Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Genetic algorithms for delays evaluation in networked automation systems

B. Addad *, S. Amari, J-J. Lesage

Automated Production Research Laboratory LURPA, ENS-Cachan, 61 av. du Président Wilson, 94235 Cachan Cedex, France

ARTICLE INFO

Article history:
Received 12 April 2010
Received in revised form
15 September 2010
Accepted 19 October 2010
Available online 4 November 2010

Keywords:
Networked automation systems
Switched packets
Client-Server protocol
Delays evaluation
Genetic algorithms

ABSTRACT

In this paper, we present an approach to evaluate end-to-end delays in packets switching networked automation systems. Since Client-Server paradigm is considered for communication between the field devices, the existing methods of network delays evaluation are hardly applicable to assess realistic upper bounds of these delays. In an effort to enhance these delays evaluation, we propose an alternative method. Two algorithms, usually used for optimization problems, exhaustive and genetic algorithms, are then developed to achieve this purpose. While a formal proof about the capacity of the former one to ensure the worst delay overestimation is given, the latter proves to provide faster and more accurate results at the same time. This is shown on a practical case study while comparing the results of the two methods.

© 2010 Elsevier Ltd. All rights reserved.

1. Introduction

Fieldbus systems have been successfully introduced in industrial automation, ensuring real time requirements on one hand and devices safety on the other hand. Nowadays, the trend is also to use the same communication technology at different levels in the industrial organizations; management and automation. A solution that supports such a vertical integration has to be able to provide high throughputs in the upper level as well as small and accurate response times in the field level. The Ethernet solutions, which were initially developed to office networks, can be considered as such a new generation of fieldbuses. Currently, many automation producers and alliances developed their own industrial Ethernet standard (Neumann, 2007). Each solution with a specific protocol is best suited to a particular application. A Client/Server protocol like Modbus over Ethernet, even not adequate for strict real time applications like motion control, is a simple and a reasonable solution for many purposes in industrial control systems. Indeed, it is an application protocol (the 7th layer of the OSI model) that is completely compliant with the standard Ethernet. Therefore, vertical integration is easily achieved. Thus, high level functions like diagnosis and device management are easily implemented. Unfortunately, with such a protocol, no global medium access scheduling is available and different delays due to waiting for resources availability or synchronization are caused. So, the evaluation of its time performances like the response time is complex and the investigations that deal with this problem are rare. The existing methods are often studies of particular systems based on model checking (Greifeneder and Frey, 2007; Witsch et al., 2006; Ruel et al., 2009) that suffers from the classical state explosion problem. Another method is based on high level colored Petri nets simulation (Marsal et al., 2006; Zaitsev, 2004). This method is time driven and very onerous of time. Moreover, it does not provide a formal analysis or proof about its capacity to sweep the worst scenarios corresponding to the worst delays. Finally, an experimental method using a logical network analyzer dedicated to delays measuring is presented in Denis et al. (2007). Hence, the aim of the current paper is to propose an adequate method to assess upper bounds of end-to-end delays of switched packets in the context of Client-Sever automation systems. While a proof about the capacity of the method to assess the worst delays is provided, a genetic algorithm is developed to achieve it much faster.

The remainder of this paper is organized as follows. Section 2 introduces the context of our investigation and the motivations to develop a method for end-to-end delays evaluation. Thereafter, two algorithms, to look for the worst delays, are developed in Section 3: an exhaustive algorithm in Section 3.1 and a genetic algorithm in Section 3.2. A case study is then considered to perform a comparison between them in Section 4. Finally, Section 5 addresses some concluding remarks.

2. Client server automation systems over switched Ethernet

The studied automation architecture works according to Client/Server protocol. It is constituted mainly of PLCs (programmable logic controllers), RIOMs (remote input output modules) and a switched Ethernet network that enables communication between

* Corresponding author. Tel.: +33 147402762.
E-mail address: boussad.addad@lurpa.ens-cachan.fr (B. Addad).

0952-1976/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2010.10.017

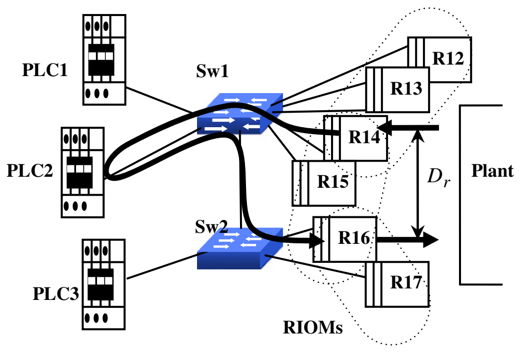

Fig. 1. Client-Server automation system.

all the components (Fig. 1). The PLCs (clients) send periodically requests to the RIOMs (servers) and wait for answers. When a RIOM receives a request, it puts it in a queue so as to process all the waiting requests according to a FIFO policy. A request is mainly to get information from the plant (e.g. is the maximal level of water reached?) or to provide orders (e.g. close the valve) or a combination of both. A major criterion of time performance of such a system is the *response time* D_r (Fig. 1). It is defined as the delay between the occurrence of an event in the plant (e.g. the maximum level is reached) and the arrival of the consequence, issued from a controller, on the controlled plant (closure of the valve). The response time can be defined intuitively as the reactivity delay of the automation system.

As a matter of fact, the evaluation of the response time of these systems is tricky. Indeed, different delays due to non synchronization of the components, resources sharing and of course the intrinsic delays (processing) are to be considered.

In Addad et al. (2010), we developed an analytic method to evaluate this response time. A formula giving an upper bound of the response time is obtained. Obviously, the delays we call the *end-to-end* network delays (including only the delays experienced in the switches) are involved in the formula. A typical end-to-end delay is the time for a request to cross the switches from its generation by a PLC until its arrival to a RIOM. Therefore, to assess an upper bound of the response time, upper bounds of these end-to-end delays are needed. It could be thought then that existing methods like the well known network calculus (Cruz, 1991; Le Boudec and Thiran, 2004; Georges et al., 2005) or worst case methods (Fan et al., 2008; Lee and Lee, 2002) can be used for this purpose. Unfortunately, in the context of switched packets, combined to Client-Server paradigm, it is not so obvious. Indeed, the flows independency supposed by the previous methods is not verified. For instance, a RIOM does only answer a received request and therefore a request with its corresponding answer cannot exist at the same time in the system. Besides this impeding fact, the formula is very sensitive to the upper bounds of the end-to-end delays. Indeed, a small overestimation of these delays may lead to a huge overestimation of the response time upper bound. As a result, the quality of control in these automation systems are dramatically degraded since the control law synthesis is based mostly on the upper bound assessment (Addad and Amari, 2008). Thereby, an adequate method to evaluate the end-to-end delays in such systems is to be investigated. This is the objective of the next section.

3. End-to-end delays evaluation

As aforementioned, the studied automation architecture works according to Client-Server protocol. The PLCs, which are the clients, send requests and the RIOMs (the servers) return answers accordingly. To explain the proposed method of delays evaluation, we use the system in Fig. 1.

- PLC1 sends a burst of three requests (series of three frames sent one after another) periodically with a period T_1 to the modules R12, R13 and R14 (Figs. 1 and 2).
- PLC2 sends a burst of three requests periodically with a period T_2 to the modules R14, R15 and R16.
- PLC3 sends a burst of two requests periodically with a period T_3 to the modules R16 and R17.

Actually, the PLCs which are the sources of frames (we use the term of “senders”) are neither synchronized nor scheduled and therefore can start sending their bursts at any time. A scenario is shown in Fig. 2 where lags τ_2 and τ_3 with respect to the start-sending date of PLC1 (chosen as a reference) are represented. Obviously, different lags (different scenarios) will lead to different interferences between the bursts of the PLCs and consequently different end-to-end delays within the system. Therefore, the main issue is then to find the critical scenario (the lags τ_i) that causes the maximum delay of a given frame. This is the purpose of the following algorithms.

3.1. Exhaustive exploration algorithm

As explained above, the interference between the bursts is entirely up to the lags τ_i between the senders. Intuitively, a sender will influence another if it starts sending frames at more or less the same time. A given frame experiences a delay depending on the *set* of frames that preceded it in a switch and are waiting to be forwarded first. In the context of our study where the pattern of generation of the frames from PLCs is known beforehand (periodically), all the possibilities of this *set* of frames can be identified *exhaustively*. In Fig. 3 for example, the number of possibilities of the set E of the frames of parallel bursts that enter before a frame f^* is equal to $4\times3=12$ (4 frames in the first burst and 3 in the second one). It is a simple combinatorial operation. As we notice in Fig. 3, there are infinite situations that correspond to the same set E, but obviously not the same delays. Therefore, the issue is to handle the variation of the delays from a scenario to another. The next lemma and theorem will show that the discrepancy from a scenario to another, corresponding to the same set E, is at worst equal to a well determined length.

Lemma. *Suppose f^* a frame that enters a switch at time t^* and experiences a delay Δ in it. Let f^i be a frame (of a parallel burst i) that comes immediately before f^* and enters the switch at time $t^*-\delta_i$.*

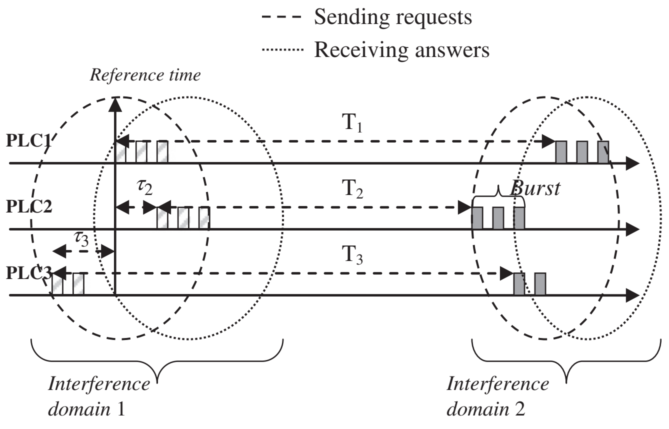

Fig. 2. Scanning mechanism and bursts interference.

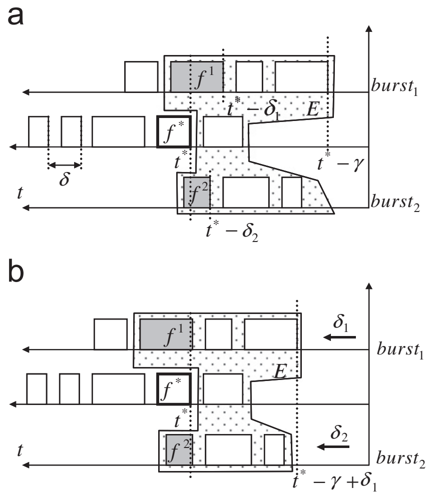

Fig. 3. Example of scenarios corresponding to the same *E*.

If the delay Δ is maximal, then the arrival time of f^i is necessarily the date $\theta_i = t^* - \varepsilon^*$ with $\varepsilon^* = 0^+$ i.e. f^* enters a bit after f^i. It follows that: $\delta_i = \varepsilon^* = 0^+$. One can equivalently write

Δ is maximal $\Rightarrow \theta = t^* - \varepsilon^*$

Proof (by contradiction). Let $\theta_i = t^* - \delta_i$ be the date of arrival of the frame f^i with $\delta_i > \varepsilon^*$. To remain under the assumption—*f^i the immediate frame entering before f^**—an enough condition on δ_i is to verify: $\delta_i < \delta$ where δ is the minimal inter-arrival time of the frames (see Fig. 3(a)). Suppose that f^i enters at date θ_i and this scenario leads to the critical situation i.e. the delay Δ is maximal. Now, suppose that f^i enters at date $\theta_i' = t^* - \varepsilon^*$ (see Fig. 3(b)) and this second scenario leads to a delay Δ'. Since the set E remains the same, from case (a) to case (b), and the arrivals of frames of this set are delayed, then the end of their processing will be delayed too. Finally, the frame f^* will experience necessarily a delay Δ' greater than Δ. This is of course absurd since from the beginning Δ is supposed to be the maximal delay. □

Actually, looking for the set E is equivalent to look for the adequate start-sending dates from PLCs or the lags τ_i (Fig. 2). Therefore, instead of tackling the problem by searching this set, we rather look for a combination of the lags corresponding to this worst scenario. The following theorem will provide an enough condition to achieve an exhaustive exploration while searching this worst scenario.

Theorem. *Suppose we look for the worst scenario by searching the lags τ_i in an enough large domain.*

If a step $\delta_e < \delta$ is used to vary the lags of the start-sending dates τ_i in a combinatorial scheme, then an exhaustive exploration is achieved and the accuracy of the result is worth δ_e. In other words, the gap between the assessed delay and the effective worst one is smaller than δ_e.

Proof. From the lemma, the maximal delay is reached if f^* enters the switch a bit after f^i. Since the step δ_e is smaller than the inter-arrival time, then we are sure to obtain the set E and the best case is when f^i enters δ_e before f^*. Hence, if every frame f^i arrives δ_e before t^*, then the processing of the set E will be finished, in the best case, δ_e earlier. Thus, the assessed delay added up to δ_e is at worst δ_e greater than the real delay. □

Remark 3.1. With Ethernet $\delta = \min_k((72+12)/C_k)$ where 72 bytes is the minimal frame length (preamble included), 12 bytes is the inter-frames gap (96 bits or 12 bytes) and C_k the physical capacity of the port k of a switch.

Thereby, we can use a step δ_e, respecting the previous theorem condition, to vary the lags τ_i in a domain noted $[-T_{Domain}, +T_{Domain}]$ where T_{Domain} is taken large enough to include the worst scenario (assessed for instance using a pessimistic method).

Hence, for each combination of the lags, we simulate the behaviour of the system and assess the corresponding delay $D_{end2end}$ (see Fig. 4). This is represented by the block *Simulator* $(\tau_1, \tau_2, \ldots, \tau_n)$ in Fig. 4 with the lags τ_i being introduced as entries. For a system with n senders (PLCs), the diagram of the exhaustive exploration method is drawn in Fig. 4. The maximum of all the obtained delays is then chosen as an upper bound. The simulation can be achieved using any suitable simulation method. In the case of the current study, a virtual queuing based simulator, already applied in Addad and Amari (2009), has been used for this purpose. It has the advantage of being an event driven simulator and therefore much faster than the usual time driven simulators. Despite this advantage, the simulation may last a long time when dealing with very large systems. Indeed, the number of times to run the block *Simulator* $(\tau_1, \tau_2, \ldots, \tau_n)$ is exactly equal to $(2 \cdot \lfloor T_{Domain}/\delta_e \rfloor)^n$ where n is the number of senders and $\lfloor T_{Domain}/\delta_e \rfloor$ the integer part of T_{Domain}/δ_e. Thus, this number grows exponentially when the number of senders grows linearly. While this exhaustive method is quite satisfactory in some cases, it can be very onerous of time in others, so other more efficient methods are to be investigated. Genetic algorithms are then considered as an alternative in our study.

3.2. Genetic algorithms

Genetic algorithms (GAs) were first introduced by Holland (1975) and have been extensively explored later in other

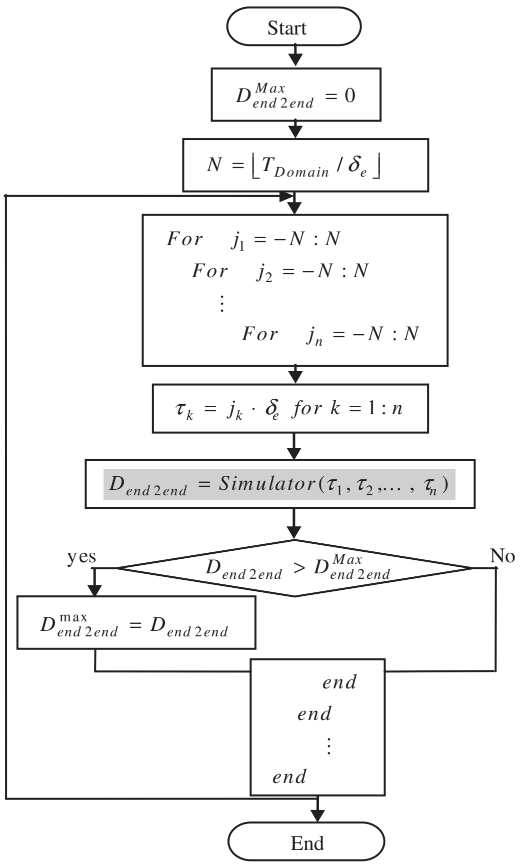
Start
$D^{Max}_{end2end} = 0$
$N = \lfloor T_{Domain} / \delta_e \rfloor$
$For \quad j_1 = -N : N$
$For \quad j_2 = -N : N$
$\vdots$
$For \quad j_n = -N : N$
$\tau_k = j_k \cdot \delta_e \; for \; k = 1 : n$
$D_{end2end} = Simulator(\tau_1, \tau_2, \ldots, \tau_n)$
yes
$D_{end2end} > D^{Max}_{end2end}$
No
$D^{max}_{end2end} = D_{end2end}$
end
end
$\vdots$
end
End

Fig. 4. Exhaustive exploration method diagram.

investigations for optimization problems resolution in almost all domains. With regard to fieldbus systems, GAs were used in Georges et al. (2006), Zhang and Zhang (2007) and Carro-Calvo et al. (2010) to optimize the partition of switched Ethernet networks into sub-networks where the intra sub-network communication traffic is maximized whereas the inter sub-networks communication is minimized. They were used in Lee et al. (2004) to help the designer to select the optimal timers in a Profibus Token Passing Protocol fieldbus so as to satisfy the maximum allowable delays of the real time data on one hand and maximize the non real time data transfer on the other hand. They were also used in Cheng and Yang (2010) and García-Nieto et al. (2010) to investigate the problem of dynamic quality of service in mobile ad hoc networks. We can also quote (Saniee Abadeh et al., 2007) where GAs were used to improve the security of networked computers while reducing the false alarms in determining intrusive activities. In the current study, we obviously use GAs to evaluate the maximum end-to-end delays of switched packets in Client-Server networked automation systems.

A GA is a structured stochastic optimum search method that mimics the process of biological evolution. At the beginning, a GA is initialized with a collection of sets of parameters. Each set is called a *chromosome* whereas a parameter a *gene*. Then, the chromosomes are evaluated according to their fitness of solving an optimization problem (maximization of $D_{end2end}$ in our case). At every generation (algorithm execution time), the fittest chromosomes (parents) are allowed to mate (crossover) and bear offspring (children). Then, the best children become the parents of the next generation. The previous steps are repeated until finding a satisfactory solution of the problem. To avoid falling in a local optimum as is often the case with the traditional hill-climbing search methods, a mutation mechanism is added. Depending on the optimization problem, many ways exist to accomplish every step of the GA: codification of the chromosomes, initialization of the population, selection of the parents, crossover and mutation. Here are the features of the applied GA in our problem:

- *Chromosomes encoding*: The parameters handled in our problem are the start-sending dates or the lags τ_k which are real numbers. Therefore, we apply a continuous GA and the genes are simply real numbers. The kth gene noted g_k^i of each chromosome i represents the lag τ_k. The ith chromosome is noted $\Theta^i=(g_2^i,g_3^i,\cdots,g_n^i)$ and its fitness $J(\Theta^i)$ where $J(\Theta^i)=D_{end2end}$.
- *Initialization of the population*: In some automation systems, we may have an idea about the worst scenario and therefore initialize the GA with the corresponding parameters. In complex systems, it is rarely the case. Therefore, a random initialization of the parameters in the domain $[-T_{Domain},+T_{Domain}]$ is considered in the current study.
- *Selection of the parents*: According to natural selection theory, the fittest members with the greatest fitness survive and the others die. One way to represent this strategy is to use the so-called roulette-wheel-spinning. A unit circumference is cut into slices, each slice being dedicated to one member according to its relative fitness. The ith member is then associated to the portion $P^i=J(\Theta^i)/\sum_k J(\Theta^k)$. Thereafter, we spin the wheel and if the pointer stops at the jth portion, the chromosome j is selected and placed into a mating pool. Obviously, the fittest members with the biggest portions are more likely to be chosen than the others.
- *Crossover* (*mating*): Pairs are chosen from the mating pool and mate with a given probability P_{Css} to form the offspring. Many approaches exist to deal with the crossover in continuous GA (Passino, 2005). One of the most efficient and simplest methods is combining two genes g_k^i and g_k^j of the parents Θ^i and Θ^j to form the children Ch^i and $Ch^{i'}$ (with genes noted Chg_k^i and $Chg_k^{i'}$) as follows:

  $$\begin{cases} Chg_k^i=\alpha g_k^i+(1-\alpha)g_k^j \\ Chg_k^{i'}=\alpha g_k^j+(1-\alpha)g_k^i \end{cases}$$

  α being a random real number in the domain [0,1].
- *Mutation*: To avoid a quick convergence of the GA and the risk of being trapped in a local optimum, we force it to explore other regions of the space by introducing random changes or mutations in the genes. The process of mutation on a gene g_k^i of a member Θ^i can be performed with a probability P_{Mt} as follows:

  $$g_k^i=T_{Domain}(2\beta-1)$$

  β being a random real number in the domain [0,1]. The mutation is applied on all the members of the population except for the best member of the generation if *elitism* option is chosen. With *elitism* option, the best member of a generation is conserved for the next generation without any change. As a consequence, the fitness function is monotonic along the generations (non-decreasing function as in Fig. 7).

The process described previously is repeated until a satisfactory solution is found or the best fitness does not change during a minimum number of generations. The GA applied for our problem resolution is implemented using Matlab software and the main steps are shown in Fig. 5. The following notations are adopted:

- *PopNum* is the total number of members of the population.
- *MaxGen* is the maximal number of generations.
- *PCss* is the probability of crossover.
- *PMt* is the probability of mutation.
- *rand* is a random number from the domain]0, 1[.

Remark 3.2. As can be seen in Fig. 5, the same block representing the simulator used with the exhaustive method is used with GA. Indeed, this block is only used to simulate the behaviour of the system for a given scenario so as to assess the corresponding delay. Note that using a simulator and introducing manually the entries (the lags τ_i) of each scenario would not be viable given the number of times the block is to be run. In this study, both the simulation block and optimization methods (either exhaustive or GA) are implemented using the same programming language (Matlab) and the whole process is therefore automated.

4. Case study

The system of the case study is shown in Fig. 6. It works as follows: three controllers, PLC1, PLC2 and PLC3 scan, respectively, the ordered sets of RIOMs: $\{R_{12},R_{13},R_{14}\}$, $\{R_{14},R_{15},R_{16}\}$ and $\{R_{16},R_{17}\}$ with 72 bytes requests. This is the automation part we used for explanation in Section 2, but as stated previously, the studied system presents the advantage of ease of high level functions integration since standard Ethernet is considered for communication. Therefore, a second part *PART II* is added for this purpose; PC1 exchanges long frames of 1008 bytes length with PC3 and PC4 whereas PC2 has the role of a supervisor of the first part *PART I*. It scans all the RIOMs every 300 ms with 72 bytes requests so as to check the good functioning of the modules. The aim is to evaluate an upper bound of the delay $D_{end2end}$ that a request, sent from PLC2 to R14, experiences in the network (Fig. 6). Any request would be chosen, but we selected this one since R14 is shared by PC2, PLC1 and PLC2. Thus, particular and not trivial scenarios lead to the worst delay. Therefore, we should find them using different methods. To evaluate the delay in question, we applied the two previously

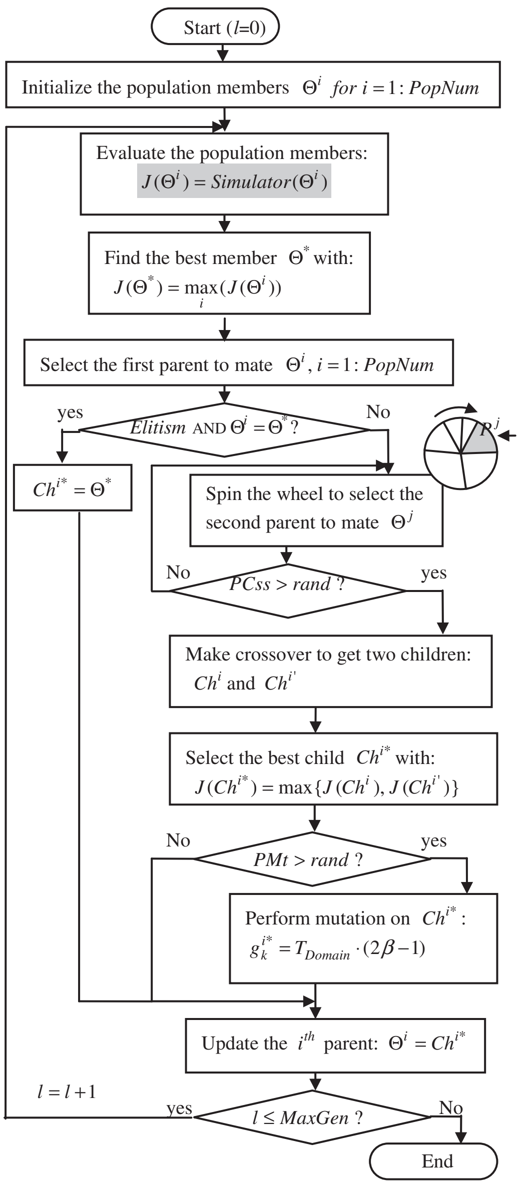
Start (l=0)
Initialize the population members Θ^i for $i=1:PopNum$
Evaluate the population members: $J(\Theta^i)=Simulator(\Theta^i)$
Find the best member Θ^* with: $J(\Theta^*)=\max_i(J(\Theta^i))$
Select the first parent to mate $\Theta^i, i=1:PopNum$
yes
No
Elitism AND $\Theta^i=\Theta^*$?
P^j
$Ch^{i*}=\Theta^*$
Spin the wheel to select the second parent to mate Θ^j
No
yes
$PCss > rand$?
Make crossover to get two children: Ch^i and $Ch^{i'}$
Select the best child Ch^{i*} with: $J(Ch^{i*})=\max\{J(Ch^i),J(Ch^{i'})\}$
No
yes
$PMt > rand$?
Perform mutation on Ch^{i*}: $g_k^{i*}=T_{Domain}\cdot(2\beta-1)$
Update the i^{th} parent: $\Theta^i=Ch^{i*}$
$l=l+1$
yes
No
$l \le MaxGen$?
End

Fig. 5. GA diagram.

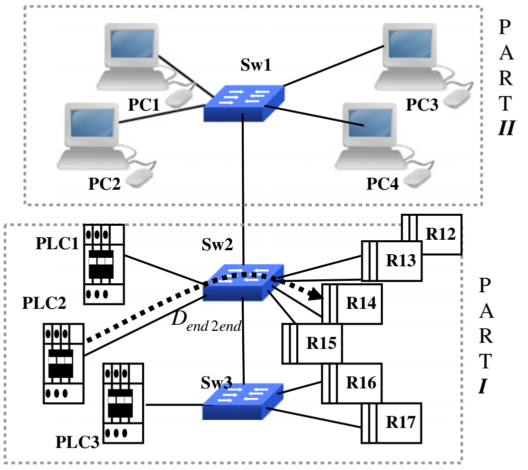

Fig. 6. Case study: evaluation of delay $D_{end2end}^{max}$.

exposed methods, exhaustive exploration and GA. We also considered two cases using the exhaustive method: Exhaustive I with step δ_e being set up to 20 μs at the beginning and to 1 μs subsequently whereas in Exhaustive II this step is progressively narrowed from 50 to 1 μs. The results of assessment using the different approaches are reported in Table 1.

As we can note, all the methods lead to nearly the same assessment of the worst end-to-end delay. The duration of simulation is however very different. First, the use of a small step in the exhaustive exploration may lead to long simulation durations (Exhaustive I). Indeed, the smaller the step δ_e is, the greater the number of situations to check is (recall that this number is equal to $(2\cdot\lfloor T_{Domain}/\delta_e\rfloor)^n$). A more interesting method is to begin with a relatively large step δ_e (50 instead of 20) if possible (it must check the condition given by the theorem) and narrow it progressively. This is made in Exhaustive II and we notice that the simulation duration is dramatically reduced from ~2 h to about 4 min. We can also note that the gap of underestimation in all the cases is smaller than the step δ_e. Indeed, the sum of the used step and the assessed delay is always greater than the final assessed delay. In Exhaustive II for instance, we have an assessed delay of 283.52 μs with a step δ_e equal to 50 μs. The sum of both is greater than the final delay, which is equal to 319.96 μs. This feature can be checked in all the results of Table 1. Naturally, this corroborates the statements of the theorem of Section 3.1.

On the other hand, the genetic algorithm not only provides an accurate upper bound of the delay (see Fig. 7), but also achieves it in relatively negligible simulation duration of about 29 s. Moreover, a gap of 1% from the final maximal delay is reached in almost 100

Table 1
Results of evaluation.

Method	Step	δ_e (μs)	Assessed worst delay (μs)	Simulation duration
Exhaustive I	Step 1	20	319.96	~2 h
	Step 2	1	**319.96**	
Exhaustive II	Step 1	50	283.52	~4 min
	Step 2	10	313.52	
	Step 3	1	**319.96**	
GA	/		**320.50**	~29 s

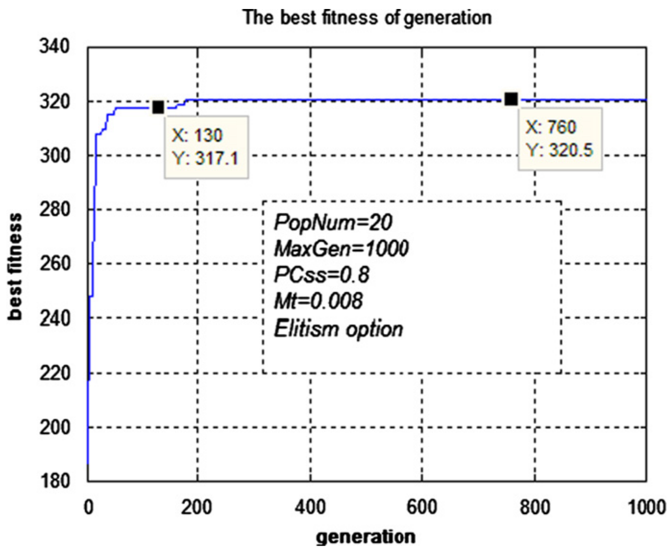

Fig. 7. The best fitness (maximal end-to-end delay) along the generations.

generations (about 2 s simulation duration). Compared to duration of 2 h or even 4 min using exhaustive search, GA achieved better results much faster. Finally, we can point out that the gap between the final assessed delays using Exhaustive (i.e. 319.96 μs) and GA (i.e. 320.50 μs) is smaller than 1 μs (the final step used in Exhaustive). Again, this is in accordance with the theorem since we can be confident about 320.50 μs of being the exact worst delay given that it does not change from generation 200 to generation 1000 (Fig. 7).

5. Conclusion

In this paper, we proposed an approach to evaluate end-to-end delays of switched packets in networked automation systems. The Client-Server paradigm being used as the protocol of communication, the existing methods of delays evaluation are indeed hardly applicable. So, an exhaustive exploration method and genetic algorithms were developed to assess upper bounds of these delays. On a case study, we showed that both methods give almost the same results but a clear advantage of using GAs has been noticed since the simulation durations are dramatically different. While the exhaustive one may be preferred when dealing with simple automation systems, given its simplicity of implementation, GAs are more suitable for evaluation when large architectures are under consideration.

References

Addad, B., Amari, S., 2008. Delays evaluation and compensation in Ethernet networked control systems. In: Proceedings of the 16th International Conference on Real Time and Network Systems, Rennes, France, pp. 139–148.

Addad, B., Amari, S., 2009. Evaluation de délais dans les systèmes de communication temps-réel en utilisant des files d'attente virtuelles. MSR09-JESA Journal Européen des Systèmes Automatisés, Nantes, France, pp. 985–1000.

Addad, B., Amari, S., Lesage, J.-J., 2010. Analytic calculus of response time in networked automation systems. IEEE Transactions on Automation Science and Engineering 7 (4), 858–869.

Carro-Calvo, Leo, Salcedo-Sanz, Sancho, Portilla-Figueras, Jose A., Ortiz-García, E.G., 2010. A genetic algorithm with switch-device encoding for optimal partition of switched industrial Ethernet networks. Journal of Network and Computer Applications 33 (4), 375–382.

Cheng, Hui, Yang, Shengxiang, 2010. Genetic algorithms with immigrant schemes for dynamic multicast problems in mobile ad hoc networks. Engineering Applications of Artificial Intelligence 23 (5), 806–819.

Cruz, R.L., 1991. A calculus for network delay, Part I: network in isolation. IEEE Transactions on Information Theory 37 (1), 114–131.

Denis, B., Ruel, S., Faure, J.-M., Marsal, G., 2007. Measuring the impact of vertical integration on response times in Ethernet fieldbuses. In: Proceedings of the 12th IEEE Conference on Emerging Technologies and Factory Automation, Patras, Greece, pp. 532–539.

Fan, X., Jonsson, M., Jonsson, J., 2008. Guaranteed real-time communication in packet switched networks with FCFS queuing. Computer Networks 53 (3), 400–417.

García-Nieto, J., Toutouh, J., Alba, E., 2010. Automatic tuning of communication protocols for vehicular ad hoc networks using metaheuristics. Engineering Applications of Artificial Intelligence 23 (5), 795–805.

Georges, J.P., Rondeau, E., Divoux, T., 2005. Confronting the performances of switched Ethernet network with industrial constraints by using Network Calculus. International Journal of Communication Systems 18 (9), 877–903.

Georges, J-P., Divoux, T., Rondeau, E., 2006. A design process of switched Ethernet architectures according to real-time application constraints. Engineering Applications of Artificial Intelligence 19 (3), 335–344.

Greifeneder, J., Frey, G., 2007. Probabilistic timed automata for modeling networked automation systems. In: Proceedings of the First IFAC Workshop on Dependable Control of Discrete Systems DCDS, Cachan, France, pp. 143–148.

Holland, J., 1975. Adaptation in Natural and Artificial Systems. Ann Arbor University of Michigan Press.

Le Boudec, J.Y., Thiran, P., 2004. Network Calculus: A Theory of Deterministic Queuing Systems for Internet. Springer-Verlag.

Lee, K.C., Lee, S., 2002. Performance evaluation of switched Ethernet for real-time industrial communications. Computer Standards and Interfaces 24 (5), 411–423.

Lee, K.C., Kim, H.K., Lee, S., Lee, M.H., 2004. Timer selection for satisfying the maximum allowable delay using performance model of Profibus Token passing protocol. IEEE Transactions on Industrial Electronics 51 (3), 701–710.

Marsal, G., Denis, B., Faure, J.-M., Frey, G., 2006. Evaluation of response time in Ethernet-based automation systems. In: Proceedings of the 11th IEEE Conference on Emerging Technologies in Factory Automation, Prague, Czech Republic, pp. 380–387.

Neumann, P., 2007. Communication in industrial automation—what is going on? Control Engineering Practice 15 (11), 1332–1347.

Passino, K.M., 2005. Biomimicry for Optimization, Control and Automation, Chapter IV: Evolution. Springer-Verlag.

Ruel, S., de Smet, O., Faure, J.-M., 2009. Finding the bounds of response time of networked automation systems by iterative proofs. In: Proceedings of the 13th IFAC Symposium on Information Control Problems in Manufacturing, Moscow, Russia, pp. 1365–1370.

Saniee Abadeh, Mohammad, Habibi, Jafar, Barzegar, Zeynab, Sergi, Muna, 2007. A parallel genetic local search algorithm for intrusiuon detection in computer networks. Engineering Applications of Artificial Intelligence 20 (8), 1058–1069.

Witsch, D., et al., 2006. Performance analysis of industrial Ethernet networks by means of timed model-checking. In: Proceedings of the 12th IFAC Symposium on Information Control Problems in Manufacturing, Saint Etienne, France, pp. 101–106.

Zaitsev, D.A., 2004. Switched LAN simulation by colored Petri nets. Mathematics and Computers in Simulation 65, 245–249.

Zhang, Qizhi, Zhang, Weidong, 2007. Network partition for switched industrial Ethernet using genetic algorithm. Engineering Applications of Artificial Intelligence 20 (1), 79–88.