Time-Complemented Event-Driven Architecture for Distributed Automation Systems

Cheng Pang, *Member, IEEE*, Jeffrey Yan, *Student Member, IEEE*, and Valeriy Vyatkin, *Senior Member, IEEE*

***Abstract*—Time-driven and event-driven control models are two fundamental design paradigms applied in distributed control systems for synchronizing decentralized activities. This paper proposes a unified architecture for combining both approaches. The combination offers the best of both worlds' properties, such as the expressiveness of event-driven programming and the determinism of time-driven logic. The foundations of this symbiosis are the IEC 61499 Function Block standard providing event-driven distributed control architecture and the IEEE 1588 Precision Time Protocol establishing the basis for highly-accurate time synchronization. The proposed time-complemented event-driven distributed control model aims at improving the modularity and flexibility of automation software with satisfactory control performance. The new control model has been compared with conventional centralized and distributed control approaches analytically and by simulation. The comparison results reveal that the proposed control model is efficient and flexible. Finally, a reference example has been used to demonstrate merits of the new approach.**

***Index Terms*—Distributed automation systems, distributed control, distributed time synchronization, IEC 61499, IEEE 1588.**

I. Introduction

TRADITIONALLY, control software has been executed in a cyclic way, recalculating control output after updating input. This stems from the computational implementation of continuous and discrete control systems. This can be clearly observed, for example, in programmable logic controllers (PLCs), which follow the classic cyclic scan paradigm of the IEC 61131-3 standard [1]. To improve a controller's reactive characteristics, the cycle duration, in general, should be as short as possible. Increasing the performance of computer-based control devices provides an opportunity to implement many independent control loops within the same control device. Based on dynamic characteristics of processes in a plant, it is possible to develop a time schedule of control loop invocations. Such a time-driven execution of control loops is a common solution in process control systems using time-driven tasks of PLCs.

Manuscript received October 25, 2013; accepted March 9, 2014. This paper was recommended by Associate Editor Z. Li.

C. Pang is with Luleå University of Technology, 971 87 Luleå, Sweden (e-mail: cheng.pang.phd@ieee.org).

J. Yan is with the University of Auckland, Auckland 1142, New Zealand (e-mail: jyan110@aucklanduni.ac.nz).

V. Vyatkin is with Luleå University of Technology, 971 87 Luleå, Sweden, and also with Aalto University, FI-00076 Aalto, Finland (e-mail: vyatkin@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2014.2326837

The development of networked control systems motivated the need for distributed process synchronization. Along with signal-based synchronization, time can also be used for synchronizing distributed processes. The implementation of this requires tight synchronization of clocks in distributed devices. This has been implemented in time-triggered networking protocols, such as TTP [2] for scheduling of communication sessions. However, these protocols are not commonly used at the application level. This is because not all the needs of a distributed control system can be fulfilled with prescheduling. Some processes may need to be synchronized dynamically by passing messages between distributed controllers. This message exchange is often modeled using event abstraction, which has given rise to the event-driven model of computation.

The IEC 61499 distributed automation architecture [3] is a well-known effort to introduce modern component-based software design into automation practice. Over the past decade, the effectiveness of the IEC 61499 standard in distributed control systems has been extensively studied in various applications, such as airport baggage handling systems [4], [5], manufacturing control [6] with closed-loop verification [7], [8], mechatronics [9], [10], building automation systems [11], machining [12], [13], process control [14], [15], and smart grids [16], [17]. These case studies have confirmed many advantages of IEC 61499 in terms of design and redesign efficiency and better interoperability and reusability [18]. The IEC 61499 architecture relies on the concept of an event-driven function block (FB), where control applications are represented as networks of FBs connected by event and data flows. The event-related nature of IEC 61499 explains why until recently time-based synchronization had not been properly addressed in that context. This limits developers of distributed systems who have chosen IEC 61499 as their development architecture.

This paper aims at providing designers of distributed systems a flexible design architecture that can compose time-driven and event-driven logics in reusable components. It takes advantage of the IEC 61499 distributed control architecture and the IEEE 1588 precision time protocol (PTP) [19]. The proposed approach is strongly motivated by the industrial trend of developing the intelligent mechatronic component (IMC) concept [20]–[22], which can include other IMCs in complex hierarchical assemblies. Some control actions in IMCs are purely reactive (i.e., event-driven) while others constitute sequences of actions that are best described by time-schedules. Lower-level actions can be triggered by commands from higher levels of the control hierarchy.

2168-2216 © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This paper is organized as follows. Section II reviews related works and defines the scope of this research. Then in Section III, a simplified sorting machine is used to illustrate the time-complemented event-driven (TCED) control model proposed in Section IV. In particular, a Petri net model is used to define and elaborate the dynamic semantics of the TCED control model. Performance metrics of the TCED control model are introduced in Section V. The qualitative comparison of different control models is presented in Section VI. The implementations of the TCED control model following the IEC 61499 standard are elaborated in Section VII. To confirm the benefits of the TCED control architecture, simulation trials have been conducted. The simulation framework and results are summarized in Section VIII. The paper is finally concluded in Section IX with future research perspectives.

II. Control Paradigms: Time-Driven and Event-Driven

The time-driven and event-driven models are two fundamental design paradigms of control systems. In general, with a time-driven model, autonomous progression of time triggers the execution of prescheduled control computations or periodic actions. In contrast, in an event-driven model, control computations or actions are triggered by occurrences of events. The temporal and synchronous nature of a time-driven model helps to guarantee deterministic behavior and required performance of control applications. On the other hand, the sporadic and adaptive nature of an event-driven model can more efficiently handle asynchronous external stimuli. Event-driven control is considered having better overall system performance than traditional cyclic scan-based control [23]–[25] due to its ability to react immediately to events that change the state of the controlled system. In other words, an event-driven model saves time and resources spent on communication and computation when no significant change in the system has occurred.

In the industrial automation domain, the time-triggered model has been partially reflected in PLC programming as the periodic task construct. The IEC 61499 standard, on the other hand, is based on an event-driven control model. While several studies reintroduced scan-based PLC semantics back to the IEC 61499 event-driven architecture for the purpose of backward compatibility [26], [27] and the migration of existing PLC code to IEC 61499 [28]–[30], no work is known to the authors on the implementation of time-driven controls in the IEC 61499 context. This paper attempts to fill this gap by combining event-driven and time-driven control models into a unified architecture.

The combination of generic time-driven and event-driven control models has been previously studied by Paoli and Tisato [31], where they proposed a unified control model. The core concepts of the unified control model are as follows.

1) Decoupling control model from controlled actions by introducing the concepts called reactive agent and control machine.
2) The integration of event-driven and time-driven control models using a planning machine and control machine.

A reactive agent (or agent for short) is an encapsulation of atomic control actions that it can execute. The actions to be

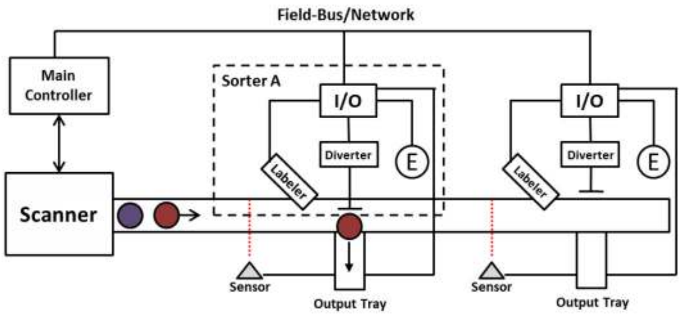


Fig. 1. Centralized control configuration of a generic sorting machine.

executed depend on an agent's status and the commands it has received. The commands are generated and dispatched to agents by a time-driven control machine (TDCM) based on a command timeline. This timeline is scheduled by an event-driven planning machine (EDPM) according to the external events received, for example, from the environment or other agents. This unified control model illustrated the complementary nature of event-driven and time-driven control semantics in a centralized configuration.

This paper proposed a TCED control architecture that extended Paoli's model to the case of component-based distributed control. Implementation of the TCED control architecture was based on the IEC 61499 framework with the clock synchronization realized by the IEEE 1588 PTP. The applications of the IEEE 1588 PTP to industrial automation systems were first introduced by Harris [32], where general comparisons between scan-based control and time-based control were theoretically discussed through a schematic example. The comparison results conformed to the claim of this paper that time-driven control has much better performance than scan-based control and can be used to reduce the necessity of event transmissions in event-driven control. However, Harris did not provide a concrete implementation of the control example and realization of the proposed time-driven control was unclear. In this paper, effects of control models on control performance were compared analytically. Moreover, concrete implementations of the IEEE 1588 PTP were provided with an illustrative control example.

III. Reference Example

The control architecture of an automation system can significantly affect its performance. An efficient control configuration can improve system performance, reduce costs, and simplify logic design. Impacts of control configuration on system performance will be illustrated using the generic sorting machine shown in Fig. 1. This machine can be used to sort items such as parcels, fruit, or baggage. In this machine, items are first scanned by a scanner and then transferred to corresponding sorters. Within each sorter, the items are labeled and diverted. Depending on the sorting criteria such as weight, size, and color, different label patterns and labeling processes can be applied and the number of sorters along the conveyor can also vary. The mechanical design of this sorting machine is highly modularized. For example, sorters can use different types of labelers such as cold-glue and self-adhesive labelers,

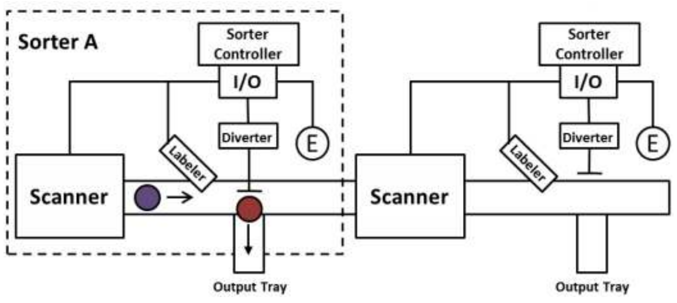


Fig. 2. Distributed control configuration.

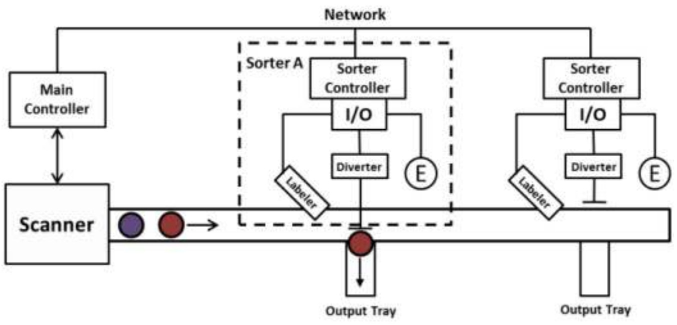


Fig. 3. TCED control configuration.

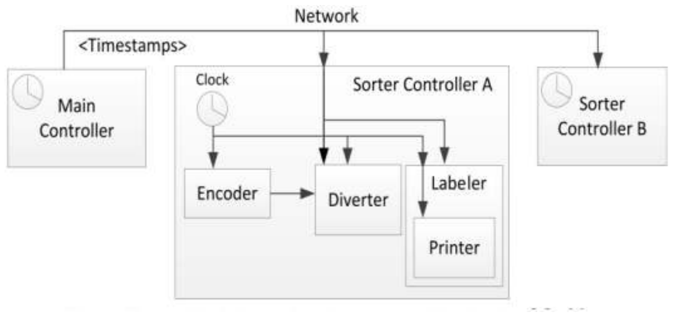


Fig. 4. Hierarchical control architecture of the sorting machine.

for different labeling processes. Labelers can again use various types of printers. Moreover, each sorter may include an encoder for adjusting action times based on actual conveyor speed. Thus, this sorting machine can be easily reconfigured to meet various sorting criteria.

The centralized control with PLC and remote I/O devices shown in Fig. 1 is common in material handling systems. In this configuration, a single controller and scanner are installed at the beginning of the conveyor with remote I/O devices attached to the sorters alongside. Based on the sensors' detection signals, the main controller can actuate the corresponding sorters to divert items. In other words, the main controller must monitor and manage operations of all sorters.

For comparison purposes, an idealized distributed control configuration was envisaged that would have performance close to perfect. In this configuration, it was assumed that each sorter could have its own scanner and controller as illustrated in Fig. 2. Each sorter's controller could make its sorting decision based on readings from its own scanner. Therefore, no communication between sorter controllers would be required.

IV. Time-Complemented Event-Driven Control Model

A. Overall Concept

The centralized control architecture shown above constrains the sorting machine's scalability. The distributed control configuration on the opposite extreme, eliminates the communication delays between the central controller and remote I/O devices. It thereby removes scalability constraints but at the cost of having a dedicated scanner and controller at each sorting point. The extra hardware increases costs that potentially go beyond practical limits. The TCED control configuration shown in Fig. 3 is a synergy of the two configurations above. It aims to achieve comparable control performance as the distributed control configuration while preserving similar costs as the centralized configuration. In the TCED control configuration, instead of having a dedicated sensor, each sorter now has a lightweight controller. The operations of sorter controllers are coordinated by the main controller based on a common notion of time.

The first target characteristic of the proposed hybrid control architecture is the modularity of control software that reflects its mechanical structure. Fig. 4 exemplifies such a modular and hierarchical control structure for the TCED configuration. In general, the control logic of each mechanical component is encapsulated into a control module. Simpler control modules can be composed to form more complex ones. For instance, the composition of the labeler and diverter control modules forms the sorter control module that again forms part of the final control system. With this hierarchical control pattern, to design the control for a sorter with a new labeling process, only the existing labeler control module must be substituted. This hierarchical composition of control modules can improve reconfigurability and reusability of control logic. However, in order to support time-driven logics located at differing levels of control hierarchy, a source of a real-time clock and a mechanism for conveying the clock readings across the hierarchies are required.

In the case of the sorting machine, the scanner and sorters operate asynchronously as conceptually depicted in Fig. 5. The main controller is responsible for analyzing scan results and making sorting decisions. Once a decision has been made, a time-stamped message will be dispatched to the corresponding sorter's controller. This message will inform the sorter controller of the scanned item's estimated time of arrival (ETA). There are two threads running on the sorter side and are as follows.

1) Thread A, with unit execution time D1, receives messages from the main controller and adds ETA to its local timetable.
2) Thread B, with unit execution time D2, repeatedly compares the sorter's current local time with the earliest time entry in its timetable. Once the time is reached, the sorter controller will actuate its labeler and diverter.

As a result, in this configuration the network traffic between the main controller and the sorter controllers is minimized to

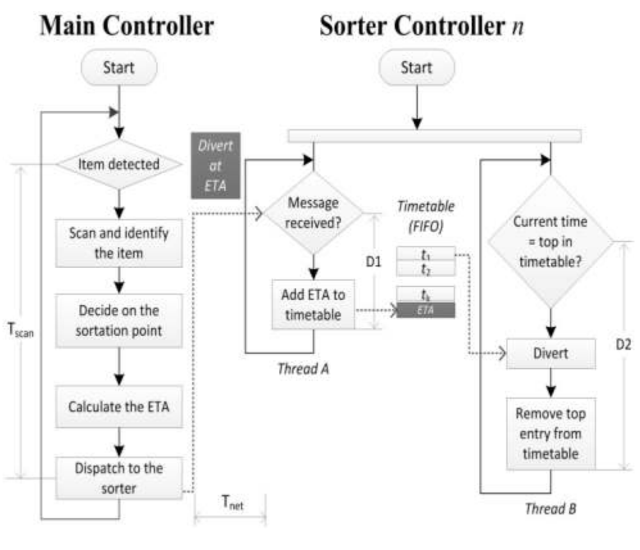


Fig. 5. Algorithms of main controller and sorter.

timestamp transmission and clock synchronization. One limitation of the proposed method is the assumption that ETA can be calculated based on plant dimensions, weight of the work piece, and motion speed. In reality, these factors may vary and impact the calculated ETA. To compensate for the possible physical impacts, the ETA can be repeatedly adjusted based on extensive measurements taken across the system such as encoder readings and other tracing information available in the system. The business practicality of this may vary from case-to-case. However, in principle it is a possible solution. By knowing the update rate of the ETA, limits of physical tolerance can be identified. At the current level of technology, the application of this TCED control approach may not be justified in many industrial applications. However, there are compelling examples, such as power plant control [33] and high-speed printing [34], where this time-stamped event-driven control approach is preferred and demonstrated [33], [35].

To give a better understanding of the TCED control model, its formal definition and dynamic behavior will be elaborated next. Performance metrics of centralized, distributed, and TCED control configurations will then be compared analytically in the following sections.

B. Formal Model of TCED Control Architecture

The formal model of the TCED control architecture is defined following a top–down approach. The basic construction unit in the TCED architecture is called a module.

Definition 1 (TCED Module):

$$m = \langle p, c, A, T, l, v\rangle$$

where

p	is an EDPM;
c	is an optional TDCM;
A	is a set of reactive agents;
T	is a set of constituent TCED modules, and $m \notin T$;
l	is a synchronized local clock;
v	is an optional action schedule.

An EDPM is a software component that reacts to asynchronous events. These events are represented as either module commands or action commands, which will be defined later in this section.

Definition 2 (EDPM): Given a TCED module, m, an EDPM, p, is a five-tuple defined as

$$p = \langle IMC, OMC, IAC, \theta, \tau\rangle$$

where

IMC	is a set of module commands this EDPM can receive;
OMC	is a set of module commands this EDPM can issue;
IAC	is a set of action commands this EDPM can receive;
θ	is a function that updates the action schedule v of m based on the received commands, $\theta\colon \langle \mathrm{IMC} \cup \mathrm{IAC}, v\rangle \mapsto v$;
τ	is a function that schedules the executions of constituent TCED modules based on the received module commands and current local time, $\tau\colon \langle \mathrm{IMC}, l\rangle \to \mathrm{OMC}$.

An action command, ac, is used to specify the action to be executed by an agent or to indicate execution results.

Definition 3 (Action Command):

$$ac = \langle e, R\rangle$$

where

e	is the action to be executed;
R	is a set of action parameters or execution results.

A module command, mc, is used to enable intermodule interactions and hierarchal propagation of commands, where upper-level action commands are further scheduled in lower-level modules.

Definition 4 (Module Command):

$$mc = \langle ac, ts\rangle$$

where

ac	is an action command;
ts	is a timestamp indicating the action's scheduled execution time.

A TDCM is used to dispatch action commands to trigger the actuations of corresponding agents based on the action schedule.

Definition 5 (TDCM): Given a TCED module, m, a TDCM, c, is a pair defined as

$$c = \langle \mathrm{OAC}, \mu\rangle$$

where

OAC	is a set of action commands this TDCM can issue;
μ	is a function that dispatches action commands scheduled in v of m, based on the local time l of m, $\mu\colon \langle v, l\rangle \mapsto 2^{\mathrm{OAC}}$.

A reactive agent (or agent for short), a, is an encapsulation of atomic control actions that it can execute. The actions to be executed depend on the agent's status and the commands it has received. During the execution, an action cannot be interrupted

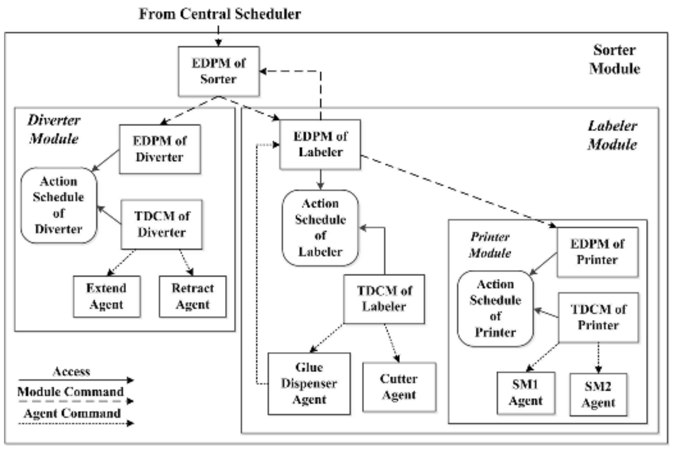


Fig. 6. TCED control module of sorter.

or preempted. Upon completion, the action's execution results can be observed.

Definition 6 (Reactive Agent):

$$a = \langle \text{IAC},\ \text{OAC}, F, S, \delta \rangle$$

where

IAC	is a set of action commands this agent can receive;
OAC	is a set of action commands this agent can issue;
F	is a set of atomic actions this agent can perform;
S	is a set of possible states of this agent, which defines the agent's status;
δ	is a function that decides the action to be executed by this agent and the action command to be issued after the execution $\delta{:}S \times \text{IAC} \rightarrow F \times \text{OAC}$.

Finally, an action schedule, v, is a set of chronologically ordered module commands.

Definition 7 (Action Schedule):

$$v = \{mc_0, mc_1, mc_2, \ldots, mc_n\}$$

where:

1) $mc_0, mc_1, mc_2, \ldots, mc_n$ are module commands;
2) $ts^{mc_0} \leq ts^{mc_1} \leq ts^{mc_2} \leq \cdots \leq ts^{mc_n}$: the timestamp of mc_0 is the earliest and timestamp of mc_n is the latest.

Fig. 6 presents the schematic composition of the TCED module for the sorting machine's first sorter, Sorter1. The formal model of this Sorter1 module, m^{s1}, can be specified as

$$m^{s1} = \left\langle p^{s1}, \left\{ m^{di}, m^{la} \right\}, l^{s1} \right\rangle$$

where

m^{di}	$= \left\langle p^{di}, c^{di}, \left\{ a^{\text{extend}}, a^{\text{retract}} \right\}, l^{di}, v^{di} \right\rangle$ is the TCED module for the diverter;
m^{la}	$= \left\langle p^{la}, c^{la}, \left\{ a^{\text{glue}}, a^{\text{cutter}} \right\}, \{m^{pr}\}, l^{la}, v^{la} \right\rangle$ is the TCED module for the labeler.

The TCED module for the printer, m^{pr}, can be again specified as

$$m^{pr} = \left\langle p^{pr}, c^{pr}, \left\{ a^{sm1}, a^{sm2} \right\}, l^{pr}, v^{pr} \right\rangle.$$

The overall dynamic behavior of a sorter is further illustrated using the place/transition Petri net model shown in Fig. 7. In particular, this Petri net model demonstrates the control flow of Sorter1. The meanings of places and transitions are listed in Table I. The control flow is initiated after an item is scanned (*T1*). Firstly, the EDPM of the main controller as a central scheduler analyzes the received scan result (*P2*). Then, module commands are dispatched to the corresponding sorter modules to schedule their actuations. In this particular case, the item is assigned to Sorter1 (*T3*), whose EDPM consequently analyzes the received module command (*P4*) and dispatches actuation schedules to its diverter and labeler modules (*P5*). Once the diverter EDPM has analyzed the received command (*P6*), it updates its action schedule (*P7*). Depending on the commands stored in the action schedule (*P8*), the diverter TDCM decides when to actuate the extend and retract agents (*P9*). As actions are scheduled based on synchronous clocks, actuations of the extend (*P10*) and retract (*P12*) agents are synchronized to the item's movement. *P11* and *P13* denote availability of the extend and retract agents, respectively. While the diverter module is processing its module command, the labeler module concurrently handles the module command it received. After the labeler EDPM analyzes the command (*P14*), it updates its action schedule (*P15*) and dispatches the action schedule to its printer module (*P22*). Similar to the diverter module, the labeler TDCM (*P17*) decides what its cutter (*P18*) and glue dispenser (*P20*) agents will do based on scheduled actions (*P16*). At last, the printer module's control flow exhibits the same pattern as that of the diverter module.

C. Implementation Considerations

The event-driven distributed automation architecture of IEC 61499 is considered in this paper as the main implementation environment for TCED. It is expected to be more efficient than traditional PLCs for the following reasons.

1) Message passing between PLCs is associated with substantial time overhead. As a result, events are delivered only in the next scan cycle. IEC 61499 applications are free from this overhead.
2) As the number of time-driven tasks in PLCs is limited and vendor-dependent, it is impossible to develop reusable software components that are clock-driven and can be placed in different levels of hierarchy as needed.
3) Reusability and reconfigurability of PLC programs are constrained due to the cyclic execution semantics as discussed in [36].

In contrast, the IEC 61499 architecture establishes a modular encapsulation of control logic in the form of a FB network, where control flows and data flows are identified by event and data connections respectively. The hierarchical structure and event-driven execution semantics of FB provide the fundamentals to develop the aforementioned TCED control model.

Moreover, the overall performance of event-driven control systems depends on event transmission time and event scheduling mechanisms. Thus, network congestion, packet loss, and runtime performance are the main factors causing response delays, which are proportional to the rate of event

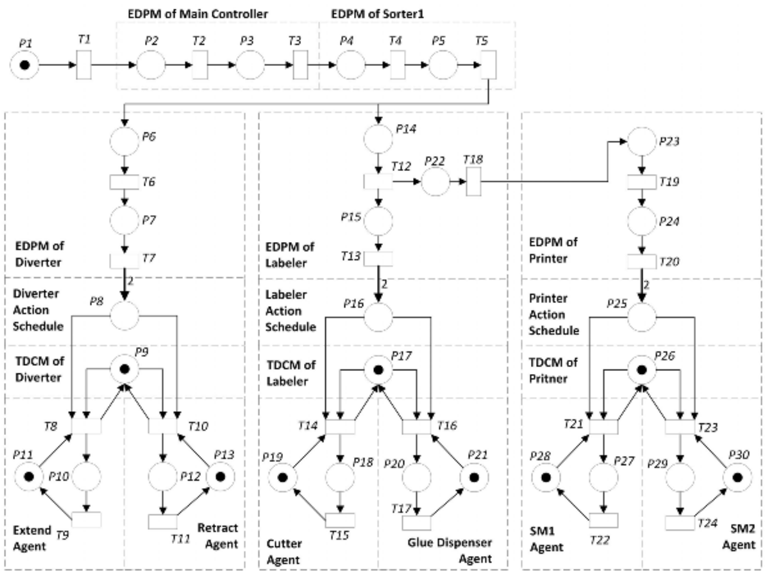


Fig. 7. Petri net model of the sorting machine's control system behavior.

TABLE I
INTERPRETATIONS OF PLACES AND TRANSITIONS

	Places		
P1	Scan result is ready		
P2	MainController_EDPM analyzes received commands		
P3	MainController_EDPM dispatches module commands		
P4	Sorter1_EDPM analyzes received commands		
P5	Sorter1_EDPM dispatches module commands		
P6	Diverter_EDPM analyzes received commands		
P7	Diverter_EDPM updates Diverter Action Schedule		
P8	Actions scheduled for Diverter		
P9	Diverter_TDCM dispatches command to agents		
P10	Diverter extends	**P12**	Diverter retracts
P11	Extend Agent is ready	**P13**	Retract Agent is ready
P14	Labeler_EDPM analyzes received commands		
P15	Labeler_EDPM updates Labeler Action Schedule		
P16	Actions scheduled for Labeler		
P17	Labeler_TDCM dispatches command to agents		
P18	Cutter actuates	**P20**	Dispenser actuates
P19	Cutter Agent is ready	**P21**	Dispenser Agent is ready
P22	Labeler_EDPM dispatches module commands		
P23	Printer_EDPM analyzes received commands		
P24	Printer_EDPM updates Printer Action Schedule		
P25	Actions scheduled for Printer		
P26	Printer_TDCM dispatches command to agents		
P27	SM1 actuates	**P29**	SM2 actuates
P28	SM1 Agent is ready	**P30**	SM2 Agent is ready
	Transitions		
T1	Start		
Others	Firing on completion of appropriate actions when corresponding resources for next actions are available.		

signal generation in the system. As control logic becomes complicated and more event signals must be transmitted and processed per time unit, the impact of these delays will be significant for applications demanding highly efficient real-time performance and accuracy. Time-driven logic can be used to reduce the necessity of event generation by simultaneously coordinating operations of distributed controllers in a synchronous way without frequently exchanging event-based messages. The IEC 61499 implementation of time-driven logic based on hierarchical timestamp passage will be further discussed in Section VII.

V. Performance Metrics

In order to analytically compare the performance metrics of different control configurations, a set of notions are defined. For each control configuration, a simplified mathematical performance model of the sorting machine is then built.

The sorting machine is considered in steady-state operation, which is defined as when the change in output throughput has reached equilibrium. In addition, an assumption is made that when an item enters the system, the travel time from scanner to target sorter is greater than the time that the scanner needs to scan an item. The following parameters will be utilized in the performance model.

1) T_{scan} is the scan time per item.
2) W is the sortation delay: the minimum time between two successful sorting actions including the time for detection, computation, labeling, and diverting, which will be further elaborated for each configuration described later in this section.
3) R is the steady-state throughput rate in items per second.

It can be assumed that the system functions correctly

$$\text{when } W \le \frac{1}{R} \text{ and } R = \frac{v}{d_{\text{obj}} + d_{\text{spa}}} \tag{1}$$

where

v	is the velocity of conveyor belt;
d_{obj}	is the length of one item;
d_{spa}	is the spacing between two items.

From (1), the upper bound on speed of the conveyor belt can be derived as follows:

$$v \le \frac{d_{\text{obj}} + d_{\text{spa}}}{W + T_{\text{scan}}}. \tag{2}$$

W can be represented as a sum of the following components reflecting hardware properties.

1) T_{act} is the physical actuation time a sorter takes to push an object to an output tray.

2) T^c_{alg}, T^d_{alg}, T^t_{alg} are the respective execution times of sortation algorithms for the centralized, distributed, and TCED configurations.
3) T_{io} is the update time of I/O configuration (i.e., I/O scan cycle).
4) T_{net} is the worst-case delay time for a message to travel between the controller and the furthest remote I/O device.

Assuming that scanning is sufficiently fast, i.e., $T_{\text{scan}} \ll W$, then (2) transforms to $v \le \frac{d_{\text{obj}}+d_{\text{spa}}}{W}$.

A. Centralized Control With Remote I/O Devices

In the centralized scan-based architecture as shown in Fig. 1, the sortation delay is calculated as

$$\begin{aligned} W_{\text{c\_sb}} &= T_{\text{io}} + T^c_{\text{alg}} + T_{\text{io}} + T_{\text{act}} + 2T_{\text{net}} \\ &= 2\,(T_{\text{io}} + T_{\text{net}}) + T^c_{\text{alg}} + T_{\text{act}}. \end{aligned} \tag{3}$$

It is notable that W_{c_sb} includes elements dependent on the computational performance of PLC and fieldbus. In general, the communication time T_{net} in a fieldbus increases with the length of cable and more remote I/O devices added. Therefore, for a sorting system it is reasonable to assume $T_{\text{net}} = O(n)$, where n is the number of sorters as adding more sorters would also mean extending the fieldbus cable.

In an event-driven system, the corresponding sortation time will be

$$\begin{aligned} W_{\text{c\_ed}} &= T_{\text{net}} + T^c_{\text{alg}} + T_{\text{net}} + T_{\text{act}} \\ &= 2T_{\text{net}} + T^c_{\text{alg}} + T_{\text{act}}. \end{aligned} \tag{4}$$

The two systems above have the same algorithm execution time, T^c_{alg}, as they execute the same sortation algorithm. The best-case steady-state throughput rates that the centralized control configuration can achieve are: $R_{\text{c\_sb}} = 1/W_{\text{c\_sb}}$ for a scan-based system and $R_{\text{c\_ed}} = 1/W_{\text{c\_ed}}$ for an event-driven system.

B. Distributed Control

The sortation delay of the distributed control configuration in Fig. 2 is mainly determined by the same factors as that of the centralized control configuration. For a scan-based system, the sortation delay is

$$\begin{aligned} W_{\text{d\_sb}} &= T_{\text{io}} + T^d_{\text{alg}} + T_{\text{io}} + T_{\text{act}} \\ &= 2T_{\text{io}} + T^d_{\text{alg}} + T_{\text{act}}\ . \end{aligned} \tag{5}$$

And for an event-driven system, the delay is reduced to

$$W_{\text{d\_ed}} = T^d_{\text{alg}} + T_{\text{act}}. \tag{6}$$

It can be noted that now the item scan time explicitly influences the system's throughput as being a part of the sortation delay. Similarly, the algorithm execution time, T^d_{alg}, for both systems is identical.

C. Time-Complemented Event-Driven Control

The sortation delay for the TCED control configuration is

$$W_{\text{tced}} \ge T^t_{\text{alg}} + T_{\text{act}} \tag{7}$$

where T^t_{alg}, in the worst case, is the sum of both threads' unit execution time indicated in Fig. 5, i.e., $T^t_{\text{alg}} = D_1 + D_2$.

VI. Performance Analysis

The throughputs of the above configurations are compared in this section to determine the potential performance gains of using a particular configuration over another. There are five configurations to be compared.

1) Centralized scan-based (*c_sb*), where
$$R_{\text{c\_sb}} = \frac{1}{W_{\text{c\_sb}}} = \frac{1}{2(T_{\text{io}}+T_{\text{net}}) + T^c_{\text{alg}} + T_{\text{act}}}. \tag{8}$$
2) Centralized event-driven (*c_ed*), where
$$R_{\text{c\_ed}} = \frac{1}{W_{\text{c\_ed}}} = \frac{1}{2T_{\text{net}} + T^c_{\text{alg}} + T_{\text{act}}}. \tag{9}$$
3) Distributed scan-based (*d_sb*), where
$$R_{\text{d\_sb}} = \frac{1}{W_{\text{d\_sb}}} = \frac{1}{2T_{\text{io}} + T^d_{\text{alg}} + T_{\text{act}}}. \tag{10}$$
4) Distributed event-driven (*d_ed*), where
$$R_{\text{d\_ed}} = \frac{1}{W_{\text{d\_ed}}} = \frac{1}{T^d_{\text{alg}} + T_{\text{act}}}. \tag{11}$$
5) TCED, where
$$R_{\text{tced}} = \frac{1}{W_{\text{tced}}} = \frac{1}{T^t_{\text{alg}} + T_{\text{act}}}. \tag{12}$$

A. Performance Comparisons

Firstly, centralized scenarios are compared. As indicated in (8) and (9), given the same fieldbus length and the transmission time, T_{net}, the event-driven scenario offers a higher throughput rate due to the elimination of the I/O scan. This also applies when comparing the distributed scenarios as indicated in (10) and (11). It can be concluded that event-driven systems have greater throughput rates than that of scan-based systems.

As indicated in (9) and (11), the main factors differentiating the centralized and distributed scenarios are T_{net} and T_{alg}. As T_{net} is eliminated in the distributed scenario, to evaluate the throughput rates, T^c_{alg} and T^d_{alg} must be analyzed by comparing their control algorithms. The execution of control algorithms depends on the sequence of events that occurs during a sortation delay. This event sequence will be referred to as the critical path. By examining the critical path, the reactions to changes in system size and complexity for each configuration scenario can be determined.

A sample control algorithm for the centralized event-driven configuration is described as follows.

1) The scanner processes the passing item upon its entry to the conveyor and captures the destination data. The item's data are placed into a first-in-first-out (FIFO) queue along with subsequent items.
2) When the first sorter's sensor detects an item, the first entry of this FIFO queue is compared to determine if the item must be sorted at the current location. If it does, then its data entry is removed from the queue. Otherwise, its data entry will be passed to another FIFO queue representing the space between the first and the second sorters.

TABLE II
CRITICAL PATH FOR CENTRALIZED EVENT-DRIVEN CONFIGURATION

Task	Complexity and Time Required
Sensor detects item and notifies the central controller.	A single I/O scan is required for sensing and passing data down the fieldbus. This requires a constant time.
Central controller iterates through a lookup table to find out which sensors have detected the item.	Table size will be of size *k* (the number of sorters in the system); thus this time is proportional to the number of sorters in the system.
Sorter actuation signals are sent to I/O devices.	A single I/O scan is required for passing data down the fieldbus. This requires a constant time.
Mechanical actuation.	Actuation time is constant.

TABLE III
CRITICAL PATH FOR DISTRIBUTED EVENT-DRIVEN CONFIGURATION

Task	Complexity and Time Required
The controller compares the scan result against its own data and sends a message to activate the sorter. The I/O device is localized on the controller itself.	A single compare operation, which requires a constant time.
Mechanical actuation.	Actuation time is constant.

The critical path for this configuration occurs when an item has reached a sensor and requires sortation. If there are assumed to be k sorters in the system, then there are also k FIFO queues with item data. The decomposition of events within the centralized configuration is detailed in Table II.

For the distributed event-driven configuration, an item arrives at the scanner and is processed. If it is to be sorted at this sorter, then it is immediately sorted; otherwise it is passed along to the next sorter's scanner. The critical path for this configuration occurs just after an item is scanned and is required to be sorted at the current sorter. The decomposition of events within the distributed configuration is detailed in Table III.

It is important to highlight that the centralized controller contains an algorithmic component, which is dependent on the number of sorters in the system when it iterates through a lookup table. Contrarily, the distributed controller has no such dependency. Thus, T^c_{alg} has an algorithmic efficiency of $O(k)$, whereas T^d_{alg} remains constant as the system scales. Thus, for larger systems, $T^d_{\text{alg}} < T^c_{\text{alg}}$ and similarly $R_{\text{c\_ed}} < R_{\text{d\_ed}}$.

For the TCED configuration, the scanner initially processes an item and calculates a timestamp for when it will arrive at its destination sorter. This timestamp is sent to the destination sorter controller and is stored within a FIFO queue. Within the sorter controller, two processes run simultaneously.

1) The first entry in the timestamp queue is repeatedly compared with the current time provided by the synchronized time source. When these times match, the sorter is actuated.
2) Another process listens for the incoming messages to add new entries into the timestamp queue.

Only the first process affects the critical path when an item arrives at the sorter. Thus, the decomposition of events within the TCED critical path is detailed in Table IV.

TABLE IV
CRITICAL PATH FOR TCED CONFIGURATION

Task	Complexity and Time Required
The controller compares current time against the topmost timestamp in its queue and activates the sorter by setting the signal at the local I/O device.	Constant time.
Mechanical actuation.	Actuation time is constant.

Comparing the distributed event-driven and TCED configurations, neither of them has algorithmic efficiency that scales with system size. Therefore, T^t_{alg} approximates T^d_{alg}.

B. Influence of Clock Skew

Clock skew in this paper is defined as the phase-shift between clocks generated in separate distributed controllers throughout a control system. As PLC and I/O scan times reduce to sub-millisecond ranges, the clock skew from applications that are not time-synchronized will have a greater effect on overall system performance. For the TCED control model, the steady-state throughput is directly affected by the sortation delay as follows $R_{\text{tced}} = \frac{1}{W_{\text{tced}}}$. The clock skew ($T_{\text{skew}}$) can be introduced into this formula and affects the sortation delay. Also, depending on the number of time-scheduled actions (n) in the current reaction, the clock skew would affect each of these independently

$$R_{\text{tced}} = \frac{1}{W_{\text{tced}} + nT_{\text{skew}}}. \tag{13}$$

The effects of clock skew on throughput can be analyzed using as listed below.

1) R_{tced}: The steady-state throughput with synchronized clocks.
2) $R_{\text{tced(skew)}}$: The steady-state throughput with addition of some clock skew.

The performance difference can therefore be found as

$$\triangle R = \left|1 - \frac{R_{\text{tced}}}{R_{\text{tced(skew)}}}\right| = \left|1 - \frac{\frac{1}{W_{\text{tced}}}}{\frac{1}{W_{\text{tced}} + nT_{\text{skew}}}}\right| = \frac{nT_{\text{skew}}}{W_{\text{tced}}}. \tag{14}$$

This results in a simple linear relationship between clock skew and sortation delay. By using IEEE 1588 PTP, distributed clocks can be synchronized with an accuracy of less than 1 μs. Since current PLC scan times and I/O scans range in the order of milliseconds, these times are already an order of magnitude greater and should result in mostly insignificant effects of clock skew in a system.

VII. TCED CONTROL IMPLEMENTATION FOLLOWING IEC 61499

This section presents the TCED control implementation for a labeler in the reference sorting machine following the IEC 61499 standard. First of all, the IEC 61499 realization of IEEE PTP is presented, which provides synchronized time for time-driven logics. Then, mappings of TCED components, such

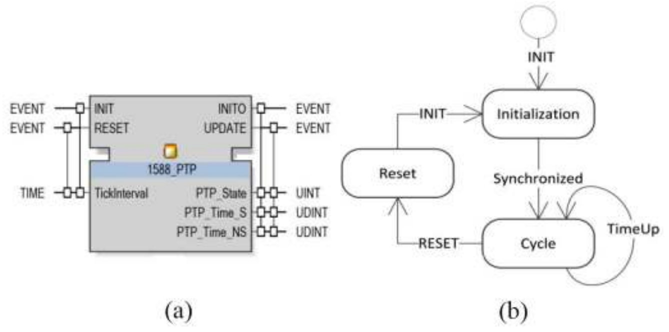


Fig. 8. 1588_PTP SIFB. (a) Interface. (b) State diagram.

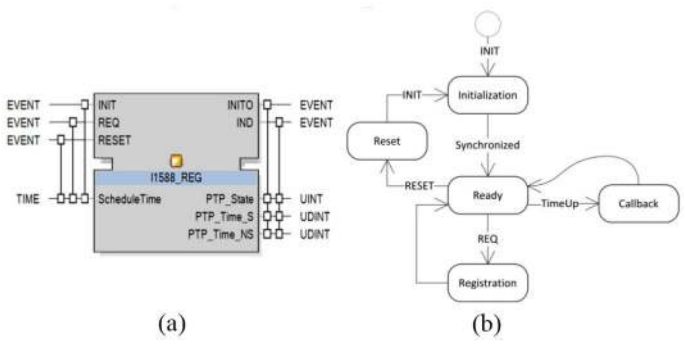


Fig. 9. 1588_REG SIFB. (a) Interface. (b) State diagram.

as commands, planning machines, and control machines, to IEC 61499 constructs are elaborated. Finally, the deployment configuration of the full control system is detailed.

A. Time Synchronization in IEC 61499

The IEC 61499 FB implementations of IEEE 1588 PTP have been investigated and discussed in previous works, where several service interface function blocks (SIFBs) were used to access the synchronous time. Both software-only [37] and hardware-supported [38] approaches were developed. The main reason of implementing the PTP as SIFBs is to have standalone and standard-compliant FBs providing synchronous time to trigger other FBs at the application level. In such a way, PTP SIFBs can be easily adapted into existing designs without affecting the event-driven execution model. Fig. 8 presents the 1588_PTP SIFB used to access hardware-supported PTP time.

This SIFBs main function is to periodically retrieve current PTP time and provide it to downstream FBs, where:

1) INIT initializes the PTP time synchronization process;
2) RESET restarts the synchronization process;
3) TickInterval specifies frequency of the UPDATE event;
4) PTP_State indicates whether local clock is synchronized or not;
5) PTP_Time_S and PTP_Time_NS hold the local time in seconds and nanoseconds since 1970, respectively.

Upon the INIT event, the 1588_PTP SIFB starts the PTP synchronization process. Once synchronized, the 1588_PTP SIFB will periodically emit UPDATE events based on its internal timer signal, TimeUp, and the frequency specified by the TickInterval input. This 1588_PTP SIFB is used as a pure time source for other FBs. On the other hand, the 1588_REG SIFB in Fig. 9 is used to provide the PTP time in a native event-driven way to the control logic.

The 1588_REG SIFB manages the PTP time synchronization process in the same way as the 1588_PTP SIFB. However, instead of periodically producing time signals, it registers the ScheduleTime on REQ events. Whenever a registered time matches current PTP time, the IND event will be emitted along with the updated PTP_Time_S and PTP_Time_NS to trigger the time-driven control logic in downstream FBs. The two SIFBs discussed in this section establish the basis for designing time-driven logic in FBs, which will be exemplified in the following sections.

B. TCED Commands

In the TCED control model, agents, EDPM, and TDCM interact with each other using one of two command types: module commands or action commands. In the FB implementation, each command type is represented as a specific combination of event and data signals.

The action commands realize TDCM-to-Agent and Agent-to-EDPM interactions. The IEC 61499 implementation of an action command consists of two parts.

1) An event signal identifies the action to be executed.
2) A set of optional data signals specifies the action parameters or execution results.

For example, in Fig. 10, for the Labeler_TDCM FB, the CUT and DISPENSE events implement the action commands for the cutter and dispenser agents.

The module command is used for EDPM-to-EDPM and EDPM-to-TCED interactions. Each module command is implemented as a signal combination as follows.

1) An event signal, which triggers the recipient EDPM and specifies the action to be executed.
2) A set of optional data signals specify the parameters.
3) A *ScheduledTime* data signal sets the scheduled time.

For instance, the Labeler_EDPM FB in Fig. 10 receives the module command for the labeling action through the signal interface LABEL+ScheduledTime, while the Pritner_TCED FB receives the module command for the printing action through the signal interface PRINT+ScheduledTime.

C. TCED Agent

An agent can be natively implemented as a basic FB, where:

1) the action commands that an agent can receive and issue are mapped to the basic FBs interface;
2) atomic actions are mapped to EC Actions;
3) possible states of an agent and action execution function are mapped to basic FBs ECC.

For example, in Fig. 10, the CutterAgent FB implements the cutter agent of the labeler, which receives an action command, CUT. Upon the action completion, it issues the DONE action command back to the Labeler_EDPM FB. This CutterAgent FB interacts with the sensors and actuators through its data signals, such as extended, extend, and so on.

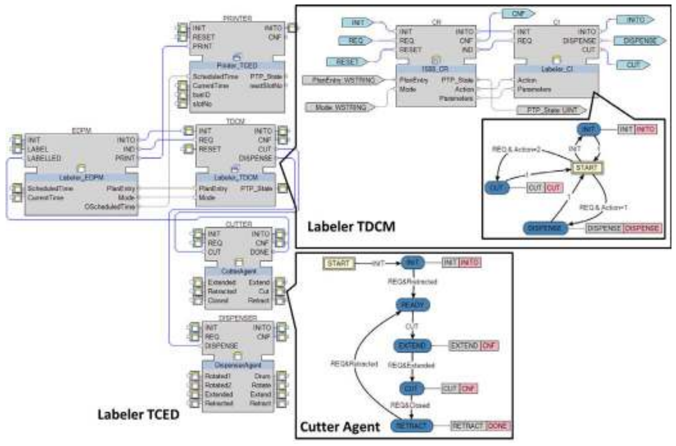
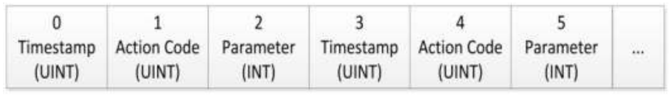


Fig. 10. Labeler TCED function block.



0 Timestamp (UINT)	1 Action Code (UINT)	2 Parameter (INT)	3 Timestamp (UINT)	4 Action Code (UINT)	5 Parameter (INT)	...

Fig. 11. Action schedule array.

D. Action Schedule, TDCM, and EDPM

The action schedule is realized as duplicated data in both of the TDCM FB and the EDPM FB due to the lack of global variables in IEC 61499. There are many data structures that can be used to store the action schedule, such as arrays, maps, and lists depending on the IEC 61499 runtime. Fig. 11 exemplifies a simple array data structure used to store the action schedule.

Each TDCM is implemented as a composite FB, whose event and data outputs specify the action commands that can be issued. The TDCM FB is composed as follows.

1) A command register (CR) SIFB that registers action schedule entries from the EDPM FB.
2) A command implementer (CI) basic FB that issues action commands to the corresponding agents according to the action schedule.

As exemplified by the Labeler_TDCM FB in Fig. 10, the signal interface REQ+PlanEntry+Mode forms the channel to receive action schedule entries from the EDPM FB. The Mode data signal can have four values: add, delete, replace, and reset indicating how to update the action schedule. The PlanEntry data signal uses a predefined syntax to specify actions and their scheduled times. For example, the plant entry below specifies that a Dispense action is scheduled at Unix time 1340083613 followed by a Cut action at 1342183613

"dispense@1340083613|cut@1342183613."

The received action schedule entries are passed to the 1588_CR SIFB. This SIFB is functionally similar to the 1588_REG SIFB shown in Fig. 9. In addition to the PTP time registration (as specified in the PlanEntry data input), this 1588_CR SIFB also manages the action schedule internally. At the time instances stored in the schedule, it triggers the CI FB to dispatch the corresponding action commands. For example, in the case of the Labeler_TDCM FB, when current PTP time matches the first schedule entry, the 1588_CR SIFB extracts the scheduled action's details for the Labeler_CI FB, where action name and parameters are specified in the action and the parameters data outputs, respectively. The Labeler_CI FB accordingly issues the DISPENSE or CUT event signals based on these details. The decoupling of command registration and implementation frees developers from repeatedly developing new SIFBs implementing the PTP time registration for every TDCM FB.

The EDPM FB is responsible for updating the action schedule based on the received commands. Fig. 12(a) illustrates the operation flow of EDPM FB. The action scheduling is demonstrated in Fig. 12(b). Upon receipt of a command at time t_0, the Labeler_EDPM FB calculates actuation times for the printer, cutter, and dispenser based on the estimated item arrival time (t_6), which is embedded in the received module command. Due to the synchronized local clock, it is able to precisely schedule the start and completion of each action and thereby optimize the overall operations. The EDPM FB interacts with the TDCM FB and TCED FB using two signal interfaces. As shown in Fig. 10, the Labeler_EDPM FB updates the action schedule stored in the Labeler_TDCM FB using the signal interface IND+PlantEntry+Mode, while the signal interface PRINT+OScheduledTime is used to schedule the actuation of the Printer_TCED FB.

E. TCED Module

Finally, the TCED module is implemented as the composition of the corresponding EDPM, TDCM, Agent, and other TCED FBs. For example, the Labeler_TCED FB shown in Fig. 10, and the Diverter_TCED FB are connected to form the Sorter_TCED FB as presented in Fig. 13.

F. Deployment Configuration of Sorting Machine

Fig. 14 illustrates the top-level deployment configuration of the FB control system for the sorting machine following its mechanical structure. The FB application is deployed to three control devices communicating over an Ethernet network. Within each control device, there is a 1588_PTP SIFB

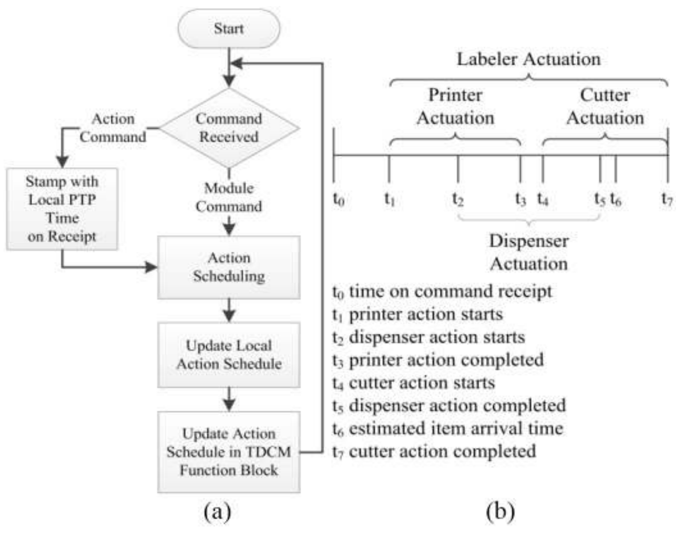

Start
Command Received
Action Command
Module Command
Stamp with Local PTP Time on Receipt
Action Scheduling
Update Local Action Schedule
Update Action Schedule in TDCM Function Block
(a)
Labeler Actuation
Printer Actuation
Cutter Actuation
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7
Dispenser Actuation
t_0 time on command receipt
t_1 printer action starts
t_2 dispenser action starts
t_3 printer action completed
t_4 cutter action starts
t_5 dispenser action completed
t_6 estimated item arrival time
t_7 cutter action completed
(b)

Fig. 12. EDPM FB. (a) Operation sequence. (b) Command scheduling.

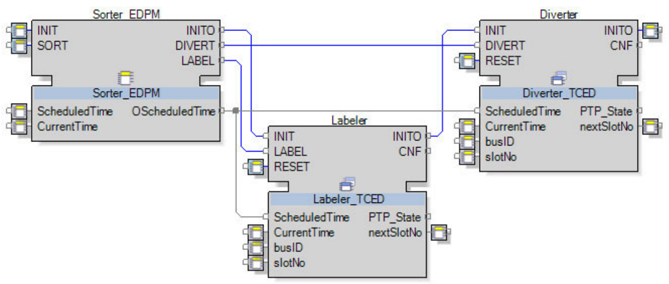


Fig. 13. Sorter module FB composition.

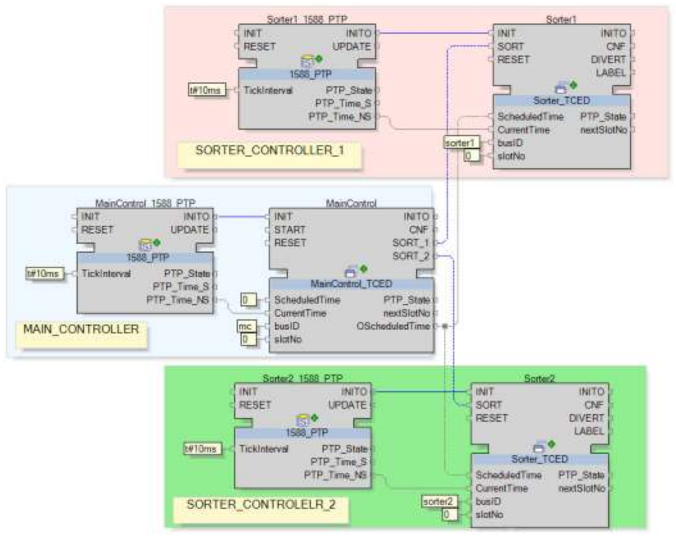


Fig. 14. FB control system for the sorting machine.

synchronizing its local clock and providing PTP time to the time-driven control logic. The MainControl_TCED FB analyzes scan results and decides at which sorter a scanned item must be diverted. Then a sortation module command will be issued to the target Sorter_TCED FB to instruct its actuation.

TABLE V
DIVERSION FAILURE RATES AT VARYING CONVEYOR SPEEDS

Conveyor Speed	Centralized	TCED
1.00 m/s	0%	0%
1.25 m/s	11%	0%
1.50 m/s	23%	3%

VIII. SIMULATION

The performance of the proposed TCED control architecture has been evaluated using simulation. An extended *Model-View-Controller* (MVC) approach [39] was utilized in order to examine the performance benefits of the TCED control model. An extra emulation layer was added between the plant model and controller to simulate timing delays. An example application was developed for the reference sorting machine as shown in Fig. 15. This architecture can be scaled up with multiple sorters by replicating the corresponding FBs. The emulation layer consists of FBs representing physical controllers and I/O network topology of a control system. These controller FBs have been designed to be parameterized with metrics such as I/O scan time, PLC cycle time, network delay, and so on.

A simulation to compare the centralized event-driven control configuration with the TCED control configuration was deployed to the nxtControl IEC 61499 runtime [40]. The speed of the conveyor was ramped up and the numbers of diversion failures were recorded. The simulation metrics for the centralized control configuration were as follows: the sortation algorithm's (as discussed in Section VI) execution time of 25 ms, the network propagation time of 5 ms, the sorter actuation time of 300 ms, the conveyor load rate of 50%, and the item size of 0.3 m. The same metrics were also used for the TCED control configuration with the corresponding omissions. Some failure rate results are listed in Table V. The TCED configuration shows a much lower failure rate when the conveyor speed is ramped up. A lower failure rate at the same speed is a consequence of better reaction time of the control system for the same plant configuration.

A second test was conducted by fixing all metrics except the execution time of the centralized sortation algorithm. This is because, as discussed in Table II, execution time would increase based on system size. The conveyor speed was then ramped up until diversion failure. The TCED algorithm execution time was fixed at 25 ms due to the independence from system size as discussed in Table IV. These results were compared to theoretical conveyor speeds provided by the analytical models. The simulation results presented in Fig. 16 shows that the TCED configuration performs consistently better than a centralized configuration as the algorithm execution time increases. The discrepancies compared to the analytical model may arise from overheads in the simulation, such as the performance of the target runtime and simulation hardware.

IX. CONCLUSION

This paper presented the TCED architecture for designing distributed controls in complex systems composed of modular objects. The architecture provides application developers a

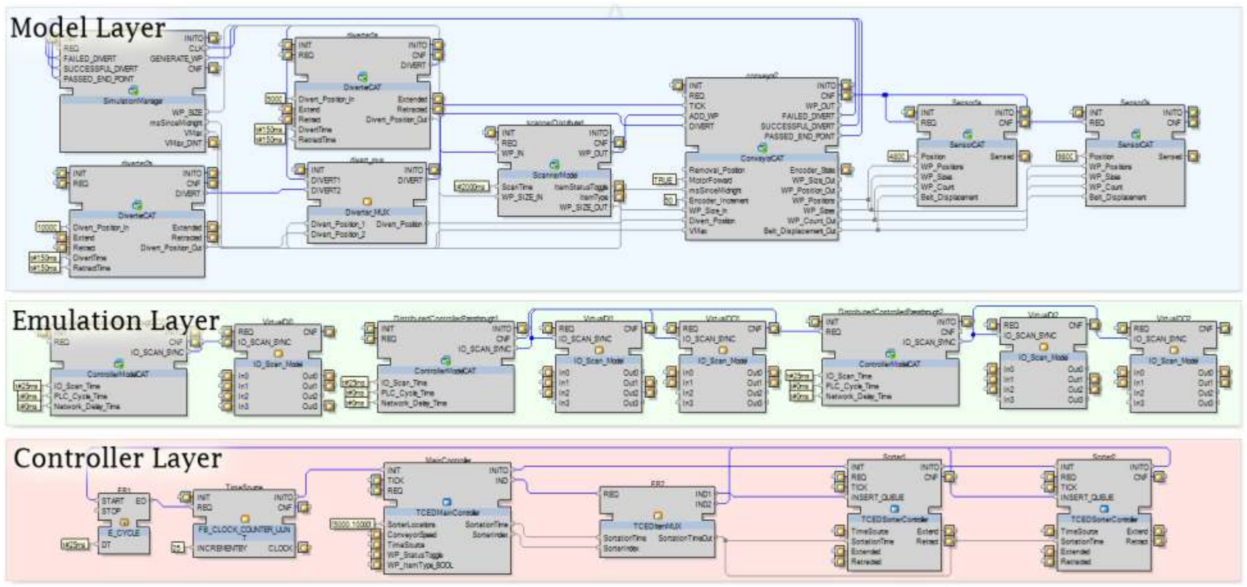


Fig. 15. TCED simulation model for the sorting machine.

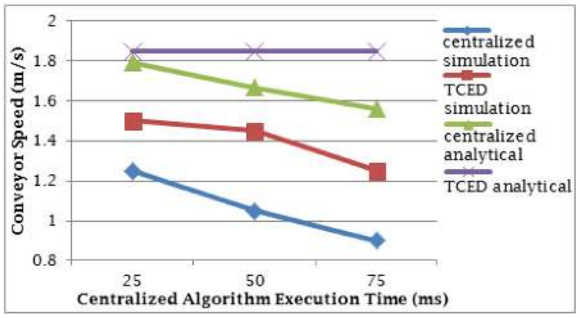


Fig. 16. Analytical performance comparison based on simulation results.

practical mechanism to combine time-driven and event-driven logic in distributed control systems. The semantics of the TCED control model have been illustrated using a Petri net model as applied to a sample material sorting system. Such a model can be used in formal methods, such as model-checking. To confirm the performance benefits of the proposed model, analytic models of performance have been developed. These models have been respectively derived for the centralized, distributed, and TCED control architectures.

A simulation architecture was proposed to integrate timing characteristics into simulation as an extension of the popular MVC architecture. The performance advantage of TCED control configuration over the centralized event-driven configuration has been confirmed in simulation using this architecture. At last, guidelines for practical implementation of the TCED control architecture have been given by presenting a complete implementation of the control system of the example sorting machine using IEC 61499 FB.

The proposed method can be applied to domains requiring distributed precision time control. In the future, the claimed benefits of the TCED control model will be experimented in other domains, such as the distributed automation of smart grids that has been modeled in IEC 61499 [16].

References

[1] IEC, *Programmable Controllers—Part 3: Programming Languages*, IEC Standard 61131-3, 2013.

[2] H. Kopetz and G. Grunsteidl, “TTP—A protocol for fault-tolerant real-time systems,” *Computer*, vol. 27, no. 1, pp. 14–23, Jan. 1994.

[3] IEC, *Function Blocks—Part 1: Architecture*, IEC Standard 61499-1, 2012.

[4] G. Black and V. Vyatkin, “Intelligent component-based automation of baggage handling systems with IEC 61499,” *IEEE Trans. Autom. Sci. Eng*., vol. 7, no. 2, pp. 337–351, Apr. 2010.

[5] J. Yan and V. V. Vyatkin, “Distributed execution and cyber-physical design of baggage handling automation with IEC 61499,” in *Proc. 9th IEEE INDIN*, Lisbon, Portugal, 2011, pp. 573–578.

[6] M. Colla, A. Brusaferri, and E. Carpanzano, “Applying the IEC-61499 model to the shoe manufacturing sector,” in *Proc. 11th IEEE ETFA*, Prague, Czech Republic, 2006, pp. 1301–1308.

[7] S. Preuße, D. Missal, C. Gerber, M. Hirsch, and H.-M. Hanisch, “On the use of model-based IEC 61499 controller design,” *IJDECS,* vol. 1, pp. 115–128, Mar. 2010.

[8] H.-M. Hanisch, M. Hirsch, D. Missal, S. Preusse, and C. Gerber, “One decade of IEC 61499 modeling and verification—Results and open issues,” in *Proc. IFAC Symp. INCOM*, Moscow, Russia, 2009, pp. 211–216.

[9] C. Sünder, A. Zoitl, F. Mehofer, and B. Favre-Bulle, “Advanced use of PLCopen motion control library for autonomous servo drives in IEC 61499 based automation and control systems,” *Elektrotech. Inf. Tech*., vol. 123, no. 5, pp. 191–196, 2006.

[10] M. Sorouri, V. Vyatkin, and S. Xie, “Distributed control design of medical devices using plug-and-play IEC 61499 function blocks,” in *Proc. 19th M2VIP*, Auckland, New Zealand, 2012, pp. 450–455.

[11] C. Pang, V. Vyatkin, Y. Deng, and M. Sorouri, “Virtual smart metering in automation and simulation of energy-efficient lighting system,” in *Proc. 18th IEEE ETFA*, Cagliari, Italy, 2013, pp. 1–8.

[12] L. Wang, G. Adamson, M. Holm, and P. Moore, “A review of function blocks for process planning and control of manufacturing equipment,” *J. Manuf. Syst*., vol. 31, no. 3, pp. 269–279, 2012.

[13] M. Minhat, V. Vyatkin, X. Xu, S. Wong, and Z. Al-Bayaa, “A novel open CNC architecture based on STEP-NC data model and IEC 61499 function blocks,” *Robot. Comput. Integr. Manuf*., vol. 25, no. 3, pp. 560–569, 2009.

[14] P. Tait, “A path to industrial adoption of distributed control technology,” in *Proc. 3rd IEEE INDIN*, Perth, WA, Australia, 2005, pp. 86–91.

[15] M. Merdan, W. Lepuschitz, B. Šahović, and M. Vallée, “Failure detection and recovery in the batch process automation domain using automation agents,” in *Proc. 2nd ACT*, Jakarta, Indonesia, 2010, pp. 113–117.

[16] G. Zhabelova and V. Vyatkin, “Multiagent Smart Grid automation architecture based on IEC 61850/61499 intelligent logical nodes,” *IEEE Trans. Ind. Electron*., vol. 59, no. 5, pp. 2351–2362, May 2012.

[17] C. H. Yang, G. Zhabelova, C. W. Yang, and V. Vyatkin, "Co-simulation environment for event-driven distributed controls of SmartGrid," *IEEE Trans. Ind. Inf.*, vol. 9, no. 3, pp. 1423–1435, Aug. 2013.

[18] V. Vyatkin, "IEC 61499 as enabler of distributed and intelligent automation: State-of-the-art review," *IEEE Trans. Ind. Inf.*, vol. 7, no. 4, pp. 768–781, Nov. 2011.

[19] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Standard 1588-2008.

[20] V. Vyatkin, "Intelligent mechatronic components: Control system engineering using an open distributed architecture," in *Proc. 9th IEEE ETFA*, Lisbon, Portugal, 2003, pp. 277–284.

[21] S. Patil, V. Vyatkin, and M. Sorouri, "Formal verification of intelligent mechatronic systems with decentralized control logic," in *Proc. 17th IEEE ETFA*, Krakow, Poland, 2012, pp. 1–7.

[22] M. Sorouri, S. Patil, and V. Vyatkin, "Distributed control patterns for intelligent mechatronic systems," in *Proc. 10th IEEE INDIN*, Beijing, China, 2012, pp. 259–264.

[23] M. Mazo, Jr., and M. Cao, "Decentralized event-triggered control with asynchronous updates," in *Proc. 50th CDC*, Orlando, FL, USA, 2011.

[24] J. H. Sandee, W. P. M. H. Heemels, and P. P. J. Van Den Bosch, "Event-driven control as an opportunity in the multidisciplinary development of embedded controllers," in *Proc. ACC*, Portland, OR, USA, 2005, pp. 1776–1781.

[25] W. Heemels, J. H. Sandee, and P. P. J. Van Den Bosch, "Analysis of event-driven controllers for linear systems," *Int. J. Control*, vol. 81, no. 4, pp. 571–590, 2008.

[26] J. L. M. Lastra, A. Lobov, and L. Godinho, "Closed loop control using an IEC 61499 application generator for scan-based controllers," in *Proc. 10th IEEE ETFA*, Catania, Italy, 2005, pp. 323–330.

[27] L. H. Yoong, P. S. Roop, V. Vyatkin, and Z. Salcic, "A synchronous approach for IEC 61499 function block implementation," *IEEE Trans. Comput.*, vol. 58, no. 12, pp. 1599–1614, Dec. 2009.

[28] W. Dai and V. Vyatkin, "Redesign distributed PLC control systems using IEC 61499 function blocks," *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 2, pp. 390–401, Apr. 2012.

[29] S. Campanelli, P. Foglia, and C. A. Prete, "Integration of existing IEC 61131-3 systems in an IEC 61499 distributed solution," in *Proc. 17th IEEE ETFA*, Krakow, Poland, 2012, pp. 1–8.

[30] W. Dai, V. N. Dubinin, and V. Vyatkin, "Migration from PLC to IEC 61499 using semantic web technologies," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 44, no. 3, pp. 277–291, Mar. 2014.

[31] F. De Paoli and F. Tisato, "On the complementary nature of event-driven and time-driven models," *Control Eng. Pract.*, vol. 4, no. 6, pp. 847–854, 1996.

[32] K. Harris, "An application of IEEE 1588 to industrial automation," in *Proc. ISPCS*, Ann Arbor, MI, USA, 2008, pp. 71–76.

[33] J. C. Eidson, E. A. Lee, S. Matic, S. A. Seshia, and Z. Jia, "Distributed real-time software for cyber-physical systems," *Proc. IEEE*, vol. 100, no. 1, pp. 45–59, Jan. 2012.

[34] The Offset Pressman. (2011, Mar.). *How a Flying Paster Works* [Online]. Available: http://offsetpressman.blogspot.se/2011/03/how-flying-paster-works.html

[35] P. Derler, J. Eidson, S. Goose, E. A. Lee, and M. Zimmer, "Deterministic execution of Ptides programs," EECS Dept., UCB/EECS, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2013-65, May 2013.

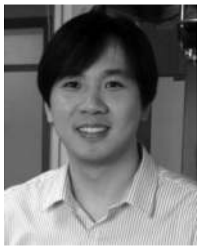
[36] V. Vyatkin, Z. Salcic, P. S. Roop, and J. Fitzgerald, "Now that's smart!" *IEEE Ind. Electron. Mag.*, vol. 1, no. 9, pp. 17–29, Dec. 2007.

[37] C. Pang and V. Vyatkin, "Time-complemented event-driven control framework for distributed motion control systems," in *Proc. 9th IEEE INDIN*, Lisbon, Portugal, 2011.

[38] C. Pang, J. Yan, V. Vyatkin, and S. Jennings, "Distributed IEC 61499 material handling control based on time synchronization with IEEE 1588," in *Proc. IEEE ISPCS*, Munich, Germany, 2011, pp. 126–131.

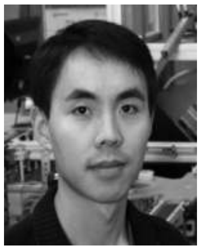
[39] J. Christensen, "IEC 61499 architecture, engineering methodologies and software tools," in *Knowledge and Technology Integration in Production and Services*, V. Mařík, L. Camarinha-Matos, and H. Afsarmanesh, Eds. New York, NY, USA: Springer, 2002, pp. 221–228.

[40] nxtControl. (2014). *nxtSTUDIO—Engineering Software for All Tasks* [Online]. Available: http://www.nxtcontrol.com/en/products/nxtstudio.html



Cheng Pang (S'08–M'13) received the B.E. (Hons.) and M.E. (Hons.) degrees in computer systems engineering and the Ph.D. degree in electrical and electronic engineering from the University of Auckland, Auckland, New Zealand, in 2005, 2007, and 2013, respectively.

He is currently a Post-Doctoral Research Fellow with the Laboratory of Advanced Computing and Communications for Industrial Applications, Luleå University of Technology, Luleå, Sweden. His current research interests include model-driven engineering for industrial automation systems, building automation and control systems, and distributed control for the Internet of things.



Jeffrey Yan (S'13) received the B.E. (Hons.) and M.E. (Hons.) degrees from the University of Auckland, Auckland, New Zealand, in 2009 and 2010, respectively, where he is currently pursuing the Ph.D. degree from the Department of Electrical and Computer Systems Engineering.

His current research interests include the application of distributed, intelligent control in the field of large-scale cyber physical systems.

Valeriy Vyatkin (M'03–SM'04) received the Engineering degree in applied mathematics, and the Ph.D. and Dr. Sci. degrees from the Taganrog State University of Radio Engineering (TSURE), Taganrog, Russia, and the Dr. Eng. degree from the Nagoya Institute of Technology, Nagoya, Japan, in 1988, 1992, 1998, and 1999, respectively.

He is on joint appointment as a Chaired Professor (Åmnesprofessor) of dependable computation and communication systems at Luleå University of Technology, Luleå, Sweden, and a Professor of information and computer engineering in automation, Aalto University, Aalto, Finland. Previously, he served as a Visiting Scholar at the Cambridge University, Cambridge, U.K., and on permanent appointments with the University of Auckland, Auckland, New Zealand, Martin Luther University of Halle-Wittenberg, Halle, Germany, and with TSURE as an Associate Professor and Professor from 1991 to 2002. His current research interests include the area of dependable distributed automation and industrial informatics, including software engineering for industrial automation systems, distributed architectures, and multiagent systems applied in various industry sectors, including smart grids, material handling, building management systems, and reconfigurable manufacturing. He is also active in research on dependability provisions for industrial automation systems, such as methods of formal verification and validation, and theoretical algorithms for improving their performance.

Dr. Vyatkin was the recipient of the Andrew P. Sage Award for the Best IEEE SMC TRANSACTIONS Paper in 2012.