POLITECNICO DI TORINO
Repository ISTITUZIONALE

System-Level Performance of an Automation Solution Based on Industry Standards

Original
System-Level Performance of an Automation Solution Based on Industry Standards / A., B., A., B., M., C., I. C., B., Durante, L., Hu, T., E., L., L., N., L., S., S., S., F., T., Valenzano, A., S., V.. - (2014), pp. 1-6. (19th IEEE Conference on Emerging Technologies and Factory Automation (ETFA) 2014).

Availability:
This version is available at: 11583/2572570 since:

Publisher:
IEEE Press

Published
DOI:

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

15 October 2026

System-Level Performance of an Automation Solution Based on Industry Standards

Andrea Ballarino[a], Alessandro Brusaferri[a], Marco Cereia[b], Ivan Cibrario Bertolotti[b], Luca Durante[b], Tingting Hu[b,c], Egidio Leo[a], Leonardo Nicolosi[a], Lucia Seno[b], Stefano Spinelli[a], Federico Tramarin[d], Adriano Valenzano[b], Stefano Vitturi[d]

[a] CNR – National Research Council, ITIA, Via Bassini 15, I-20133 Milano, Italy
[b] CNR – National Research Council, IEIIT, C.so Duca degli Abruzzi 24, I-10129 Torino, Italy
[c] Politecnico di Torino, DAUIN, C.so Duca degli Abruzzi 24, I-10129 Torino, Italy
[d] CNR – National Research Council, IEIIT, Via Gradenigo 6/b, I-35131 Padova, Italy
{andrea.ballarino, alessandro.brusaferri, egidio.leo, leonardo.nicolosi, stefano.spinelli}@itia.cnr.it, {marco.cereia, ivan.cibrario, luca.durante, tingting.hu, lucia.seno, federico.tramarin, adriano.valenzano, stefano.vitturi}@ieiit.cnr.it

***Abstract*— The flexibility and reconfigurability requirements of factories and manufacturing plants of the future can be partially met by adopting technologies and solutions already available for testing and experimentation. Openness and adherence to international standards are becoming increasingly important in modern distributed production and automation systems, especially when they have to cope with ever-increasing product differentiations and short product lifecycles. However, the increased flexibility and openness should not come to detriment of the system real-time characteristics. This paper deals with a pilot mechatronic architecture for agile transport systems, which has been specifically developed to enable the study of the aforementioned aspects in the framework of the "Factory of the Future" Italian flagship project. In particular, the paper focuses on possible bottlenecks and pitfalls at the operating system and communication levels, and provides preliminary indications on how to address or mitigate them by means of solutions already available on the market.**

Keywords—reconfigurable manufacturing systems, factory of the future, real-time performance, operating systems.

I. INTRODUCTION

To face new consumer-centered manufacturing paradigms, like mass customization and personalization, factories must be capable to adapt themselves in real time to continuously changing market demands. Therefore, modern automation systems shall be able to conjugate increasing complexity of controlled processes with agile reconfiguration and flexibility of manufacturing systems. Indeed, reliable and agile automation systems represent a crucial issue for competitiveness of modern manufacturing systems.

In such a context, new paradigms based on the distribution of control solutions onto a network of embedded components have been widely considered, thus enhancing the rapid design, modification, integration and reconfiguration of the resulting systems. The IEC 61499 standard [1], which defines function blocks for industrial process measurement and control systems, provides considerable support for developing complex distributed control applications. In fact, the organization of the control functionalities into a network of interconnected function blocks at application and sub-application layers provides an effective high-level view of the distributed application, supporting quick integration, deployment and reconfiguration of the production system control architecture.

Advanced factory data acquisition and aggregation represent the second major element to be considered, when addressing the increasing complexity of modern production systems as well as enhanced flexibility and re-configurability requirements. OPC Unified Architecture (OPC UA) [2] represents an important step ahead in combining specific needs of the automation industry with a standard interface for secure and more efficient information exchange of complex data. From this point of view, a major advance introduced in OPC UA is represented by the transition from the OPC original COM/DCOM technology to the cross-platform capable Web Service technology. These changes in OPC allow a brand new approach to information integration, namely the unification of several OPC data models (such as Data Access, Alarms & Events, or Historical Data Access, as a single set of services) and their extension to other domains such as manufacturing, production, maintenance and business applications.

Due to such potentialities, IEC 61499 and OPC UA have been considered as backbone technologies in the development of the automation solution for mechatronic products, such as the De-manufacturing pilot plant described in [3]. Nonetheless, as the complexity of the system increases, the distributed system framework could explode into many levels, so that evaluating and guaranteeing real-time performance of the overall system are not usually immediate tasks. The subproject "Generic Evolutionary Control Knowledge-based module" (GECKO) of the three-year Italian flagship project "Factory of the Future" is aimed at investigating these aspects and a number of enabling technologies that can be of significant help in pursuing the above-mentioned goals of reconfigurability, flexibility and rapid adaptability to changing production requirements goals for a new generation of factories and automation systems.

During the first year of the project, several different aspects of the GECKO architecture and capabilities have been

This work was partially supported by the National Research Council of Italy (CNR) in the framework of the flagship project "Factory of the Future", subproject "Generic Evolutionary Control Knowledge-based mOdule" (GECKO).

investigated, spanning from all-important security issues [4][5] to error handling characterization [6] and effect of jitter-reduction techniques on the error detection capability of popular fieldbus-based industrial networks [7], up to analyzing the effects of internal components behavior on the performance of the systems/applications that use the components themselves. This paper, instead, focuses on the real-time performance of a mechatronic pilot plant revolving around an industrial Ethernet network and, in particular, on how the underlying operating systems hosted in the plant controllers affect actuation delays and jitters. Moreover, suggestions are also given about mitigating jitters by means of an appropriate choice and configuration of the operating system.

The paper is organized as follows: Section II first describes the hardware and software architecture of the GECKO pilot plant, while Section III presents experimental results about the actuation jitter derived from the plant itself, which are further analyzed with the help of a simplified, laboratory test-bed. The analysis aims at both assessing the relative impact of different sources of jitter (namely, task activation and communication jitters) and understanding how they can be mitigated by a suitable configuration of the execution environment. Finally, Section IV contains some concluding remarks and ideas for future work.

II. System Architecture

The pilot plant considered in this paper is an integrated automation solution based on the adoption of IEC 61499 and OPC UA, which implements control and monitoring functions for an experimental mechatronic architecture aimed at agile transport.

A. Hardware Architecture

The transport system has been designed by composing a number of modular, Plug-and-Play mechatronic elements. Each element, in its turn, integrates dedicated sensors and actuators as well as the related control system. Furthermore, advanced hardware and software solutions have been adopted to simplify the integration of mechatronic elements within the overall system.

Actually, each transport module has been designed to support two main (straight) transfer services and possibly one cross-transfer service or more. Fig. 1 shows two possible example configurations for the transport module. Configuration 1 provides forward (F) and backward (B) transfer capabilities as well as a specific position with left and right cross-transfer capabilities. Configuration 2 extends Configuration 1 by integrating one additional cross-transfer element. Transport system modules can then be connected back to back to form the desired conveyor layout. The controller cabinet of the transport modules has been designed to support comparative laboratory benchmark tests of alternative hardware architectures, either centralized or distributed, such as legacy (e.g. Siemens, Rockwell Automation) IEC 61131-compliant devices or open source PC-based control solutions. To such an aim, the input/output signals of each conveyor module have been connected to a dedicated Moxa ioLogik E1214 Ethernet remote I/O module, as shown in Fig. 2. In this way, the integration and testing of new controllers do not require signals re-cabling, waste of time and additional effort and costs.

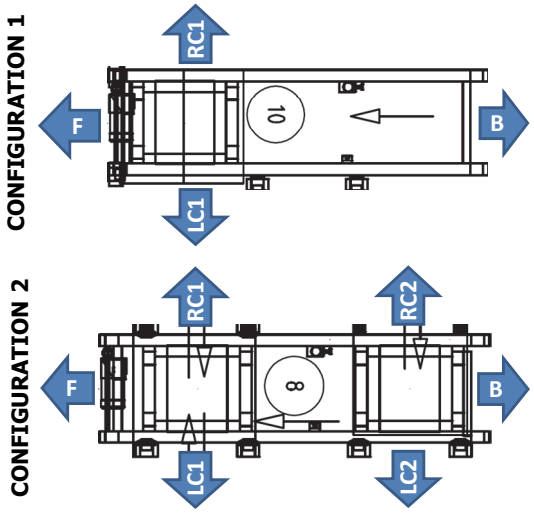

Fig. 1. Transport module configuration examples.

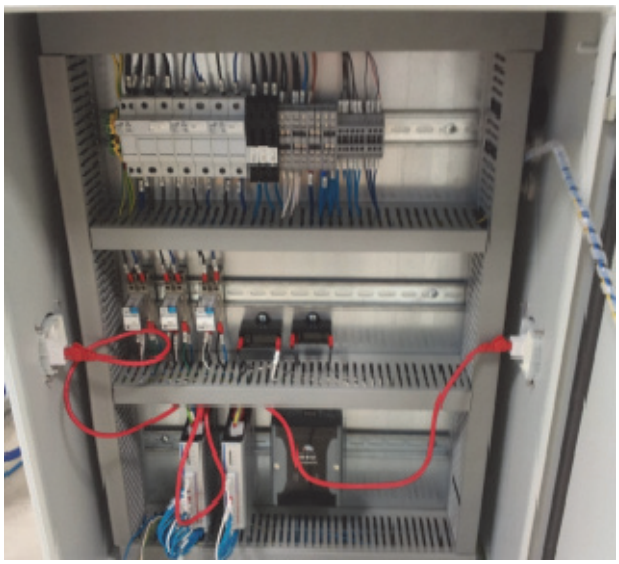

Fig. 2. Control module cabinet.

In the first pilot system set-up, in order to guarantee openness, interoperability, and international standard compliancy, embedded controllers from different brands have been considered for running the control applications and, in particular:

- Moxa IA261-I-LX embedded computer [8];
- ICP DAS Programmable Automation Controller LinPAC 5000 [9];
- Tecnint Leonardo PC BOX embedded PC [10].

The communication framework between the controllers has been implemented by selecting a suitable industrial Ethernet-based communication technology (Modbus TCP). Indeed, the

adoption of such technological solution guarantees the resiliency and network security of traditional fieldbus solutions, as well as the improved bandwidth, open connectivity, and standardization that Ethernet provides.

In order to properly optimize synchronous data access, Moxa EDS 405A Industrial Ethernet switches have been considered, which provide suitable support for multicast control (IGMP Snooping), Quality of Service (QoS), and virtual LANs (VLANs) configurations. Furthermore, in order to obtain a reasonable tradeoff between the robustness of the communication network and the easiness of cable connection, a ring topology has been implemented and one Moxa EDS switch included in each cabinet to simplify the integration of cross translators when reconfigurations of the transport system are needed. Such a solution also enables the definition of QoS policies, so that the real-time information exchange at the conveyor module layer (that is between components inside each conveyor module) occurs at the maximum priority, while the communication at the system level (that is between conveyor modules and/or between conveyor modules and plant machines) happens at a priority immediately lower than the highest one.

The industrial Ethernet network also supports service-oriented, OPC UA-based interoperable data exchanges between automation system components (up to the HMI, SCADA and MES layers) which are managed at a lower priority as defined by the QoS policies.

B. Software Architecture

All embedded controllers share the common feature of using the Linux operating system as a base to execute an IEC 61499 control solution, which has been implemented by means of the ISaGRAF workbench [11]. This environment has been chosen for different reasons. By supporting IEC 61499, such an environment acts as a sort of backbone for the overall application development, spanning from the design to the implementation and validation phases. Furthermore, the distributed hardware architecture can be defined by the proper assignment of resources (virtual PLCs) to each network device. The overall control application (CA) can then be designed as a single program. Furthermore, CA can also be generated through hardware-independent C code, which can be executed by a virtual machine running on the target device (i.e. industrial PC, embedded controller and so on). Portability and scalability of the developed application are also guaranteed in this case.

The transport configuration considered for the pilot plant implementation includes 15 conveyor modules configured as shown in Fig. 1. The control application has been designed and developed by foreseeing a composite IEC 61499 function block for each conveyor module, which encapsulates the structured control logic. Function blocks have then been downloaded to the dedicated hardware controllers in the distributed architecture. As mentioned before, the GECKO experimental plant includes a Modbus TCP real-time communication layer, and an OPC UA-based interoperability layer for flexible high-level information exchange, which makes use of the software development kit (SDK) provided by Unified Automation [12]. Actually, each automation component (i.e. conveyor module) includes an embedded OPC UA server, which works in cooperation with the ISaGRAF soft PLC firmware to provide run-time data access. All servers together offer to the applications a general data abstraction layer, which is independent from the underlying control technology. A main advantage of this choice is that the automation technology (adopted for controlling the plant) can be dynamically reconfigured during the whole plant lifecycle. At the same time data classes, which are designed to structure raw data of the automation system, can be kept unchanged or quickly readapted to different installation scenarios in the worst case, independently from the underlying control technology adopted.

In the experimental plant case, each automation component exposes one specific pallet transport service (that depends on the conveyor module configuration) and additional services devoted to the management and monitoring of task parameters that can be configured by either human-machine interfaces (HMIs) or the supervisory control and data acquisition (SCADA) system. With such architecture, a node is dynamically added to the server address space for each data class instance included in the hardware component that exposes the service. The address space nodes are based on an object-oriented approach thanks to the OPC UA dedicated support. Therefore, any change in a node class description is automatically reflected in each instance within the address space.

III. Analysis and Architectural Improvements

A. Performance Issues in the Pilot Plant

As a first laboratory experimental scenario, standard Linux kernels have been installed on the configurable hardware controllers, so as to obtain a preliminary evaluation of the system performance, when no particular care is paid to the real-time behavior of software components. To this purpose, a dedicated test program has been developed and integrated within the conveyor control software to better identify the jitter between the controller and the remote input/output signals implemented in the distributed hardware architecture. In particular, a square wave with period equal to 40 milliseconds has been generated on a plant Moxa ioLogik E1214 free channel and measured by means of an oscilloscope during the conveyor system operation.

The average value measured for the jitter was about 20 milliseconds, thus motivating the use of a dedicated test-bench scenario for analyzing possible improvements, which can be obtained by introducing real-time extensions to the controller operating systems. From a general point of view, this kind of practical result confirms the validity of the all-important question on how much the Linux operating system is able to support the concurrent execution of tasks with very different characteristics. In particular, the ISaGRAF-based control application, the OPC UA server, and the operating system tasks must be hosted on the same physical machine and still meet their timing requirements. It is worth noting that, at least for the third class of tasks listed above, this aspect is made even more difficult to evaluate because the operating system's processes are not under control of the application programmer and cannot be modified or removed at will.

In order to better analyze the shortcomings identified directly on the pilot plant, further experiments have then been performed on a simplified laboratory test bench, based on commercial instead of industrial-grade equipment, as discussed in the next sections.

B. Experimental Test Bench

The contribution of the underlying operating system to the actuation jitters highlighted in the previous section was studied by considering the raw performance of Linux, for what concerns the activation jitter of a periodic real-time task, while the Modbus TCP network communication jitters were evaluated by means of synthetic test programs. To this purpose, a prototype Modbus TCP master/slave communication system has been developed, which is based on the following hardware and software components:

- Two Intel-based PCs running at a CPU frequency of 2.66GHz on the slave side and 3.10GHz on the master side.
- The open-source FreeMODBUS Modbus TCP protocol stack [13] on the slave side and its commercial counterpart [14] on the master side.

Even if both CPUs offer dual-core support, only a single core has been enabled on the master side. This ensures that interfering loads are scheduled on the same CPU and are readily visible. On the slave side, the CPU has been left running to its maximum performance level, because the aim of this test is evaluating the performance of the master, and hence, any jitter due to the slave side should be maintained as negligible as possible. When the test software is considered, the master application cyclically sends a Write Single Register Modbus command, with a cycle time of 100 ms. At cycle i, as soon as the master task is activated, a timestamp $t_{i,1}$ is taken, and the Modbus command is sent to the slave. The master waits for the slave reply and, as just as it is received, another timestamp $t_{i,2}$ is taken. Using these timestamps two performance indexes can be computed:

- The difference $A_i = t_{i,1} - t_{i-1,1}$ is the time elapsed between two subsequent activations of the master task. Ideally, it should be equal to the cycle period, so it is representative of the jitter suffered by the master task due to scheduler and activation latencies.
- The difference $D_i = t_{i,2} - t_{i,1}$ represents the end-to-end communication roundtrip delay between the transmission of the Modbus command and the reception of the corresponding answer returned by the Modbus TCP slave.

In order to minimize measurement noise, the Modbus slave application has been kept as simple as possible: it only sends replies to master requests, without performing any other activity.

C. Experiments on Standard Linux

The first set of results was derived using the standard Linux kernel version 3.2. In order to determine the sensitivity of the operating system to interfering lower-priority tasks, the evaluation has been carried out in two different load conditions:

- *Without interfering loads:* no tasks except those functional to the execution of the operating system are executed with the real-time test application.
- *With I/O load*: an interfering I/O-bound task is executed together with the real-time test application.

Experimental results are shown in Fig. 3 and Fig. 4, as well as in the upper half of Table 1. The figures plot A_i as a function of the cycle/sample number i, while the table contains summary statistics for the same data. Without interfering load, the activation jitter is within ±50 µs of the nominal time, thus confirming that the operating system behavior is acceptable in this case. All these results pertain to short-term experiments involving 10,000 samples and corresponding to about 15 minutes of experiment time, but longer-term experiments confirmed the same behavior. However, the scenario is radically different when an interfering I/O load is added to the system. Namely, the results plotted in Fig. 4 clearly show that, albeit with low probability (several samples out of 216,000, corresponding to 6 hours of experiment time), the activation time achieved by the standard Linux scheduler may even exceed the cycle time of the communication task. Even more importantly, the experimental results show that the activation jitter under I/O load represents a significant part of the overall jitter measured in the pilot plant and discussed in Section III.A.

TABLE I. SUMMARY STATISTICS OF A_i

Kernel Type	Samples	Load	[us]			
			Mean	Std. Dev.	Min.	Max.
Standard	10,000	None	99,999.6	15.0	99,952	100,053
Standard	216,000	I/O	99,999.6	887.5	98	448,861
RT patch	10,000	None	99,999.6	12.0	99,960	100,041
RT patch	216,000	I/O	99,999.6	19.0	99,955	100,044

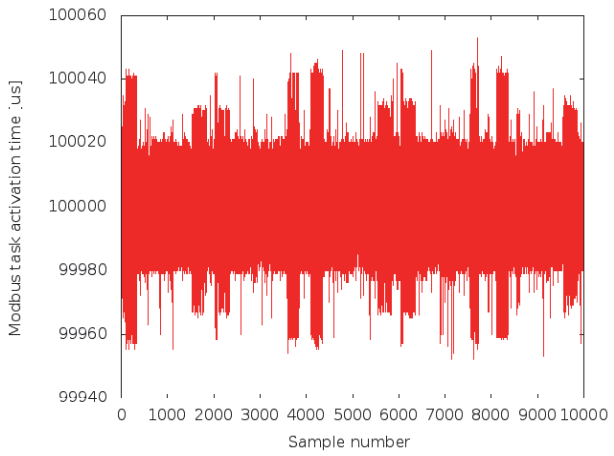

Fig. 3. A_i on Linux, no interfering load.

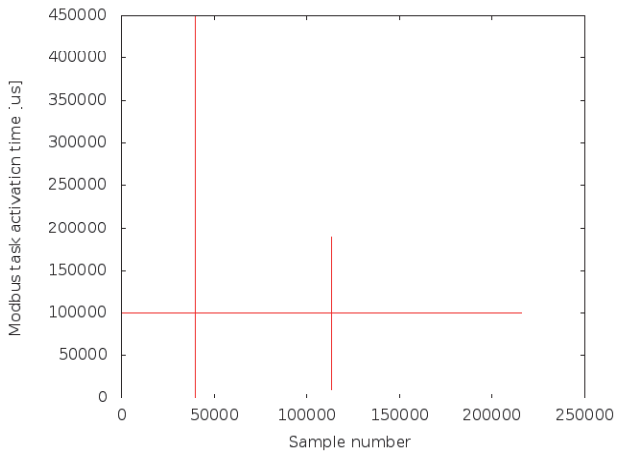

Fig. 4. A_i on Linux, with I/O load.

As the upper part of Table 2 shows, the same is true also for the end-to-end communication roundtrip delay D_i, which may also exceed the communication cycle time when an interfering I/O load is present.

D. *Use of a Real-Time Extension to Linux*

The large task activation and communication roundtrip jitters mentioned in the previous section are both due to shortcomings in the standard Linux operating system scheduler. In recent years, several solutions such as RTAI [15], Xenomai [16], and RT Patch [17][18] have appeared in the literature, proposing extensions to the Linux operating system to better support real-time applications. RT Patch, in particular, has been considered in this paper for two main reasons. First of all, unlike the other approaches mentioned above, it is based on the *single-kernel* approach, which simplifies system configuration and software development. Secondly, it is likely that the features provided by RT Patch will be included in the standard Linux kernel in the near future. This will simplify system deployment because virtually all off-the-shelf Linux software distributions will become capable of real-time execution in this way.

A new set of experiments was performed using the same test software discussed in the previous section but running, this time, on a Linux system extended with RT Patch. The summary statistics of the experimental results pertaining to A_i and D_i are shown in the lower half of Table 1 and Table 2, respectively. Results concerning the activation time A_i, clearly show that its jitter has become independent of the interfering I/O load, and its magnitude is now limited to about ±50 µs in all cases. This is further confirmed by the time traces of the activation time, plotted in Fig. 5 and Fig. 6. Very similar results have been obtained concerning the communication round-trip delay D_i. As also shown in the time trace plotted in Fig. 7 (with a significant I/O load), the observed worst-case jitter of D_i is limited to less than 500 µs, even in a long-term experiment which collected a total of 216,000 samples. The combination of those results shows that, at least in the test environment, the adoption of the RT Patch at the operating system level is quite effective to remove two significant sources of jitter that emerged in the pilot plant.

TABLE II. SUMMARY STATISTICS OF D_i

Kernel Type	Samples	Load	[us]			
			Mean	Std. Dev.	Min.	Max.
Standard	**10,000**	**None**	149.4	4.7	112	171
Standard	**216,000**	**I/O**	141.0	1,048.0	89	448,847
RT patch	**10,000**	**None**	139.8	6.3	124	184
RT patch	**216,000**	**I/O**	132.46	14.6	78	458

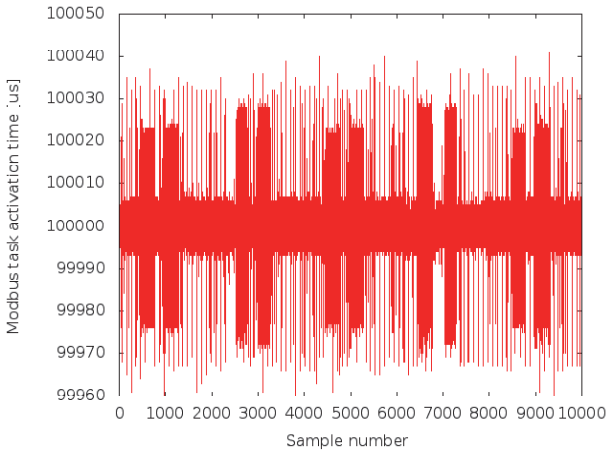

Fig. 5. A_i on RT Patch, no interfering load.

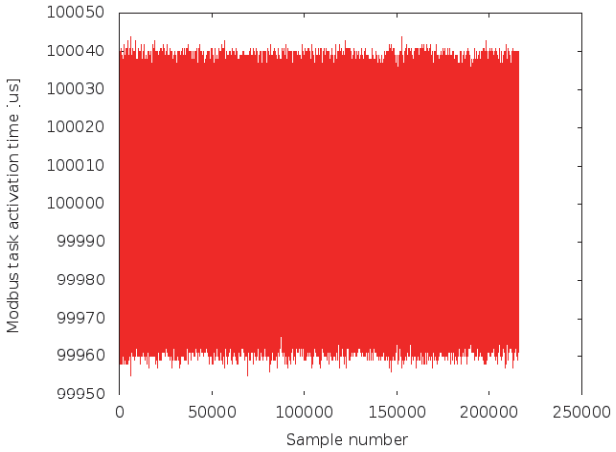

Fig. 6. A_i on RT Patch, with I/O load.

IV. CONCLUDING REMARKS

As modern automation systems are becoming more and more complex, in order to be able to conjugate high sophistication of controlled processes with agile reconfiguration and flexibility requirements, keeping the real-time performance of the overall system under control is of utmost importance in perspective. Unfortunately, this is not a straightforward task.

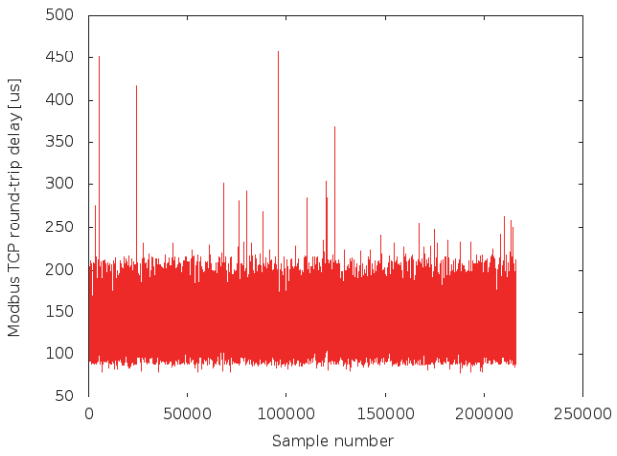

Fig. 7. D_i on RT Patch, with I/O load.

On the one hand, at the system design level, the adoption of the IEC 61499 standard [1] provides useful support for developing complex distributed control applications. On the other hand, OPC Unified Architecture (OPC UA) [2] enables secure and efficient information exchange of complex data through a standard, vendor-independent interface. However, once those facilities are deployed on the system embedded controllers, the issue of the coexistence of very different software tasks on the same hardware (namely, the IEC 61499 runtime, the OPC UA server, as well as other application and operating system tasks) becomes of concern. This is especially true if the embedded controllers run an open-source operating system such as Linux, which has not been originally specifically designed for real-time execution.

In this paper, the impact on the overall system performance brought by shortcomings in the operating system real-time scheduling mechanisms (namely, task activation and communication jitter) has been analyzed, and found to represent a significant part of the overall actuation jitter that affects an experimental pilot plant. Afterwards, a significant performance improvement has been verified to be achievable by means of relatively straightforward techniques applied at the operating system level. In particular, the adoption of the RT Patch extension of the Linux kernel was proven to be quite effective in limiting the worst-case amount of activation and communication jitter in a laboratory test-bed. Even more importantly, the same property holds true even if the operating system is subject to interfering loads representative of other software tasks running concurrently with the control application, thus paving the way towards an orderly coexistence of disparate tasks on the same embedded controller.

As a future work, the system-level optimization techniques just described, which have been implemented on a laboratory test-bed at present, will be evaluated in the context of the experimental pilot plant as well. It is worth mentioning that the careful characterization of network components represents a further, important issue that has to be addressed. Indeed, it is well known that the internal behavior of the components may heavily influence the whole system performance, leading to the introduction of both communication delays and jitter [19]. Although some preliminary activities have been already carried out in this context, more work has to be scheduled in the future, mainly based on practical measurements on the components that will be actually employed.

REFERENCES

[1] International Electro-technical Commission, (IEC), International Standard IEC 61499, Function Blocks, part 1-4, edition 1.0, Jan. 2005.

[2] W. Mahnke, S-H. Leitner, M. Damm, "OPC Unified Architecture", Springer, 2009.

[3] G. Copani, A. Brusaferri A, M. Colledani, N. Pedrocchi, M. Sacco, T. Tolio, "Integrated De-Manufacturing Systems as New Approach To End-Of-Life Management Of Mechatronic Devices", Proc. 10th Global Conference on Sustainable Manufacturing — Towards Implementing Sustainable Manufacturing, pp. 332–339, Oct. 2012.

[4] I. Cibrario Bertolotti, L. Durante, T. Hu, and A. Valenzano, "A model for the analysis of security policies in industrial networks," in Proc. 1st International Symposium for ICS & SCADA Cyber Security Research, pp. 66–77, BCS Learning and Development Ltd., Sept. 2013.

[5] M. Cheminod, L. Durante, L. Seno, A. Valenzano, "On the Description of Access Control Policies in Networked Industrial Systems", Proc. 10th IEEE International Workshop on Factory Communication Systems (WFCS), pp. 1–10, May 2014.

[6] G. Cena, I. Cibrario Bertolotti, T. Hu, and A. Valenzano, "Software-based assessment of the synchronization and error handling behavior of a real CAN controller," Proc. 18th IEEE Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1–9, 2013.

[7] G. Cena, I. Cibrario Bertolotti, T. Hu, A. Valenzano, "Effect of Jitter-Reducing Encoders on CAN Error Detection Mechanisms", Proc. 10th IEEE International Workshop on Factory Communication Systems (WFCS), pp. 1–10, May 2014.

[8] Moxa co., "IA261-I/IA262-I RISC-based fanless computers". Available online, at http://www.moxa.com/.

[9] ICP DAS co., "LinPAC-5000 Linux-based Programmable Automation Controller". Available online, at http://www.icpdas.com/.

[10] Tecnint HTE, "Leonardo PC BOX". Available online, at http://www.tecnint.it/.

[11] ISaGRAF inc., "ISaGRAF control software environment". Available online, at http://www.isagraf.com/.

[12] Unified Automation GmbH, "OPC UA Software Development Kit". Available online, at http://www.unified-automation.com/.

[13] C. Walter, "FreeMODBUS – Modbus ASCII/RTU and TCP implementation". Available online, at http://freemodbus.berlios.de/.

[14] Embedded Solutions, "Modbus Master". Available online, at http://www.embedded-solutions.at/.

[15] P. Mantegazza, E. Bianchi, L. Dozio, S. Papacharalambous, S. Hughes, and D. Beal, "RTAI: Real-time application interface", Linux Journal, no. 72, pp. 142–148, Apr. 2000.

[16] P. Gerum, "Xenomai—Implementing a RTOS emulation framework on GNU/Linux, 2004. Available online, at http://www.xenomai.org/.

[17] S. Rostedt, D. V. Hart, "Internals of the RT Patch", Proc. of the Ottawa Linux Symposium, vol. 2, pp. 161–172, June 2007.

[18] A. Garg, "Real-time Linux kernel scheduler", Linux Journal, no. 184, pp. 54–60, Aug. 2009.

[19] L. Seno, F. Tramarin, S. Vitturi, "Performance of Industrial Communication Systems: Real Application Contexts," IEEE Industrial Electronics Magazine, vol. 6, no. 2, pp. 27–37, June 2012.