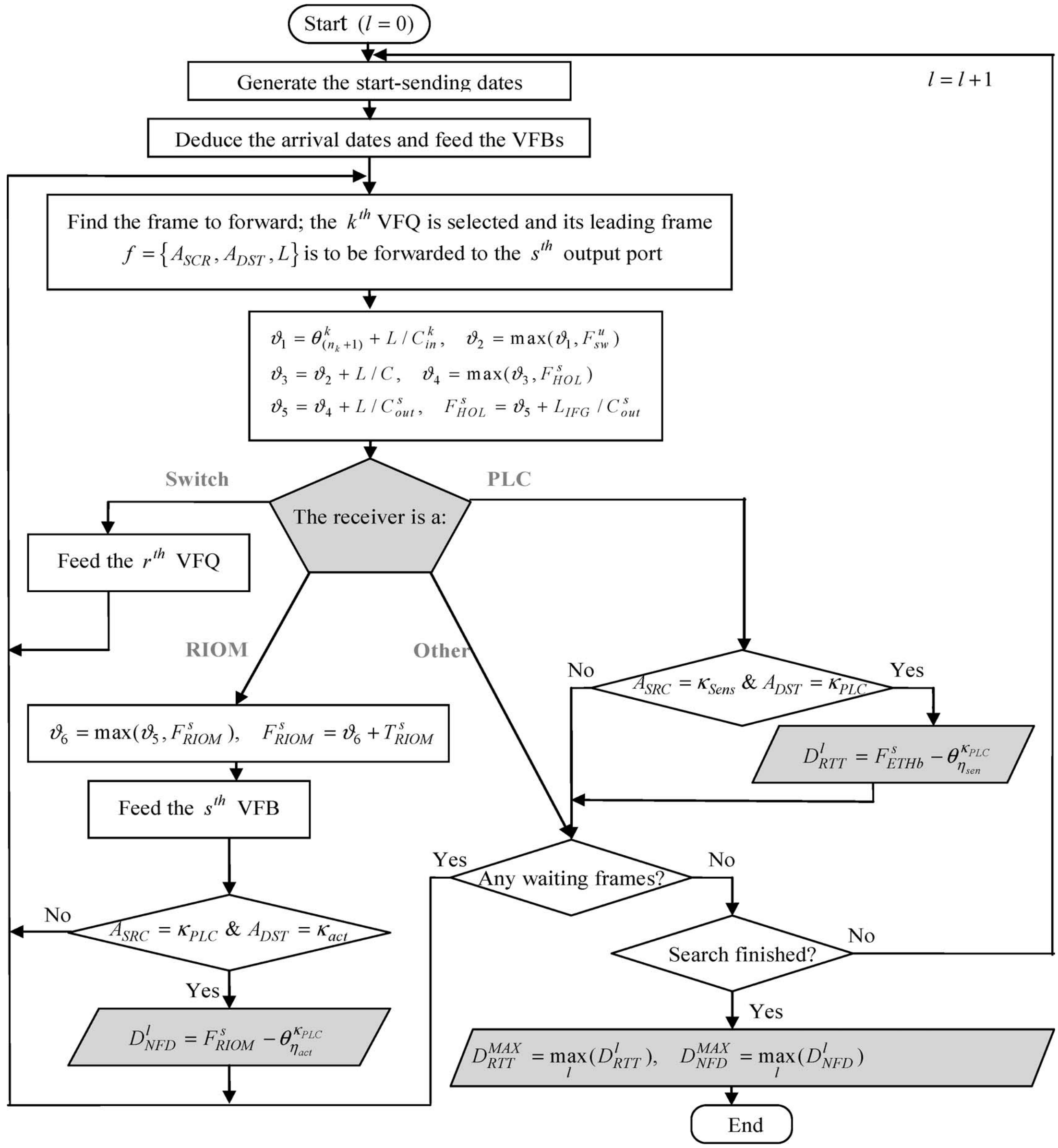

Fig. 6. Flowchart of VQA.

3) F^s_{RIOM}: FAD to the processor of the RIOM connected to the port s (a RIOM can process only one request at a time).

The main notions we handle being defined, we will explain step by step the algorithm VQA.

B. VQA

The flowchart drawn in Fig. 6 displays all the steps to achieve delay evaluation using VQA. The variables ϑ_i in the flowchart are only introduced for a better structuring of the algorithm and improving its clarity.

The idea behind VQA is to look at every step, among all the competing VFQs, which one will have access to the shared resources. When a VFQ is selected, its leading frame is forwarded until it reaches another VFQ or simply quit the system. Meanwhile, the FADs and also the contents of the competing VFQs are updated. The mechanism is repeated until all the VFQs are empty.

To explain the method as clearly as possible, a relatively simple system shown in Fig. 7(a) is used. The corresponding scanning mechanism is in Fig. 7(b).

1) PLC1 sends a burst of two requests periodically with a period T_1 to modules R12 and R13.
2) PLC2 sends a request periodically with a period T_2 to module R11.

We can note from Fig. 7(a) that all the ports of the switches are numbered differently (Sw1: 1, 2, 3; Sw2: 4, 5, 6, 7). Thus, we talk about the kth port of the system that ascribed the kth VFQ. In such a way, we can use as an address of each station only the number of the port that connects it to the network. Thus, we adopt the following notations.

1) κ_{PLC} is the address of the controller of the loop in question. In Fig. 7(a), it is PLC1; thus, $\kappa_{\mathrm{PLC}} = 2$.
2) κ_{sen} is the address of the sensor. In Fig. 7(a), the sensor is connected to R12, and therefore, $\kappa_{\mathrm{sen}} = 6$.

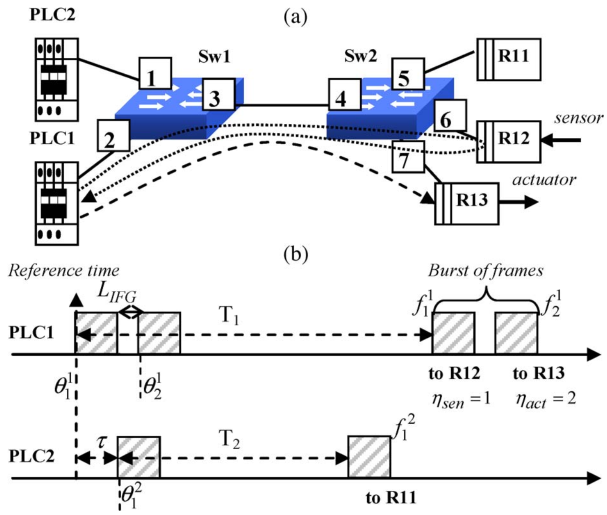

Fig. 7. (a) Example of NCS. (b) Scanning mechanism.

3) κ_{act} is the address of the actuator. In Fig. 7(a), the actuator is connected to R13, and therefore, $\kappa_{\text{act}} = 7$.

Since the controller may send a burst of several requests to the RIOMs [e.g., PLC1 in Fig. 7(b)], we have to identify the frames that are involved in the loop in question: the request sent to the sensor and the one sent to the actuator. Therefore, we identify them by their rank in the burst.

1) η_{sen} is the rank of the frame sent to the sensor ($\eta_{\text{sen}} = 1$ in the example in Fig. 7).
2) η_{act} is the rank of the frame sent to the actuator ($\eta_{\text{act}} = 2$ in the example in Fig. 7).

With the notations being defined, we can explain VQA as follows.

1) *Step0 (initialization of VQA)*: First of all, we generate the dates of start sending the frames from the PLCs: θ_1^1 and θ_1^2. We put the date of start sending of one of them as a reference, and the others are generated with lags from it. In Fig. 7(b), the reference is PLC1, and the start-sending date of PLC2 is generated with a lag τ from it.

   With the start-sending dates of the leading requests in each burst being known, the arrival dates to the switch Sw1 of the following frames are deduced. In the example, θ_2^1 is deduced using the length of frame f_2^1, the interframe gap L_{IFG}, and the bit rate of port 1.

   Thus, we can already feed the VFQ numbers 1 and 2 (of the ports linking the PLCs to Sw1). The counters of the VFQs are all set to zero since no frame has already been forwarded ($n_k = 0\,\forall k$).
2) *Step1*: Among all the nonempty VFQs (#1 and #2 at this step), we select the one whose HOL (leading frame that waits to be forwarded first) has the minimal date. If it is the kth VFQ of the switch number u, then it must verify $\theta^k_{(n_k+1)} = \min\limits_p(\theta^p_{(n_p+1)})$. The frame to be forwarded is supposed to be $f = \{A_{\text{SCR}}, A_{\text{DST}}, L\}$, and the egress port that will receive it is the sth port (to be determined using a lookup table). If we look at the scenario drawn in Fig. 7(b), the selected frame is f_1^1 since $\theta_1^1 = \min\{\theta_1^1, \theta_2^1, \theta_1^2\}$. The egress port is $s = 3$.
3) *Step2*: The frame is selected, but the forwarding is possible only at date $\vartheta_2 = \max(\vartheta_1, F^u_{\text{sw}})$ with $\vartheta_1 = \theta^k_{(n_k+1)} + L/C^k_{\text{in}}$, i.e., the frame is completely received (store-and-forward switch) and the dispatcher u is free.
4) *Step3*: The frame is received completely at the egress line at date $\vartheta_3 = \vartheta_2 + L/C$, but once again, the frame can exit the switch only after the HOL of egress port s is free, i.e., at date $\vartheta_4 = \max(\vartheta_3, F^s_{\text{HOL}})$. Hence, the frame exits completely the switch at date $\vartheta_5 = \vartheta_4 + L/C^s_{\text{out}}$. The FAD of the HOL of egress port s is updated using this date and the interframe gap $L_{IFG} : F^s_{\text{HOL}} = \vartheta_5 + L_{\text{IFG}}/C^s_{\text{out}}$.
5) *Step4*: At this point, four cases can be analyzed depending on the type of the receiver of the previous frame: a switch, a RIOM, a PLC, or another station (e.g., a PC that does not return answers).
   a) Switch (e.g., a frame that exits from port 3 to enter port 4): If the port of this receiving switch is ascribed the number r, then the rth VFQ is fed with the previous frame. Go to *Step1)*.
   b) RIOM: In this case, the frame will wait in the FIFO buffer of requests until the access is free to the RIOM processor: $\vartheta_6 = \max(\vartheta_5, F^s_{\text{RIOM}})$. Then, it is processed, and a response is returned. This answer is received at the input port s (which was an output port previously). Thus, the VFQ s is fed with the response frame, and the FAD of the RIOM is updated. Hence, we make a test if this RIOM is the control signal destination R13 and the source address of the frame is PLC2. If it is the case, the NFD can be calculated. Go to *Step1)*.
   c) PLC: Here, also, we make a test if this PLC is the sender of the control signal (PLC2) and the RIOM that returned the answer is R12. If it is the case, the RTT can be calculated. Go to Step 4d).
   d) A station without answer (including PLCs): In this case, we only check if there are waiting frames in the different VFQs of the NCS. If it is the case, then go to *Step1)*; else, the algorithm is terminated.

The previous algorithm is run for a particular set of start-sending dates. The algorithm is repeated as many times as necessary according to the theorem in Section II so as to find the critical scenario. This is represented using the exploration loop (index l) on the flowchart which finishes by the calculus of the maxima of the delays RTT and NFD

$$D^{\text{MAX}}_{\text{RTT}} = \max_l \left(D^l_{\text{RTT}}\right) \qquad D^{\text{MAX}}_{\text{NFD}} = \max_l \left(D^l_{\text{NFP}}\right).$$

Remarks IV.1: While the client/server protocol (e.g., Modbus TCP [12]) displays many advantages that justify its widespread in industry automation, the continuous periodic scanning of the remote modules would be a disadvantage in some cases. Indeed, when a client asks continuously for a sensor data while this latter does not change during many cycles, it results unnecessarily in an overcrowded network. Other protocols like producer/consumer (e.g., ProfiNet [3]) bypass this problem by constraining the data emission; a sensor does send a message only if its state changes.

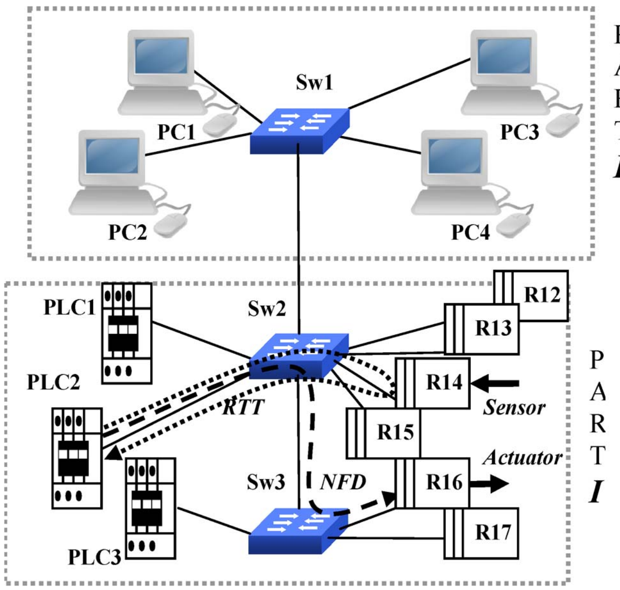

Fig. 8. Client/server NCS: Case study.

TABLE I
REQUEST PROCESSING DURATIONS IN THE RIOMS (IN MILLISECONDS)

R12	R13	R14	R15	R16	R17
0.50	0.60	0.70	0.55	0.60	0.50

TABLE II
RESULTS OF EVALUATION USING VQA

		Case 0	Case 1	Case 2	Case 3	Case 4
Removed components		PC2, PLC1, PLC3	PC2, PLC1	PLC1	PC2	
RTT (ms)	***64B***	1.10	1.10	1.86	1.80	2.56
	89B	1.23	1.23	1.99	1.90	2.72
NFD (ms)	***64B***	0.91	1.51	1.51	2.11	2.11
	89B	0.96	1.53	1.53	2.19	2.19

Fortunately, the concepts of VFQ and FAD are generic enough to be extended easily to these protocols without changing the core of VQA. This can be achieved quite straightforwardly by inverting the roles of the controllers and the sensors. A sensor can be considered as a client that sends requests, whereas a controller can be considered as a server that returns answers. The necessary time to a request to be processed by a controller is the time to execute the user program, noted as d_4 previously in Section II. Nevertheless, in considering such protocols, one has to take care about the information broadcast by a sensor. In the event that there are many consumers of this information, multicast has to be taken into account in the models of the NCS components, and VQA has to be adapted accordingly.

V. CASE STUDY: APPLICATION AND VALIDATION

The system is shown in Fig. 8 and includes two parts: *PART I* and *PART II*. *PART I* works as follows: Three controllers, namely, PLC1, PLC2, and PLC3, scan the ordered sets of RIOMs, respectively: $\{R_{12}, R_{13}, R_{14}\}$, $\{R_{14}, R_{15}, R_{16}\}$, and $\{R_{16}, R_{17}\}$. Minimal length Ethernet frames of 64 B are used for scanning the RIOMs in the first case and 89-B frames in the second case. The two cases are considered so as to figure out the influence of the frame length on the delays. *PART I* is the automation part, as the one we used for the explanation in Section II, but as stated previously, the studied system presents the advantage of easiness of high-level function integration (since standard Ethernet is used for communication). Thus, the second part *PART II* with such a role is connected to this subnetwork; PC1 sends long frames of 1000-B length to the stations PC3 and PC4. PC2 has the role of a supervisor in *PART I*. It scans all the RIOMs every 300 ms so as to check the good functioning of the NCS. All the links of the switches are configured to work at a full duplex mode with a speed of 10 Mb/s. The processing times T^s_{RIOM} in the different RIOMs are very slightly variable in practice. Indeed, the experimental measures performed on a RIOM [21] led to a maximal value of 0.5471 ms and a minimal one of 0.5428 ms, i.e., less than 0.8% jitter. Thus, the rounded-up values are used in our study, as given in Table I.

Now, the aim is to evaluate the upper bounds of the delays RTT and NFD relative to the control loop shown in Fig. 8; PLC2 is the controller, R14 is the source of information (sensor), and R16 is the destination of the control signal (actuator). The evaluation was performed with different conditions of load so as to analyze the influence of the parallel traffic on this loop. Many cases were considered by removing one or more stations from the system. In each case, the removed components are mentioned in Table II. Case 0, where no load is present (all the flows affecting the loop are removed), was the reference of our analysis. Hence, VQA was applied by taking into consideration the previous theorem conditions. With Ethernet, we can easily calculate the minimal interarrival time as $\delta = (64 + preamble + L_{\mathrm{IFG}})/(10\ \mathrm{Mb/s}) = 67.2\ \mu$s. Thus, the exploration step δ_e must be smaller than this value. A progressive narrowing of the step δ_e is performed with the condition checked: $\delta_e = 50$, then 10, and, finally, 1 μs. The final results are reported in Table II.

As expected, whatever is the length of the frames, the delays RTT and NFD had their smallest values in Case 0. Depending on the removed traffic, either RTT or NFD changed while the following observations were noticed.

1) RTT: As we can notice, PLC3 had no influence on RTT since no effect was observed when it was added from Case 0 to Case 1. This is simply due to the fact that no intersection exists between the RTT path and PLC3 request path. When PLC1 or PC2 was added however (Cases 2 and 3), an additional delay of about 0.7 ms was noticed. When they were present at the same time, twice this value was added (Case 4), and a variation of the RTT of about 130% from Case 0 to Case 4 was observed. The little difference between Cases 2 and 3 is due to the effect of sharing the resource Sw2. As a matter of fact, a value of 0.7 ms is the necessary delay to process a request in

R14 (see Table I). Thus, the more it is shared by clients, the more the request from PLC2 suffers when waiting for the module R14 processor availability.

2) NFD: Almost the same phenomenon was observed with NFD. Indeed, when a client of the module R16 was added (Cases 1 and 2), an extra delay of about 0.6 ms was noticed and twice this value when two clients were present (Cases 3 and 4). It followed a variation of about 128% from Case 0 to Case 4. The time to process a request in R16 is also equal to 0.6 ms.
3) *Network delay* (experienced exclusively in the switches): When all the flows were present and, therefore, the network maximally crowded (Case 4), the pure-network-delay-to-RTT ratio was less than 20%. The same proportion can be estimated with respect to NFD. Hence, we get to the same conclusion stated previously in Section II; the sole pure network delay is not enough to carry out an efficient evaluation of this NCS time performances. The whole architecture is to be considered in the study.
4) *Frame length*: The effect of changing the length of the frames was maximal in Case 0 with ratios of about 5% for NFD and 10% for RTT. Hence, we can conclude that the impact of the frame length and that of the parallel traffic (affecting the pure network delays) are much smaller than the effect of sharing the RIOMs.
5) *Response time*: By setting the scanning period of PLC2 to 10 ms and the CPU period to 5 ms, the use of (2) led to upper bounds of the response time of about 22.6 ms in Case 0 and 23.7 ms in Case 4 (both cases using 89-B frames). This means a tight overestimation of the experimental value (22.3 ms [22]) with only 1.5%. Thus, both overestimation and accuracy were achieved at the same time. We can also point out that the effect, on the response time, of sharing the RIOMs was of about 6% (calculated as $(2.19 - 0.96)/22.3$). Therefore, all the conclusions of this study fit the experimental ones in [22], where it was stated that the effects of the network components and the frame length are not striking whereas sharing the RIOMs causes less than 10% effect.

VI. Conclusion

In this paper, we have proposed a method to assess switched-packet delivery delays in the context of client/server NCSs. A VQA, taking into account both the field devices and the switches, was developed for this purpose. Moreover, a formal proof was provided about the capacity of VQA to achieve exhaustive exploration and sweep the worst delays. Thereafter, it was shown that VQA fits different experimental observations and provides satisfactory results. Thus, instead of making thousands of onerous experimental measures, VQA is a satisfactory alternative with efficient delay evaluation.

References

[1] P. Neumann, “Communication in industrial automation: What is going on?” *Control Eng. Pract.*, vol. 15, no. 11, pp. 1332–1347, Nov. 2007.

[2] J. Jasperneite, J. Imtiaz, M. Schumacher, and K. Weber, “A proposal for a generic real-time Ethernet systems,” *IEEE Trans. Ind. Informat.*, vol. 5, no. 2, pp. 75–85, May 2009.

[3] J. Kjellson, A. E. Vallestad, R. Steigmann, and D. Dzung, “Integration of a wireless I/O interface for PROFIBUS and PROFINET for factory automation,” *IEEE Trans. Ind. Electron.*, vol. 56, no. 10, pp. 4279–4287, Oct. 2009.

[4] L. Seno, S. Vitturi, and C. Zunino, “Analysis of Ethernet powerlink wireless extensions based on the IEEE 802.11 WLAN,” *IEEE Trans. Ind. Informat.*, vol. 5, no. 2, pp. 86–98, May 2009.

[5] K. Natori and K. Ohnishi, “A design method of communication disturbance observer for time-delay compensation, taking the dynamic property of network disturbance into account,” *IEEE Trans. Ind. Electron.*, vol. 55, no. 5, pp. 2152–2168, May 2008.

[6] X. Tian, X. Wang, and Y. Cheng, “Self-tuning fuzzy controller for networked control system,” *Int. J. Comput. Sci. Netw. Security*, vol. 7, no. 1, pp. 97–102, 2007.

[7] G. P. Liu, Y. Xia, J. Chen, D. Rees, and W. Hu, “Networked predictive control of systems with random network delays in both forward and feedback channels,” *IEEE Trans. Ind. Electron.*, vol. 54, no. 3, pp. 1282–1297, Jun. 2007.

[8] K. Natori, T. Tsuji, K. Ohnishi, A. Hace, and K. Jezernik, “Time delay compensation by communication disturbance observer for bilateral teleoperation under time-varying delay,” *IEEE Trans. Ind. Electron.*, vol. 57, no. 3, pp. 1050–1062, Mar. 2010.

[9] R. L. Cruz, “A calculus for network delay, Part I: Network in isolation,” *IEEE Trans. Inf. Theory*, vol. 37, no. 1, pp. 114–131, Jan. 1991.

[10] J. P. Georges, E. Rondeau, and T. Divoux, “Confronting the performances of switched Ethernet network with industrial constraints by using network calculus,” *Commun. Syst.*, vol. 18, no. 9, pp. 877–903, Nov. 2005.

[11] K. C. Lee, S. Lee, and M. H. Lee, “Worst case communication delay of real time industrial switched Ethernet with multiple levels,” *IEEE Trans. Ind. Electron.*, vol. 53, no. 5, pp. 1669–1676, Oct. 2006.

[12] *MODBUS Messaging on TCP/IP Implementation Guide V1.0b*, Oct. 2006. [Online]. Available: http://www.modbus-ida.org/specs.php

[13] H. 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. 12th IFAC INCOM*, St-Etienne, France, 2006, pp. 101–106.

[14] D. A. Zaitsev, “Switched LAN simulation by colored Petri nets,” *Math. Comput. Simul.*, vol. 65, no. 3, pp. 245–249, Apr. 2004.

[15] G. Marsal, B. Denis, J.-M. Faure, and G. Frey, “Evaluation of response time in Ethernet-based automation systems,” in *Proc. ETFA*, Prague, Czech Republic, 2006, pp. 380–387.

[16] J. Greifeneder and G. Frey, “Probabilistic timed automata for modeling networked automation systems,” in *Proc. 1st IFAC Workshop DCDS*, Cachan, France, 2007, pp. 143–148.

[17] S. Ruel, O. De Smet, and J.-M. Faure, “Finding the bounds of response time of networked automation systems by iterative proofs,” in *Proc. 13th IFAC INCOM*, Moscow, Russia, 2009, pp. 1365–1370.

[18] B. Addad and S. Amari, “Delay evaluation and compensation in Ethernet networked control systems,” in *Proc. 16th Int. Conf. RTNS*, Rennes, France, 2008, pp. 139–148.

[19] B. Addad, S. Amari, and J.-J. Lesage, “Analytic calculus of response time in networked automation systems,” *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 4, pp. 858–869, Oct. 2010.

[20] B. Addad, S. Amari, and J.-J. Lesage, “Evaluation de délais dans les systèmes de communication temps réel en utilisant des files d’attente virtuelles,” *MSR—J. Europeen des Systemes Automatisé (JESA)*, vol. 43, no. 7–9, pp. 985–1000, 2009.

[21] G. Marsal, “Evaluation of time performances of Ethernet based automation systems by simulation of high level Petri nets,” Ph.D. dissertation, Kaiserslautern Univ., Kaiserslautern, Germany, 2006.

[22] B. Denis, S. Ruel, J.-M. Faure, and G. Marsal, “Measuring the impact of vertical integration on response times in Ethernet fieldbuses,” in *Proc. ETFA*, Patras, Greece, 2007, pp. 532–539.

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, where he is currently working toward the Ph.D. degree.

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 with the University of Paris XIII, Paris, France. He carries out research at the Automated Production Research Laboratory, École Normale Supérieure de Cachan, 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 and the "Habilitation à Diriger des Recherches" from the University Nancy 1 Henri Poincaré, Nancy, France, in 1989 and 1994, respectively.

He is currently a Professor of automatic control with 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.