Game Theoretic Feedback Control for Reliability Enhancement of EtherCAT-Based Networked Systems

Liying Li, Peijin Cong, Kun Cao, Junlong Zhou, Tongquan Wei, *Member, IEEE*, Mingsong Chen, Shiyan Hu, and Xiaobo Sharon Hu, *Fellow, IEEE*

***Abstract*—EtherCAT has become one of the leading real-time Ethernet solutions for networked industrial systems, where a reliable communication infrastructure is needed due to highly error-prone environments. However, existing work on EtherCAT mainly focuses on clock synchronization and timeliness improvement. The reliability of EtherCAT-based networked systems has largely been ignored. In this paper, we present a proportional integral derivative (PID)-based feedback control scheme that aims at enhancing reliability of networked systems under timing and system resource constraints. Instead of retransmitting data upon error detection, we use forward error control technique based on inequality of arithmetic and geometric means to achieve the required system reliability at a low deadline miss rate of messages. We further optimize the forward error control technique and design a fast and fair error resilient mechanism by using a cooperative game. In addition to reliability enhancement, our PID-based error control scheme can also improve the stability of a system in terms of deadline miss rate in the presence of burst errors. Simulation results show that the proposed scheme can achieve reliability enhancement of up to 91% compared to benchmarking methods.**

***Index Terms*—Embedded systems, EtherCAT, feedback control scheme, game theory, real-time, reliability.**

I. Introduction

A CYBER physical system (CPS) of increasing importance in the era of industry 4.0 is composed of various physical and computing components that interact through embedded communication capabilities. The connectivity between physical entities and cyber components must ensure accurate and reliable data acquisition from the physical world and real-time information feedback from the cyber space. Networked machines are expected to work more efficiently and reliably under convergence of information and automation technology over the connectivity, which is enabled by the powerful technology of EtherCAT [1].

EtherCAT is an industrial Ethernet technology standardized by ISO [2]–[5]. It is one of the fastest real-time Ethernet networks superior to existing networks adopted in industry. Most existing industrial real-time networks are mainly designed to meet applications' timing constraints, and are not suitable for transmitting large data [6]. For instance, controller area network (CAN) [7] is a popular real-time communication network designed to ensure the communication between micro-controllers and devices in applications without a host computer. CAN is widely used in various fields, such as robot systems, but supports only 1 Mb/s of bandwidth, which is not well-suited for systems that need to transmit large data in a short period. On the contrary, EtherCAT provides high data transmission efficiency at high speed. This is due to the fact that frames transmitted in EtherCAT networks are processed based on an "on the fly" mechanism that ensures the master and multiple slaves can exchange data in a very short time. EtherCAT frames are sent by the master to slaves cyclically. During each cycle time, every slave reads and/or writes its data from/into the EtherCAT frame and no buffering is required. Thanks to the unique way to transmit data, high speed in EtherCAT networks are achievable. For example, by using the full-duplex features of 100BASE-TX, the data rates of EtherCAT can reach more than 100 Mb/s [8]. Fig. 1 illustrates a CPS system, where multiple components are connected together by an EtherCAT cable for machine and plant control in various CPS applications.

Extensive research efforts have been made to investigate EtherCAT and its deployment in high performance industrial applications. Nguyen *et al.* [10] proposed the design and implementation of a closed-loop stepper motor drive control system using EtherCAT. Specifically, they presented the details on the embedded EtherCAT telegram and CiA402 motion profile, and implemented the open-loop control stepper motor based on EtherCAT. Yan *et al.* [11] built a micro-grid control system and used EtherCAT as a communication protocol to ensure the high communication speed for this system. The ring topology of EtherCAT is adopted to exert control over devices.

Manuscript received January 24, 2018; revised May 6, 2018; accepted June 29, 2018. This work was supported in part by the Shanghai Municipal Natural Science Foundation under Grant 16ZR1409000, and in part by the Natural Science Foundation of China under Grant 61672230. The work of X. S. Hu was supported by the U.S. National Science Foundation under Award CNS-1319904. This paper was recommended by Associate Editor Q. Zhu. *(Corresponding authors: Tongquan Wei; Mingsong Chen.)*

L. Li, P. Cong, K. Cao, T. Wei, and M. Chen are with the Department of Computer Science and Technology, East China Normal University, Shanghai 200062, China (e-mail: tqwei@cs.ecnu.edu.cn).

J. Zhou is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China.

S. Hu is with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, MI 49931 USA.

X. S. Hu is with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46656 USA.

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/TCAD.2018.2859241

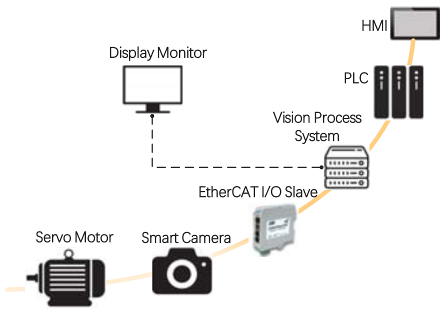

Fig. 1. Example EtherCAT-enabled CPS system [9].

Ma *et al.* [12] proposed an EtherCAT-based multi degree of freedom motion control system including nine rectilinear and rotation. EtherCAT has also been used in the design of modular multilevel converters for high voltage conversion in power electronics [13], [14], and various assisted devices that target people with disabilities following a stroke [15]. EtherCAT is used in these applications as a high-speed, high accuracy clock synchronization, and low-overhead communication platform.

Precise clock synchronization is a key feature that makes EtherCAT appealing in applications above and many other domains like motion control [16]. The clock synchronization mechanism of EtherCAT, known as distributed clock, enables networks to be synchronized within several tens of nanoseconds and guarantees the timeliness of applications [17]. Distributed clock can also effectively reduce implementation costs of EtherCAT devices. Lee *et al.* [6] designed a software architecture for a rescue robot to rescue wounded people and move dangerous objects in disaster situations. Distributed clock for EtherCAT is executed in order to ensure that all joint controllers in the rescue robot can react in time. Xu *et al.* [18] presented a distributed power quality monitoring method, where mass data exchange between monitoring terminal and monitoring center is conducted over high-speed EtherCAT of accurate clock synchronization. The EtherCAT synchronization performance can be improved by using various techniques, such as drift compensation [19], and can be evaluated by conducting extensive experimental measurements [16].

The timeliness of EtherCAT networked system is of particular importance to real-time applications like compliance control in robotics. Bello *et al.* [20] proposed a swapping-based approach to lower the cycle time of transmitting EtherCAT frames. A shorter cycle time entails lower response times, thus increasing the number of messages delivered within their deadlines. In [21], a networked soft motion control system with EtherCAT was designed and evaluated. The timeliness of the presented control method is experimentally validated. Wu and Xie [22] explored end-to-end delays of EtherCAT-based control systems under free-running, frame-driven, and clock-driven schemes. They found that free-running and frame-driven methods fit in traditional automation applications and clock-driven method achieves better results in networked control systems, where deterministic data communication is required. Jia *et al.* [23] designed a new type of wear-resistant coating testing system based on EtherCAT. EtherCAT is used in the design of hardware platform to enhance the timeliness of the proposed system, which is developed with various functions, such as information display, manual operation, and offline simulation.

EtherCAT networks are typically deployed in harsh environments, where transmission links and processing nodes are very likely to suffer from errors. This necessitates a system design approach that takes into account reliability in addition to timeliness. Although EtherCAT has been investigated from various perspectives including its applications, synchronization schemes, and timeliness performance, the reliability of EtherCAT network has not been thoroughly investigated in the literature. The current reliability scheme of EtherCAT can be divided into backward and forward control mechanism. For backward control mechanism, unlike the scheme used in common wireless networks that sends the same frames continuously until the frame is correctly received, EtherCAT masters generally retransmits frames upon a failure detection or timeout. However, backward control mechanism leads to low channel utilization, and requires receivers to send acknowledgments to confirm whether data is received correctly, which increases network overheads and reduces the transmission speed. As to forward control mechanism, redundancy has been widely used to improve reliability. Maruyama and Yamada [17] presented a reliable communication architecture for EtherCAT masters by using the port redundancy. In the presented architecture, an EtherCAT master is equipped with two network interface controllers (i.e., ports). The EtherCAT master sends duplicated frames from both ports, and the frames are received at the other port. Then the master determines which frame can be used by taking a logical OR of data area of two frames. The presented approach enables highly accurate cyclic communications with high reliability. However, this technique only considers the time synchronization failure. In addition, extra hardware is required for EtherCAT masters and slaves, which incurs a significant amount of costs.

In this paper, we propose a feedback control-based scheme to enhance system reliability under the timing constraint and reliability requirement for messages as well as the resource constraint for network channels. The major contribution of this paper is summarized as follows.

1) We investigate reliability modeling of EtherCAT networks from aspects of transmission links and processing nodes, and propose a proportional integral derivative (PID)-based feedback control loop that aims at improving system reliability under the constraint of message deadline miss rate and channel utilization.
2) We improve the proposed PID-based error control scheme with respect to convergence speed and fairness by using a cooperative game and Nash bargaining solution. System reliability, message deadline miss rate, and channel utilization are also improved.
3) Extensive simulations show that the proposed control scheme can enhance system reliability by up to 91% and increase channel utilization by up to 69% when compared to benchmarking methods.

The rest of this paper is organized as follows. Section II introduces EtherCAT system architecture and models. Section III formalizes the problem studied in this paper and

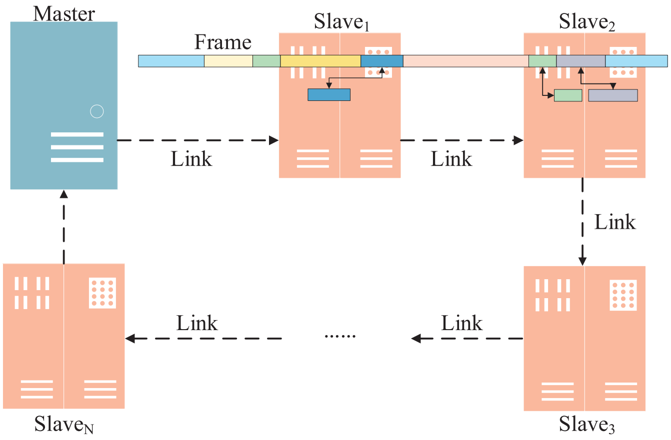

Fig. 2. Ring topology of an EtherCAT system.

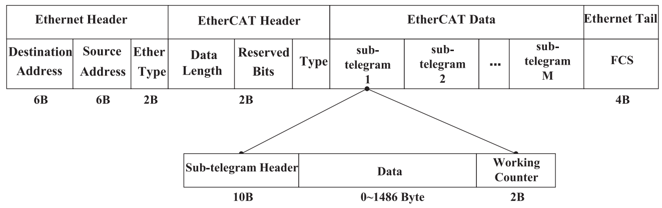

Fig. 3. Structure of an EtherCAT frame.

TABLE I
COMPARE MESSAGE TRANSMISSION WITH TASK EXECUTION

Items	Messages	Tasks
period	minimum interval between two transmissions	task peirod
time	time of transmitting a message through a topology	task execution time
deadline	deadline of finishing the transmission of a message	task deadline
utilization	network utilization	CPU utilization
preemption	non-preemptive	preemptive

provides an overview of the proposed scheme. Section IV describes in details the proposed feedback control scheme. Section V improves of the channel allocation mechanism based on a cooperative game theory. Section VI presents the experimental results, and Section VII concludes this paper.

II. SYSTEM ARCHITECTURE AND MODELS

The focus of this paper is on reliability enhancement of an EtherCAT system in the presence of transient faults. Below, we present the various models used in this paper.

A. System Architecture

EtherCAT is one of the real-time Ethernet communication technologies and is included as a part of the ISO standards [5]. It enables a multitude of network topologies, including line, tree, ring, star, or any combination. In this paper, we adopt the ring topology as depicted in Fig. 2. The system is composed of one master and N slaves connected by the standard Ethernet cable. The master cyclically sends a standard Ethernet frame containing several subtelegrams or messages (see Fig. 3) to slaves. The frame transmits through all slaves. As the frame passes through slaves on the fly, every slave is responsible for reading or/and writing the frame. Specifically, each slave distinguishes subtelegrams addressed to itself by address parameter in the header, then takes an action specified by command parameter (read and/or write data) in the header without buffering a frame. For those subtelegrams that are not addressed to a slave, the slave only need to forward them. After the last slave in the topology transmits the frame back to the master, the next cycle starts again. We refer to the master and the slaves as computing nodes in the topology.

In fact, the scheduling for message transmission through a topology is similar to the scheduling for task execution in a CPU. That is, both processes determine the transimission/execution sequence of message/tasks that compete for shared resources. The difference between the two processes is that task execution in a CPU can be preemptive, while message transmitting through a network topology cannot be interrupted once it starts. Table I gives a brief comparison between message transmission and task execution. The two processes are compared in terms of the period, transimission/execution time, deadline, network/CPU utilization, and preemption. In this paper, we extend task scheduling methods for a CPU node to message processing/transmission in an EtherCAT network.

B. Message Model

The EtherCAT protocol is optimized for processing data. The payload of an EtherCAT frame is encapsulated in the standard IEEE 802.3 Ethernet frame and is typically composed of several subtelegrams (or messages) [24]. Fig. 3 illustrates the fields of a standard IEEE 802.3 Ethernet frame of Ethertype 0x88a4. As shown in the figure, each Ethernet frame contains 10 bytes of Ethernet header, 2 bytes of EtherCAT header, an EtherCAT data field, and 4 bytes of Ethernet tail field. The data field of EtherCAT frame may consists of multiple EtherCAT messages. Each EtherCAT message consists of 10 bytes of header, a messages data field which is up to 1486 bytes and 2 bytes of working counter. The working counter is a mechanism for EtherCAT master to monitor slaves' behavior cyclically and synchronously. It is incremented by the slaves every time they read and/or write data into a telegram successfully. EtherCAT master can monitor the slaves in the topology by checking the working counter value contained in the periodic frames.

We consider a message set Γ, which consists of M independent messages and is denoted by Γ: $\{\tau_1, \tau_2, \ldots, \tau_M\}$. A message in Γ corresponds to a subtelegram in the EtherCAT frame, and we use messages and subtelegrams interchangeably in the following sections. Real-time message τ_i $(1 \leq i \leq M)$ is associated with $\{T_i, D_i, L_i, \mathrm{RG}_i\}$, where T_i is the period of τ_i, D_i represents the deadline of the τ_i, L_i denotes the length of τ_i, and RG_i is the reliability target of τ_i. The reliability requirement of each message may be different, so different reliability target can be set according to the different reliability requirement, determined by the number of different message's backups.

C. Reliability Model

A forward error control technique [26] is adopted in this paper to provide fault-tolerance. Unlike the automatic repeat request (ARQ) technique that resends messages when a fault occurs [25], the forward error control technique sends and executes original messages and their backups at the same time [26]. Since messages in an EtherCAT system are likely to suffer transient faults at nodes and over links, we first discuss the soft error model for nodes, and then introduce the bit error model for links.

D. Soft Error Model for Nodes

The master of EtherCAT transmits a frame that passes through all the slave in topology. When the frame is transmitted forward, each slave recognizes the relevant commands and executes them accordingly while the frames are forwarded to the next device [28], [29]. Since there are multiple EtherCAT frames composed of serval messages for different slaves, these relevant commands are usually executed for many times. For instance, Delgado *et al.* [30] presented a real-time motion control system using EtherCAT protocol. More specifically, they conducted an established trajectory planning algorithm presented in [31] to generate a large number of velocity commands and send them to slaves. The slaves recognizes the relevant commands and executes them just like CPU execute tasks. Thus, soft errors may occur when messages are processed in slaves. Soft errors mainly result from transient faults. Poisson distribution is widely used to model the occurrences of transient faults in computing nodes [27]. Let λ_j be the average fault occurrence rate at computing node j for $0 \leq j \leq N$,[1] then it is given by

$$\lambda_j = \gamma_j \cdot e^{-\alpha_j \cdot f_j} \tag{1}$$

where γ_j and α_j are node dependent constants, and f_j is the operating frequency of node j.

EtherCAT computing nodes process frames on the fly. Specifically, the incoming frame of a node is divided into multiple fragments of equal length, each of which is processed by the node in a unit time. A key characteristic of the EtherCAT on the fly processing is that the processing time of a fragment is equal to its forwarding time, thus, there is no need to buffer the frame. Let Δl denote the fragment length of a frame that a node can process at a time, and E_j denote the processing time of a fragment length message at node j. E_j is calculated as $E_j = \Delta l / f_j$. The probability that no faults occur at node j during the processing of message τ_i, denoted by P_{ij}, is hence expressed as

$$P_{ij} = \left(e^{-\lambda_j \cdot E_j}\right)^{\frac{L_i}{\Delta l}} \tag{2}$$

where L_i is the length of the message τ_i. Since each message passes through all the $N+1$ nodes (including the master and slaves) in the EtherCAT topology, the probability that message τ_i is processed and forwarded successfully at all nodes, denoted by $P_{i,\text{nodes}}$, is calculated as

$$P_{i,\text{nodes}} = \prod_{j=0}^{N} P_{ij} = e^{-\frac{L_i}{\Delta l}\sum_{j=0}^{N} E_j \cdot \lambda_j}. \tag{3}$$

[1] N is the number of nodes in the system and the concerned node is the master when $j = 0$.

E. Bit Error Model for Links

In digital transmission, bit errors are induced by noise, interference, distortion, or bit synchronization errors over links. Let t_i be the transmission time of message τ_i through all links of the topology. Then the probability that message τ_i is successfully transmitted over links, which is denoted by $P_{i,\text{links}}$, can be modeled as [32]

$$P_{i,\text{links}} = e^{-\theta \cdot t_i} \tag{4}$$

where θ is the constant bit error rate.

Let P_i be the probability that message τ_i is successfully processed and transmitted in a given EtherCAT system when no messages are replicated for tolerance. P_i is obtained by

$$P_i = P_{i,\text{nodes}} \cdot P_{i,\text{links}} = e^{-\theta \cdot t_i - \frac{L_i}{\Delta l}\sum_{j=0}^{N} E_j \cdot \lambda j}. \tag{5}$$

The reliability of a message is defined as the probability that the message issued by the master is successfully processed, and routed back to the master in the presence of errors. Assume that k_i backups are used for message τ_i to achieve the required reliability. The reliability, denoted by $R_i(k_i)$, is expressed as

$$R_i(k_i) = 1 - (1 - P_i)^{k_i+1}. \tag{6}$$

The reliability of the system of M messages, defined as the product of the reliability of individual messages and denoted by R_{sys}, is thus given by

$$R_{\text{sys}} = \prod_{i=1}^{M} R_i(k_i). \tag{7}$$

III. Problem Definition and Overview of the Proposed Scheme

Our goal is to design a fault-tolerance message scheduling scheme in order to enhance the overall reliability of the EtherCAT system (i.e., R_{sys}). We first formulate in this section the problem to be tackled, followed by an overview of the proposed control scheme. We assume a scenario that messages transmitted in an EtherCAT system are periodic and independent, and the characteristics of the messages are known *a priori*. The forward error control technique is used in the EtherCAT system to achieve fault tolerance.

A. Problem Definition

Given an EtherCAT system of a ring topology that contains $N+1$ nodes (one master and N slaves), and a set of M messages, find the number of backups for each message such that the system reliability, R_{sys}, is maximized under the timing and message reliability constraint. That is

$$\begin{aligned} &\text{Maximize}: R_{\text{sys}} \\ &\text{Subject to}: \text{MissRate} \leq \varepsilon \\ &\qquad R_i \geq \text{RG}_i \\ &\qquad \text{NET} \leq 1 \end{aligned}$$

where MissRate is the deadline miss rate of messages during one sampling period of the proposed controller, ε is a positive constant that indicates the threshold for deadline miss

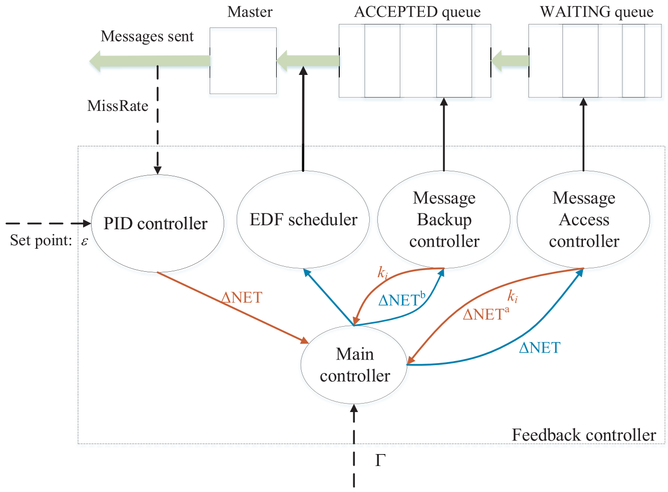

Fig. 4. Overview of the proposed feedback control system.

rate, R_i is the reliability of message τ_i, and NET is the total channel utilization of messages in the network. The message τ_i is required to meet its reliability target RG_i. The objective function R_{sys} is given in (7).

B. Overview of the Proposed Control Scheme

Fig. 4 shows the overall structure of our proposal feedback control system. It consists of a main controller, PID controller, message access (MA) controller, message backup (MB) controller, and earliest deadline first (EDF) scheduler. Two queues, ACCEPTED and WAITING, are maintained for messages admitted into the system and messages that have not yet been accepted by the systems, respectively. The PID controller periodically samples the current deadline miss rate MissRate of messages and returns the required control action ΔNET to the main controller according to (8). ΔNET is the total amount of channel utilization that should be added into (when $\Delta\mathrm{NET} > 0$) or reduced from (when $\Delta\mathrm{NET} < 0$) the system. Channel utilization is defined as the percentage of the net bit rate (in bit/s) of a digital communication channel used for the actually achieved throughput [33]. The main controller calls the MA and MB controller sequentially to accommodate the channel utilization of ΔNET. The MA controller can accommodate the channel utilization of the system by controlling message flow into the ACCEPTED queue. If the MA controller cannot accommodate all of the ΔNET, the main controller calls the MB controller to accommodate the channel utilization of the system by increasing/decreasing the number of backups of messages in the ACCEPTED queue. Finally, the EDF scheduler schedules the accepted messages along with their backups using the EDF policy and dispatches the accepted messages to the master for processing. We describe in detail the PID controller, MA controller, MB controller, and main controller below.

IV. Feedback Control Scheme for Reliability Enhancement

In this section, we present in details the working mechanism of the proposed controller that integrates a PID controller, MA controller, MB controller, and EDF scheduler.

Algorithm 1: PID Control Algorithm

Input: Threshold ε for deadline miss rate.
Output: Total channel utilization to be accommodated, ΔNET.

```
do
    PID controller samples messages to derive MissRate;
    Calculate ΔNET using Equation (8);
    return ΔNET;
while (MissRate > ε);
return ΔNET;
```

A. PID Controller

PID controller is a control loop feedback mechanism that improves robustness of a control process against external disturbances. The operation of our PID controller is outlined in Algorithm 1. Taking as input the threshold ε for deadline miss rate, the PID controller periodically samples messages to derive the process variable MissRate. Note that the PID controller only considers messages that have entered the EtherCAT system. Messages rejected from entering the system are not taken into account when sampling. The PID controller then computes the control variable ΔNET in terms of requested channel utilization using the control equation given by [34]

$$\Delta\mathrm{NET} = -C_P \cdot \mathrm{err}(t) - C_I \cdot \sum_{\mathrm{IW}} \mathrm{err}(t) - C_D \cdot \frac{\mathrm{err}(t) - \mathrm{err}(t - \mathrm{DW})}{\mathrm{DW}} \quad (8)$$

where $\mathrm{err}(t)$ is the difference between the threshold for system deadline miss rate and the current system deadline miss rate, that is, $\mathrm{err}(t) = \varepsilon - \mathrm{MissRate}$. The C_P, C_I, and C_D are coefficients of the PID controller. IW is the time window for the last IW time units over which the errors are summed. Similarly, DW is the time window for the last DW time units over which the derivative error is calculated as $(\mathrm{err}(t) - \mathrm{err}(t - \mathrm{DW}))/\mathrm{DW}$.

The PID controller returns the computed ΔNET to the main controller, which in turn sends ΔNET to the MA and MB controller for allocation. When $\Delta\mathrm{NET} > 0$, the channel utilization should be increased, hence more messages and/or message backups are admitted into the system to allocate the ΔNET. On the contrary, when $\Delta\mathrm{NET} < 0$, the channel utilization should be decreased, hence some messages and/or message backups will be dismissed from the system to distribute the ΔNET. The procedure repeats until $\mathrm{MissRate} \le \varepsilon$.

B. Message Access Controller

The MA controller is responsible for controlling the admission of original messages into the EtherCAT system. When a new message τ_i is submitted to the WAITING queue, the MA controller decides whether it can be accepted into the system. Messages in the WAITING queue are sorted according to the EDF scheduling policy. Let $\Delta\mathrm{NET}^a$ be the portion of the channel utilization ΔNET that can be allocated by the MA controller. As shown in Algorithm 2, the MA controller takes ΔNET as input and returns $\Delta\mathrm{NET}^a$ to the main controller. Given $\Delta\mathrm{NET} > 0$, the MA controller admits message τ_i if the condition $\Delta\mathrm{NET}^a - \mathrm{NET}_{i(e+1)} > 0$ holds (lines 3–18). e denotes the minimum number of backups to

Algorithm 2: MA Control Algorithm

Input: ΔNET; M, the number of messages in ACCEPTED queue; W, the number of messages in WAITING queue.
Output: The portion of ΔNET that can be accommodated by the MA controller.

`// initialization`
1 $\Delta NET^a = 0$;
2 **if** $\Delta NET > 0$ **then**
3   Sort messages in WAITING queue according to the EDF policy;
4   **for** $i = 1; i \leq W; i++$ **do**
5   **if** $\Delta NET^a \geq \Delta NET$ **then**
6   break;
7   **end**
8   Calculate minimum number of backups that can meet τ_i's reliability target (e);
9   **if** $\Delta NET^a - NET_{i(e+1)} \geq 0$ **then**
      `// ` NET_{ie} ` is given by Equation (9)`
10       $\Delta NET^a = \Delta NET^a + NET_{i(e+1)}$;
11   Dequeue head message from WAITING queue;
12   Enqueue the message to ACCEPTED queue;
13   Update the number of message τ_i's backup;
14       $M++$;
15   **end**
16   **end**
17 **end**
18 **else**
19   Messages remain in WAITING queue;
20 **end**
21 **return** ΔNET^a, the portion of ΔNET that can be accommodated by the MA controller; the number of messages' backup;

meet the reliability target of a message, which can be calculated by setting R_i in (6) to RG_i. $\mathrm{NET}_{i(e+1)}$ is the channel utilization of message τ_i with e backups, and can be calculated by using (9) by setting $c = (e+1)$. Once τ_i is admitted, $\Delta\mathrm{NET}^a$ is updated to $\Delta\mathrm{NET}^a + \mathrm{NET}_{i(e+1)}$. The admitted message is dequeued from the WAITING queue, and in turn enqueued to the ACCEPTED queue. The admission request of the message τ_i is denied if $\Delta\mathrm{NET} \leq 0$ or available channel resources cannot meet τ_i's reliability target. The rejected messages remain in the WAITING queue (lines 19–21).

C. Message Backup Controller

The MB controller functions as a tuner to regulate the channel utilization of the EtherCAT system. It changes the channel utilization by adjusting the number of backup of messages, which to be transmitted via the communication channel. When it increases/decreases the number of backups, the channel utilization of the EtherCAT system increases/decreases accordingly. Once the number of backups of message τ_i is determined, the message's reliability, R_i, can be derived by using (6). Since every message's reliability is non-negative, according to the inequality of arithmetic and geometric means [35], we have

$$\left(\frac{R_1 + R_2 + \cdots + R_M}{M}\right)^M \geq R_1 \cdot R_2 \cdot \ldots \cdot R_M = R_{\mathrm{sys}}$$

where M is the number of messages in the system and R_{sys} represents the overall system reliability. Equality in the above relation holds if and only if $R_1 = R_2 = \cdots = R_M$. Therefore, in order to enhance system reliability, R_{sys}, we aim to balance the reliability of each message equal and make each as large as possible.

Algorithm 3: MB Control Algorithm

Input: The portion of allocated channel utilization (ΔNET^b).
Output: The number of messages' backup.

1 **if** $\Delta NET^b > 0$ **then**
2   Compute mean R_{avg} of message reliabilities in ACCEPTED queue;
3   Determine the number m of messages for $R_i < R_{avg}$ in the queue;
4   Sort the m messages in the queue in the ascending order of reliability, R_i;
5   $i = 1$;
6   **while** $\Delta NET^b > 0$ **do**
7   Increment the number of message τ_i's backup by 1;
8     $\Delta NET^b = \Delta NET^b + NET_{i1}$;
    `// ` NET_{i1} ` is given by Equation (9)`
9     $i++$;
10   **if** $i = m + 1$ **then**
11       $i = 1$;
12   Recalculate R_{avg} and update m;
13   **end**
14   **end**
15 **end**
16 **else**
17   Derive mean R_{avg} of message reliabilities in ACCEPTED queue;
18   Derive the number m of messages for $R_i > R_{avg}$ in the queue;
19   Sort the m messages in the queue in ascending order of reliability, R_i;
20   $i = m$;
21   **while** $\Delta NET^b < 0$ **do**
22   Decrement the number of message τ_i's backup by 1;
23     $\Delta NET^b = \Delta NET^b - NET_{i1}$;
    `// ` NET_{i1} ` is given by Equation (9)`
24     $i--$;
25   **if** $i = 0$ **then**
26       $i = m$;
27   Recalculate R_{avg} and update m;
28   **end**
29   **end**
30 **end**
31 **return** the number of messages' backup;

The MB controller is designed based on the above principle to enhance the system reliability. It first calculates the average message reliability in the ACCEPTED queue and selects messages with reliability below/above the average. It then iteratively increases/decreases the number of backups of the selected messages to improve system reliability R_{sys}. As shown in Algorithm 3, the MB controller takes $\Delta\mathrm{NET}^b$ as input. $\Delta\mathrm{NET}^b$ is the portion of the channel utilization that can be allocated by the MB controller, which is calculated by the main controller.

Algorithm 3 works as follows. For the case of $\Delta\mathrm{NET}^b > 0$, the algorithm calculates the average reliability of messages in the ACCEPTED queue (denoted by R_{avg}), picks the m messages for $R_i < R_{\mathrm{avg}}$ and $1 \leq i \leq m$, and sorts the m messages in the queue in the ascending order of reliability (lines 2–5). When not all of $\Delta\mathrm{NET}^b$ is allocated by MB controller (i.e., $\Delta\mathrm{NET}^b > 0$), the algorithm increments the number of message τ_i's backup by 1, updates $\Delta\mathrm{NET}^b$ to $\Delta\mathrm{NET}^b+\mathrm{NET}_{i1}$ and increments i by 1. If all the m messages have been updated by increasing a backup and $\Delta\mathrm{NET}^b$ is not used up yet, the algorithm resets $i = 1$, recalculate the average reliability, update the value of m and repeats the accommodation process (lines 7–14). Assume that τ_i is the message selected during the accommodation process. Let NET_{ic} be the incurred channel utilization due to the admission of message τ_i and its c

copies, then NET_{ic} is given by

$$\mathrm{NET}_{ic} = \frac{c\left(\sum_{j=0}^{N} E_j + t_i\right)}{T_i} \tag{9}$$

where t_i is the total time needed to transmit a message over all the links of the ring topology, T_i is the period of message τ_i, E_j is the processing time of unit length message at node j, and N is the number of nodes in the system. NET_{i1} in line 9 can be easily derived using (9).

In the case of $\Delta\mathrm{NET}^b < 0$, the algorithm works the same as in the case of $\Delta\mathrm{NET} > 0$ except that backups of messages satisfying $R_i > R_{\mathrm{avg}}$ for $1 \le i \le m$ are iteratively dismissed from the system (lines 16–29).

The control scheme of the MB controller above simply considers the average of message reliability (R_{avg}), and ignores a single message's reliability target (RG_i). However, messages may have different reliability targets. It is expected that the MB controller can allocate different channel resources for messages according to their respective reliability targets. Further, incrementally increasing the backup of messages during each iteration takes a long time for the EtherCAT system to converge. Note that the time complexity of Algorithm 3 is at least $O(M)$. Thus, we propose an optimization strategy based on the cooperative game theory and Nash bargaining solution for the MB controller. The time complexity of the game-based optimization strategy is $O(M)$, and is introduced in Section V.

D. EDF Scheduler and Main Controller

EDF is a dynamic scheduling algorithm that dictates the arrangement of messages in a priority queue [42]. When the EDF scheduler is called, the EDF scheduler first finds the task with the earliest deadline and then executes the task [43]. In our feedback control scheme, once a message is admitted into the system by the MA controller and the number of its backups is adjusted by the MB controller, it is delivered to the ACCEPTED queue. When the MB controller returns the number of message backups, ACCEPTED queue is ready and the main controller invokes the EDF scheduler. Thus, the EDF scheduler is scheduled every PID controller's sampling period, and this scheduling frequency has no relation to the period of messages. The EDF scheduler dynamically arranges the execution order of messages in the ACCEPTED queue. The message with the earliest deadline is selected by the EDF scheduler, and is dispatched to the EtherCAT master for processing.

The main control algorithm integrates the PID, MA, and MB controllers to form a closed loop that effectively improves the robustness of the control process against external disturbances. It is called periodically for the enhancement of system reliability, the period of which is determined by the minimum sampling interval.

Algorithm 4 describes the operation of the main control algorithm. It takes as input the message set (Γ), the total channel utilization returned by the PID controller for allocation ($\Delta\mathrm{NET}$), the portion of channel utilization allocated by the MA controller ($\Delta\mathrm{NET}^a$), and the updated numbers of all the messages' backups. If this is the first time the algorithm is called, it first determines the number of backups for

Algorithm 4: Main Control Algorithm

Input: The message set Γ; ΔNET, total channel utilization to be accommodated; ΔNET^a, the portion of accommodated channel utilization; The updated number of message backups.

```
// Initialize the number of messages' backups
1  if the algorithm is called for the first time then
2      for τ_i ∈ Γ do
3          Calculate the number of message τ_i's backup based on τ_i's
           reliability target (RG_i) and Equation (6);
4      end
5  end
6  Call PID (Algorithm 1) to get ΔNET;
7  Call MA (Algorithm 2) to get ΔNET^a;
8  Calculate ΔNET^b using ΔNET^b = ΔNET − ΔNET^a;
9  if ΔNET^b ≠ 0 then
10     Call MB (Algorithm 3) to allocate ΔNET^b and updated message
       backups;
11 end
12 Call EDF scheduler to dispatch messages;
```

each message τ_i based on the reliability target (RG_i) and (6) (lines 2–6). It then calls the PID controller to calculate the deadline miss rate MissRate and $\Delta\mathrm{NET}$ (line 7), calls the MA algorithm to derive $\Delta\mathrm{NET}^a$ (line 8), and calculates $\Delta\mathrm{NET}^b$ in (line 9). Afterward the MB algorithm is called to allocate $\Delta\mathrm{NET}^b$ if $\Delta\mathrm{NET}^b \neq 0$ and update messages' backups (lines 10–12). In the end, the EDF scheduler is called (line 13) to dispatch messages to the master for processing.

V. Game Theory-Based Refinement of Message Backup Control

Due to the slow convergence and unfairness of the MB control mechanism described in Section IV, we propose a game theoretic approach to refining the channel allocation process for further reliability enhancement. In this section, we first introduce the concepts of cooperative game and Nash bargaining, then we model the channel allocation game among multiple messages, and finally refine our MB control mechanism based on a game theory.

A. Cooperative Game and Nash Bargaining

A cooperative game consists of M players, a performance function f, and an initial agreement point $\boldsymbol{RG}$. The M players are represented by a 3-tuple of nonempty, closed, and convex set $\{\ \boldsymbol{\kappa},\ \aleph,\ \Re\ \}$, where $\boldsymbol{\kappa}$ is the set of strategies, $\aleph$ denotes the states of the assigned resource, and $\Re$ gives the states of the M players. The performance function f maps $\boldsymbol{\kappa}$ to $\Re$. The vector $\boldsymbol{RG} = (\mathrm{RG}_1, \mathrm{RG}_2, \ldots, \mathrm{RG}_M)$ is defined as the initial agreement point, where RG_i indicates the minimum value of performance function f. RG_i is the minimal performance required for the player i to enter the game without any cooperation. The above cooperative game is in general resolved by Nash bargaining, and the generated solutions to the cooperative game are called Nash bargaining solutions.

Nash bargaining solution (NBS solution) [38] is defined as follows. A mapping $f : (\boldsymbol{\kappa}|\boldsymbol{RG}) \to \Re$ is an NBS solution if $f(\boldsymbol{\kappa}|\boldsymbol{RG}) \in \Re$, where $\boldsymbol{\kappa}$ is the set of strategies, i.e., the set of possible bargaining agreements that M players may reach. $\boldsymbol{RG}$ is the set of initial agreement point. $\Re$ represents the set of players' current states and $f(\boldsymbol{\kappa}|\boldsymbol{RG})$ is Pareto optimal and satisfies the fairness axioms [39].

In the modeling of our channel allocation, $\boldsymbol{\kappa} = (\Delta k_1, \Delta k_2, \ldots, \Delta k_M)$ denotes the set of possible bargaining agreements that M messages may reach. $\mathfrak{R}$ represents the set of M messages' current reliability. The performance function f maps allocation strategies (i.e., $\boldsymbol{\kappa}$) to messages' current reliability (i.e., $\mathfrak{R}$). The initial agreement point $\boldsymbol{RG}$ is the set of minimum guarantee (i.e., the reliability target of messages) that system must satisfy. We assume that message τ_i $(1 \leq i \leq M)$ involved in the cooperative game can achieve its initial performance requirement (RG_i) without any cooperation. Thus, we have $\mathfrak{R} = \{R_i | R_i \geq \mathrm{RG}_i\}$. Under these definitions and the assumption, we can derive an NBS solution with its strategy $\Delta k_i \in \boldsymbol{\kappa}$, which is obtained by solving the following optimization problem [37]:

$$\text{Problem}: \max \prod_{i=1}^{M} (f(\Delta k_i | \mathrm{RG}_i) - \mathrm{RG}_i). \tag{10}$$

In the NBS solution above, multiple players (typically more than two) enter the cooperative game with their corresponding initial performance requirements in $\boldsymbol{RG}$ satisfied. The messages (players) cooperate in the game to achieve a win-win solution, which enhances the performance given in (10) and leads to a relative fairness among all messages. Using the logarithm of the objective function, an equivalent problem can be derived as

$$\text{Problem}': \max \sum_{i=1}^{M} \ln(f(\Delta k_i | \mathrm{RG}_i) - \mathrm{RG}_i) \tag{11}$$

where Problem$'$ is a convex optimization problem and has a unique solution [40], [41]. The unique solution to the problem is the NBS solution.

B. Channel Allocation Refinement for MB Controller

We consider a cooperative game in which M messages are competing for the shared available channel resource $(\Delta\mathrm{NET})$. In the context of our channel allocation, we define the performance function f that maps the change in the number of message τ_i's backups (i.e., Δk_i) to the reliability of message τ_i (i.e., R_i). The performance function is formulated as

$$f(\Delta k_i | \mathrm{RG}_i) = \left(1 - \left(1 - e^{-\theta t_i - \sum_{j=0}^{N} E_j \lambda_j}\right)^{k_i + \Delta k_i + 1}\right) \tag{12}$$

where k_i denotes the original number of message τ_i's backups and Δk_i represents the change in the number of τ_i's backups. t_i, $\sum_{j=0}^{N} E_j$, and T_i denotes the time that τ_i transmits over links in EtherCAT topology, the time τ_i processed in $N+1$ computing nodes (N slaves plus 1 master), and τ_i's period, respectively. Suppose that each message τ_i has an initial reliability requirement RG_i, with RG_i we can derive the minimal number of τ_i's backups that need to be guaranteed without any cooperation. We also assume that the M messages can achieve the same or better performance (i.e., $R_i \geq \mathrm{RG}_i$).

Our goal is to enhance system reliability and improve the reliability of individual messages under the messages' reliability requirements. The problem can be described as follows. Given the shared available channel resource $(\Delta\mathrm{NET}^b)$ and reliability requirements $(\boldsymbol{RG})$, M messages cooperate in the game to obtain a win-win solution described by $(\Delta k_1, \Delta k_2, \ldots, \Delta k_M)$. Therefore, this optimization problem can be formulated as

$$\begin{aligned} \text{Maximize} \quad & \prod_{i=1}^{M} (f(\Delta k_i | \mathrm{RG}_i) - \mathrm{RG}_i) \\ &= \prod_{i=1}^{M} \left(1 - \left(1 - e^{-\theta t_i - \sum_{j=0}^{N} E_j \lambda_j}\right)^{k_i + \Delta k_i + 1} - \mathrm{RG}_i\right) \end{aligned} \tag{13}$$

$$\text{Subject to} \quad \sum_{i=1}^{M} \frac{\left(t_i + \sum_{j=0}^{N} E_j\right) \cdot \Delta k_i}{T_i} \leq \Delta\mathrm{NET}^b \tag{14}$$

where (14) indicates all of the available channel can be allocated to enhance reliability.

In the above formulation, $\max \prod_{i=1}^{M} (f(\Delta k_i | \mathrm{RG}_i) - \mathrm{RG}_i)$ is selected as the objective rather than $\max \sum_{i=1}^{M} (f(\Delta k_i | \mathrm{RG}_i) - \mathrm{RG}_i)$. This is because the former formulation not only demonstrates the capability of maximizing system reliability, but also shows the expectation of the M messages for maximizing their respective reliability. According to the analysis given in the end of Section V, the objective in (13) is equivalent to $\max \sum_{i=1}^{M} \ln(1 - (1 - e^{-\theta t_i - \sum_{j=0}^{N} E_j \lambda_j})^{k_i + \Delta k_i + 1} - \mathrm{RG}_i)$, which can be converted into

$$-\min \sum_{i=1}^{M} \ln\left(1 - \left(1 - e^{-\theta t_i - \sum\limits_{j=0}^{N} E_j \lambda_j}\right)^{k_i + \Delta k_i + 1} - \mathrm{RG}_i\right). \tag{15}$$

Equation (13) is an optimization problem that attempts to maximize system reliability under the constraint of channel resources [i.e., (14)]. Since Lagrange multiplier is powerful for solving this type of problem with low computation complexity, we adopt it to obtain the best solution to our problem. The Lagrangian of this problem is expressed as

$$\begin{aligned} \iota(\Delta k_i, \alpha) = &- \sum_{i=1}^{M} \ln\left(1 - \left(1 - e^{-\theta t_i - \sum_{j=0}^{N} E_j \lambda_j}\right)^{k_i + \Delta k_i + 1} - \mathrm{RG}_i\right) \\ &+ \alpha \left(\sum_{i=1}^{M} \frac{\left(t_i + \sum_{j=0}^{N} E_j\right) \Delta k_i}{T_i} - \Delta\mathrm{NET}^b\right) \end{aligned} \tag{16}$$

where $\alpha \in \mathbb{R}$, and it is the Lagrange multiplier associated with the constraints given in (14).

It is clear that the optimal solution is derived when the derivative of $\iota(\Delta k_i, \alpha)$ with respect to Δk_i equals zero. In this case, the expression

$$\nabla \iota(\Delta k_i, \alpha) = 0 \Leftrightarrow \nabla \iota_1 + \alpha \nabla \iota_2 = 0 \tag{17}$$

and the Karush–Kuhn–Tucker conditions [40] holds. In (17), $\nabla \iota_1 = (1 - e^{-\theta t_i - \sum_{j=0}^{N} E_j \lambda_j})^{k_i + \Delta k_i + 1} \ln(1 - e^{-\theta t_i - \sum_{j=0}^{N} E_j \lambda_j}) / (1 - (1 - e^{-\theta t_i - \sum_{j=0}^{N} E_j \lambda_j})^{k_i + \Delta k_i + 1})$ and $\nabla \iota_2 = (E_i + t_i)/T_i$. Therefore, the best solution to the optimization problem can be derived from (17), and it can be given by

$$\Delta k_i = ((\ln(1 - \mathrm{RG}_i) \cdot \nu) - \ln((1 - \nu) \cdot \omega)) / \omega - k_i - 1 \tag{18}$$

where ν denotes $(t_i + \sum_{j=0}^{N} E_j) \cdot \lambda_j \cdot \alpha / T_i$ and ω represents $\ln(1 - e^{-\theta \cdot t_i - \sum_{j=0}^{N} E_j \cdot \lambda_j})$.

Algorithm 5: Refined MB Control Algorithm

```
   Input: The portion of accommodated channel utilization (ΔNET^b).
   Output: The number of messages backup.
1  for i = 1 to M do
2  |   Calculate the number of backups of message τ_i that need to be
   |   changed (Δk_i) using Equation (18);
3  |   if ΔNET^b > 0 then
4  |   |   Increase the number of message τ_i's backup by Δk_i
5  |   end
6  |   else
7  |   |   Decrease the number of message τ_i's backup by Δk_i;
8  |   end
9  end
10 return the number of messages' backup ;
```

As indicated in (18), we can improve the original MB controller algorithm (Algorithm 3) by using the method above, the refined MB control algorithm is shown as follows.

Algorithm 5 works as follows. For each message τ_i in ACCEPTED queue, it calculates Δk_i, the change in the number of message τ_i's backups (Δk_i), by using (18) (line 3). In the case of $\Delta \mathrm{NET}^b > 0$, the algorithm increases the number of message τ_i's backup by Δk_i (lines 4–6). In the case of $\Delta \mathrm{NET}^b < 0$, it decreases the number of message τ_i's backup by Δk_i (lines 7–9). The time complexity of Algorithm 5 is $O(M)$.

VI. Simulation-Based Evaluation

Extensive simulation-based experiments have been conducted to validate the effectiveness of the proposed scheme. In this section, we first describe simulation settings in detail and then verify the effectiveness of the refined channel allocation mechanism proposed in Section V. To evaluate the performance of the proposed scheme, we compare the original feedback control scheme with two benchmarking methods in terms of deadline miss rate, channel utilization, and system reliability. Finally, we compare the refined channel allocation mechanism with the original one in order to validate the effectiveness of the refined mechanism.

A. Simulation Settings

The simulations are conducted on a machine equipped with 2.4 GHz Intel i7 quad-core processor and 8 GB DDR4 memory, and running a Windows version of MATLAB_x64 and OMNeT++. OMNeT++ is an extensible, modular, and component-based C++ simulation library and framework, primarily for building network simulators [36]. We use OMNeT++ to simulate the EtherCAT ring topology and MATLAB_x64 to simulate the message scheduling process of the proposed feedback scheme. Two different scales of EtherCAT ring topologies are considered in the simulation for a better comparison study. The first topology has 1 master and 10 slaves, while the second topology contains 1 master and 20 slaves. We use three message sets, each of which contains 5, 10, and 20 messages, respectively.

Similar to the work presented in [34], coefficients C_P, C_I, and C_D of the PID controller are set to 0.5, 0.005, and 0.1, respectively. The time window IW and DW are set to 100 and 1 units of time, respectively. The PID controller samples the network once every 500 time units. The values of RG reflect the difference in reliability targets of messages. We randomly generate reliability target RG_i for message τ_i in the interval of (0,1). The period T_i (in time units), deadline D_i (in time units), and length L_i (in bytes) of message τ_i are randomly generated in the interval of (200, 500), (200, 800), and (12, 1498), respectively. The fragment length a node can process on the fly at a time (Δl) is set to 4 Bytes [20].

In order to prove the effectiveness of our proposed methods, we compare the proposed methods with three benchmarking methods in various aspects. The three benchmarking methods are referred to as no-backup (NBK), ARQ, and allocating channel equally (ACE), respectively. The first method, referred to as NBK, sends messages with no backups even if errors occur, that is, no error control technique is taken for reliability enhancement. The second method is ARQ, which also sends messages with no backup, however, it resends messages when a fault occurs. The third one is called ACE. This method assigns available channel utilization equally to each message.

For the sake of easy presentation, our proposed reliability enhancement methods are referred to as Proposed_Mean and Proposed_Game, respectively. Our Proposed_Mean method derives the initial number of each message's backup according to our reliability model, then allocates channel utilization based on inequality of arithmetic and geometric means, and obtains the ultimate number of backups for each message through multiple iterations. Our Proposed_Game method refines the Proposed_Mean method in the way that channel utilization is allocated. It distributes channel utilization to each message based on the game theory and Nash bargaining solution, and calculates the final number of message's backups by using Lagrange multiplier.

B. Proposed_Mean Versus Benchmarking NBK and ARQ

We compare the Proposed_Mean method with the NBK method and the ARQ method in terms of message deadline miss rate, channel utilization, and system reliability, respectively. Fig. 5 shows the deadline miss rate of the three methods. It can be seen from the figure that the MissRate of the Proposed_Mean method is higher than that of the NBK method in two different topologies. This is primarily due to the fact that NBK method does not use any backup for reliability enhancement. On the contrary, the MissRate of the Proposed_Mean method is lower than that of the ARQ method. In terms of stability, the MissRate variance of ARQ method is 24, while the MissRate variance of NBK method and the Proposed_Mean method is 1.12 and 0.64, respectively. This is because the ARQ method does not send message backups until an error occurs, resulting in burst transmission of message backups, thus an increased deadline miss rate.

Fig. 6 shows the channel utilization of the three methods. Compared with the NBK and the ARQ method, the Proposed_Mean method consumes up to 52% more channel utilization in topology with 10 slaves and 63% in topology with 20 slaves. This is because the Proposed_Mean method sends backups together with messages while the NBK and the ARQ method do not. Results also show that the variance of NBK, ARQ and proposed method in channel utilization is

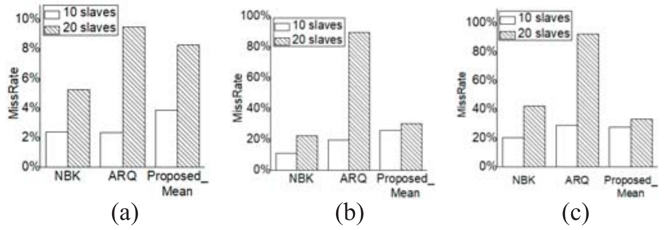

Fig. 5. Compare three methods in deadline miss rate. (a) $M = 5$. (b) $M = 10$. (c) $M = 20$.

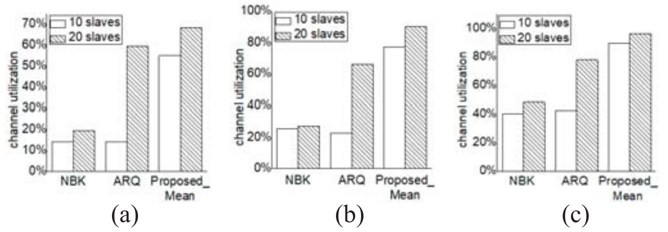

Fig. 6. Compare three methods in channel utilization. (a) $M = 5$. (b) $M = 10$. (c) $M = 20$.

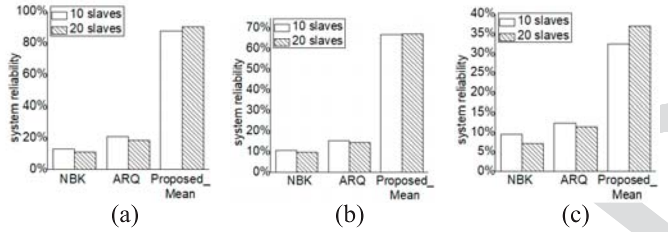

Fig. 7. Compare three methods in system reliability. (a) $M = 5$. (b) $M = 10$. (c) $M = 20$.

2.04, 28.97, and 1.69, respectively. Therefore, ARQ method is not suitable for systems of high stability requirements.

Fig. 7 shows that the Proposed_Mean method can effectively enhance system reliability by up to 74% when compared to the NBK and the ARQ method in 10 slave-topology and 79% in 20 slave-topology. The figure also shows that the system reliability slightly gets lower when the number of slaves, i.e., the system complexity, increases.

C. Proposed_Game Method Versus Proposed_Mean Method

Before we compare the Proposed_Game method with the Proposed_Mean method, we first verify the effectiveness of proposed schemes. We set channel utilization to be allocated (i.e., ΔNET) to 15%, 20%, 25%, and 30%, respectively. Then we compare the Proposed_Mean method and the Proposed_Game method with the ACE method in term of system reliability.

As shown in Fig. 8, the proposed schemes outperform the ACE method in terms of system reliability improvement, and the Proposed_Game is more effective in allocating channel utilization to enhance the system reliability. The reason is that the Proposed_Game method can allocate Pareto optimal channel resources to messages, while the ACE method just equally allocates available channel utilization to messages, which makes limited channel resources unavailable to messages of greater impact on system reliability.

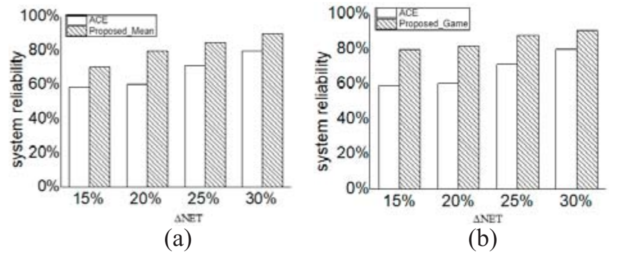

Fig. 8. Compare proposed mechanisms with allocate equally method in R_{sys}. (a) ACE versus Proposed_Mean. (b) ACE versus Proposed_Game.

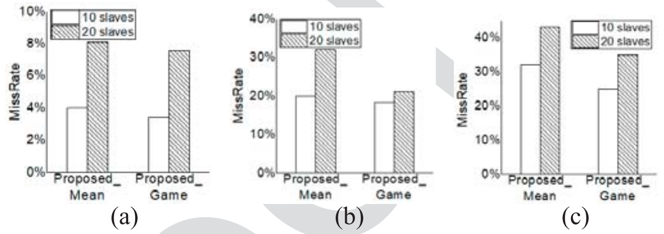

Fig. 9. Compare two proposed methods in deadline miss rate. (a) $M = 5$. (b) $M = 10$. (c) $M = 20$.

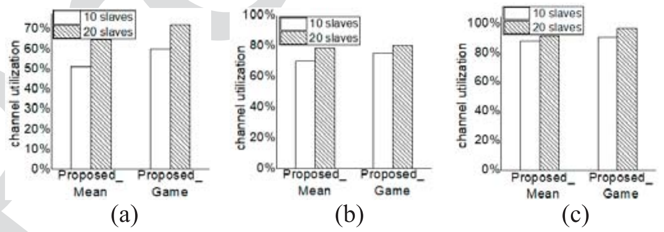

Fig. 10. Compare two proposed methods in channel utilization. (a) $M = 5$. (b) $M = 10$. (c) $M = 20$.

Then we compare the Proposed_Game method and Proposed_Mean method with respect to message deadline miss rate, channel utilization and system reliability. Fig. 9 shows the deadline miss rate of the Proposed_Mean method and the Proposed_Game method in two different topologies. It can be seen from the figure that compared to the Proposed_Mean, the Proposed_Game can reduce the message deadline miss rate by up to 11%. This is primarily due to the fact that the Proposed_Game method has better control over messages' backups, thus, fewer messages miss their deadlines.

Fig. 10 shows the channel utilization of the Proposed_Mean method and the Proposed_Game method in two different topologies. As compared to the Proposed_Mean, the Proposed_Game method consumes up to 9.2% more channel utilization in topology with 10 slaves and 5.3% in topology with 20 slaves. This is because the Proposed_Game method has better control over messages' backups and thus make better use of the available channel resources. Fig. 11 shows that when compared to the Proposed_Mean method, the Proposed_Game method can effectively enhance system reliability by up to 16% for 10 slave-topology and 9.1% for 20 slave-topology. As compared to the benchmarking methods, the Proposed_Game method can improve system reliability by up to 91% in topology with 10 slaves and 86% in topology with 20 slaves. The reason is that the Proposed_Game method allocates Pareto

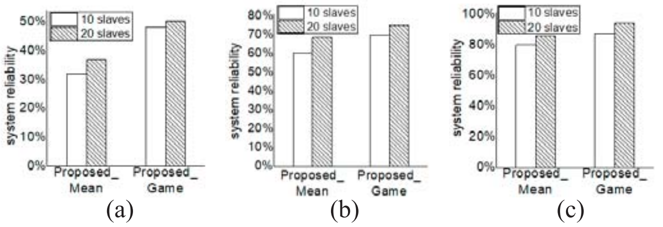

Fig. 11. Compare two proposed methods in system reliability. (a) $M = 5$. (b) $M = 10$. (c) $M = 20$.

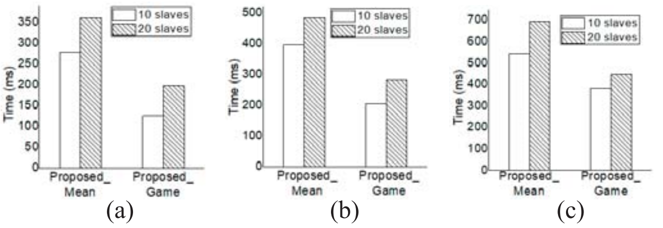

Fig. 12. Compare two proposed methods in execution time. (a) $M = 5$. (b) $M = 10$. (c) $M = 20$.

optimal channel resources to messages of greater impact on system reliability.

We also compare the execution time of the two proposed schemes. As shown in Fig. 12, the execution time of the Proposed_Mean method is up to 2.1 times of the Proposed_Game method for 10 slave-topology, and 1.8 times for 20 slave-topology. According to Section IV and V, the time complexity of Algorithm 5 is $O(M)$ and that of Algorithm 3 in the best case is $O(M)$. This is because the Proposed_Mean method needs to derive the number of messages' backup in multiple iterations, while the Proposed_Game can quickly obtain a Pareto optimal result by using the cooperative game.

VII. Conclusion

In this paper, we aim to enhance EtherCAT system reliability while meeting the timing constraint of real-time messages and resource constraint of the network channel. Our Proposed_Mean scheme adopts a PID-based feedback control mechanism that improves system reliability by adjusting the number of messages and their backups admitted into the system using inequality of arithmetic and geometric means method. In addition, in order to allocate channel resource faster and fairer, we design the Proposed_Game method, which optimize the Proposed_Mean method by using cooperative game theory and Nash bargaining solution. Simulation results show that the Proposed_Mean and Proposed_Game method improves system reliability by up to 79% and 91%, respectively, when compared to two benchmarking schemes. In addition, the Proposed_Game takes only about half the time of Proposed_Mean to derive the ultimate number of message backups.

References

[1] *IMS_CPS*. Accessed: Jul. 31, 2018. [Online]. Available: http://www.imscenter.net/cyber-physical-platform

[2] *Standard for Sensor/Actuator Network Communications for EtherCAT*, SEMI Standard E54.20, 2007.

[3] *Industrial Communication Networks—Fieldbus Specifications—Type 12 Elements (EtherCAT)*, IEC Standard 61158-2/3/4/5/6-12, 2007.

[4] *Industrial Communication Networks—Profiles—Part 2: Additional Fieldbus Profiles for Real-Time Networks Based on ISO/IEC 8802-3, Ed. 1.0*, IEC Standard 61784-2, 2007.

[5] *Industrial Automation Systems and Integration—Open Systems Application Integration Framework, EtherCAT Profiles*, ISO Standard 17545-4, 2007.

[6] Y. Lee, W. Lee, B. Choi, G. Park, and Y. Park, "Reliable software architecture design with EtherCAT for a rescue robot," in *Proc. IEEE Int. Symp. Robot. Intell. Sensors (IRIS)*, 2016, pp. 34–39.

[7] *CAN Bus*. Accessed: Jul. 31, 2018. [Online]. Available: https://en.wikipedia.org/wiki/CAN_bus

[8] M. Rostan, J. E. Stubbs, and D. Dzilno, "EtherCAT enabled advanced control architecture," in *Proc. IEEE/SEMI Adv. Semicond. Manuf. Conf. (ASMC)*, 2010, pp. 39–44.

[9] *PLC_EtherCAT*. Accessed: Jul. 31, 2018. [Online]. Available: www.advantech.com

[10] V. Q. Nguyen, N. V. P. Tran, H. N. Tran, K. M. Le, and J. W. Jeon, "A closed-loop stepper motor drive based on EtherCAT," in *Proc. Annu. Conf. IEEE Ind. Electron. Soc.*, 2017, pp. 3361–3365.

[11] S.-T. Yan, L. Zhang, Z.-X. Chen, J. Wang, and B. Wang, "A micro-grid control system based on EtherCAT," in *Proc. Inf. Technol. Comput. Eng. Manag. Sci.*, 2011, pp. 385–388.

[12] J. Ma, B. Xing, and S. Chen, "Multi-DOF motion control system design and realization based on EtherCAT," in *Proc. Int. Conf. Instrum. Meas. Comput. Commun. Control (IMCCC)*, 2016, pp. 727–732.

[13] L. Mathe, P. D. Burlacu, and R. Teodorescu, "Control of a modular multilevel converter with reduced internal data exchange," *IEEE Trans. Ind. Informat.*, vol. 13, no. 1, pp. 248–257, Feb. 2017.

[14] P. Burlacu *et al.*, "Implementation of fault tolerant control for modular multilevel converter using EtherCAT communication," in *Proc. IEEE Int. Conf. Ind. Technol.*, 2015, pp. 3064–3071.

[15] B. Allouche, A. Dequidt, L. Vermeiren, and P. Hamon, "Design and control of a sit-to-stand assistive device via EtherCAT fieldbus," in *Proc. IEEE Int. Conf. Ind. Technol.*, 2017, pp. 761–766.

[16] G. Cena, I. C. Bertolotti, S. Scanzio, A. Valenzano, and C. Zunino, "Evaluation of EtherCAT distributed clock performance," *IEEE Trans. Ind. Informat.*, vol. 8, no. 1, pp. 20–29, Feb. 2012.

[17] T. Maruyama and T. Yamada, "Spatial–temporal communication redundancy for high performance EtherCAT master," in *Proc. Emerg. Technol. Factory Autom. (ETFA)*, 2017, pp. 1–6.

[18] W. Xu, G. Xu, Z. Xi, and C. Zhang, "Distributed power quality monitoring system based on EtherCAT," in *Proc. China Int. Conf. Electricity Distrib.*, 2012, pp. 1–5.

[19] S.-M. Park, H. Kim, H.-W. Kim, C. Cho, and J.-Y. Choi, "Synchronization improvement of distributed clocks in EtherCAT networks," *IEEE Commun. Lett.*, vol. 21, no. 6, pp. 1–8, Jun. 2017.

[20] L. Bello, G. Patti, G. Alderisi, V. Patti, and V. Mirabella, "A flexible mechanism for efficient transmission of aperiodic real-time messages over EtherCAT networks," in *Proc. IEEE Workshop Factory Commun. Syst.*, 2014, pp. 1–8.

[21] S. Lim, L.-K. Jung, and J.-H. Kim, "A performance evaluation and task scheduling of EtherCAT networked soft motion control system," in *Proc. IEEE Int. Symp. Robot.*, 2014, pp. 1–5.

[22] X. Wu and L. Xie, "End-to-end delay evaluation of industrial automation systems based on EtherCAT," in *Proc. Conf. Local Comput. Netw. (LCN)*, 2017, pp. 70–77.

[23] H. Jia, P. Yao, B. Li, and X. Tian, "Four axes wear-resistant coating testing system based on EtherCAT," in *Proc. Chin. Autom. Congr. (CAC)*, 2017, pp. 2842–2844.

[24] *EtherCAT*. Accessed: Jul. 31, 2018. [Online]. Available: https://en.wilipedia.org/wiki/EtherCAT

[25] S. T. Andrew and J. Wetherall, *Computer Networks*, 5th ed. New York, NY, USA: Morgan Kaufmann, 2011, p. 225.

[26] J. Alzubi, O. Alzubi, and T. Chen, *Forward Error Correction Based on Algebraic-Geometric Theory*, 1st ed. Berlin, Germany: Springer, 2014, p. 12.

[27] T. Wei, P. Mishra, K. Wu, and J. Zhou, "Quasi-static fault-tolerant scheduling schemes for energy-efficient hard real-time systems," *J. Syst. Softw.*, vol. 85, no. 6, pp. 1386–1399, 2012.

[28] D. E. Lee, Q. V. Nguyen, T.-W. Kim, J. Y. Moon, and J. W. Jeon, "Development of independent EtherCAT slave module and application to closed loop step motor drive with multi-axis," in *Proc. IEEE Int. Conf. Comput. Commun. Autom.*, 2017, pp. 912–917.

[29] *EtherCAT Specification—Part 3: Data Link Layer Service Definition*, EtherCAT Technol. Group, Nuremberg, Germany, 2017.

[30] R. Delgado, S.-Y. Kim, B.-J. You, and B.-W. Choi, "An EtherCAT-based real-time motion control system in mobile robot application," in *Proc. 13th Int. Conf. Ubiquitous Robots Ambient Intell. (URAI)*, 2016, pp. 710–715.

[31] G. Yang, R. Delgado, and B. W. Choi, "Convolution-based time optimal path planning for a high-curvature Bezier curve considering possible physical limits," *Int. J. Control Autom.*, vol. 8, no. 9, pp. 325–336, 2015.

[32] *Bit Error Rate*. Accessed: Jul. 31, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Bit_error_rate

[33] S. T. Andrew and J. Wetherall, *Computer Networks*, 5th ed. New York, NY, USA: Morgan Kaufmann, 2011, p. 264.

[34] C. Lu, J. A. Stankovic, G. Tao, and S. H. Son, "Design and evaluation of a feedback control EDF scheduling algorithm," in *Proc. IEEE Real Time Syst. Symp.*, 1999, pp. 56–67.

[35] E. Beckenbach and R. Bellman, *Introduction to Inequalities*. Dallas County, TX, USA: Random House, 1975, p. 54.

[36] *OMNeT++*. Accessed: Jul. 31, 2018. [Online]. Available: https://omnetpp.org/

[37] H. Yaiche, R. R. Mazumdar, and C. Rosenberg, "A game theoretic framework for bandwidth allocation and pricing in broadband networks," *IEEE/ACM Trans. Netw.*, vol. 8, no. 5, pp. 667–678, Oct. 2000.

[38] A. Muthoo, *Bargaining Theory With Applications*. Cambridge, U.K.: Cambridge Univ. Press, 1999.

[39] J. Nash, "The bargaining problem," *Econometrica*, vol. 18, no. 2, pp. 155–162, 1950.

[40] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[41] S. Qin and X. Xue, "A two-layer recurrent neural network for nonsmooth convex optimization problems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 6, pp. 1149–1160, Jun. 2015.

[42] *Earliest Deadline First Scheduling*. Accessed: Jul. 31, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Earliest_deadline_first_scheduling

[43] R. M. Pathan, "Design of an efficient ready queue for earliest-deadline-first (EDF) scheduler," in *Proc. Design Autom. Test Europe Conf. Exhibit. (DATE)*, 2016, pp. 293–296.

Liying Li received the B.S. degree from the Department of Computer Science and Technology, East China Normal University, Shanghai, China, in 2017, where she is currently pursuing the master's degree.

Her current research interests include cyber physical systems and IoT resource management.

Peijin Cong received the B.S. degree from the Department of Computer Science and Technology, East China Normal University, Shanghai, China, in 2016, where she is currently pursuing the master's degree.

Her current research interest includes power management in mobile devices and edge computing.

Kun Cao is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, East China Normal University, Shanghai, China.

His current research interests include high performance computing, multiprocessor systems-on-chip, and cyber physical systems.

Junlong Zhou is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, East China Normal University, Shanghai, China.

He was a Research Visitor with the University of Notre Dame, Notre Dame, IN, USA. His current research interests include real-time embedded systems, cyber physical systems, and cloud computing. He has published a dozen of papers in the above areas.

Mr. Zhou was a recipient of the Reviewer Award from the *Journal of Circuits, Systems, and Computers* in 2016. He is an Active Reviewer of several international journals, including the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, the *Journal of Systems and Software*, and the *Journal of Circuits, Systems, and Computers*.

Tongquan Wei (M'11) received the Ph.D. degree in electrical engineering from Michigan Technological University, Houghton, MI, USA, in 2009.

He is currently an Associate Professor with the Department of Computer Science and Technology, East China Normal University, Shanghai, China. His current research interests include Internet of Things, cloud computing, edge computing, and design automation of intelligent and CPS systems.

Dr. Wei has been serving as a Regional Editor for the *Journal of Circuits, Systems, and Computers* since 2012.

Mingsong Chen received the B.S. and M.E. degrees from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2003 and 2006, respectively, and the Ph.D. degree in computer engineering from the University of Florida, Gainesville, FL, USA, in 2010.

He is currently a Full Professor with the Department of Embedded Software and Systems, East China Normal University, Shanghai, China. His current research interests include design automation of cyber-physical systems, formal verification techniques, and mobile cloud computing.

Shiyan Hu received the Ph.D. degree in computer engineering from Texas A&M University, College Station, TX, USA, in 2008.

He is an Associate Professor with Michigan Technological University, Houghton, MI, USA, and he was a Visiting Associate Professor with Stanford University, Stanford, CA, USA, from 2015 to 2016. His current research interests include cyber-physical systems (CPS), CPS security, data analytics, and computer-aided design of very large scale integration circuits. He has published over 100 refereed papers in the above areas. He is a fellow of IET.

Xiaobo Sharon Hu (S'85–M'89–SM'02–F'16) received the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1989.

She is a Professor with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, USA. Her current research interests include real-time embedded systems, low power system design, and computing with emerging technologies. She has published over 250 papers in the above areas. She is a fellow of the IEEE.