Response time evaluation in Ethernet-based automation architectures

B. Addad, S. Amari
Lurpa ENS-Cachan, 61 av. Du Président Wilson, 94235 Cachan Cedex, France
{boussad.addad, said.amari}@lurpa.ens-cachan.fr

Abstract

In this paper, a new method to evaluate the response time in switched Ethernet automation architectures is developed. It is based on the modeling of the whole system in the form of timed event graphs and on the resulting state representation in Max-Plus algebra. After the resolution of the state equations and their fusion according to the working of the system, we obtained an algorithm giving the reaction delay of the architecture. With deep analysis of these equations we got to analytical formulas for direct calculus of the response time as a function of the features of the architecture. The minimal and maximal bounds of response time are also calculated. To check the validity of the results, simulations of the algorithm and experimental measurements on a laboratory platform are used. Finally, a comparison with results obtained using a classical method, shows the interest and the effectiveness of this new approach.

Keywords: Ethernet-based automation architectures, response time evaluation, Max-Plus algebra, timed event graphs (TEGs).

1. INTRODUCTION

Ethernet is more and more used as a fieldbus in automation architectures. Indeed, it offers many advantages either of interoperability or performance. To benefit from these advantages, many suppliers and researchers worked on this alternative. Unfortunately, they define specific protocols not always compatible with the standard networks. In this study on the contrary, we are interested in switched Ethernet networks that use open standard protocols and particularly the client server Modbus TCP/IP protocol [1]. However, the introduction of new elements like switches in the network makes it difficult to evaluate the architecture performances. Because of resource sharing by many parallel flows, it is not obvious to predict the delay a message suffers in a node of the network. Different investigations are made to assess these delays by the use of network calculus [2] and simulation [3]. However, like the majority of studies, these works focus on the end-to-end delay of the network and ignore both the PLCs (programmable logic controllers) and the RIOMs (remote input output modules). As a matter of fact, the modules of the PLC are not synchronized and the RIOMs are shared by many applications. This leads to considerable delays that have to be taken into account. To our knowledge, the studies that consider the whole architecture are so far, often based on simulation or experimental measurements [4]. For instance, a method based on exhaustive state space exploration and model-checking [5], [6], [7], enables to know if a bound of time is reached or not. Its disadvantage is that it does not provide the distribution of response delays and also the explosion of the number of states in the model. Another method relies upon a model in form of hierarchical timed colored Petri nets and its simulation with CPNTools [8], [9]. This last provides good evaluations of the delays but is onerous of time and the critical states are not surely scanned.

In our study, we propose a formal method to assess the time delay by the use of Max-Plus algebra equations representing the dynamics of the whole system. Our work is organized as follows. First in section 2.1, the automation architecture is described. Then, a classical evaluation method based on extreme assumptions is presented. It is used in a comparison to show the improvements that our results bring. After in section 3, we remind some fundamentals about the Max-Plus algebra and the timed event graphs (TEGs). Then, we move on to the modeling of the system using these tools in section 4. Different configurations are considered. We begin by a first case in section 4.1, with one PLC and one RIOM before to consider a more complex architecture in section 4.2. After the resolution of the equations in section 4.1.2, we obtained a calculus algorithm that scans all the states of the system if the length of simulation is over a critical period précised in section 4.1.3. After in section 4.1.4, by following the same principle and a deep analysis of the solutions, we got to analytical formulas giving the response time of the architecture. Then in section 5, simulations of the algorithm and experimental measurements on a laboratory platform PRISME [10] are used to check the validity of the analytical results we developed before. By comparison with the classical method results, it is shown how our results are interesting.

2. AUTOMATION ARCHITECTURE RESPONSE TIME

2.1 Switched Ethernet-based automation architectures

In this study, the messages are exchanged in the automation architecture according to Modbus TCP/IP client server protocol. The communication module of the PLC is the client and the RIOMs are the servers. The considered PLC is made up of two modules, the CPU (central processing unit) that executes the user program and the Ethernet board that sends requests (combined requests: read and write data) to the RIOMs. They operate cyclically but are not synchronized, the fact that makes it difficult to set the values of their periods so as to reach the desired temporal performances. The CPU accomplishes periodically the tasks: reading inputs, execution of user program and up date of the outputs. The only constraint is the respect of the cycle period. Regardless of the CPU, the Ethernet board sends requests to RIOMs and waits until the cycle time elapses. If all answers arrived, it begins a new cycle, else it waits until all answers come back. The PLC and the RIOMs of this study are linked via store-and-forward Ethernet switches. They are without quality of service and check only the validity of a frame before to forward it. Finally, the frames loss is not considered and no time-out has to be taken into account.

2.2 Classical calculus of the response time bounds

The response time is the delay between the occurrence of an event on the controlled process (plant) and the occurrence of the reaction event issued from the controller, on the plant. This delay results from both the behaviour of the PLC and the communication protocol. The messages moving in the system suffer a delay at each visited device of the architecture (FIGURE 1).

FIGURE 1: Response time decomposition.

At the top of FIGURE 1, we find the plant which is the source of the event and the destination of the consequence. On the figure we assigned a number to each important transition of the information in the system. The arrow (1) represents the occurrence of an event from the plant. The delay d_f is due to the filtering of the data coming from the sensor. This information is taken into account once the next request arrives. It is used in processing during T_{IO} before to return the corresponding response (arrow 2). The response suffers a network delay d_{NETb} before to arrive to the Ethernet Board. Once in the input buffer of the Ethernet board, it is copied to the shared memory during a time T_{CPY} . At the next cycle of the CPU, it is read and used in calculus (during T_{CLC}) to update the output (arrow 5). It is taken into account at the next beginning of scanning cycle (arrow 6). Again, this event consequence suffers of a network delay d_{NETf} before it gets to the destination (RIOM'). After a processing time $T_{IO'}$ in the destination remote module RIOM', the consequence gets to the plant (arrow 8). During this trip, the delays between two transitions are not only the grey lines mentioned on the figure but also the gaps. They can be distinguished according to their origins [9] as follows:

- Intrinsic processing delay.
- Waiting for resource availability (load): a shared resource is free only after processing all waiting frames.
- Waiting for synchronization.

Thus, the delay d_{RIOM} that the data suffer from their generation from the plant to their forwarding to the network is:

$$d_{RIOM} = d_f + T_{IO} + d_{load}^1 + d_{asyn}^1,$$

where d_{load}^1 is the delay due to concurrent requests that share the module RIOM and d_{asyn}^1 is the time to wait for the arrival of the corresponding request to the module.

The delay between the arrival of the response to the Ethernet board (arrow 3) and its reading to be used in calculus in the CPU (arrow 4) is:

$$d_{SCN} = T_{CPY} + d_{load}^2 + d_{asyn}^2,$$

where d_{load}^2 is the time to process waiting frames and d_{asyn}^2 the delay before the beginning of the next CPU cycle.

Once in the CPU, the information is affected by a delay:

$$d_{CPU} = T_{CLC} + d_{load}^3 + d_{asyn}^3,$$

where $d_{load}^3 + d_{asyn}^3$ the time before the result is sent during the next scanning cycle.

When a request is sent, each visited switch of the network affects it with a delay. The total delay is written:

$$d_{NETf} = T_{NETf} + d_{load}^4 + d_{asyn}^4.$$

Also when the response comes back, the network causes it a delay given:

$$d_{NETb} = T_{NETb} + d_{load}^5 + d_{asyn}^5,$$

with $d_{load}^5 + d_{asyn}^5$ (resp. $d_{load}^4 + d_{asyn}^4$) the time to wait for the switch availability and T_{NETf} (resp. T_{NETb}) the intrinsic delay to process a frame in the switches.

Finally, when the result arrives to the destination, it is affected with the delay:

$$d_{RIOM'} = T_{IO'} + d_{load}^6$$

where d_{load}^6 is the delay due to concurrent requests that share the module RIOM'.

The recapitulation of the previous decomposition, leads to the response time as the sum of all delays:

$$D_r = d_{RIOM} + d_{NETb} + d_{SCN} + d_{CPU} + d_{NETf} + d_{RIOM'}. \quad (1)$$

In the classical method, the dependences between the components of the architecture are ignored. The bounds of time are obtained with extreme considerations. So, the response time is maximal (resp. minimal) if the parallel processes are maximally non synchronized (resp. totally synchronized) and loads are maximal (resp. minimal) in all the previous calculated delays.

In the case of our work, we consider an architecture with one PLC and N remote modules RIOMs. All the modules are scanned in an invariant order by only this PLC. Using the previous method, the bounds of response time are:

$$\begin{cases} D_{MIN} = T_{IO} + d_{NETbmin} + T_{CPU} + d_{NETfmin} + T_{IO'} + d_f \\ D_{MAX} = 2T_{SCN} + T_{IO} + d_{NETbmax} + T_{CPU} + T_{CLC} + d_{NETfmax} + T_{IO'} + d_f \end{cases} \quad (2)$$

Where $d_{NETbmin}$ and $d_{NETfmin}$ (resp. $d_{NETbmax}$ and $d_{NETfmax}$) the extreme network induced delays.

We will see later in this study that the results (2) are not realistic and are far from the real bounds of response time. Indeed, even the fact that the processes are not synchronized, there is always a dependence that constrains the system to operate accordingly. This is what is proven in our work by the use of a formal method based on Max-Plus algebra and timed event graphs.

3. FUNDAMENTALS OF MAX-PLUS ALGEBRA AND TIMED EVENT GRAPHS

3.1 Max-Plus algebra

A set D endowed with an internal law $\oplus$ is a monoid if $\oplus$ is associative and has a neutral element ε , $\forall a \in D, a \oplus \varepsilon = \varepsilon \oplus a = a$. A semiring is a commutative monoid with another internal law $\dot{\vee}$ that is distributive on $\oplus$, has a neutral element e and admits ε as an absorbent element, $\forall a \in D, a \otimes \varepsilon = \varepsilon \otimes a = \varepsilon$. A dioid is a semiring with idempotent addition, " $a \dot{\wedge} D, a \dot{\wedge} a = a$ ". It is commutative if $\dot{\vee}$ is commutative.

In the Max-Plus formalism, the structure $(\bar{i}_{max}, \max, +)$ where $\bar{i}_{max} = i \in \mathbb{Z} \cup \{-\infty\}$, is considered. It is a commutative dioid with usual maximum and addition operations for respectively $\oplus$ and $\dot{\vee}$. The neutral element of the maximum is $\varepsilon = -\infty$ and of the addition is $e = 0$.

This algebra is extended to vectors and matrix. Then, for $n \in \mathbb{N}$ and $v, w \in \bar{i}_{max}^n$, the vector $v \dot{\wedge} w$ has the components $v_i \oplus w_i = \max(v_i, w_i)$ for $i = 1$ to n . With $p, q \in \mathbb{N}$, $A \in \bar{i}_{max}^{p \times n}$ and $B \in \bar{i}_{max}^{n' \times q}$, the matrix multiplication in

$$\bar{i}_{max} \text{ is defined by } A \dot{\wedge} B \text{ or } A.B \text{ where: } (A.B)_{ij} = \dot{\bigvee}_{k=1}^n (A_{ik} \dot{\wedge} B_{kj}) = \max_k (A_{ik} + B_{kj}).$$

The Kleene star of a square matrix $M \in \bar{\square}_{\max}^{n \times n}$ is $M^* = \bigoplus_{i=0}^{\infty} M^i$ where M^0 is the unit matrix with only ε on the diagonal and ε elsewhere. Then, for $v \in \bar{\square}_{\max}^n$, $x = M^* \cdot v$ is the minimal solution of the inequality $x \leq M \cdot x \oplus v$ and the equality $x = M \cdot x \oplus v$ [11].

3.2 Timed event graphs

An event graph is an ordinary Petri net where all the places have at most one upstream and one downstream transition. An event graph is timed if the transitions or the places are affected with delays. We note n the number of transitions with at least one place upstream and m the number of source transitions t_i . The only place relying the transitions t_j and t_i is noted p_{ij} and its delay τ_{ij} . In the modeling of our study (section 4), we assign simply the delay τ_i to the place p_i .

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

Example (FIGURE 2):

FIGURE 2: Example of TEG.

The timed graph of the example leads to the equation:

$$\theta_3(k) = \max(\tau_{31} + u_1(k - m_{31}), \tau_{32} + u_2(k - m_{32})),$$

which is a linear equation in Max-Plus algebra:

$$\theta_3(k) = (\tau_{31} \otimes u_1(k - m_{31})) \oplus (\tau_{32} \otimes u_2(k - m_{32})).$$

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

$$\theta(k) = \bigoplus_{m>0} (A_0^* \cdot A_m \cdot \theta(k-m) \oplus A_0^* \cdot B_m \cdot u(k-m)), \quad (3)$$

where the vectors $\theta(k)$ and $u(k)$ components are the firing times of the n and m transitions of system for the k^{th} time. The matrix A_m belongs to the dioid $\bar{\square}_{\max}^{n \times n}$ with the element $A_{m,ij}$ representing the delay τ_{ij} associated to the place p_{ij} if it exists and the neutral element ε else. Similarly for $B_m \in \bar{\square}_{\max}^{n \times m}$, it corresponds to the delays of the places downstream of the source transitions. A_0^* is the Kleene star of A_0 .

In an analogous manner as in usual linear systems, this form can be brought to state space representation by replacing all the places with markings $m_{ij} > 1$ by m_{ij} other places with $(m_{ij} - 1)$ intermediate transitions. Hence, we obtain an extended system with a state vector $x(k)$ that belongs to $\bar{\square}_{\max}^N$, where $N = n + n'$ and n' the number of transitions added. The new system is described by the equation:

$$x(k) = \bar{A}_0 \cdot x(k) \oplus \bar{A}_1 \cdot x(k-1) \oplus \bar{B} \cdot u(k), \quad (4)$$

Can be rewritten:

$$x(k) = A \cdot x(k-1) \oplus B \cdot u(k), \quad (5)$$

where $A = \bar{A}_0^* \cdot \bar{A}_1$ and $B = \bar{A}_0^* \cdot \bar{B}$. The last formulations permit to point out that the behaviour of a timed event graph is determinist, depending only on the source transitions and the initial conditions [12]. This dependence can be clarified by the following equation:

$$x(k) = A^{k-1} \cdot x(1) \oplus \left[\bigoplus_{i=0}^{k-2} A^i \cdot u(k-i) \right]. \quad (6)$$

4. AUTOMATION ARCHITECTURE MODELING AND RESPONSE TIME EVALUATION

In this paper, two main cases are considered. The first to determine the response time of architectures with one PLC and one RIOM as in FIGURE 3 and a more complex system is studied after in the other case.

4.1 Case 1: one PLC and one RIOM

Two cases are to be considered in control architectures. It may be, for instance, the delay between the detection of danger and the triggering of an alarm. In this case the evaluation of the maximal bound of reaction time is of top priority. However, if the control concerns for example a robot arm speed, it is the distribution rather than the bounds of the response time that is more important to assess. In our work, we consider the general case. The response delay is calculated using only the state of the system and the time of occurrence of an event, regardless to its consequences.

FIGURE 3: Automation architecture (case 1).

4.1.1 TEG model and Max-Plus equations

According to the previous description of the architecture in section 2.1, we got to the model of FIGURE 4. It comprises two independent TEGs: one at the left to model the CPU and another at the right for the rest of the system. As a matter of fact, the model is abstract and represents the state of the two parts of the architecture independently by making abstraction of the meaning of the tokens in circulation in the TEGs. Therefore, the link between the CPU and the Ethernet board is hidden. By the fusion of the equations of the two TEGs and introduction of secondary variables, we can follow the frames along their route in the whole architecture to get the time delay during each step.

FIGURE 4: Automation architecture model with TEGs (case 1).

The places p_1, p_2, p_3 , and p_4 with delays τ_1, τ_2, τ_3 and τ_4 of the CPU, model respectively the phases of waiting, user program execution during T_{CLC} (reading and writing included), CPU busy and finally CPU idle. The place p_3 is not necessary. It is represented only to show more clearly the end of a CPU cycle by firing the transition t_3 . It is also the case of p_{15} . So, we easily note the periodical operating of the CPU with cycle period τ_3 equal to T_{CPU} . Similarly, τ_{14} models the scanning period T_{SCN} of the Ethernet board and a token in the place p_{14} means it is busy during at least this period. The sending of a request starts by firing the transition t_4 and finishes by firing t_5 , τ_6 or T_{EM} being the required time to send a frame. A token in p_{13} means the request is sent and the Ethernet board is waiting for the answer. The places p_7 and p_{12} model the network (switch) delays imposed to the sent requests and the returned answers. The separation of these places is made possible since the lines are full duplex and no situation of conflicts or collision is possible. To avoid overcrowding the model, τ_{12} includes also the necessary time (T_{CPY}) to copy the response from the input buffer of the Ethernet board to the shared memory with the CPU (indeed T_{CPY} is by far smaller than the other delays of the architecture). The places p_8, p_9, p_{10} , and p_{11} represent the RIOM. This module stays waiting in p_9 until a request arrives to its input buffer p_8 . By firing t_7 , the processing

starts and goes on for a time τ_{10} equal to $T_{I/O}$. At the end, it puts the answer in its output buffer p_{11} before forwarding it to the network.

The grey arrows represent the source (data coming from the sensor) and output (data toward the plant). They are not considered at this stage since the system is not constrained and data are available at the output of the sensor as long as it is functional.

By applying the method of the section 3.2 to the model of the architecture with the initial conditions on FIGURE 4, we got to the Max-Plus equations:

$$\begin{cases} \theta_1(k) = (\theta_2(k-1) \otimes \tau_1) \oplus (\theta_3(k-1) \otimes \tau_4) \\ \theta_2(k) = \theta_1(k) \otimes \tau_2 \\ \theta_3(k) = \theta_1(k) \otimes \tau_3 \end{cases} \quad (7)$$

$$\begin{cases} \theta_4(l) = (\theta_{10}(l-1) \otimes \tau_5) \oplus (\theta_{11}(l-1) \otimes \tau_{15}) \\ \theta_5(l) = \theta_4(l) \otimes \tau_6 \\ \theta_6(l) = \theta_5(l) \otimes \tau_7 \\ \theta_7(l) = (\theta_6(l) \otimes \tau_8) \oplus (\theta_8(l-1) \otimes \tau_9) \\ \theta_8(l) = \theta_7(l) \otimes \tau_{10} \\ \theta_9(l) = \theta_8(l) \otimes \tau_{11} \\ \theta_{10}(l) = (\theta_5(l) \otimes \tau_{13}) \oplus (\theta_9(l) \otimes \tau_{12}) \\ \theta_{11}(l) = \theta_4(l) \otimes \tau_{14} \end{cases} \quad (8)$$

The systems (7) and (8) are linear in Max-Plus algebra and can be rewritten in the form (5). We assigned them different indexes (k and l) to mean that they are not synchronized, exactly like the CPU and the Ethernet board. It is the additional and the main difficulty of this study.

4.1.2 Equations resolution and simulation algorithm

The resolution of the equations in $\bar{\square}_{\max}$ leads to:

$$\begin{cases} \theta_1(k) = (k-1) \cdot T_{CPU} \\ \theta_2(k) = (k-1) \cdot T_{CPU} + T_{CLC} \\ \theta_3(k) = k \cdot T_{CPU} \end{cases} \quad (9)$$

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

In these solutions, only the equations representing the following events interest us:

- Reading and beginning of processing in the CPU (θ_1).
- End of processing in the CPU and output update (θ_2).
- Beginning of scanning and sending a request (θ_4).
- Reception of an answer in the shared memory (θ_{10}).

Indeed, they are the events that link the CPU and the Ethernet board. When an answer arrives (θ_{10}), it is taken into account in the next beginning of the CPU cycle (θ_1). It is read and used in the calculus in the CPU. Once the processing finishes, the result is written in the memory of the Ethernet board (θ_2). It is taken into account at the next beginning of the scanning cycle (θ_4) and sent to the RIOM.

Let us put the time to wait for the reception of an answer: $T_r = \tau_6 + \tau_7 + \tau_8 + \tau_{10} + \tau_{11} + \tau_{12}$, then we obtain the following equations:

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

$$\begin{cases} \theta_4(l) = (l-1) \cdot T_{SCN} \\ \theta_{10}(l) = (l-1) \cdot T_{SCN} + T_r \end{cases} \quad (12)$$

At the l^{th} scanning cycle, the answer is received at time $\theta_{10}(l)$ and to be taken in account, it has to wait for m^{th} (but the immediate next one with respect to $\theta_{10}(l)$) beginning of the processing cycle of the CPU. The condition to check is: $m = \arg \min_{i/\theta_1(i) > \theta_{10}(l)} (\theta_1(i) - \theta_{10}(l))$ where "arg" is the converse function which gives the index that minimizes

the positive expression: $(\theta_1(i) - \theta_{10}(l))$. Hence, we introduce a new variable $\hat{\theta}_1$ where: $\hat{\theta}_1(l) = \theta_2(m) = \theta_1(m) + T_{CLC}$.

As soon as the result of calculus is obtained, it is written in the memory of the Ethernet board. At the next scanning cycle, the n^{th} one, it is encapsulated in the request frame and sent to the RIOM. Again, we add another new variable $\hat{\theta}_2$ with the conditions: $\hat{\theta}_2(l) = \theta_4(n)$ and $n = \arg \min_{j/\theta_4(j) > \hat{\theta}_1(l)} (\theta_4(j) - \hat{\theta}_1(l))$. The date of arrival of the event

consequence to the controlled process is therefore: $\hat{\theta}_f(l) = \theta_8(n)$. So, for the p^{th} event generated at time $v(p)$ and taken into account at the l^{th} scanning cycle, the associated response time of the architecture is given by:

$$D_r = \hat{\theta}_f(l) - v(p) \quad (13)$$

The delay is minimal if the data coming from the sensor are used in processing in the RIOM immediately after they are generated. So the minimal delay (not global minimum) for the p^{th} event is:

$$D_{MIN} = \hat{\theta}_f(l) - \theta_7(l) + d_f, \quad (14)$$

where d_f is the delay due to the data filtering in the sensor.

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

$$D_{MAX} = \hat{\theta}_f(l) - \theta_7(l-1) + d_f \quad (15)$$

This delay is always valid and it is the case if the frequency of update of the sensor output is smaller than the frequency of scanning. Else, some events may be erased and not used in any processing. This case is not considered in study even the formula (13) is not limitative. Indeed, it is not interesting to be considered since no response time is assigned to the erased events and it is only loss of processing time.

Thus, we have an algorithm for the evaluation of the response time of the architecture to any occurring event. It is fast and easily implemented. The delays and the features of the components of the system are introduced as parameters. So, it is flexible for use for different configurations of the architecture. Simulations of this algorithm are used to check the validity of the formulas, obtained later in this study.

4.1.3 Critical period of simulation

In the architecture, we have two cyclical applications but not synchronized. Despite of this constraint, the whole system remains cyclical with a period T_{CR} that verifies:

$$T_{CR} = k_1 \cdot T_{SCN} = k_2 \cdot T_{CPU} \text{ where } k_1, k_2 \in \mathbb{N} \text{ always exist.}$$

Proof: Let us put: $T_{SCN} = r \cdot T_{CPU}$ with $r = [r] + \varepsilon$, $[r]$ being the entire part of r and ε fractional part.

Since we can write $\varepsilon = n_1/n_2$, it is enough to take: $k_1 = n_2$ and $k_2 = n_2 \cdot [r] + n_1$ to get to:

$$T_{CR} = (n_2 \cdot [r] + n_1) \cdot T_{CPU} \quad (16)$$

This period is minimal if n_1 and n_2 are chosen prime numbers (of course always possible).

We can conclude that the algorithm is formal and all possible states of the system are scanned if the simulation length is over the previous critical period T_{CR} . So, the global maximum and minimum are the maximum and minimum of those calculated using the simulation of this algorithm during at least T_{CR} .

4.1.4 Analytical calculus of response time

We use the results (11), (12) and the principle of the algorithm. Let us put: $T_r = \alpha \cdot T_{CPU} + \tau_r$, $T_{CLC} = \beta \cdot T_{CPU}$ where $\beta < 1$. α and τ_r are respectively the ratio and the remainder of euclidean division of T_r by T_{CPU} .

For calculus complexity proven after, we begin with the case $r \in \mathbb{N}$ and generalize later (recall: $T_{SCN} = r \cdot T_{CPU}$).

- Case: $r \in \square$ ($\varepsilon = 0$)

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

$$\theta_{10}(l) = (l-1) \cdot r \cdot T_{CPU} + \alpha \cdot T_{CPU} + \tau_r \quad (17)$$

$$\text{For } k-1 = (l-1) \cdot r + \alpha + 1 \text{ we obtain: } \theta_1(k) = \theta_{10}(l) + T_{CPU} - \tau_r \quad (18)$$

Since $0 < T_{CPU} - \tau_r < T_{CPU}$ then:

$$\hat{\theta}_1(l) = \theta_1(k) + T_{CLC} = [1 + \alpha + \beta + (l-1) \cdot r] \cdot T_{CPU} \quad (19)$$

We have also: $\theta_4(n) = (n-1) \cdot r \cdot T_{CPU}$ and for $n = l+1$ then:

$$\theta_4(n) = l \cdot r \cdot T_{CPU}, \quad (20)$$

$$\text{Rewritten: } \theta_4(n) = \hat{\theta}_1(k) + [r - (1 + \alpha + \beta)] \cdot T_{CPU} \quad (21)$$

Thus on the condition C_1 : $r > (1 + \alpha + \beta)$, (it is the optimal case), $\hat{\theta}_2(l) = \theta_4(n) = \theta_4(l+1)$. This implies:

$$\begin{cases} \hat{\theta}_f(l) = \theta_8(l+1) \\ D_{MIN} = \theta_8(l+1) - \theta_7(l) + d_f \\ D_{MAX} = \theta_8(l+1) - \theta_7(l-1) + d_f \end{cases} \quad (22)$$

Finally:

$$\begin{cases} D_{MIN} = T_{SCN} + T_{I/O} + d_f \\ D_{MAX} = 2T_{SCN} + T_{I/O} + d_f \\ D_r(p) = \theta_8(l+1) - v(p) \end{cases} \quad (23)$$

In practice the condition C_1 is often respected and the results (23) are valid because the scanning period is by far longer than the period of the CPU. The results are very interesting since the calculated bounds in (23) are constant and therefore are global minimum and maximum. Indeed, the network and the RIOM processing delays are supposed constant. It is not the same in architectures where the resources are shared with acyclic traffic and the imposed delays variable. This is so far out of our study.

The results (23) are generalized for a condition: $r \cdot q > (1 + \alpha + \beta) > r \cdot (q-1)$ and we found:

$$\begin{cases} D_{MIN} = q \cdot T_{SCN} + T_{I/O} + d_f \\ D_{MAX} = (q+1) \cdot T_{SCN} + T_{I/O} + d_f \\ D_r(p) = \theta_8(l+q) - v(p) \end{cases}, \quad (24)$$

- Case : $r \in \square^+$ ($\varepsilon \neq 0$)

Let us put: $\tau_r = \gamma \cdot T_{CPU}$ (where obviously $\gamma < 1$).

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

$$\theta_{10}(l) = (l-1) \cdot r \cdot T_{CPU} + (\alpha + \gamma) \cdot T_{CPU} \quad (25)$$

Let us take $i \in \square$ where: $i \leq \gamma + \varepsilon \cdot (l-1) < (i+1)$ (*)

For $k-1 = (l-1) \cdot [r] + \alpha + 1 + i$ we get:

$$\theta_1(k) = \theta_{10}(l) + [i + 1 - (\gamma + \varepsilon \cdot (l-1))] \cdot T_{CPU} \quad (26)$$

and since in (*) $i \leq \gamma + \varepsilon \cdot (l-1) < (i+1)$, then:

$$\hat{\theta}_1(l) = \theta_{10}(l) + [i + 1 + \beta - (\gamma + \varepsilon \cdot (l-1))] \cdot T_{CPU}. \quad (27)$$

We have also:

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

$$\theta_4(n) = \hat{\theta}_1(k) + [r - [\alpha + \beta + i + 1 - \varepsilon \cdot (l-1)]] \cdot T_{CPU}. \quad (28)$$

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

Let us put: $\Gamma_{l,i} = i + 1 - \varepsilon \cdot (l-1)$ with $\gamma < \Gamma_{l,i} \leq 1 + \gamma$

$$\Gamma_{MIN} = \min_{i \in \square, l \in \square} (\Gamma_{l,i})$$

$$\Gamma_{MAX} = \max_{i \in \square, l \in \square} (\Gamma_{l,i})$$

Thus, on the condition C_2 : $r > (\alpha + \beta + \Gamma_{MAX})$, we obtain :

$\hat{\theta}_2(l) = \theta_4(n) = \theta_4(l+1)$ i.e. $\hat{\theta}_f(l) = \theta_8(l+1)$. As we note, this leads to the same results as in (23). But if this condition is not respected, the bounds of time would depend on $\Gamma_{l,i}$. So, the conditions of bounds calculus are global (absolute) and the delay condition relative to the p^{th} event is local.

On the following global and local conditions:

$$r \cdot q_1 > (\Gamma_{MIN} + \alpha + \beta) > r \cdot (q_1 - 1)$$

$$r \cdot q_2 > (\Gamma_{MAX} + \alpha + \beta) > r \cdot (q_2 - 1),$$

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

the global bounds of time and local delay are given by:

$$\begin{cases} D_{MIN} = q_1 \cdot T_{SCN} + T_{I/O} + d_f \\ D_{MAX} = (q_2 + 1) \cdot T_{SCN} + T_{I/O} + d_f \\ D_r(p) = \theta_8(l + q_3) - v(p) \end{cases} \quad (29)$$

In this general case, we point out that the optimality condition $r > (\alpha + \beta + \Gamma_{MAX})$ is more restrictive than in case $r \in \square$. Indeed, for $\varepsilon = n_1/n_2$, it is enough to take $(l-1) = n_2$ and $i = n_1$ to obtain $\Gamma_{l,i} = 1$. This implies $\Gamma_{MAX} \geq 1$ and the condition C_2 more restrictive than C_1 .

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

4.2 Case 2: One PLC and N RIOMs

The modeling of the PLC does not change and only the number of RIOMs that is different. The requests are sent from the Ethernet board in an invariant order and the switch with FIFO policy is without quality of service. In this more general case, the TEG of the CPU remains the same but the other one becomes more complex. Indeed, we introduce other equations to model the FIFO policy and solve the problem of sharing resources: the switch and the Ethernet board. The scanned RIOMs are affected with indexes according to the order of sending the request relative to each one. We associate the index i to the RIOM receiving the i^{th} request from the Ethernet board. Particularly, N_S and N_D are the indexes assigned to respectively the event source (S) and destination (D) of its consequence. On FIGURE 5 for example, $N_S = 4$ and $N_D = 5$.

With similar analysis and same notations as in the first case with one RIOM, the following results are obtained:

$$\begin{cases} D_{MIN} = q_1 \cdot T_{SCN} + (N_D - N_S) \cdot T_{EM} + \tau_7^D - \tau_7^S + T_{I/O}^D + d_f \\ D_{MAX} = (q_2 + 1) \cdot T_{SCN} + (N_D - N_S) \cdot T_{EM} + \tau_7^D - \tau_7^S + T_{I/O}^D + d_f \end{cases} \quad (30)$$

where: $r \cdot q_1 > \Gamma_{MIN} + \alpha + \beta > r \cdot (q_1 - 1)$

$$r \cdot q_2 > \Gamma_{MAX} + \alpha + \beta > r \cdot (q_2 - 1).$$

$$T_r = \max(N \cdot T_{EM}, N_S \cdot T_{EM} + \tau_7^S + \tau_{12}^S + T_{I/O}^S), \quad (31)$$

$T_r = (\alpha + \gamma) \cdot T_{CPU}$, is the necessary time to wait for the reception of the answer from the event source.

$T_{I/O}^S$ and $T_{I/O}^D$ are processing times of RIOMs (S) and (D).

τ_7^S , τ_{12}^S and τ_7^D are the network delays imposed respectively to the request going to the event source, the response coming from the event source and the request going to the event consequence destination. For a given architecture with RIOMs with constant processing times, these delays are invariant and we have to calculate them only once. Indeed, the scanning is cyclical and the frames (either the requests or the answers) always arrive to the switch or the Ethernet board in the same order.

FIGURE 5: Automation architecture (case 2).

The results are very interesting and to get small response time, we should assign great index to the source and small one to the destination: the order of RIOMs is important. However, we have to keep in mind that the condition of calculus of the delay depends on T_r or α and we should decrease N_S (see (31)). So, the optimal case is got by increasing N_S and stop just before the condition changes (before q_2 changes).

5. VALIDATION OF THE METHOD

To check the validity of the model and the results developed previously, we consider two configurations of architectures (FIGURE 3 and FIGURE 5). We compare the results of the algorithm and the formulas with experimental measurements taken on the patented laboratory platform [10].

In the second configuration, we are interested in the causality delay between an event generated on the input of the RIOM R4 and its consequence on the output of the RIOM R5. The histograms of FIGURE 6 represent a series of 10,000 measurements and simulations of the algorithm for this configuration. This architecture is more general than those of the study (two switches). This is made to show the possibility to extend the results to more complex systems. The CPU period is set up to 5 ms and scanning to 10 ms with a jitter of 15%. The jitter is considered in the algorithm and the formulas. We used the maximal (resp. minimal) scanning period to calculate the maximal (resp. minimal) bound of response time. We obtained the results of TABLE 1.

		Response delays in ms		
		Min	Max	Mean
Case 1	Measures	10.40	21.90	16.10
	Simulation	10.06	22.24	15.82
	Formulas	10.06	22.24	/
	Classical	4.91	31.39	/
Case 2	Measures	10.65	22.25	16.40
	Simulation	10.31	22.49	16.07
	Formulas	10.31	22.49	/
	Classical	5.91	32.44	/

TABLE 1: Results of response time calculus

FIGURE 6: Histograms of simulated and measured delays (case 2).

In both cases, we already can conclude about the validity of the formulas since the maximal delays are greater than all measured delays and the minimal delays are smaller than all. As expected, the results of the simulation and formulas are exactly the same in all cases. Indeed, they are based on the same principle. The gaps of delays, with respect to measurements, are in all cases smaller than 3.27% for analytical formulas or simulations. This gap is in both cases, smaller than 2.01% in the calculus of the mean of responses times. A random event generator is used in simulation to obtain realistic distribution of delays (to offset effects of the jitter). Indeed, the shapes of the measurements and simulations histograms are very similar (FIGURE 6). However, the results of calculus using the classical method are very far from the real bounds. We can note a gap of more than 50% with respect to the measured minimal or maximal bounds. These results are not relevant to any conclusion about the performance of the architecture. Considering such result in the synthesis of controller for instance, would lead to unsatisfactory performances.

If we compare the results of both configurations (case 1 and case 2), we note that there is about a difference of 0.25 ms between the maximal bounds (resp. minimal bounds). It is exactly the value of transmission time of a frame T_{EM} and since the switches are very fast, the considered configurations are very similar. The main difference is the use of a RIOM for event source and another for consequence destination with a difference of one in order (R4 and R5). This result consolidates the general formulas of the section 4.2.

6. CONCLUSION

In this work, we modeled different configurations of automation architectures by the means of TEGs and Max-Plus algebra. We obtained an algorithm and analytical formulas for formal calculus of the response time. The

comparison of the results with experimental measurements, allowed us to check the validity either of the algorithm or the formulas. Their comparison with the classical method results, made clearer their effectiveness. Thus, by the use of these interesting formulas, it is easy to choose the adequate configuration of the components of an architecture to fulfil the desired temporal requirements. For further studies, it would be interesting to consider more general automation architectures with many PLCs and RIOMs. Finally, a study of an overall networked control system to evaluate the parts that our results can play in the synthesis of control strategy, is prospected.

REFERENCES

- [1] MODBUS Messaging on TCP/IP Implementation Guide V1.0b, (October 2006), [Online]. Available: www.modbus-ida.org/specs
- [2] J. P. Georges, E. Rondeau, and T. Divoux, (2002) How to be sure Ethernet networks will satisfy the real-time requirements? *Proceedings of IEEE Int. Symposium on Industrial Electronics*. July.
- [3] N. Kakanakov, M. Shopov, G. Spasov and H. Hristev (2007) Performance evaluation of switched Ethernet as communication media in controller networks. *International Conference on Computer Systems and Technologies (CompSysTech'07)*, pp. IIIA.8-1-6.
- [4] K. C. Lee and S. Lee, (2002) Performance evaluation of switched Ethernet for real-time industrial communications. *Computer standards & interfaces*, (24):411–423.
- [5] J. Greifeneder, G. Frey, (2006) Optimizing Quality of Control in Networked Automation Systems using Probabilistic Models. *Proceedings of the 11th IEEE Int. Conf. on Emerging Technologies and Factory Automation*, Prague, Czech Republic, September.
- [6] B. Ben-Hédia, F. Jumel and J.-P. Babau, (2005) Formal evaluation of quality of service for data acquisition systems. *Proceedings. of FDL'05*.
- [7] D. Witsch, B. Vogel-Heuser, J. Faure, and G. Poulard-Marsal, (2006) Performance analysis of industrial Ethernet networks by means of timed model-checking. *Proceedings of the 12th IFAC Symposium on Information Control Problems in Manufacturing*, pp 101–106.
- [8] D. A. Zaitsev, (2004) Switched LAN simulation by colored Petri nets. *Mathematics and Computers in Simulation*, 65(3):245–249.
- [9] G. Marsal, B. Denis, J.-M. Faure, G. Frey, (2006) Evaluation of response time in Ethernet-based automation systems. *Proceedings of 11th IEEE Int. Conf. on Emerging Technologies and Factory Automation*, Prague, Czech Republic, September.
- [10] B. Denis, O. De Smet, J.-J. Lesage, J.-M. Roussel, "Process of performance analysis and identification of systems as finite state automata", FR. Patent 01 110 933, August 2001.
- [11] F. Baccelli, G. Cohen, G.-J. Olsder, and J.-P. Quadrat, (1992) *Synchronization and Linearity: An algebra for Discrete Event Systems*, Wiley.
- [12] F. Baccelli, G. Cohen, and B. Gaujal, (1992) Recursive equations and basic properties of timed Petri nets. *Discrete Event Dynamic Systems: Theory and Applications*, 1 (4).