Preventing Data Loss in Linux-Based EtherCAT Master

Cheol-Jin An, Hyun-Chul Yi, Hyoung-Woo Kim, Sung-Mun Park, and Joon Young Choi

***Abstract*—A receiving and sending algorithm is proposed to prevent the packet loss in Linux-based EtherCAT master. The algorithm is implemented by modifying the IgH EtherCAT master stack module, and can be used in the application program for EtherCAT master. The performance of the modified EtherCAT master is verified by experimenting it with the EtherCAT slave hardware provided from Texas Instruments Inc. Experiment results show the improved data throughput as well as the compensation of communication jitter compared to the existing IgH EtherCAT open master stack.**

***Index Terms*—Communication jitter, data receiving and sending algorithm, EtherCAT, packet loss, throughput.**

I. Introduction

In recent years, the software-based network master stacks with non-dedicated computer have been increasingly used as industrial automation systems [1]. In some cases, this trend led to various approaches related to both the implementation and evaluation of software-based master using open-source operating system (OS) supporting the real-time capability [2], [3].

As for an industrial fieldbus, EtherCAT, an Ethernet-based high-performance fieldbus system, has been widely used for the industrial network solution [4]. It enables the network control system to be built without additional customized interface card on the controller. Moreover, this kind of systems can be built entirely from open-source components due to the availability of open-source EtherCAT master components like IgH EtherLab [5].

Several different real-time extensions of Linux have been chosen for the EtherCAT network applications in the recent past [6]. Their real-time performance has been thoroughly evaluated because Linux has sometimes been used as a platform to develop and benchmark novel scheduling paradigms and algorithms [7], [8]. However, they do not consider the improvement of the inner stack program and the compensation of communication jitter incurred by a periodic networked control task for receiving and sending process of EtherCAT frame.

In this study, we propose a receiving and sending algorithm to improve the data throughput of Linux-based EtherCAT master. The modified master stack eliminates the data skip which occurs in process data handling of high-speed EtherCAT communication. The performance of Linux-based master stack applied with the proposed receiving and sending algorithm is experimented with the EtherCAT slave controllers, TMDSICE3359, provided in Texas Instrument. The experimental results show that the proposed solution not only enhances data throughput but also have a positive effect on the communication system in terms of transfer period compared to existing IgH EtherCAT master stack.

Manuscript received Aril 1, 2015; revised June 25, 2015. This work was not supported by any organization

The authors are with the Department of Electrical and Computer Engineering, Pusan National University, Busan, 609-735, Korea (e-mail: cja@pusan.ac.kr, hcy@pusan.ac.kr, hwkim0314@pusan.ac.kr, psm2689@pusan.ac.kr, jyc@pusan.ac.kr).

II. Description of the EtherCAT Protocol

EtherCAT is an industrial protocol built on the Ethernet specifications. The particular method of EtherCAT to process frames makes it possible to be the fastest industrial Ethernet technology [4], [9]. Key functional principle lies in how its nodes process Ethernet frames. Each node reads the data addressed to it and writes its data back to the frame all while the frame is moving downstream. This leads to improved bandwidth utilization while also eliminating the need for switches or hubs [4].

Data communication structure of EtherCAT network can be characterized as Master and Slave. The EtherCAT master sends a telegram that passes through each node. Each of EtherCAT slave devices reads the data addressed to it expeditiously and inserts its data in the frame moving downstream. The frame is delayed only by hardware propagation delay times. The last node in a segment or branch detects an open port and sends the message back to the mater using full duplex feature of Ethernet.

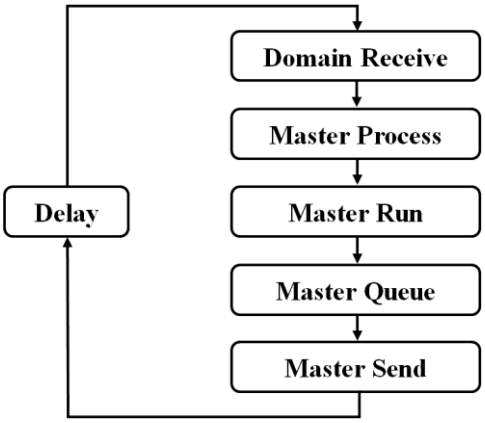


Fig. 1. Flowchart of receiving and sending algorithm at EtherCAT master.

The receiving and sending algorithm of master software between the data-link layer and the application layer can be explained as in Fig. 1 [10]. In this study, IgH EtherCAT master stack which is the qualified open-source master software is used as the basis of analysis. First, the program receives the datagrams sent from the slaves periodically. Second, the program processes the received process data and queues the domain datagrams. After all of the applications in the master finish, the program put the new process data in the send datagram queue. Then, the master sends the queued

datagrams to the slaves. After then, master sleeps for the established time by a Linux function, *clock_nanosleep()*, and one cycle of the master process is finished. This cyclic process continuously repeats until the program is finished.

III. Problem Statement

The transfer period of EtherCAT master is affected by the communication jitter depend on the hardware performance [5], [11], [12]. Communication jitter can be classified into two kinds; the transfer period to be shorter than the defined value and to be longer than the defined value. Both kinds of jitters imply a potential risk on network system and lead to instability of the communication between master and slaves.

The focus of this paper is the case of transfer period being shorter than defined value. When the transfer period is shorter than the previous cycle time, the receive command is performed in the state that master have not yet received new data from slaves. Then, the master is not able to receive the data consequentially. Cycle time is the time taken for data to pass through the network and return to the master [13].

Fig. 2 shows that the difference in data exchange sequence between a normal circumstance and the shorten transfer period circumstance. In this case, the process data keeps being the same as in the last cycle because it is not erased by the domain. When the domain datagrams are queued again, the master notices that they are already queued. It is noticed by domain process as the working counter of processing datagrams is not increased. In the case of IgH EtherCAT master stack, the master is programed to print warning messages that inform datagrams were skipped [5]. It is able to be confirmed by the system log as shown in Fig. 3.

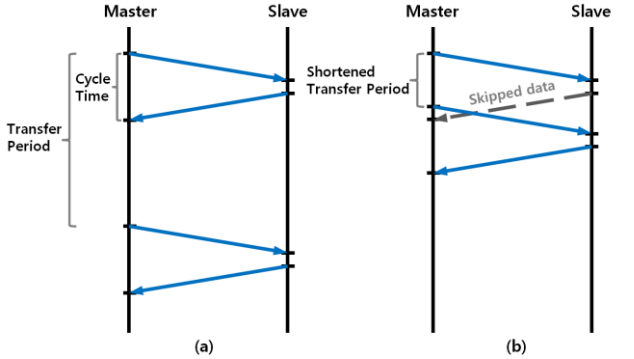


Fig. 2. Data exchange sequence when (a) the network works normally (b) transfer period is shorter than cycle time.

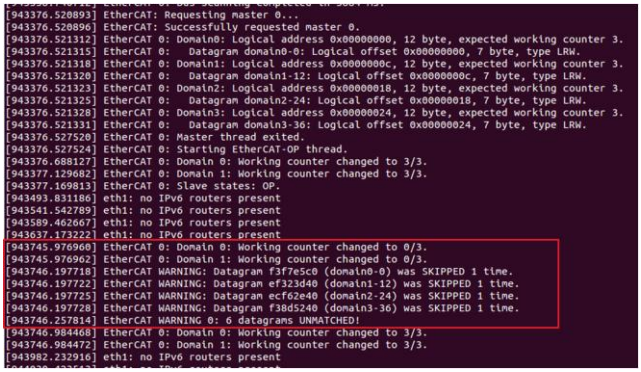


Fig. 3. System log including data warning messages.

Although the transfer period is set as a constant value, the problem usually occurs since the performance of hardware is not able to perform the programmed tasks. Therefore, the problem could get solved with sufficiently high-performance hardware [5]. However, we suggest the solution of this problem only with modifying the master stack by proposed receiving process.

IV. Proposed Solution

In this section, we propose a simple method to overcome the defect of EtherCAT master outlined above. The proposed algorithm is depicted in Fig. 4.

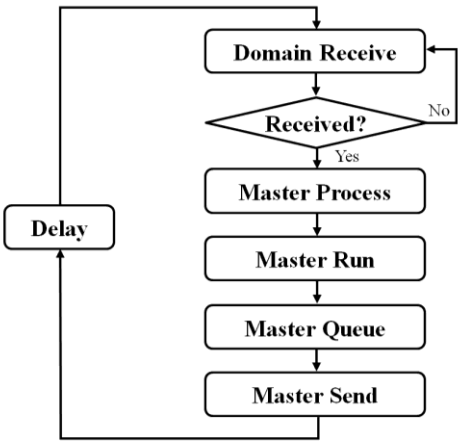


Fig. 4. Flowchart of the proposed algorithm.

A step for decision whether receiving has been completed or not is added to the receiving and sending algorithm. Under this algorithm, if the master have not received all of data from the slaves, it has master wait and execute the receive function in succession until master receives all response frames sent from master to slave just before.

This additional process is implemented by checking the working counter. The working counter enables information in each datagram to be monitored for consistency within the frames [4], [5]. Every node addressed by the datagram and whose memory is accessible increments working counter automatically. If the working counter has a different value than it should, the master waits until working counter change to the value it should. The master device is then able to automatically detect the reason for the unexpected behavior with help from status and continue the job. The pseudo code for decision on receiving data to prevent data skip is shown in Fig. 5.

In the proposed algorithm, the additional receiving process works during the working counter being not three only if the state of master is operation mode. The working counter being three means slaves has finished reading and writing data in the arrived frame. For that condition, the program continue to attempt to receive frame until the working counter being changed to the expected working counter value using the three functions; *ec_master_receive*, *ecrt_domain_process* and *check_domain_state*.

When this algorithm is applied to the existing master stack, it is possible to send and receive frames stably without data loss because master do not send next packet until finishing the processing for previous packet. Consequently, the data throughput of EtherCAT network master is improved without

hardware which has remarkable performance.

```
                    ⋮
ecrt_master_receive(master);
ecrt_domain_process(domain);
check_domain_state();

if State of Master is 8
{
    while( Working Counter is not 3)
    {
        Print "While-Received";
        ecrt_master_receive(master);
        ecrt_domain_process(domain);
        check_domain_state();
    }
}
                    ⋮
```

Fig. 5. Pseudo code for data receiving algorithm.

V. Experiment

A. The Hardware and Software Platform

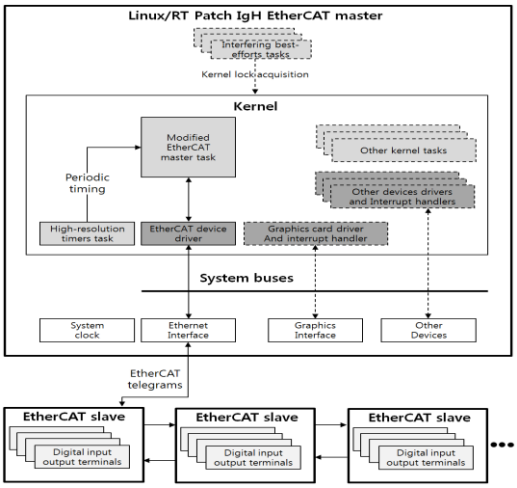


Fig. 6. Block diagram of experimental station.

The experiment is conducted with one PC master and four slaves. The block diagram of experimental station is presented in Fig. 6 [14]. The blocks can be classified into a master and slaves. Solid boxes and arrows represent hardware components and causality relationships within the modules. Dashed boxes and arrows signify interfering components and concurrent activities. In order to reduce the interferences caused by operating system tasks not required by the control application, unnecessary tasks for implementing an EtherCAT network had been removed and the static priority of the control application for experiment is set to 49, the highest value.

The EtherCAT system was implemented and tested on a general purpose PC equipped with an Intel Core i7 running at 2.67GHz and 4GB of DDR3 RAM as the master. The network interface card carrying EtherCAT traffic is based on the Intel gigabit network card. The PC has been set up with a Linux kernel version 3.2.0 patched with the RT version 10 components which are calibrated to schedule and synchronize hard real-time tasks. The Linux kernel with RT patch offers real-time performance comparable with commercial RTOS as well as IgH EtherCAT master stack installed is optimally designed for Linux kernel module. As for the slave device, TMDSICE3359 provided by Texas Instrument was used as slave devices.

B. Experimental Results

The experiment is conducted using program a master send process data to slaves on a regular period. Executing the program for 20 minutes in each transfer period, we measure the number of skipped EtherCAT frames. As depicted in Fig. 7, the number of skipped EtherCAT frames is in inverse proportion to the transfer period before collecting the process. On the other hand, there is no skipped frame when the proposed solution is applied. It means that there was any data loss in the process of communication as the proposed process clearly had got rid of the skipping data problem.

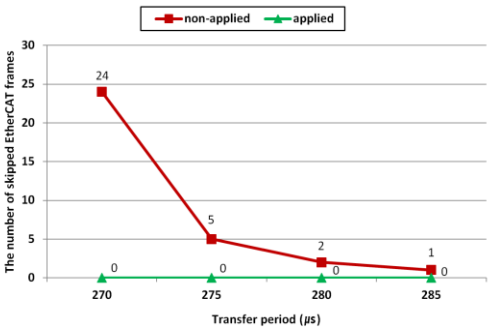


Fig. 7. The number of data skip for each transfer period.

In addition, a point to be considered for this solution is whether it has an adverse effect on real-time communication system. In order to verify it, the average duration of the receiving step with the proposed algorithm and change of transfer period are measured. All of the values are measured by the Master PC using the *clock_gettime()* function which gets the current time from CPU.

The average duration of the receiving step as it is repeated more than twice on each condition is measured and the results are shown in Fig. 8.

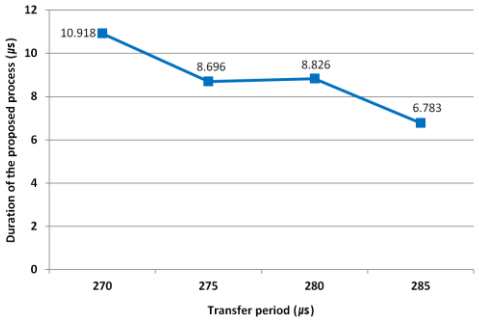


Fig. 8. Duration of the proposed process.

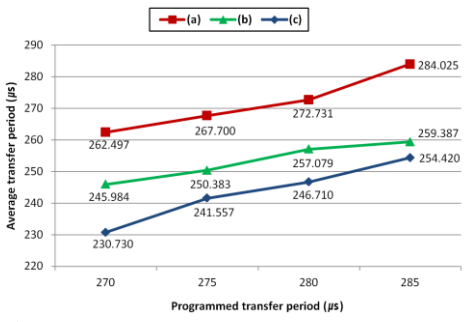


Fig. 9. Transfer period (a) in normal situation (b) when the receiving process is repeated with the proposed algorithm (c) when data are skipped without the proposed algorithm.

According to the results of Fig. 8, the receiving process takes much less time comparison with the programmed transfer period and the maximum time of the receiving process also shorter than time for the programmed transfer period.

Fig. 9 is the measured transfer period of the system in the same condition. According to the results, the transfer periods in the situation that the receiving process being repeated do not exceed the real transfer periods in normal situation which are different with the programmed values since an actual system is not able to work ideally as it had been programmed by external factors. It means that the proposed solution doesn't have an adverse effect on the communication system at all in terms of time.

The additional receiving process of the proposed algorithm is also able to compensate a shorten time by a quick execution of the first data receiving function. It is also can be known by the result of Fig. 9. The transfer period in case of the receiving process being repeated with the proposed algorithm is longer than the transfer period in the case of data being skipped without the proposed algorithm and is closer to the programmed transfer period.

VI. Conclusion

In this paper, we proposed a method to solve the weakness of EtherCAT protocol that EtherCAT master skips data in the case of transfer period being shorter than defined value. It had been verified through the experiments with EtherCAT network system applying the receiving and sending algorithm. We also confirmed that the algorithm is able to prevent data loss effectively between master and slaves as well as does not influence on existing program aspect of delay and even the short delay helps to maintain the cycle time of communication. This proposal is all the more meaningful in that the EtherCAT network can be improved with a simple code modification without any hardware upgrades.

References

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

[2] P. Gerum, "Xenomai — Implementing a RTOS emulation framework on GNU/Linux," 2004.

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

[4] Introduce of EtherCAT. [Online]. Available: http://www.ethercat.org/pdf/ethercat e.pdf

[5] IgH EtherLab EtherCAT master. [Online]. Available: http://www.etherlab.org/en/ethercat

[6] L. Abeni, L. Palopoli, C. Scordino, and G. Lipari, "Resource reservations for general purpose applications," *IEEE Trans. Ind. Inf.*, vol. 5, no. 1, pp. 12-21, Feb. 2009.

[7] L. Palopoli and L. Abeni, "Legacy real-time applications in a reservation-based system," *IEEE Trans. Ind. Inf.*, vol. 5, no. 3, pp. 220-228, Aug. 2009.

[8] A. Rowe, K. Lakshmanan, H. Zhu, and R. Rajkumar, "Rate-harmonized scheduling and its applicability to energy management," *IEEE Trans. Ind. Inf.*, vol. 6, no. 3, pp. 265-275, Aug. 2010.

[9] I.-S. Song, Y.-H. Jeon, J.-H. Kim, S.-H. Seo, K.-H. Kwon, J.-H. Chun, and J.-W. Jeon, "Implementation and analysis of the embedded master for EtherCAT," in *Proc. Inst. Elect. Eng., in Proc. Int. Conf. Control Automation and Systems*, Oct. 2010, pp. 2418-2422.

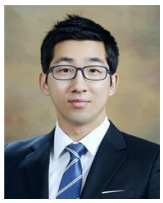
[10] IgH EtherCAT master reference manual. [Online]. Available: http://www.etherlab.org/en/ethercat/

[11] S. Potra and G. Sebestyen, "EtherCAT protocol implementation issues on an embedded Linux platform," in *Proc. Autom., Quality and Testing Robotics*, May 2006, pp. 420-425.

[12] AM3359 industrial communications engine (ICE) schematic. [Online]. Available: http://www.ti.com/tool/tmdsice3359

[13] D. Hristu-Varsakelis and W. S. Levine, "Handbook of networked and embedded control system," 2005.

[14] M. Cereia, I. Cibrario-Bertolotti, and S. Scanzio, "Performance of a real-time EtherCAT master under Linux," *IEEE Trans. Ind. Inf.*, vol. 7, no. 4, pp. 679-687, Nov. 2011.



Cheol-Jin An received the B.S. degree in electronic and electrical engineering from Pusan National University, Pusan, Korea in 2014. He is now a M.S. candidate in electrical and computer engineering at Pusan National University, Korea since 2014. His research interests are nonlinear control and embedded systems.

Hyun-Chul Yi received the B.S. degree in computer science engineering from Pusan National University, Pusan, Korea in 2010 and the M.S. degree in electronic and electrical engineering from Pusan National University, Pusan, Korea in 2012. He is now a Ph.D. candidate in electrical and computer engineering at Pusan National University, Korea since 2012. His research interests are embedded system and wireless sensor network.

Hyoung-Woo Kim received the B.S. and M.S. degrees in electronics engineering from Pusan National University (PNU), Pusan, Korea in 2013 and 2015, respectively. He is currently a Ph.D. candidate in electronics engineering at PNU since 2015. His research interests include nonlinear control, embedded systems, and power systems.

Sung-Mun Park received the B.S. degree in electronic and electrical engineering from Pusan National University, Pusan, Korea in 2014. He is now a M.S. candidate in electrical and computer engineering at Pusan National University, Korea since 2014. His research interests are nonlinear control and embedded systems.

Joon Young Choi received the B.S., M.S. and Ph.D. degrees in electronic and electrical engineering from Pohang University of Science and Technology (POSTECH), Pohang, Korea in 1994, 1996 and 2002, respectively. From 2002 to 2004, he worked as a senior engineer at Electronics and Tele-Communication Research Institute (ETRI), Daejeon, Korea. From 2004 to 2005, he worked as a visiting associate in the Departments of Computer Science and Electrical Engineering at California Institute of Technology (CALTECH), Pasadena, CA. Since 2005, he has been with the Department of Electronics Engineering, Pusan National University, Pusan, Korea, where he is currently a professor. His research interests include nonlinear control, internet congestion control, embedded systems, and automation.