Measures vs. Analytic Evaluation of Response Time of Networked Automation Systems

Boussad Addad, Saïd Amari, Jean-Jacques Lesage, Bruno Denis

Measures vs. Analytic Evaluation of Response Time of Networked Automation Systems

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

***Abstract*— In this paper, we present a study that deals with response time of Networked Automation Systems (NAS). The main aim is to verify experimentally the validity of an analytic formulation of NAS response time, obtained in our past investigations [12]. To that purpose, we carried out, under different conditions, a lot of measurements on a laboratory facility. Each time, we compared the observed measures to the predicted values. Overall, the study reveals clear agreement between the real observations and the theoretic predictions of our formula. It therefore promotes the formula to be used confidently for NAS response time evaluation while avoiding tedious and onerous experiments.**

I. Introduction

NOWADAYS, the industrial organizations are more and more sophisticated, consisting of many intelligent devices connected via communication networks. While the benefits of these networks are considerable, several issues have emerged too (delays, jitter, information loss …). So, the performances of systems subject to real-time constraints like Networked Automation Systems (NAS) are therefore affected [1]. So, a beforehand evaluation of these time features is needed before operating a NAS on a real site. In literature, many methods have been proposed to deal with network delays [2], [3], [4], [5]. In this paper, we investigate the evaluation of a major time feature of a NAS, the *response time*. It is the delay between the occurrence of an event in a plant (sensor) and the occurrence of its consequence on the controlled plant (actuator).

Depending on the communication protocol of the NAS, the response time can be evaluated more or less difficultly. In client/server automation systems over standard Ethernet, it is quite tricky to achieve this evaluation. This is mainly due to the non synchronization between the field devices of the NAS and the absence of a global scheduling of the shared resources. Nonetheless, many methods have been proposed to assess the response time of these NAS; non exhaustive methods based on simulation [6], [7], [8] or experimentation [9], and exhaustive ones, using model-checking [10], [11]. Lately, we exposed an analytic method in [12] and provided formulae to calculate the bounds of response time. A probabilistic version of the method is also exposed in [13] to calculate the response time distribution. In this paper, we propose a further experimental investigation in an effort to verify if the maximal bound formula in [12] fits reality.

The remainder of this paper is structured as follows: in Section II, an overview of a NAS, considered along the paper, is described. The formula that gives its maximal response time is recalled and explained. Thereafter, a comparison between predictions of the formula and a lot of real measures is performed in Section III. Finally, some concluding remarks and perspectives are given in Section IV.

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).

B. Denis is with Automated Production Research Laboratory, 61 av. du Président Wilson, 94235 Cachan Cedex, France (phone: +33- 0147402413; e-mail: bruno.denis@lurpa.ens-cachan.fr).

II. NAS Overview and Response Time Evaluation

A. NAS composition and functioning

The networked automation system we consider along this paper is represented in Fig. 1. It is constituted by the following devices:

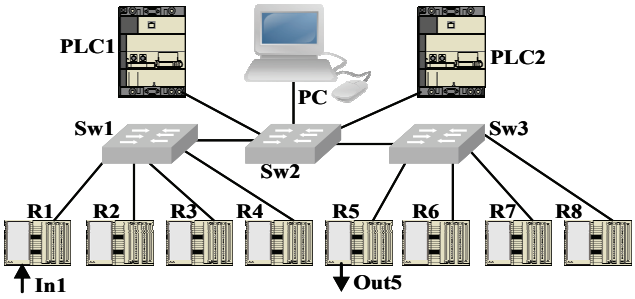

Fig. 1. Networked automation system.

i) Remote I/O modules (R1 to R8) to get/provide information from/to the plant. They are connected to the field instruments like sensors and actuator, etc. Each module includes an Ethernet Interface that enables it to communicate with the other field devices. Also, each remote I/O does only answer the requests it receives, according to client/server paradigm. No asynchronous transmission from a remote I/O is considered in our study.

ii) Controllers: they can be PLC (Programmable Logic Controller) or any PC including a control application. These controllers poll periodically the remote I/O modules to

request some information (e.g. read input, read memory, etc) or provide information (e.g. write output, write memory, etc). This periodic poll is also called *I/O scanning*. It is achieved thanks to an Ethernet Interface. The exchanged variables with the field are stored in a shared a memory (Fig. 2). Obviously, the main role of a controller is to execute the user program using the input variables to update the outputs. This task is performed either cyclically or periodically, but in both modes, independently from the scanning period used by the Ethernet interface. The PLC and its Ethernet interface are not synchronized (see Fig. 2).

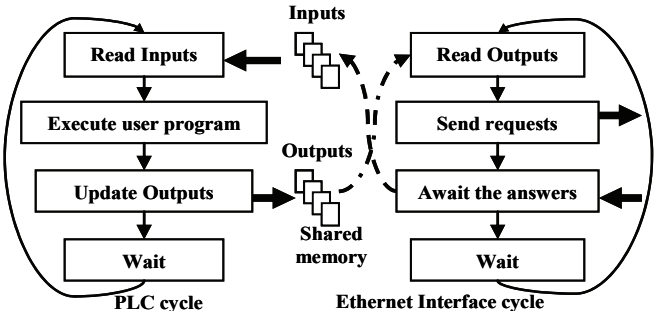

Fig. 2. A PLC and its Ethernet interface are not synchronized.

iii) Network devices (e.g. Ethernet switches): they link the previous field components altogether.

A simple automation function was implemented in the controller PLC1 of Fig. 1. It consists of transmitting the value of the input of R1, noted In1, to the output of R5, noted Out5. To achieve this purpose, PLC1 polls periodically module R1 to request the value of input In1. When this value is returned, the Ethernet interface puts it in the shared memory. At the beginning of a PLC cycle, this stored value is copied into another shared memory ascribed to output Out5 (Fig. 2). Also, the Ethernet interface polls periodically module R5. At the beginning of every scanning cycle, it copies the value of the memory associated to Out5 and sends it to module R5.

Note by the way that, in the considered NAS, PLC1 polls also modules (R2, R3, R4, R6), one after another during the same scanning cycle. Modules R7 and R8 are polled by PC.

B. Analytic evaluation of response time

Suppose that when input In1 is equal to 1, it means that a leak in a gas pipe is detected and changing output Out5 from 0 to 1 means closing a valve to stop this leak. This brings naturally to question: what time elapses between the occurrence of the leak detection and the date of stopping it? This time is actually what is called *response time*, noted D_r in Fig. 3.

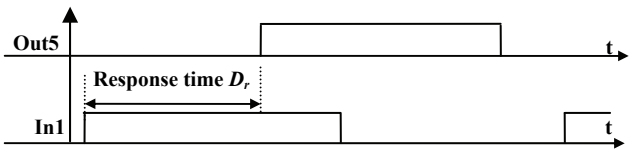

Fig. 3. Response time of a NAS (delay between the rising edge of input In1 and the rising edge of output Out5)

Given the previous client/server paradigm, the value of input In1 is not transmitted directly to PLC1 when its state changes but waits until the arrival of a request from PLC1. So, a delay due this non synchronization affects the response time. Also, when the answer returned by R1 gets to the Ethernet interface of PLC1, it is not taken into account immediately but waits until the beginning of a new cycle of PLC1. This independence of PLC1 and its interface affects the response time as well. So, the response time of these NAS is not easy to evaluate and depends on many factors.

For obvious room reasons, we will recall in this paper only the key points of the analytic approach of response time evaluation we developed in [12]. The whole procedure we followed to achieve this evaluation is summarized in the scheme below (Fig. 4):

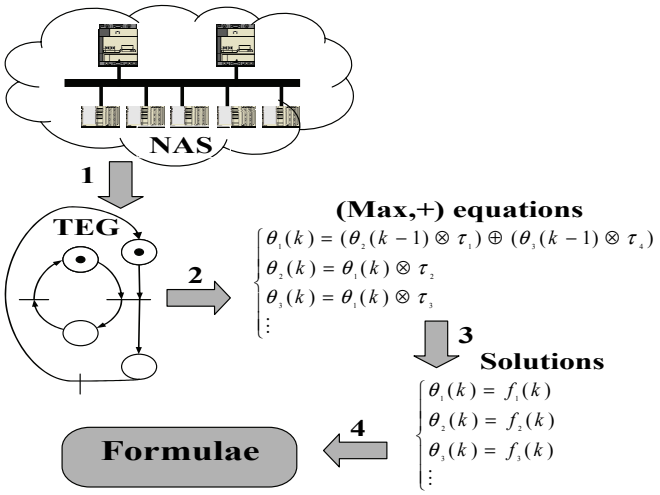

Fig. 4. Steps to achieve analytic evaluation of response time [12].

- *Step 1:* The First step of the method was to model the NAS using Timed Event Graphs (TEG). A TEG is a Petri net whose places all display at most one upstream transition and one downstream transition. An example of such a model is shown in Fig. 5. It is the model of a NAS with one PLC and one I/O module, given only for illustration. The general model is much larger and is therefore not given here (See [12]).

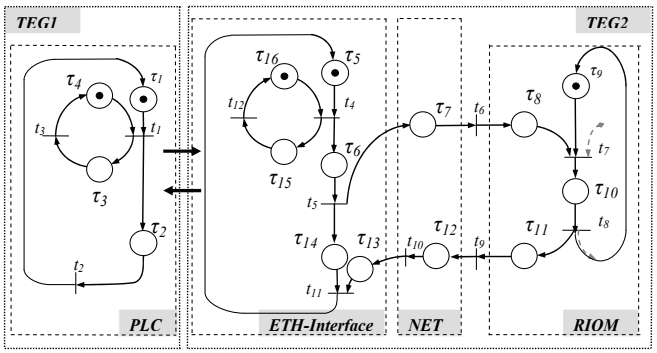

Fig. 5. TEG-based model of a mono-PLC mono-I/O NAS [12].

- *Step 2*: The second step of the method is the representation of the TEG dynamics using linear equations in (max,+) algebra (equations involving the classical maximum noted $\oplus$ and classical addition noted $\otimes$). Timed event graphs can indeed be represented using such linear equations. For more details about TEG and (max,+) algebra, we invite the reader to see the reference book [14].

For illustration, the obtained (max,+) equations of the model of Fig. 5 are:

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

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

The variables θ of these equations are daters. They are associated to the transitions of the TEGs. Each dater $\theta_i(k)$ represents the date of firing transition t_i (occurrence of an event) for the *k-th* time. For instance, $\theta_1(k)$ is the date of starting the *k-th* PLC cycle. Delays τ_i are simply the durations of different tasks in the NAS. For example, τ_2 is the time to execute the user program in the PLC.

- *Step 3*: the third step is the resolution of the previous equations to get the dates as only functions of the cycles' indices, *k* and *l*.

- *Step 4*: we used these dates (solutions) to track every event occurring in a sensor along its traveling in the NAS until its arrival to the actuator. Finally, the response time is calculated as the difference between two dates: the date of event occurrence in the sensor and the date of its arrival to the actuator. At the end, formulae that provide the response times, relative to occurred events, are obtained.

Besides that, a formula of the worst response time is given. In considering the NAS of our study, the max response time can be expressed as follows:

If the maximal response time occurs at the l-th scanning cycle, then it is given as:

$$D_r = (q+1) \cdot T_{ETH} + T_{Out}(l+q) - T_{In}(l) + T_{Proc} + T_{filt} \qquad (1)$$

The different parameters of formula (1) stand for:

- T_{ETH} is the scanning period of PLC1 Ethernet interface.

- $T_{Out}(l+q)$ is the delay $PLC1 \xrightarrow{(l+q)-th\ cycle} Out5$, experienced by a request during its trip, from its generation by PLC1, at the *(l+q)-th* scanning cycle, to its arrival to output Out5 (number $q \in \mathbb{N}$ is explained below).

- $T_{In}(l)$ is the delay $PLC1 \xrightarrow{l-th\ cycle} In1$, experienced by a request during its trip, from its generation by PLC1 at the *l-th* scanning cycle to its arrival to input In1.

- T_{Proc} is the necessary time to process, in module R1, the request coming from PLC1.

- T_{filt} is the necessary time to filter the data coming from the field (sensor) in module R1.

- Finally, q is the minimal integer number that verifies the following condition:

$$q > (T_{RTT}(l) + T_{PLC} + T_{Exc}) / T_{ETH} \qquad (C1)$$

The involved parameters in (C1) stand for:

- $T_{RTT}(l)$ is the round trip time $PLC1 \xrightleftharpoons{l-th\ cycle} In1$, experienced by a request during its trip, from its generation by PLC1, at the *l-th* scanning cycle, to its arrival to R1 + the delay experienced by the returned answer, from its generation by R1 to its arrival to PLC1.

- T_{PLC} is the functioning period of PLC1 (in periodic mode).

- T_{Exc} is the necessary time to execute the user program implemented in PLC1 (Inputs read and outputs update included).

Hence, the main scope of the current paper is to check the validity of formula (1) using experimentation.

Remarks II.B.1:

- A PLC functions either cyclically or periodically. In periodic mode, the phase "Wait" in Fig. 2 means that the PLC finishes updating the outputs, then waits for the period T_{PLC} to elapse before starting another cycle. In cyclic mode however, the PLC starts a new cycle as soon as outputs update is accomplished. So, the "Wait" phase is null and therefore $T_{PLC} = T_{Exc}$. Thus, the previous condition (C1) becomes:

$$q > (T_{RTT}(l) + 2 \cdot T_{Exc}) / T_{ETH} \qquad (C2)$$

This condition is used later in the study to analyze and discuss the experimental results with regard to formula (1).

- To calculate an upper bound of response time using formula (1), one can use the maximal bounds of T_{RTT} and T_{Out} along with the minimal bound of T_{In} [12].

III. Experimental Evaluation vs. Analytic Calculus

A. Bench test overview

In this sub-section, we describe the platform we used to measure the response time. It is mainly constituted by three components (Fig. 6):

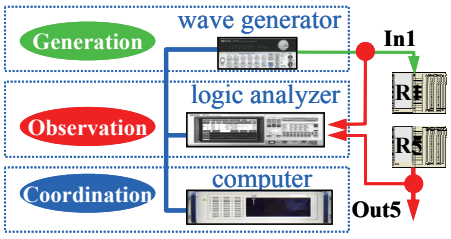

Fig. 6. Experimental platform composition.

i) A wave generator: its role is to generate a logic signal so as to excite input In1. In our test bench, we generate a periodic square signal whose period T_G is chosen much larger than the response time to avoid any ambiguity in measuring the response time (Fig. 7(b)). If it were not the case, one would measure a delay d that does not correspond to the real response time as in Fig. 7(a). Indeed, the first rising edge of Out5 may be due to the change in state of In1 during cycle #1 rather than cycle #2.

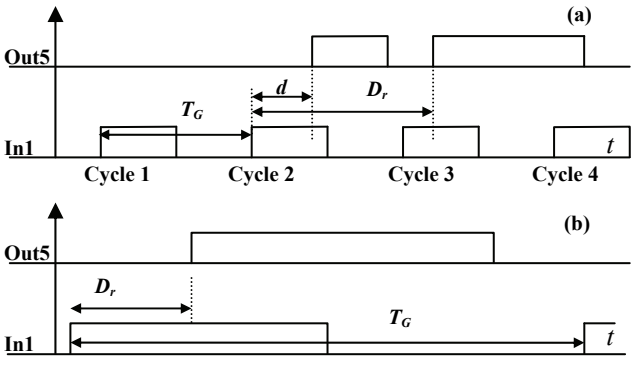

Fig. 7. Response time measurement: (a) possible misevaluation of response time D_r (b) Evaluation of D_r without ambiguity.

ii) A logic analyzer: this tool is similar to an oscilloscope but better dedicated to logic signals. It is used to capture simultaneously input In1 and output Out5 and store them in its memory. This analyzer has an interesting advantage; instead of storing the amplitude of a digitalized signal at every sampling period, it stores only the dates when the logic signal state changes. This results in a much bigger storage capacity.

iii) A computer: this computer is used for setting the parameters of both the wave generator and the logic analyzer. Also, it is used to process the acquired data into the analyzer memory and calculate the response times. All these operations were automated and executed via a program written in Python[1] code.

B. Example of response time measurement

In this sub-section we present an example of the NAS in Fig. 1 configuration and the corresponding response time measurement. PLC1 functions in cyclic mode and its Ethernet interface period is set up to 10ms. To generate parallel traffic, PLC2 polls module R5 every 5ms whereas the PC, that emulates a PLC, does it every 2ms. Both PLC2 and PC read 10 holding registers (memory words) in module R5. To emulate a PLC that communicates according to Modbus/TCP protocol in the PC, we utilized software called Modbus Poll[2].

So, about 10200 response times were evaluated for this fixed set of parameters. The histogram of Fig. 8 represents their distribution.

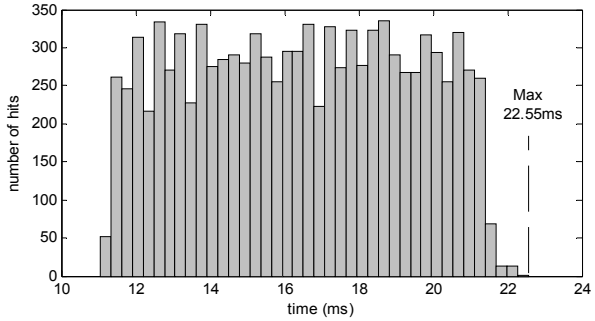

Fig. 8. Example of measured response times and their distribution.

As can be seen in Fig. 8, the maximal measured response time is 22.55ms. What about the evaluation using formula (1) ? To do that, we obviously need the values of the parameters involved in formula (1). They are either known from configuration or assessed theoretically or experimentally, according to the fixed settings of the example: $T_{ETH} = 10ms$, $T_{Exc} = 3ms$, $T_{Proc} = 0.7ms$, $T_{Out} = 2.1ms$, $T_{In} = 0.15ms$, $T_{filt} = 0.06ms$ and $q = 1$.

So, one can check that the maximal bound of response time using formula (1) with these values is equal to 22.71ms. This value corresponds to about 0.7% overestimation of the measured one. This theoretic evaluation is therefore quite satisfactory since it overestimates the real bound on one hand and is enough accurate (a gap of only 0.7%) on the other hand.

This result is indeed important but is only an example. It is not sufficient to validate the formula. So, several other configurations of the NAS were considered throughout the experimentation, as will be seen in the next section. In each case, about 10,200 response times were measured and processed. This single task takes about a half an hour. The total accumulated duration of data acquisition and processing was of about 40h for the whole investigation.

C. Formula vs. a series of measures

To validate objectively the formula, we carried out many measurements by changing the configuration of NAS. The followed procedure is as follows: for each measurement, we changed the traffic in the NAS in an appropriate way by targeting one parameter in formula (1) at a time. Then, we analyze if the impact of changing the traffic that way on response time is the one predicted by the formula.

One has to keep in mind that some of the parameters are directly tuned by the user (e.g T_{ETH}) whereas others are not. For instance, to change delay T_{Out}, we did not do it by just pushing a button. We rather changed the amount of traffic (throughput) in the network in an adequate pattern. More precisely, we loaded module R5 using different throughputs so as to increase or decrease delay T_{Out}. Indeed, more frequent requests sent to R5 means that the request from PLC1 is more likely to wait a longer time to be processed.

[1] See www.Python.org.

[2] Modbus Poll is a Modbus master simulator for testing and debugging slave devices. (See www.modbustools.com).

Fortunately, in targeting a delay in that way, the other delays are not strikingly affected since the delays experienced exclusively in the switches are not enough influent as we will see in sub-section 4. We obviously limited the throughputs to avoid saturating the buffers in the switches. One of the hypotheses we considered while we obtained our formula is indeed the absence of any packet loss.

So, the parameters of formula are handled as follows:

i) T_{ETH} : tuned directly by the user

ii) T_{Out} : by loading module R5

iii) T_{In}, T_{RTT} : by loading module R1

iv) None but the network: by loading R4 for instance

Remark III.1: Note that we do not mention the impact of term $(T_{Proc} + T_{filt})$. It is actually indirectly included in analyzing T_{Out}. So, we analyze the impact of all the delays except for T_{Exc}. This comes down indeed to changing the user program implemented in PLC1.

1) Effect of changing the scanning period T_{ETH}

The scanning periods of PLC2 and the PC remain unchanged (10ms) while the period of PLC1 varies from 10ms to 60ms. The results are depicted in Fig. 9.

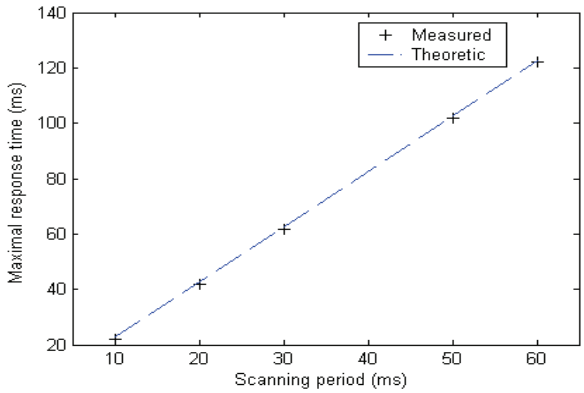

Fig. 9. Effect of varying the scanning period on the maximal response time.

As it can be seen in Fig. 9, the variation of the maximal response time is almost linear. The maximal bound is always around twice the scanning period. This fits well formula (1) for two reasons:

-i) delays T_{RTT} and T_{Exc} are much smaller than T_{ETH}. So, the higher is T_{ETH}, the smaller is the right hand-side of condition (C2). Number q remains therefore equal to one in all the cases. So, the maximal bound using (1) is almost equal to twice T_{ETH} since delays T_{Out} and T_{In} do not change considerably and they are much smaller than T_{ETH}.

- ii) the longer is the scanning period, the less crowded is the network and therefore the shorter is the delay T_{RTT}. So, the previous conclusion about (C2) is reinforced and consequently $q = 1$.

- *Consequence:* since networked automation systems are often not very crowded given the small amounts of exchanged data, one can suppose that: $T_{Out}(l+q) - T_{In}(l) < 2ms$. So, a simple linear approximation of (1) can be expressed as:

$$D_r \approx (2 \cdot T_{ETH} + 2 + T_{Proc} + T_{filt})ms \quad (2)$$

The curve corresponding to this linear approximation is depicted in Fig. 9 (dashed line). We can notice that the measures are quite close to this curve and the approximation is acceptable. However, one has to take care when using (2) and be sure that the condition (C2) is verified for $q = 1$ (by the way, check that $T_{Exc} \ll T_{ETH}$).

2) Effect of loading module R1

Loading R1 is achieved by polling R1 with requests at different rates (throughputs) using a traffic generator called Iperf [3]. Contrary to Modbus Poll, it enables to generate packets of variable size with a much wider range of throughputs. In our tests, we used 1470Bytes UDP packets (maximal) and varied the throughput from 0 to 2000Kbps. The impact on the max response time is depicted in Fig. 10.

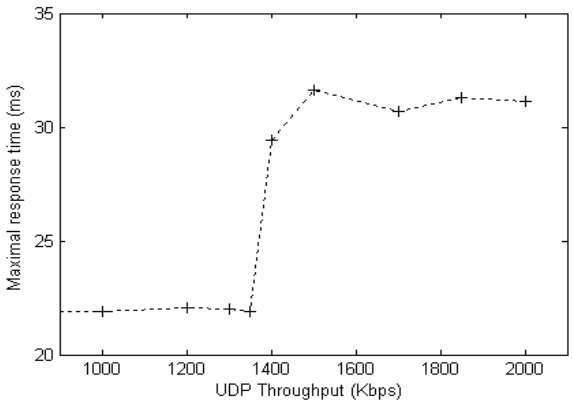

Fig. 10. Effect of loading module R1.

We can notice in Fig. 10 that the maximal response time remains almost unchanged around 22ms when changing the throughput from 0 Kbps to 1350Kbps. Then at 1400Kbps, the bound rises sharply to about 32ms. This phenomenon agrees with formula (1) and it can be explained as follows:

When module R1 is polled at higher rates, the worst delay T_{Out} does not change strikingly (R5 load is not affected). Also, the minimal delay T_{In} does not change too. However, the more loaded module R1 is, the worse the delay T_{RTT} is. According to Formula (1), the increase of delay T_{RTT} has no impact on the maximal response time as long as condition (C2) is respected with $q = 1$. So, the response time remains around twice T_{ETH}. This situation corresponds to throughputs from 0Kbps to 1350Kbps. However, if T_{RTT} goes past the threshold that keeps $q = 1$, then this number

[3] Iperf is a testing tool used for traffic generation in networks and for some performances measurements like jitter, packets loss, and throughput. It enables the user to generate TCP or UDP data streams along with setting many other parameters. (See http://iperf.fr)

changes suddenly from 1 to 2 (since it is an integer number). This threshold corresponds to throughput 1400Kbps. So, according to formula (1), the maximal response time changes abruptly too, from around twice T_{ETH} to three times T_{ETH} (around 30ms), exactly as it can be noticed in Fig. 10.

3) Effect of loading event destination R5

We increased the rate of polling module R5 and measured its impact on the max response time. The results are depicted in Fig. 11.

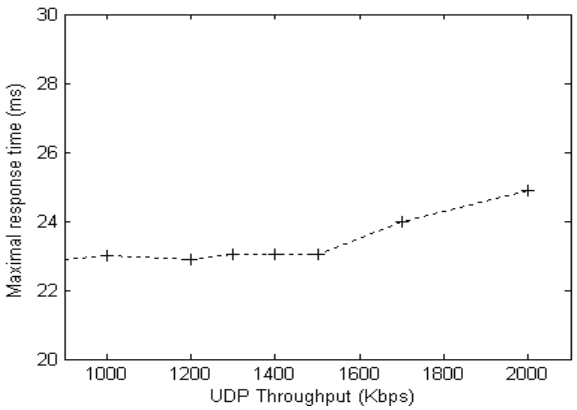

Fig. 11. Effect of loading R5.

As we can see in Fig. 11, the maximal response time varies much more smoothly when varying the polling rate of R5 than the polling rate of R1. This fits the prediction of formula (1) too. Indeed, in increasing the load on R5, the maximal bound of delay T_{Out} varies almost proportionally but delay T_{RTT} is not affected considerably. So, number q remains equal to 1 and therefore only term $T_{Out}(l+q)-T_{In}(l)$ varies in (1). So, formula (1) predicts a proportional variation too.

4) Effect of loading another module: neither R1 nor R5

The results are similar while loading another module, neither the event source nor the destination. The results of loading R4 for instance are shown in Fig. 12.

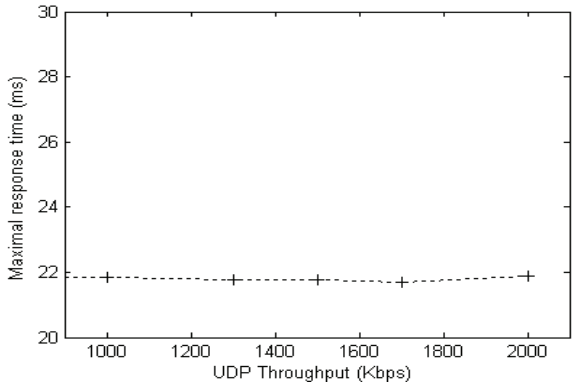

Fig. 12. Effect of loading module R4.

As it can be seen, almost no effect is noticed on the maximal the response time. This is also expected from formula (1). The slight variation can be explained by a slight change in term $T_{Out}(l+q)-T_{In}(l)$ due to the variation of amount of traffic in the switches. Given the relatively limited traffic with regard to the speed of the switches, no striking effect was expected.

IV. Conclusion

In this paper, we investigated the reactivity of networked automation systems. We first recalled a theoretic formula that provides the maximal bound of response time and subsequently checked its validity using experimentation. Several cases have been considered along with interesting results deduced. So, this investigation can be considered as a first step towards the use of this formula in real applications with enough confidence. For further studies, it would be interesting to look for the possibilities of extending the applicability of these results to other NAS, working according to other protocols (e.g. producer/consumer).

References

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

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

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

[4] K. C. Lee, S. Lee, and M. H. Lee, "Worst case communication delay of real time industrial switched Ethernet with multiple levels", *IEEE Trans. on Industrial Electronics,* 53(5), pp. 1669-1676, 2006.

[5] 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.

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

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

[8] G. Marsal, B. Denis, J.-M. Faure, and G. Frey, "Evaluation of response time in Ethernet-based automation systems", In *Proc. of ETFA06,* Prague, Czech Republic, pp. 380-387, 2006.

[9] 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 ETFA07,* Patras, Greece, pp. 532-539, 2007.

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

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

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

[13] B. Addad, S. Amari, and J. J Lesage, "Client/server networked automation systems reactivity: deterministic and probabilistic analysis", IEEE Trans. on Automation Science and Engineering, DOI**:** 10.1109/TIE.2010.2098358, 2011.

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