Evaluation of Response Time in Ethernet-based Automation Systems

Gaëlle Marsal, Bruno Denis, Jean-Marc Faure, Georg Frey

Evaluation of Response Time in Ethernet-based Automation Systems

Gaëlle Marsal, Bruno Denis, Jean-Marc Faure
ENS de Cachan
61, av. du Président Wilson
94235 Cachan Cedex, France
{marsal, denis, faure}@lurpa.ens-cachan.fr

Georg Frey
University of Kaiserslautern
Erwin-Schrödinger-Str. 12
67663 Kaiserslautern, Germany
frey@eit.uni-kl.de

Abstract

This paper presents a method to assess response time of automation system architectures including industrial switched Ethernet networks using client/server protocols. The method relies upon modeling the behavior of the components of these architectures in the form of Hierarchical Timed Colored Petri Nets and upon simulation of these models. A case study exemplifies the method and shows how it can facilitate design of automation systems including this kind of industrial Ethernet networks.

1. Introduction

Many Ethernet-based solutions for distributed automation systems are currently available or under development [1, 2, 3, 4]. Each one of these solutions can be ranked into one of the following two categories. Either it is based on a specific proprietary protocol developed by an automation solution supplier, using master/slave or producer/consumer mechanisms [5], or it relies on the generic TCP/IP client/server protocol [4]. Time performances of networks of the first class can be assessed by using methods developed for former non-Ethernet industrial networks which used the same protocols, whereas time performances evaluation of client/server-based solutions, in which resource sharing is a new concern, requires new investigations. This paper presents results of such investigations that focus on automation systems distributed over Ethernet using the standard TCP/IP client/server protocol.

A network can connect controllers either to a continuous process, modeled as a set of differential equations, or to a discrete process, modeled as a Discrete Event System (DES), e.g. using state automata or Petri nets. In these two cases, network performances impact different features. If the network connects a continuous process to controllers, the main feature that is impacted is stability [6] of the closed loop system while if focusing on control of DES it is the reactivity of the automation system. That includes both controllers and the network, which is modified when network performances change. This feature, on which this paper focuses, can be evaluated by the response time of the automation system, defined as the delay between the occurrence of an event from the controlled system and the occurrence of the resulting event produced by a controller back to this system.

This response time includes delays in all the components of the automation system. Analysis of the delay in each component of this system results – for the architectures that are studied in this paper – in a decomposition into three elementary delays: firstly delays caused by data processing, secondly delays caused by waiting for synchronization and thirdly delays caused by waiting for resource availability.

Basically, there are to approaches to determine the response time: analytically – using e.g. Network Calculus [7] or Model Checking [8] – and simulation. Given that all these delays are not independent, it is not trivial to determine the response time analytically [9]. Furthermore, the analytical approaches only deliver worst-case bounds for the response time. However, from the application point of view in automation the distribution of the response time is also of interest. Hence, a simulation approach has been chosen. For this, a generic model of Ethernet-based automation systems is first designed in a high-level Petri Nets formalism. The simulation of an instantiation of this generic model, that models a particular Ethernet-based architecture, enables to determine the whole distribution of its response time. Given this distribution, it is possible to check whether the control architecture fits requirements such as the maximum response time or its range.

The rest of this paper is organized as follows: the next section gives an overview on architectures of Ethernet-based automation systems and details those on which focus is put. Section 3 analyzes the problems in response time determination. Then, in section 4 the proposed modeling approach is explained. Thereafter, section 5 shows how the generic models introduced so far can be instantiated for a given architecture. A case study in section 6 demonstrates the benefits of the presented approach. Finally a short summary and some prospects are given in the concluding section.

2. Ethernet-based fieldbuses in automation

The introduction of Ethernet technology at field level enables great prospects both for interoperability and flexibility of automation systems. Indeed, as Ethernet is a widely used standard [10] in enterprise networking, it is possible to communicate over all enterprise levels, to use Commercial Off The Shelf (COTS) devices for automation and to take benefit of the "plug and play" feature of Ethernet. However, the CSMA/CD [8] access method implemented in Ethernet is not deterministic. This implies that collisions are possible on the network, and so that variable and potentially infinite delays may be introduced.

A radical solution to this problem is to implement a specific protocol reproducing the behavior of former fieldbuses with master/slave or producer/consumer mechanisms [11] on Ethernet. This solution often requires the use of hub devices, more adapted to broadcast messages needed in such protocols. This is the case for ProfiNet protocol by Siemens or EtherNet/IP by Rockwell Automation. These protocols are neither open nor standard and so they do not enable to take full advantage of the Ethernet standard. Another solution is to use switches to avoid collisions and TCP/IP standard protocols [12, 13]. This is the case when using soft controllers with Modbus clients or commercial solutions like Modbus TCP/IP by Schneider Electric [14]. This solution really provides new possibilities concerning flexibility and interoperability.

However, as TCP/IP implements a client/server protocol, no global resource management is available. With such a protocol, all devices can send a frame on the network at the same time. As a consequence, each device can receive several frames though it is not available to treat them and in this case the order of their treatment is decided locally. Ethernet frames that are not treated immediately when they are received in a component are waiting in a buffer. The consequent waiting time is variable depending on the load of the component. In this case, it is not trivial to assess the delay induced by Ethernet and TCP/IP protocols.

This study is limited to controllers, as Programmable Logic Controllers (PLCs) or industrial PCs, and Distributed Input Output Devices (DIODs) interconnected by a switched Ethernet network. Only common commercial devices for automation purposes are studied, such as modular controllers, DIODs and "store and forward" switches. In the controller, one board is dedicated to the cyclic execution of the user program (inputs reading, user program execution, and outputs updating), while another one is dedicated to the cyclic scan of DIODs through Ethernet. These two boards, which have their own processor, are not synchronized and communicate via a shared memory accessible through a backbone bus. In the following, the term "device" is used for controllers, DIODs and switches, while "components" is used for subparts of devices, such as Ethernet boards and CPU boards in controllers. Fig. 1 illustrates the type of studied architecture, with an example composed of one controller (PLC1; PLC stands for controller in the models), three switches (Switch1 to Switch3) and seven DIODs (DIOD1 to DIOD7).

The communication over Ethernet follows a client/server protocol, where controllers are clients and DIODs are servers. So the DIODs send data to the controller only after being requested. Hence, the controller has to request regularly the DIODs to update its Input images and the Output values on DIODs. This procedure is called IO scanning. Here only cyclic IO scanning is considered, with a minimum cycle time set up in each client which is, in the studied architectures, the controller. The IO scanning cycle studied is the following: Each Ethernet board of a controller sends requests to every DIOD concerned by this cycle. Then it waits until the minimum cycle time is finished. If all answers arrived, then it begins a new cycle, else it waits until all answers come back. In this study, frame losses are not considered, so no time-out has to be taken into account.

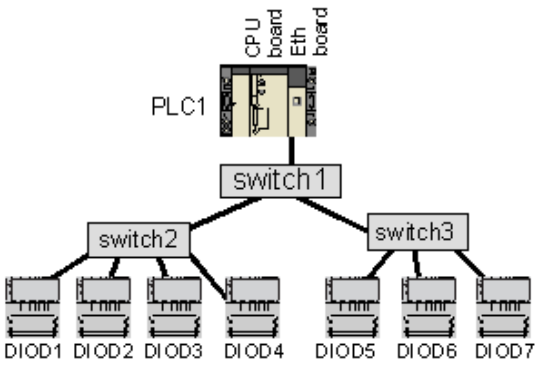

Figure. 1. Example of the studied Ethernet-based automation architecture.

3. Response Time in Switched Ethernet-based Automation Systems

The response time is defined as the delay between the occurrence of an event on the process, and the occurrence of the reaction event, issued from the controller, on the process. It results in data traveling between and data processing inside the different devices. A better understanding of the response time is possible by decomposing it into elementary delays in each component depending on their causes. Every component delay may be decomposed in the three following delays:

- delay caused by data processing,
- delay caused by waiting for synchronization,
- and delay caused by waiting for the availability of a resource (for shared resources).

The first two types of delay are present whatever fieldbus and protocol. Delays of the first type are characteristics of the components and can be known from technical documentation. There are two different waiting delays for synchronization, one called “internal” and the other one “external”. Internal synchronization delay refers to synchronization between components within a device, e.g. a controller, while external synchronization delay refers to synchronization between components. Both synchronization delays can be evaluated thanks to techniques developed for former fieldbuses [15, 16]. The third type of delay is specific to the client/server communication mechanism. The aim of the proposed simulation approach is to take into account the three sources of delays.

Fig. 2 illustrates the response time between input I1 and output O1 occurring respectively on DIOD4 and DIOD5 of the architecture given in Fig. 1. The component delays are represented for each component on its timing diagram by a grey rectangle for the processing delay, by a white rectangle for the waiting for synchronization and by a black rectangle for waiting for resources availability. To understand the relationship between the overall response time and elementary components delays, let us follow the data along its route in the automation system, from sensor to actuator. First, the digital signal coming from a sensor (connected to input I1) is filtered in DIOD4. This implies a processing delay given in technical documentation. The information on the state of I1 waits in a buffer for the request coming from a controller (delay of waiting for external synchronization). Then, it waits for availability of the DIOD CPU resource (delay of waiting for resource). When the resource is available, the information is encapsulated in a Modbus TCP/IP Ethernet frame and sent as a response (delay of processing data). This frame, which is now on the network, has to be forwarded by two switches, Switch2 and Switch1. When the frame arrives in a switch, it first waits for the resource before to be treated. Finally, the response enters the input buffer of the controller’s Ethernet board. When the resource is free, the board processes the frame and writes the new data into the shared memory. However, the user program takes it into account not before its next reading phase, resulting in a waiting phase for internal synchronization. There half of the route is done, the other half consists of sending the result of the user programs calculation (O1:= I1) back to the process issuing the corresponding reaction at the actuator (noted O1). So from bottom to top, firstly, the output O1 is written into the shared memory, secondly, its value is brought into the Ethernet board at the next IO scanning cycle, and finally, when the CPU of the Ethernet board is available, the modbus TCP/IP Ethernet frame is built and sent over the network to DIOD5, via Switch1 and Switch3. After that, when the DIOD’s resource is free, its treatment begins to send a signal on the process to cause the desired reaction.

Several authors [17, 18, 5] have addressed the problem of evaluation of time performances of Ethernet-based networks by focusing on only delays caused by switches, without taking into account the delays provoked by controllers and DIODs. On Fig. 2, this corresponds to the delay for crossing the cascade of Switch2 and Switch1 and the cascade of Switch1 and Switch3 but independently from the rest of the system. However, even when this delay is determined and when all the processing times are known, the overall response time is not directly reachable because both the waiting delays for synchronization and the waiting delays for resources in DIODs and controllers are not known. Indeed, this is the conjunction between the three delays in all components of the route which has the major influence on the response time. That is the reason why it is necessary to study the whole automation system, including controllers, network devices, and software as it has been initiated in process control area [6].

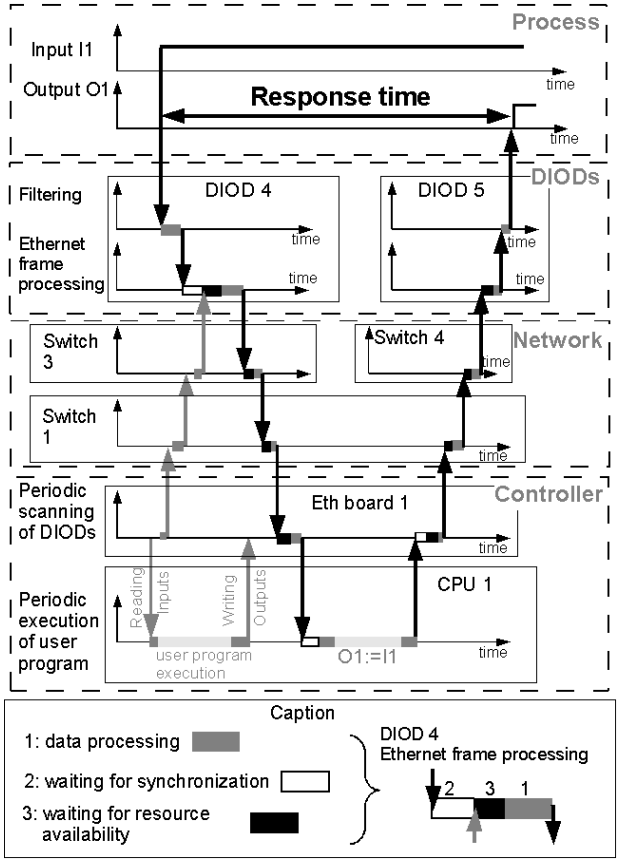

Figure 2. Response time of the system in Fig. 1 from input I1 to output O1.

However, if Fig. 2 illustrates clearly the different delays in each component, it is very difficult to evaluate all the delays of waiting for synchronization and resource along a route, given the different periodic processes running in parallel that induce complex relationships. That is why, even if analytical computation or formal methods can provide an upper-

bound of the worst-case or a lower-bound of the best case of response time, it is really complex with these approaches to obtain accurate and realistic values [9]. A more appropriate method for such complex systems is the simulation of a dynamic model of the whole automation architecture. The network research community has developed dedicated simulation tools as OPNET, OMNET or NS-2. These tools provide very detailed and complex models for switches and protocols. However, at the best of our knowledge, nothing exists concerning controllers or DIODs whose delays are also to be taken into account. Moreover, potential users are automation specialists who are not supposed to be familiar with network simulation tools. Finally, the use of given black-box models of network components makes it hard to analyze the reasons for delays in specific settings. This analysis however is of interest to derive more abstracted models for formal verification. These drawbacks lead to the necessity to develop new models in a language that is adapted to DES modeling with existing implementation in simulation tools, well-known from automation community and with formal semantics. Given the objective of designing a generic model for time performance evaluation, the Hierarchical Colored Timed Petri Nets (HCTPN) class totally complies with modeling requirements [19]. In the presented work CPNtools from the University of Aarhus is used to design and simulate the HCTPN models.

4. Modeling with Hierarchical Timed Colored Petri Nets

Based on knowledge of the isolated devices behaviors, a knowledge-based component-oriented modeling approach is chosen. Moreover, the model presented is generic for the class of studied architecture. Indeed, it enables to model any system composed of modular controllers and DIODs distributed over a switched Ethernet network, employing a client/server protocol. The complete model is too large and complex to be presented in detail in a paper. So this section focuses first on its hierarchical structure and its dynamics and then explains a part of the model, the IO scanning performed by Ethernet boards of controllers.

4.1. Structure of the model

Hierarchical modeling enables to use refinement design. On Fig. 3, the highest level *Global* contains the model of the whole architecture class. It is refined in the lower level in one sub-model per component type. Indeed, in any given architecture there is one *Process*, a set of *DIODs*, a set of *Switches*, a set of *Ethernet boards of PLC* and a set of *CPUs of PLC*. The last two modules constitute a controller (PLC). The last level of hierarchy is devoted to communication stacks in components as *Ethernet TCP IP communication stacks*.

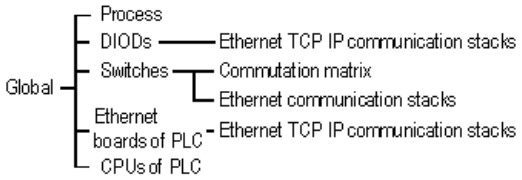

Figure 3. Structure of the HTCPN model.

Fig. 4 shows the HTCPN representation of the highest hierarchical level *Global*, each component is modeled as a substitution transition (*Process*, *DIODs*, *Switches*, *Ethernet boards of PLC* and *CPUs of PLC*). This structure can remain the same for each architecture thanks to colors. These enable to distinguish different types of tokens, here, one color per component and two colors for data exchange are used. For instance a token colored as *Ethboard* can only represent an Ethernet board of a controller, and if there are several tokens of this type, they describe several components. Tokens for data exchange are colored as "events" or "frames", depending if these represent events in the process and in controllers or Ethernet frames on the network. To model the information transport from one component to another, tokens are moved from one component model to another via the interface places *Shared event data*, *Process events* and *Ethernet frames*. The tag *I/O* on these places notifies the role of the interface among several sub-models. Output tag means receiving tokens from sub-models while input tag means sending tokens to sub-models. The places *I/O* have the two roles of input and output.

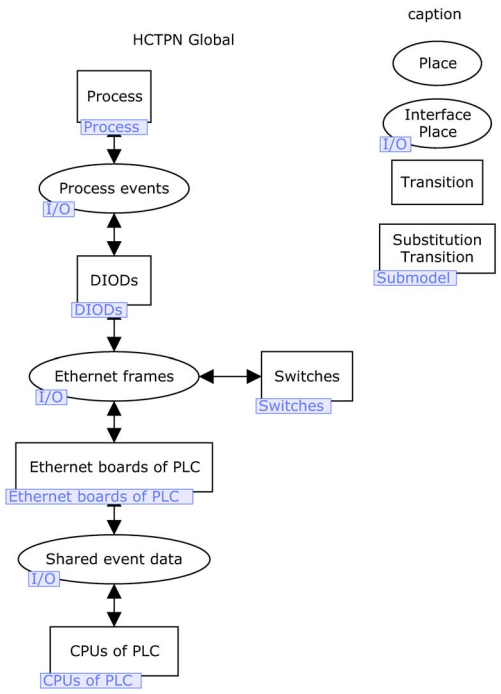

Figure 4. Global HTCPN model

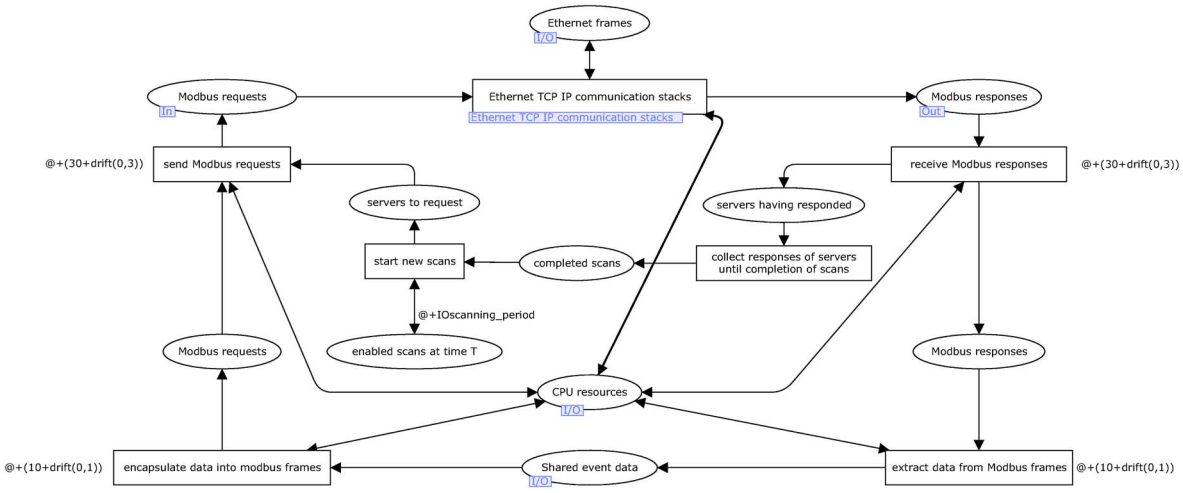

Figure 5. HTCPN model of IO scanning cycle.

4.2. Dynamics of the model

The elementary processing delays as well as the durations of the scan cycles are given as parameters to timed transitions in the HCTPN models. To give realistic results, the time delays are built by adding a small jitter to a constant delay. This jitter also avoids unwanted synchronization effects in the model. As an example see the transition *encapsulate data into modbus frames* in Figure 5 (bottom left corner). The encapsulation is assumed to take 10 µs and a jitter of 1 µs is added. Another point to be considered in the simulation is the initial state of the model. Different offsets between the various cyclic processes will lead to different results. A long simulation time is not sufficient to account for these effects. Therefore, the elementary models contain a setup process to set initial conditions. Based on this, a simulation can be easily started from a set of different initial condition. This set can be either given as fixed or stochastically determined in the model itself.

The following subsection details the sub-model underlying the transition *Ethernet boards of PLC*, and in particular the IO scanning behavior model.

4.3. Example: Ethernet boards of PLC model

The Ethernet board of a controller has two functions. The first one is the manipulation of network frame into a communication stack, and the second one is the IO scanning. Fig. 5 shows a simplified representation of the model *Ethernet board of PLC* underlying the transition of the same name in the *Global* model. The first function is modeled by the transition *Ethernet TCP IP communication stacks*, and is not detailed here. The second is modeled by the set of places and transitions that are not respectively of interface or of substitution.

The studied IO scanning cycle has two phases; the first phase is to send Modbus requests to a DIOD (server), on the left from bottom to top, and the second to receive Modbus responses, on the right from top to bottom. It also includes a set of places and transitions to initiate IO scans at the right time, and a place modeling one CPU resource per board, in the middle.

Sending of request is a two steps action. The first corresponds to the bottom left transition, *encapsulate data into modbus frames*, and performs the building of Modbus frames containing data address and values and instructions (read data or write data) for each DIOD (server) scanned by each controller (client). The second step – modeled by the transition *send Modbus requests* consists – of sending the request at the right time. This time is given by the transition *start new scans* which produces tokens in the place *servers to request* only if the former scan is completed (and if all responses from scanned servers arrived, i.e. if the corresponding token is present in the place *completed scans*) and also if the default scanning period has elapsed (if the corresponding token is available in the place *enabled scans at time T*). On the right part of the Petri net, similar actions are done for receiving Modbus frames and extracting data from them. The received responses are stored in the place *servers having responded* to check the completion of scans and enable new scans only in this case. There is one token of color Ethernet board in the place *CPU resources* for each Ethernet board, and each action (encapsulate data into frame, send requests, receive responses, extract data) needs this resource. This resource place enables to take into account the influence of load on delay.

Before simulations could be performed, this generic model has to be instantiated for a given architecture to be evaluated as presented in the next section.

5. Instantiation of the generic model

The instantiation is carried out on the architecture presented in Fig. 6. It is composed of one controller

(PLC1), three switches (SW1 to SW3) and eleven DIODs (DIOD1 to DIOD11). Each component has an unique and absolute identifier (positive integer) in the architecture. These identifiers are given on Fig. 6, as 0 for the CPU of controller and 60 its Ethernet board. The response time to evaluate here is the delay between input I1 and output O1 occurrences, PLC1 executing a program including the instruction "O1 equals I1". The image of I1 value in the controller as well as the value of O1 in the process is updated cyclically every 5 ms.

To instantiate the model for this architecture, the initial marking of some places and tokens have to be set up. For instance, on the model presented Fig. 5, it concerns the places *CPU resources*, *enabled scans at time T* and *completed scans*. The initial marking of these three places is the same and it is one token per Ethernet board on the architecture, colored as Ethernet boards identifier. In this example, there is only one Ethernet board identified by *60*, and so the initial marking of the three places is for each one token *60*. At this point, the Ethernet board of the studied architecture is set up, identical customization must be carried out for each other component present in this architecture.

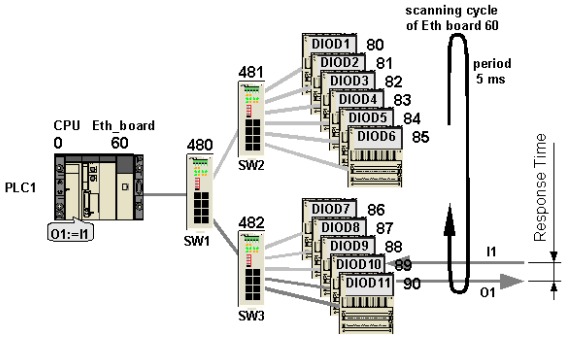

Figure 6. Example of an Architecture.

When they are all correctly set up, connections among all components in devices and among devices in the network must be modeled. In order to get a generic model structure, these connections are modeled by the tokens representing events or frames. A color has been defined for event token which is a set of:

- an event identifier,
- a source component identifier,
- a destination component identifier, and
- an "event route" which is a list of component identifiers.

The event identifier is incremented for each occurring event (input) that enables to follow it until its consequence on the process. The route in event token, called "event route", concerns only automation components. One event route [89, 60, 0, 60, 90] is used in this example as shown Fig. 7 b. For instance, the first event token, which have the identifier (id) 1, is (1, 89, 90, [89, 60, 0, 60, 90]).

Tokens representing frames have not exactly the same structure:

- a source component identifier,
- a destination component identifier,
- a "frame route" which is a list of component identifiers, and
- a list of events encapsulated in the Ethernet frame.

The "frame route" concerns only Ethernet components. Figures 7 c and 7 d show the two used frame routes ([89, 482, 480, 60] and [60, 480, 482, 90]) in this example. The first one is for the first network crossing, from DIOD 89 to the Ethernet board of the controller, and the second one is for the second network crossing, when data is sent from controller to DIOD. One event token in the model corresponds to one input or one output while one DIOD groups several inputs and outputs. For that reason, a frame token addressing a DIOD encapsulates the list of all related events. The "route" term of the two colors describes the remaining route in the network. So, a frame between DIOD 89 and switch 482 which should arrive at the Ethernet board 60 is modeled by the frame token containing the previous event token, (89, 60, [482, 480, 60], [(1, 89, 90, [60, 0, 60, 90])]), where the first item of each route is the next component to cross.

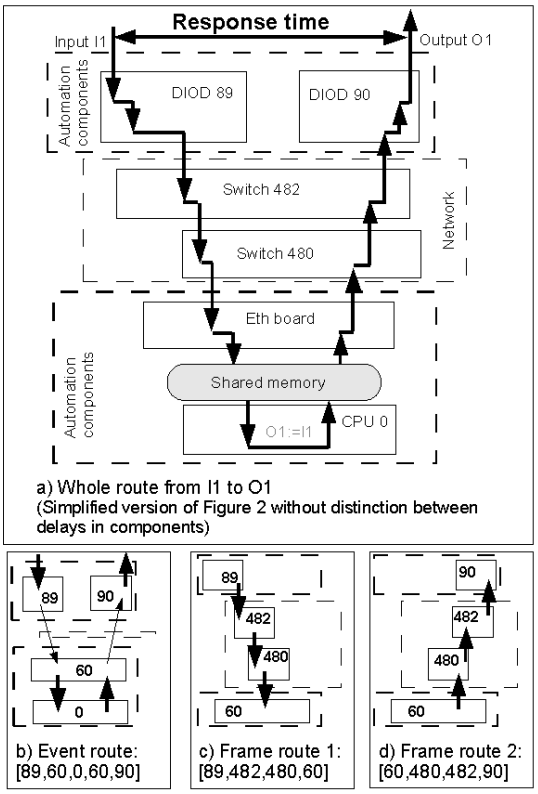

Figure 7. Routes for events and frames.

6. Case Study

6.1. Controlled sub-process

The previously presented control architecture (Fig. 6) has a part of its function dedicated to control of a linear transfer system. The sub-process considered is composed of a trolley, a motor and a position sensor (Fig. 8) and the particular function studied is to stop the trolley at the position set point (to stop motor by acting on O1) when it is detected by the sensor (change of value of I1). In this stop position, the part transferred by the trolley must be picked by a rotary gripper. This operation allows a deviation of 0.8 mm around the stop position. As the trolley has a constant velocity of 100mm/s, the effective stop position relative to the sensor position is directly related to the response time of the control architecture, and so this position is variable. The sensor position can be adjusted to have the position set point corresponding to the average value of all stop positions, so the average value of response time is not constrained. However, the range of response time must enable the trolley to stop in the acceptable range of 1.6 mm. The next subsection details the evaluation of the response time to check the adequacy of the architecture to the process requirements.

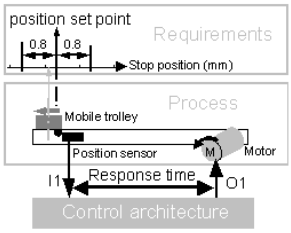

Figure 8. Application example.

6.2. Response time assessment using simulation

Simulation of the instantiated model has been carried out during 2 hours on a Pentium IV 2.4GHz PC for 10,000 delays representing about 100 seconds of the automation systems operation. The result obtained for the response time is presented on the topmost histogram in Fig. 10 with time in milliseconds sampled in 30 intervals of 0.5 ms on abscissa and percentage of response time measured in each interval on ordinate.

The histogram shows that the range of response time is 15 ms, and so the range of stop position of the trolley is 1.5 mm. Hence, the control architecture complies with the requirement for stop position of the trolley.

6.3. Adding new automation devices

A major feature of Ethernet with client/server protocol is to enable a high flexibility, and in particular to permit easy adding of devices and functions to an existing network. This feature enables to automate a new part of the process or to add a new sub-process with its control architecture without changing the existing parts. It is therefore of considerable practical interest to study the impact of such upgrades on the response time. In the following to upgrades to the of the previous architecture called “reference configuration” are analyzed.

In “Configuration 1”, an additional sub-process of the transfer system is automated. For this, another controller and seven DIODs are added to the reference architecture (Fig. 9a). To synchronize the controllers four DIODs (87, 88, 89, 90) are now shared, i.e. these DIODs are scanned by both controllers. The parameters for the second controller are the same as for the first one (CPU cycle of 2 ms and IO scanning cycle of 5 ms).

A major feature of using TCP/IP protocol is to increase interoperability. In “Configuration 2”, a PC is added as supervisor. It scans all the DIODs every 300 ms to have images of inputs and outputs values (Fig. 9b).

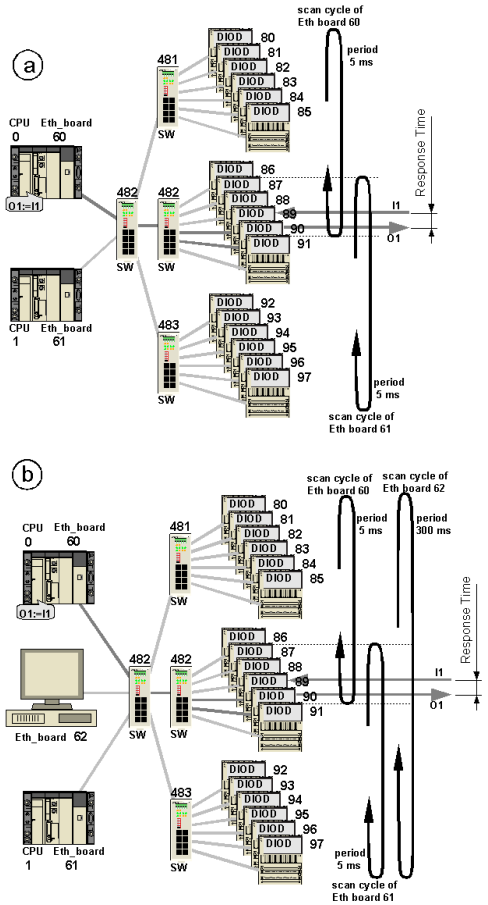

Figure 9. Two upgraded configurations.

The simulation results for the three configurations are given in Fig. 10. As expected, the response time increases when adding devices. In configuration 1, only a small increase is observed compared to the reference configuration. Indeed, the switches are far from overload, and the four shared DIODs can cope with two requests from the controllers in about 5 ms if necessary. On the contrary, in Configuration 2, the range increased to 20 ms. In this case, the supervisor scans all DIODs, so four DIODs are shared by three resources and from time to time they should reply to three requests which is not

possible within one controller scan cycle of 5 ms. Configuration 2 therefore does no longer fulfill the requirements of the controlled process.

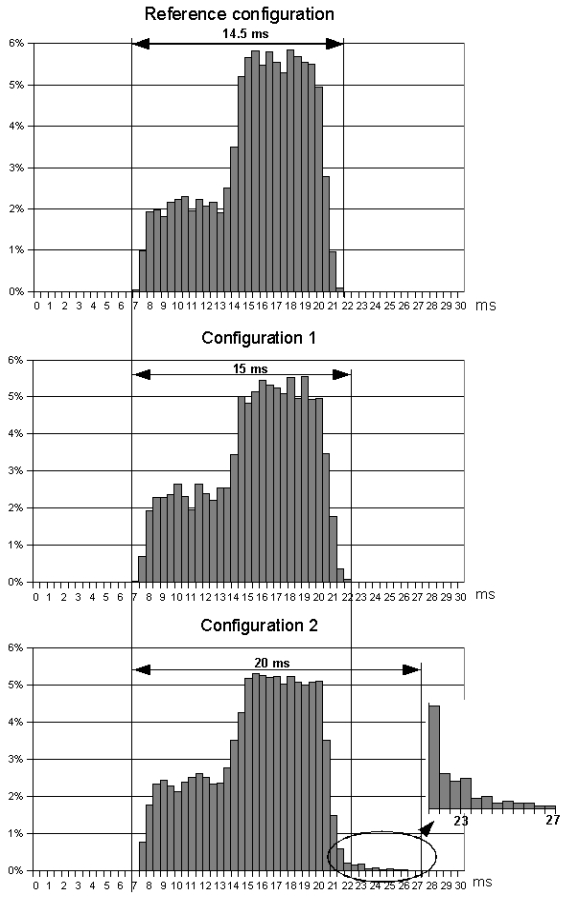

Figure 10. Histograms for response time in the three configurations.

7. Conclusion

Ethernet-based automation architectures using the generic TCP/IP client/server protocol increase flexibility and interoperability. However, these solutions do not offer guaranteed time performances. For the control of processes modeled as Discrete Event Systems, the major time characteristic of an automation system is its response time, defined as the delay between the occurrence of an event on the process and the occurrence of the corresponding reaction on the process.

To evaluate this delay, a simulation model designed in Hierarchical Timed Colored Petri Net is proposed. This model is generic for a class of automation architecture distributed on switched Ethernet networks. This enables to assess response time of such architectures, and from this to determine whether the system fulfils the application requirements.

The obtained results will lead to extensions of the generic model in various aspects and to the derivation of simplified – more abstract – models.

References

[1] The Online Industrial Ethernet Book (Jan. 2006), [Online]. Available: http://ethernet.industrialnetworking.com/ieb/fieldbus.asp

[2] Beckhoff, (Jan. 2006), EtherCAT - Ethernet for Control Automation Technology [Online]. Available: http://www.ethercat.org/

[3] Siemens, (Jan. 2006), Industrial Ethernet [Online]. Available: http://www.automation.siemens.com/net/html

[4] Schneider Electric, (Jan. 2006), Transparent Ready [Online]. Available: http://www.transparentfactory.com/en/index.htm

[5] P. Ferrari, A. Flammini, D.Marioli, and A.Taroni, "Experimental evaluation of PROFINET performance", in Proc. *WFCS 2004*, Sept. 2004.

[6] G. Juanole, "Quality of service of communication networks and distributed automation: models and performances", in Proc. of *15th Triennial IFAC World Congress*, July 2002.

[7] J.-Y. Le Boudec and P. Thiran. "Network calculus: a theory of deterministic queuing systems for the Internet", In *Lecture Notes in computer science*, volume 2050, Springer Verlag, 2001.

[8] B. Ben Hédia, F. Jumel and J-P. Babau, "Formal Evaluation of Quality of Service for Data Acquisition Systems", In Proc. of FDL'05, 2005.

[9] N. Pereira, E. Tovar, and L. M. Pinho, "Timeliness in COTS Factory-floor distributed systems: what role for simulation?", in Proc. *WFCS 2004*, Sept. 2004.

[10] "Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications", IEEE standards 802.3, IEEE computer society, Mar. 2002.

[11] J. Jasperneite and P. Neumann, "How to guarantee real-time behavior using Ethernet", in Proc. *11th IFAC Symposium on Information Control Problems in Manufacturing,* Apr. 2004.

[12] "Transmission Control Protocol", RFC 793, Information Science Institute, University of Southern California, Marina del Rey, CA, USA, Sept. 1981.

[13] "Internet Protocol", RFC 791, Information Science Institute, University of Southern California, Marina del Rey, CA, USA, Sept. 1981.

[14] Modbus application protocol specification v1.1a, (June 2004), [Online]. Available: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1a.pdf

[15] S. Vitturi, "Some features of two fieldbuses of the IEC 61158 standard", *Computer Standards & Interfaces*, vol. 22, no. 3, pp. 203–215, Aug. 2000.

[16] S. Vitturi, "On the use of Ethernet at low level of factory communication systems", *Computer Standards & Interfaces*, vol. 23, no. 4, pp. 267–277, Sept. 2001.

[17] J.-P. Georges, E. Rondeau, and T. Divoux, "How to be sure that Ethernet networks will satisfy the real-time requirements ?", in Proc. *ISIE'2002, IEEE International Symposium on Industrial Electronics,* July 2002.

[18] K.C. Lee and S. Lee, "Performance evaluation of switched Ethernet for real-time industrial communications", *Computer Standards & Interfaces*, vol. 24, no. 5, pp. 411–423, Nov. 2002.

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