Feedback Control of Real-Time EtherCAT Networks for Reliability Enhancement in CPS

Liying Li[†], Peijin Cong[†], Kun Cao[†], Junlong Zhou[*], Tongquan Wei[†], Mingsong Chen[§], Xiaobo Sharon Hu[‡]
[†]Department of Computer Science and Technology, East China Normal University
[*]School of Computer Science and Engineering, Nanjing University of Science and Technology
[§]Shanghai Key Lab of Trustworthy Computing, East China Normal University
[‡]Department of Computer Science and Engineering, University of Notre Dame

***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 PID-based feedback control scheme that aims at enhancing reliability of networked systems under timing and system resource constraints. Instead of automatic repeat request method (ARQ), a forward error control technique is introduced to achieve the required system reliability at a lower deadline miss rate of messages. The PID-based feedback control scheme can also improve the stability of a system in terms of deadline miss rate in the presence of bursty errors. Simulation results show that the proposed scheme can achieve reliability enhancement of up to 79% compared to benchmarking methods.**

I. Introduction

A cyber physical system (CPS) is composed of various physical and computing components that interact through embedded communication capabilities and is gaining importance in the era of industry 4.0. 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 can be enabled by powerful EtherCAT [1].

EtherCAT is an industrial Ethernet technology standardized by ISO [2]. It is one of the fastest real-time industrial Ethernet networks available. EtherCAT provides high data transmission efficiency combined with high speed and accuracy clock synchronization, which are two key features for CPS applications.

Extensive research efforts have been made to investigate EtherCAT and its deployment in high performance industrial applications. For example, EtherCAT has been used in the design of modular multilevel converters for high voltage conversion in power electronics [4], [5], and various assisted devices that target people with disabilities following a stroke [6]. EtherCAT is used in these applications as a high-speed and low-overhead communication platform.

This work was partially supported by Shanghai Municipal Natural Science Foundation (Grant No. 16ZR1409000) and Natural Science Foundation of China (Grant No. 61672230). T. Wei is the corresponding author, Email: tqwei@cs.ecnu.edu.cn.

Precise clock synchronization is a key feature that makes EtherCAT appealing in applications above and many other domains like motion control [7]. Synchronization in EtherCAT is achieved by means of a distributed clock mechanism, which can effectively reduce the implementation costs of EtherCAT devices. The EtherCAT synchronization performance can be improved by using various techniques such as drift compensation [8], and can be evaluated by conducting extensive experimental measurements [7].

The timeliness of EtherCAT networked system is of particular importance to real-time applications like compliance control in robotics. Bello et al. [9] propose 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 [10], a networked soft motion control system with EtherCAT is designed and evaluated. The timeliness of the presented control method is experimentally validated.

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. 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 considers reliability in addition to timeliness.

In this paper, we propose a feedback control based scheme to enhance system reliability under the timing constraint for messages and the resource constraint for network channels. We first investigate reliability modeling of EtherCAT networks from the aspects of transmission links and processing nodes, then 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 utilziation. Extensive simulation based experiments show that the proposed control scheme can achieve reliability enhancement of up to 79% compared to benchmarking methods.

The rest of the paper is organized as follows. Section II introduces EtherCAT system architecture and models, Section III describes in details the proposed feedback control scheme, Section IV presents the experimental results, and Section V concludes the paper.

978-3-9819263-0-9/DATE18/©2018 EDAA

II. System Architecture and Models

The focus of this paper is on reliability enhancement of an EtherCAT system in the presence of transient faults. We are interested in the ring topology which includes one master and multiple slaves connected by the standard Ethernet cable. Below, we present the various models used in this paper.

A. System Architecture

EtherCAT is one of the real-time Ethernet communication topologies and is included as a part of the ISO standards [2]. 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. 1. The system is composed of one master and N slaves connected by the standard Ethernet cable. We refer to the master and the slaves as computing nodes. The master cyclically sends a standard Ethernet frame containing several sub-telegrams or messages (see Fig. 2) to slaves, while the slaves are responsible for reading or/and writing the data addressed to them when the frame passes through them, and processing data on the fly. After the last slave transmits the frame back to the master, the cycle starts again.

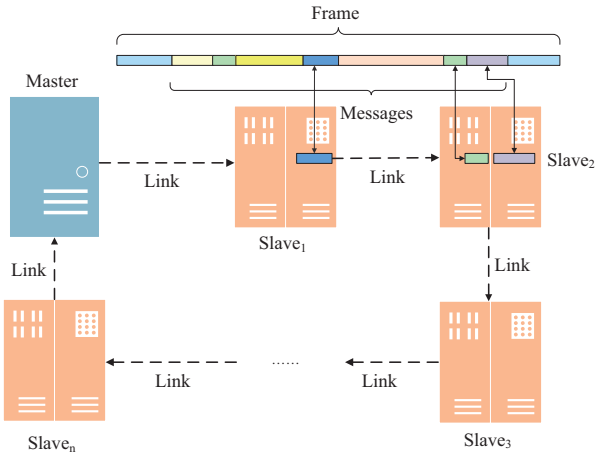


Fig. 1: The ring topology of an EtherCAT system.

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 sub-telegrams (or messages) [11]. Fig. 2 illustrates the fields of a standard IEEE 802.3 Ethernet frame of Ethertype 0x88a4. 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 sub-telegram in the EtherCAT frame, and we use messages and sub-telegrams interchangeably in the following sections. Real-time message τ_i $(1 \leq i \leq M)$ is associated with $\{T_i, D_i, L_i, RG_i\}$, where T_i is the period of τ_i, D_i represents the deadline of τ_i, L_i denotes the length of τ_i, and RG_i is the reliability target of the message τ_i. The reliability requirement of each message may be different, so different reliability target can be determined according to the different reliability requirement, that is, the number of different message's backups.

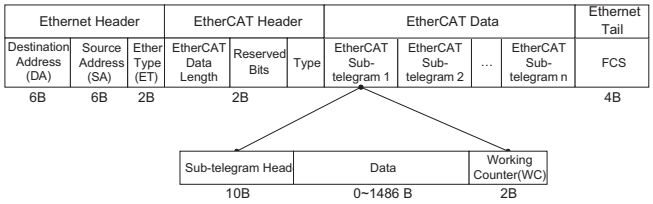


Fig. 2: The structure of an EtherCAT frame.

C. Reliability Model

A forward error control technique [13] is adopted in this paper to provide fault-tolerance. Unlike the automatic repeat request (ARQ) technique that re-sends messages when a fault occurs [12], the forward error control technique sends and executes original messages and their backups at the same time [13]. Since messages in a 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.

Soft error model for nodes: Soft errors mainly result from transient faults. Poisson distribution is widely used to model the occurrences of transient faults in computing nodes [14]. Let λ_j be the average fault occurence rate of computing node j for $0 \leq j \leq N^*$, 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. Let Δl and E_j denote the unit length of a message that a node can process on the fly at a time, and the processing time of a unit length message at node j, respectively. 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

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

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 successfully at all nodes, denoted by $P_{i,nodes}$, is calculated as

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

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

*N is the number of nodes in the the system and the concerned node is the master when j=0.

links, which is denoted by $P_{i,links}$, can be modeled as [15]

$$P_{i,links} = e^{-\theta \cdot t_i}, \quad (4)$$

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. P_i is obtained by

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

The reliability of a message is defined as the probability that the message issued by the master is 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, then the reliability, which is denoted by $R_i(k_i)$, is expressed as

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

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

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

III. Feedback Control Scheme for Reliability Enhancement

Our goal is to design a fault- tolerance message scheduling scheme in order to enhance the overall reliability of the EtherCAT system, R_sys. In this section, we first formulate the problem to be tackled, followed by an overview of the proposed control scheme. We then present in details the working mechanism of the proposed controller that integrates a PID controller, Message Access (MA) controller, Message Backup (MB) controller and Earliest Deadline First (EDF) scheduler.

A. Problem Definition and Overview of the Proposed Scheme

As explained above, we focus on designing a fault-tolerance message scheduling strategy for EtherCAT systems of ring topology. 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.

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,

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

where $MissRate$ is the deadline miss rate of messages during one sampling period of the proposed controller, ε is a user-specified positive constant that indicates the threshold for deadline miss rate, R_i is the reliability of message τ_i, and NET is the total channel utilization of messages in the network. Channel utilization is defined as the percentage of the net bit rate (in bit/s) of a channel used for the actually achieved throughput [16]. Message τ_i is required to meet its reliability target RG_i. The objective function R_{sys} is given in Eq. (7).

Overview of the proposed control scheme: Fig. 3 shows the overall structure of our proposed feedback control system. It consists of a main controller, PID controller, MA controller, MB controller and 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 Eq. (8). ΔNET is the total amount of channel utilization that should be added into (when $\Delta NET>0$) or reduced from (when $\Delta NET<0$) the system. The main controller calls the MB and MA controller sequentially to accommodate the channel utilization of ΔNET. The MB controller can accommodate the channel utilization of the system by increasing/decreasing the number of backups of messages in the ACCEPTED queue. If the MB controller can not accommodate all of the ΔNET, the main controller calls the MA controller to control message flow into the ACCEPTED queue such that the remaining ΔNET is accommodated. 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.

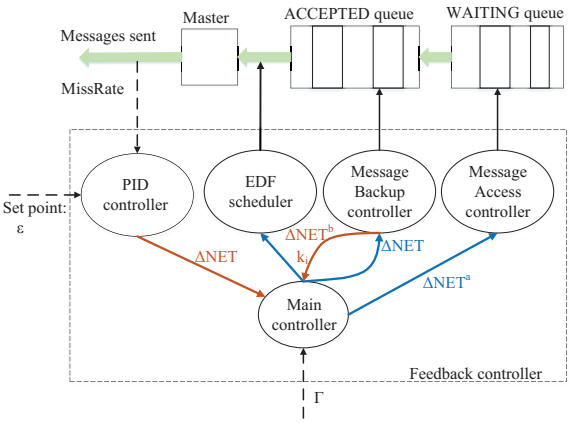


Fig. 3: Overview of the proposed feedback control system.

B. Feedback Control Mechanism for Reliability Enhancement

In this subsection, we first detail the working mechanism of the PID controller, MB controller and MA controller, respectively, then describe the mechanism of the Main controller which combines operations of these three controllers to achieve a comprehensive control strategy for enhancing the system reliability.

PID controller: A general 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 $MissRate$. It then computes the control variable ΔNET in terms of requested channel utilization using the control equation given by [17]

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

where $err(t)$ is the difference between the threshold for system deadline miss rate and the current system deadline miss rate, that is, $err(t) = \varepsilon - MissRate$. 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 $(err(t) - err(t - DW))/DW$.

Algorithm 1: PID control algorithm

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

```
1  do
2      PID controller samples messages to derive MissRate;
3      Calculate ΔNET using Eq. (8);
4      return ΔNET;
5  while (MissRate>ε);
6  return ΔNET;
```

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

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 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 reliability of τ_i, R_i, can be derived by using Eq. (6). Since every message's reliability is nonnegative, according to the inequality of arithmetic and geometric means [18], we have

$$\left(\frac{R_1 + R_2 + ... + R_M}{M}\right)^M \geq R_1 \cdot R_2 \cdot ... \cdot R_M = R_{\text{sys}},$$

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

The MB controller is designed based on the above principle to maximize 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 2, the MB controller takes ΔNET as input and returns ΔNET^b to the main controller. ΔNET^b is the maximum incremental channel utilization that can be accommodated by the MB controller at most.

Algorithm 2: Message Backup control algorithm

Input: Total channel utilization to be accommodated, ΔNET.
Output: The portion of accommodated channel utilization, ΔNET^b.

```
   // initialization
1  ΔNET^b = 0;
2  if ΔNET > 0 then
3      Compute mean R_avg of message reliabilities in ACCEPTED queue;
4      Determine the number m of messages for R_i < R_avg in the queue;
5      Sort the m messages in the queue in the ascending order of R_i;
6      i = 1;
7      while ΔNET^b<ΔNET do
8          Increment the number of message τ_i's backup by 1;
9          ΔNET^b = ΔNET^b + NET_i1;
           // NET_i1 is given by Eq. (9)
10         i++;
11         if i = m + 1 then
12             i = 1;
13         end
14     end
15 end
16 else
17     Compute mean R_avg of message reliabilities in ACCEPTED queue;
18     Determine the number m of messages for R_i > R_avg in the queue;
19     Sort the m messages in the queue in the ascending order of R_i;
20     i = m;
21     while ΔNET^b > ΔNET do
22         Decrement the number of message τ_i's backup by 1;
23         ΔNET^b = ΔNET^b − NET_i1;
           // NET_i1 is given by Eq. (9)
24         i−−;
25         if i = 0 then
26             i = m;
27         end
28     end
29 end
30 return ΔNET^b, the number of messages' backup ;
```

Algorithm 2 works as follows. It initializes ΔNET^b to 0 in line 1. For the case of $\Delta NET > 0$, the algorithm calculates the average reliability of messages in the ACCEPTED queue (denoted by R_{avg}), picks the m messages for $R_i < R_{avg}$ and $1 \leq i \leq m$, and sorts the m messages in the queue in the ascending order of R_i (line 3 to line 5). When not all of ΔNET is accommodated by MB controller (i.e. ΔNET^b<ΔNET), the algorithm increments the number of message τ_i's backup by 1, updates ΔNET^b to $\Delta NET^b + NET_{i1}$ and increments i by 1. If all the m messages have been updated by increasing a backup and ΔNET is not used up yet, the algorithm resets $i = 1$ and repeats the accommodation process (line 7 to line 15). 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 c copies,

NET_{ic} is given by

$$NET_{ic} = \frac{\left(\sum_{j=0}^{N} E_j + t_i\right)(c+1)}{T_i}, \quad (9)$$

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 Eq. (9).

In the case of $\Delta NET < 0$, the algorithm works the same as the case of $\Delta NET > 0$ except that backups of messages satisfying $R_i > R_{avg}$ for $1 \leq i \leq m$ are iteratively dismissed from the system. In the end, the MB controller returns to the main controller the updated number of messages' backups and the portion of channel utilization that can be accommodated by the MB controller at the most (ΔNET^b).

Message Access controller: The MA controller is responsible for controlling the flow of messages with no backups 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 ΔNET^a be the portion of the channel utilization ΔNET that is left for the MA controller to accommodate. ΔNET^a is calculated as $\Delta NET^a = \Delta NET - \Delta NET^b$. Given ΔNET^a, the MA controller admits message τ_i with no backups if the condition $\Delta NET^a - NET_{i0} > 0$ holds. NET_{i0} is the channel utilization of message τ_i with no backups, and can be calculated by using Eq. (9) by setting c to 0. Once τ_i is admitted, ΔNET^a is updated to $\Delta NET^a - NET_{i0}$. 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 τ_i cannot be admitted even with no backups. The rejected messages remain in the WAITING queue. Algorithm 3 below describes the working mechanism of the MA controller.

Algorithm 3: Message Access control algorithm

```
Input: ΔNET^a, the portion of ΔNET that cannot be accommodated
       by the MB controller; M, the number of messages in
       ACCEPTED queue.
1  if ΔNET^a>0 then
2      while ΔNET^a>0 do
3          Sort messages in WAITING queue according to the EDF
           policy;
4          for head message in WAITING queue do
5              if ΔNET^a − NET_i0>0 then
                   // NET_i0 is given by Eq. (9)
6                  ΔNET^a = ΔNET^a - NET_i0 ;
7                  Dequeue head message from WAITING queue;
8                  Enqueue the message to ACCEPTED queue;
9                  M++;
10             end
11         end
12     end
13 end
14 else
15     Messages remain in WAITING queue;
16 end
```

Main controller: 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 and the period is determined by the minimum sampling interval.

Fig. 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 accommodation (ΔNET), the portion of channel utilization accommodated by MB controller (ΔNET^b), and the updated numbers of message backups. The algorithm first determines the number of backups for each individual message τ_i based on the reliability targets (RG_i) and Eq. (6) (Line 1-3). It then calls the PID controller to calculate the deadline miss rate $MissRate$ and ΔNET (Line 4), calls the MB algorithm to derive ΔNET^b and updated message backups (Line 5), and calculates ΔNET^a (Line 6). The MA algorithm is finally called to accommodate ΔNET^a if $\Delta NET^a \neq 0$. In the end, the EDF scheduler is called (Line 10) to dispatch messages to the master for processing.

Algorithm 4: Main control algorithm

```
Input: The message set Γ; The total channel utilization to be
       accommodated (ΔNET); The portion of accommodated
       channel utilization (ΔNET^b); The updated number of message
       backups.
   // Initialize the number of messages' backups
1  for τ_i ∈ Γ do
2      Calculate the number of message τ_i's backups based on τ_i's
       reliability target (RG_i) and Eq. (6);
3  end
4  Call PID (Algorithm 1) to get ΔNET;
5  Call MB (Algorithm 2) to get ΔNET^b and updated message backups;
6  Calculate ΔNET^a using ΔNET^a = ΔNET − ΔNET^b;
7  if ΔNET^a ≠ 0 then
8      Call MA (Algorithm 3) to accommodate ΔNET^a;
9  end
10 Call EDF scheduler to dispatch messages;
```

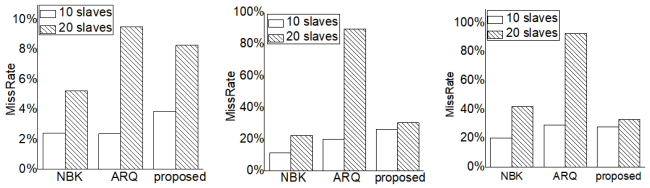
IV. Simulation-based Evaluation

Extensive simulation-based experiments have been conducted to validate the effectiveness of the proposed scheme. The simulations were conducted on a machine equipped with 2.4GHz Intel i7 quad-core processor and 8GB 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 [19]. 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 [17], coefficients C_P, C_I and C_D of the PID controller are set to 0.5, 0.005 and 0.1, respectively, and 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 reliability target RG_i, period T_i (in time units), deadline D_i (in time

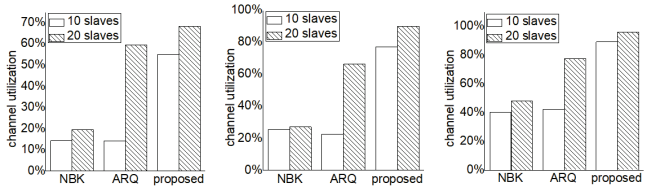
units) and length L_i (in bytes) of message τ_i are randomly generated in the interval of (0,1), (200,500), (200,800) and (12,1498), respectively. The unit length that a computing node can process at a time on the fly (Δl) is set to 4 Bytes [9].

We compare the proposed scheme with two other methods in terms of message deadline miss rate, channel utilization and system reliability, respectively. The first method, which is referred to as NBK, does not take any anti-error measures like message backups when errors occur. The second method is automatic repeat request mechanism, referred to ARQ. It resends messages only when errors occur.



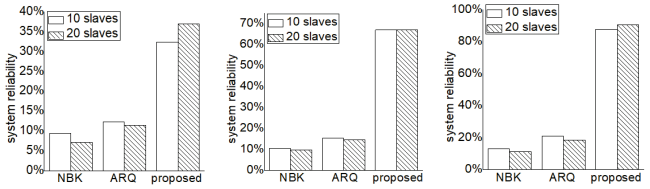
(a) M = 5. (b) M = 10. (c) M = 20.

Fig. 4: Compare three methods in deadline miss rate.



(a) M = 5. (b) M = 10. (c) M = 20.

Fig. 5: Compare three methods in channel utilization.



(a) M = 5. (b) M = 10. (c) M = 20.

Fig. 6: Compare three methods in system reliability.

Fig. 4 shows the deadline miss rate of the three methods. It can be seen from the figure that the $MissRate$ of the proposed scheme is higher than that of the NBK method in both topologies. This is primarily due to the fact that NBK method does not use any backups for reliability enhancement. On the contrary, the $MissRate$ of the proposed scheme 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 proposed scheme is 1.119 and 0.64, respectively. This is because the ARQ method does not send message backups until an error occurs, resulting a burst transmission of message backups, thus an increased deadline miss rate.

Fig. 5 shows the channel utilization of the three methods. Compared with NBK and ARQ method, the proposed feedback control method consumes up to 51.8% more channel utilization in topology with 10 slaves and 62.9% in topology with 20 slaves. This is because the proposed method sends backups together with messages while NBK and ARQ do not. These results also show that the variance of NBK, ARQ and proposed method in channel utilization is 2.04, 28.97 and 1.69, respectively. Therefore, ARQ method is not suitable for systems of high stability requirements.

Fig. 6 shows that the proposed scheme can effectively enhance system reliability by up to 74% when compared to the NBK and ARQ method in 10 slaves topology and 79% in 20 slaves topology. The figure also shows that the system reliability slightly gets lower when the number of slaves, i.e., the system complexity, increases.

V. Conclusion

In this paper, we aim to enhance EtherCAT system reliability while meeting the timing constraint of real-time messages and resource constraint of network channel. Our proposed 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. Extensive simulation experiments have been conducted to validate the effectiveness of our scheme. Simulation results show that the proposed scheme can improve system reliability by up to 79% when compared to two alternative schemes.

References

[1] "IMS_CPS"[Online].Available:http://www.imscenter.net/cyber-physical-platform.

[2] ISO 17545-4 Industrial automation systems and integration - Open systems application integration framework, EtherCAT profiles.

[3] "PLC_EtherCAT" [Online]. Available: www.advantech.com.

[4] L. Mathe, P. Burlacu, and R. Teodorescu, "Control of Modular Multilevel Converter with reduced internal data exchange," *IEEE Transactions on Industrial Informatics*, pp. 1-8, 2017.

[5] P. Burlacu, L. Mathe, M. Rejas, H. Pereira, A. Sangwongwanich, and R. Teodorescu, "Implementation of fault tolerant control for modular multilevel converter using EtherCAT communication," *IEEE International Conference on Industrial Technology*, pp. 3064-3071, 2015.

[6] B. Allouche, A. Dequidt, L. Vermeiren, and P. Hamon, "Design and control of a sit-to-stand assistive device via EtherCAT fieldbus," *IEEE International Conference on Industrial Technology*, pp. 761-766, 2017.

[7] G. Cena, I. Bertolotti, S. Scanzio, A. Valenzano, and C. Zunino, "Evaluation of EtherCAT Distributed Clock Performance," *IEEE Transactions on Industrial Informatics*, pp. 20-29, 2012.

[8] S. Park, H. Kim, H. W. Kim, C. Cho, and J. Choi. "Synchronization Improvement of Distributed Clocks in EtherCAT Networks," *IEEE Communications Letters*, pp. 1-8, 2017.

[9] 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," *IEEE Workshop on Factory Communication Systems*, pp. 1-8, 2014.

[10] S. Lim, K. Jung, and J. Kim, "A Performance Evaluation and Task scheduling of EtherCAT Networked Soft Motion Control System," *IEEE International Symposium on Robotics*, pp. 1-5, 2014.

[11] "EtherCAT"[Online].Available:https://en.wilipedia.org/wiki/EtherCAT.

[12] S. Andrew, Tanenbaum, and J. Wetherall, Computer Networks, 5th Edition, Morgan Kaufmann, pp. 225, 2011.

[13] J. Alzubi, O. Alzubi, and T. Chen, Forward Error Correction Based On Algebraic-Geometric Theory, 1st Edition, Springer, pp. 12, 2014.

[14] T. Wei, P. Mishra, K. Wu and J. Zhou, "Quasi-static fault-tolerant scheduling schemes for energy-efficient hard real-time systems," *Journal of Systems & Software*, pp. 1386-1399, 2012.

[15] "Bit error rate" [Online]. Available: https://en.wikipedia.org/wiki/Bit_error_rate.

[16] S. Andrew, Tanenbaum, and J. Wetherall, Computer Networks, 5th Edition, Morgan Kaufmann, pp. 264, 2011.

[17] C. Lu, J. Stankovic, G. Tao, and S. Son, "Design and Evaluation of a Feedback Control EDF Scheduling Algorithm," *IEEE Real-Time Systems Symposium Proceedings*, pp. 56-67, 1999.

[18] E. Beckenbach, and R. Bellman, Introduction to Inequalities, Random House, pp. 54, 1975.

[19] "OMNeT++" [Online]. Available: https://omnetpp.org/.