MESSAGE SCHEDULING ON CAN BUS FOR SHIP ENGINE SYSTEMS

Favian Dewanta[1], Dong-Sung Kim[2], Rendy Munadi[3]

[1,3]School of Electrical Engineering, Telkom University
[2]Kumoh National Institute of Technology
[1]favian@telkomuniversity.ac.id, [2]dskim@kumoh.ac.kr, [3]rendymunadi@telkomuniversity.ac.id

Abstract

This paper reports novel algorithm for handling real time message scheduling on CAN bus which consider buffer occupancy and message type (alarm, periodic real time and non real time). This algorithm is derived to reduce burden of the distributed control unit (DCU) which have obligation to deliver message within strict time and also manage buffer storage occupancy. The algorithm works by changing the message priority which is attached in the identifier field of message frame. The priority adjustment is performed by online calculation prior to send the message on the CAN bus. Finally, this proposed algorithm is applied on ship engine systems which consist of certain number of DCUs. In this networked control system, the proposed algorithm can decrease buffer overflow. Furthermore, the important message alarm and periodic real time message can be delivered within the deadline.

Keywords: CAN Bus, message scheduling, DCU

1. Introduction

CAN bus have been used for networked control systems on ship, automotive, and other applications [4, 10, 12]. The use of CAN bus for those applications can simplify the cabling from main controller to sensors and actuators. Then, by using CAN bus, all sensors and actuators which are closed each other can be handled by the DCU.

Implementation of CAN bus for networked control system requires strict scheduling and timing allocation. Those requirements are reasonable because CAN bus has low bit rate among other protocols [5].

Since CAN bus was emerged and widely used, many studies have been pointed to put priority in occupying CAN bus by manipulating the identifier field of CAN message [1, 8]. By giving priority, important messages, e.g. alarm message and periodic real time message, can be delivered within bounded delay or deadline. For that reason, message scheduling on CAN bus became one of the interesting topics to discuss and investigate among scholar, engineer, practitioners, and others.

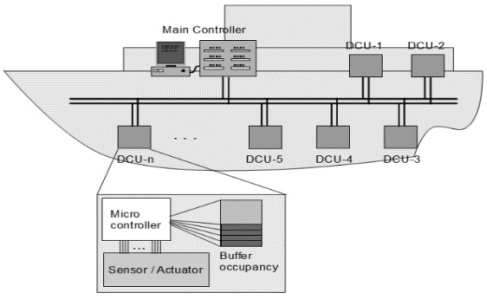


Figure 1. Distributed Control Network for Ship Engine System

Several works and studies have been done to show and analyze the performance of message scheduling on CAN bus. In [7], authors tried to construct deterministic task activation and message scheduling based on priority CAN bus. Then they analyzed the probability distribution of end-to-end latencies in message scheduling. In their result, they can show the worst case response time. However, they did not show clearly how to adjust the identifier field to set the priority.

Comparison between early deadline first (EDF) and rate monotonic (RM) for FTT-CAN protocol has been presented in the reference [9]. In simulation part, EDF showed better performance for utilization factor and jitter than RM based message scheduling and fixed priority. However, their approach did not show the effect of message identifier field adjustment to give maximum delay bound.

Another message scheduling based on dynamic priority promotion also has been presented by Peng, et. al. [13]. They proposed message scheduling which considers message waiting time, i.e., the time for which a message experiences lost during arbitration until successfully arriving at the receiver. In simulation part, their algorithm showed better performance than static priority.

Dynamic distributed message scheduling method (DDSM) was proposed in [11] to deal with time constraints. This method considers the life time, i.e., the expected time taken by a message to arrive at the receiver, and waiting time of a message in the network.

However, no previous study considered the buffer occupancy and message type (alarm, periodic real time message, and non-real time message) concurrently. Therefore, this paper proposes a new algorithm to solve the message scheduling issue for

the CAN bus. The proposed algorithm exploits 29 bits identifier field of CAN 2.0B [2, 6]. The proposed algorithm not only gives both the delay bounds for a message, but also minimizes the buffer overflow.

This paper is organized as follows. Section 2 presents message scheduling method. Section 3 presents a mathematical analysis of time delay on CAN bus. Section 4 covers simulation result. Section 5 covers measurement results. The last, conclusion and future work is presented in section 6.

2. Message Scheduling Algorithm

2.1. System Model

This paper investigates the performance of networked control system on ship engine system as shown in Figure 1. The networked control system consists of some DCUs, CAN bus, and main control system (MCS). The DCUs and MCS communicates each other by sending certain fixed message through CAN bus.

In common networked control system, the message sent by DCU or MCS are classified into three types. First is real time periodic (RTP) message which delivers important periodic sensor information or actuator command. This kind of message should be delivered within bounded time to make the system work well. Second is alarm message which delivers information of broken system that happen on certain DCU. This kind of message should be given highest priority to occupy the network. Third is non-real time (NRT) message which delivers message that does not require to arrive in the destination on time.

CAN bus, however, is naturally deterministic network which allows lowest message identifier field to occupy the network. Then, identifier field of the message should be adjusted to make the most important message can occupy the network if many messages want to occupy the network.

Because of postponing unimportant message to make important message occupying the network, buffer occupation on the DCU can increase if another message is produced by that DCU. The buffer can be full even overflow that will lead some data lost before being sent to the main controller.

Figure 2 show the example of CAN bus occupation by some types of message (RTP, alarm, and NRT message). In that figure, first, CAN bus is occupied by alarm message that is sent by DCU 3. Second, eventhough there are two PRT messages that attempt to occupy CAN bus, but message from DCU 1 can occupy CAN bus because it has shorter delay than DCU 4 after being lost in arbitration in the previous slot. Slot 3 and 4 show the same phenomena that already explained before.

Slot 5 is slightly different than previous slot which is NRT message can win the arbitration eventhough PRT message exists. This thing can happen because buffer occupation in DCU 2 increase. As shown in that example, it is possible to adjust the priority based on message type and buffer occupation. The alarm message sould be given highest priority and the PRT and NRT message can be given flexible priority depending the buffer and message deadline. Therefore, the proposed algorithm change the message priority based on the message type and buffer occupation.

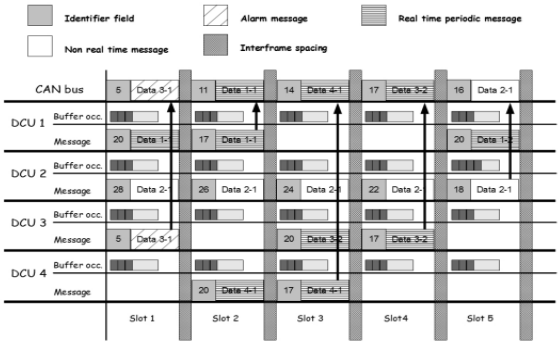


Figure 2. CAN Bus Occupied by Some Messages Type

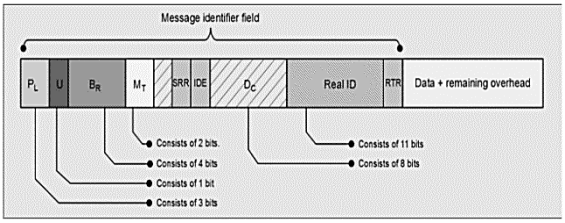


Figure 3. Segmentation on Identifier Field of Message Frame on CAN Bus

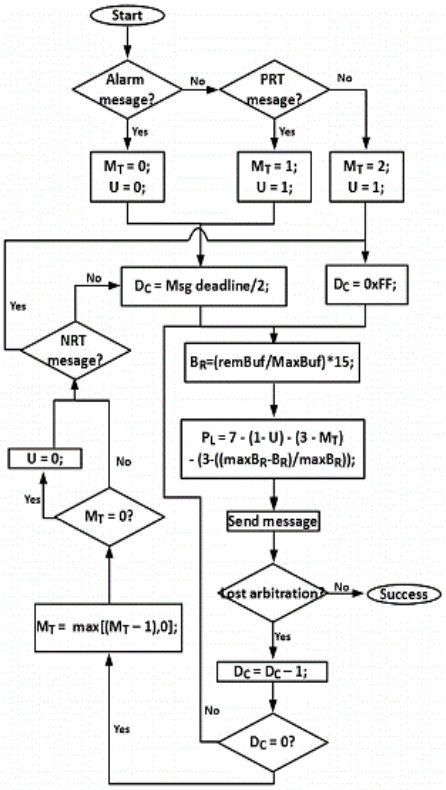


Figure 4. Flowchart of the Proposed Algorithm

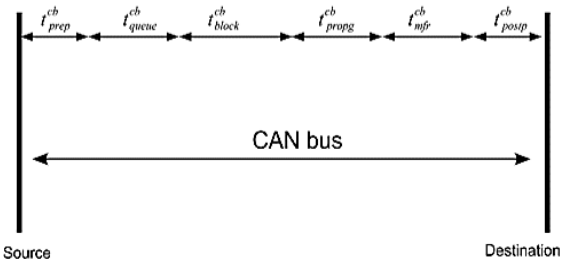


Figure 5. Model of CAN Bus End-to-end Delay Component

2.2. Implemented Algorithm

To accomodate both message type and buffer occupation in message scheduling, this algorithm is proposed to adjust the value of message identifier field by dividing into some parts that are shown in Figure 3. Real ID means the ID of DCU. D_C means deadline counter and its value is decreased whenever a message experience lost in arbitration. M_T means message type and its initial value is asigned based on message type (alarm, PRT, or NRT message). B_R means remaining buffer and its value depends on the buffer occupation in the DCU. U means urgency and its initial value is assigned based on the type of message. The last, P_L means priority level which is gotten from the value of U, M_T, and D_C.

Figure 4 shows the flow of message identifier field adjustment. At the first, M_T and U are assigned based on message type (alarm, PRT, or NRT message). After that, D_C is assigned based on the deadline of the message and the type of message. Then, prior to send the message, B_R and P_L are always calculated.

Whenever a message experience lost in arbitration, the value of D_C is always decreased to promote the priority. If the value of D_C equal to 0, the value of M_T will be adjusted to maximum value between 0 and $M_T - 1$. If the value of M_T is equal to 0, the value of U will be assigned to 0. Then, the value of D_C will be re-assigned either to half of message deadline for alarm and PRT message or to 0xFF for NRT message.

With this kind of method, the priority can adjusted flexibly to bound the delay of important message (alarm and PRT message) without sacrifice the buffer occupation on the DCU. Hence, the system can work well and the buffer occupation is fair among all DCUs.

3. End-to-end Delay Calculation

Prior to analyze the end to end delay of CAN bus, the following assumptions are used to simplify the calculation. In addition, to make better understanding, Figure 5 show the end to end delay components for CAN bus.

a. Due to the fact that computation process happens in application layer, so for simplification, t^{cb}_{prep} can be ignored.
b. CAN bus message frame has total 17 bytes length, including overhead and data.
c. The propagation time (t^{cb}_{propg}) for CAN bus is 1 µs. This value represents the use of 100 m cable length in CAN bus. Considering that t^{cb}_{propg} is much less than the frame transmission (t^{cb}_{mfr}) (around 0.005 %), so t^{cb}_{propg} can be ignored.

Message waiting time in DCU (t^{cb}_{queue}) can be derived from the total time of all messages in front of the arrived message in the queue.

The total time of all messages means the sum of delay (blocking time and message frame time) experienced by each message. The t^{cb}_{queue} is expressed by the equation 1.

$$t^{cb}_{queue} = \sum_{N_{queue}} t^{cb}_{mfr} + t^{cb}_{block} \quad (1)$$

Message blocking time in DCU (t^{cb}_{block}) can be derived from the total time of high priority message frame and the residu time of the current transmission (t^{cb}_{residu}). Residu time occurs if a message arrives while the CAN bus is still occupied by other message transmission. Residu time is calculated from a message arrive in the queue until the current message transmission is finished. The following equation show the message blocking time.

$$t^{cb}_{block} = t^{cb}_{residu} + \sum_{N_{hpm}} t^{cb}_{mfr} \quad (2)$$

Message frame time in CAN bus (t^{cb}_{mfr}) can be calculated from the total bits (N_{data}, N_{ovhd}, N_{stuff}) divided by the bit rate of message transmission (R_{cb}) as shown in equation 3. In that equation, N_{data} means the number of data in the message frame. N_{ovhd} means the number of overhead in the frame, including the start of frame, CRC, ACK, etc. N_{stuff} means the number of stuffing bits due to the same five consecutive bits [6].

$$t^{cb}_{mfr} = \frac{N_{data} + N_{ovhd} + N_{stuff}}{R_{cb}} \quad (3)$$

Then, the total delay of CAN bus is shown in equation 4.

$$D^{cb}_{total} = t^{cb}_{queue} + t^{cb}_{block} + t^{cb}_{mfr} \quad (4)$$

4. Simulation Result

The simulation environment was created using the C code and GCC compiler. This system consists of 15 DCUs that are connected through a CAN bus. The bit rate used in this CAN bus simulation model is 250 kbps. Each DCU uses event triggered to activate the transmission mode and send the message through the CAN bus. Otherwise, if there is no event, each DCU will stay in the reception mode. An event will

generate 8 bytes of data, which are given different identifier labels. The maximum buffer size used in this simulation is 1024 bits.

The DCUs will send the data continuously until there is no data in the buffer. If all the DCUs send the message at the same time, it will cause a collision. In such a situation, arbitration will determine which message should be sent. A busy networks due to many the messages in the CAN bus is used to analyze the performance of the algorithm by determining whether it can handle the issues of delay and buffer.

To simplify end-to-end delay calculation, the simulations capture delay of message which accumulates only message frame time (t^{cb}_{mfr}) and message blocking (t^{cb}_{block}). In addition, to evaluate the performance of the RTMS method, this section is partitioned into two subsections that emphasize on buffer occupation and message type performance.

4.1. Scenario 1 - Buffer Occupation Evaluation

In order to create and analyze such a busy network situation, some parameters are set and varied for each simulation, such as buffer storage rate (λ) and initial buffer for each DCU. The simulation will capture buffer occupation for each DCU. The detailed parameters for this simulation are listed in Table 1.

Figure 6 shows that the proposed and dynamic priority algorithm give the best result for preventing buffer over flow owing to their capability to maintain the fairness for each DCU's buffer size. The fixed priority algorithm is worst because this method makes the highest identifier always lose in arbitration and wait until there is no lowest ID. DDSM gives better results than the fixed priority algorithm because of its capability to switch the message transmission among all DCUs in the network.

For an alarm message, the proposed algorithm gives the best result as shown in Figure 7. Still, the proposed algorithm guarantees no buffer overflow even in heavy traffic. In addition, the fixed priority algorithm still shows worse performance than other algorithms because of the same reason already explained before. The other algorithms, dynamic priority [6] and DDSM [2] are still better than the fixed priority algorithm because of its capability to schedule the message dynamically.

4.2. Scenario 2 - Case Study of Networked Control System on Ship Engine System

In this part, a case study of ship engine networked control system is presented. The system maintains the stability of the ship in facing turbulence caused by wind, waves and current by employing some accelerometers. The system consists of several DCUs as noted in Table 2 based on the previous work in [10].

The number of node that generate NRT message is varied from 15 to 150 nodes to evaluate the performance of some message scheduling algorithm. The evaluation parameter in this scenario is worst case delay for DCU ACC-1 until ACC-6. To evaluate the worst case delay performance, the real ID of DCU ACC-1 until ACC-6 is assigned in the last number. For example, if the total DCU is 15, so the real ID of the DCU ACC-1 until ACC-6 will be 9, 10, 11, 12, 13, 14, and 15 respectively.

Figure 8 shows that proposed method can maintain the maximal delay of observed DCUs below the deadline. Even though the number of DCU is increased until ten times from the beginning, the worst case still remains same under the deadline. This results prove that the proposed method is robust and stable even the number of DCU, that want to occupy the CAN bus increase ten times.

Table 1. Parameter for Simulation

Simulation Parameter	Value
Initial buffer	512 bits
Number of node	15 nodes
Bit rate	250 kbps
Maximum buffer	1024 bits
NRT message arrival rate	5 - 15 ms
Alarm message arrival rate	100 ms

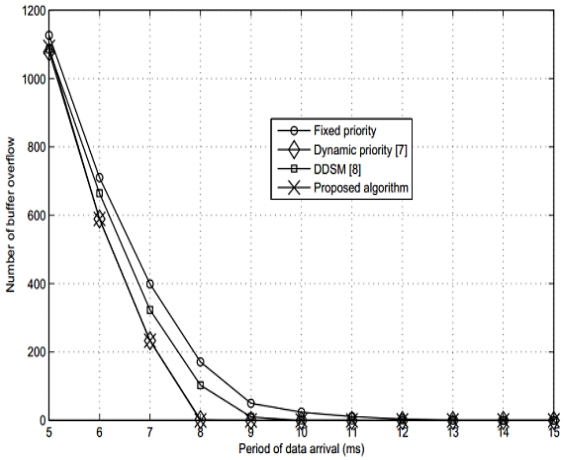


Figure 6. Number of NRT Message Drop in Perspective of Data Arrival Period

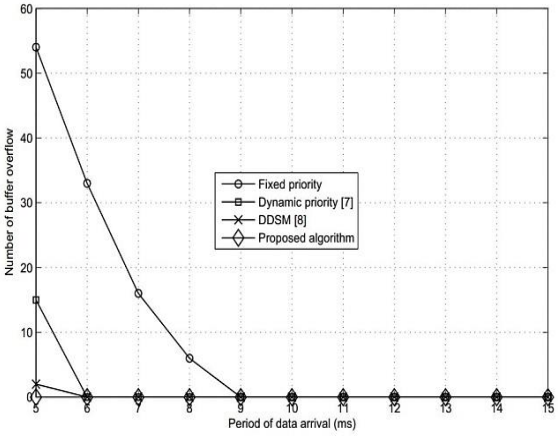
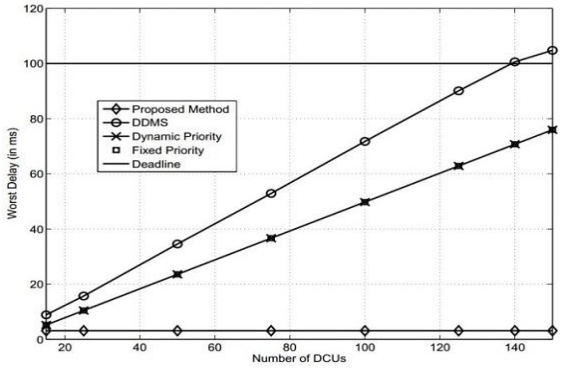


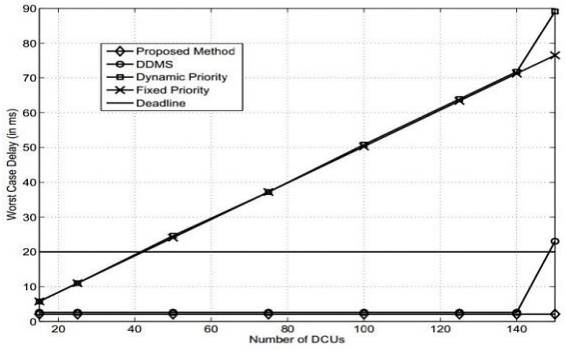
Figure 7. Number of Alarm Message Drop in Perspective of Data Arrival Period

Table 2. Example of DCUs Used to Simulate Networked Ship Engine Control System

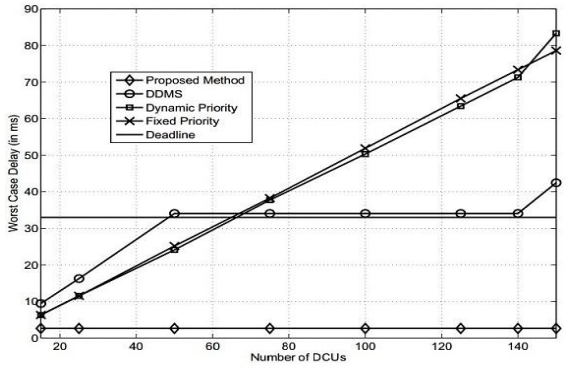
Variable	Period (ms)	Deadline (ms)	Type	Size (bytes)
ACC-1	100	100	PRT	8
ACC-2	20	20	PRT	8
ACC-3	33	33	PRT	8
ACC-4	6.5	6.5	PRT	8
ACC-5	6.5	6.5	PRT	8
ACC-6	6.5	6.5	PRT	8
Other DCUs	100	100	NRT	8



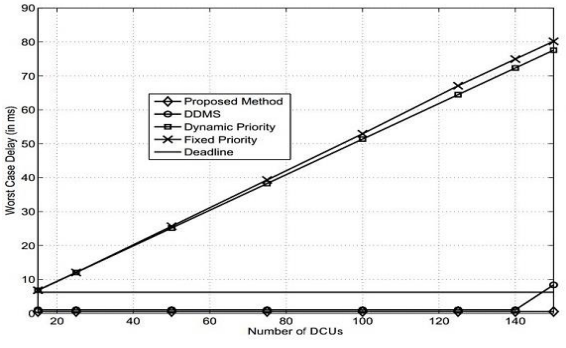
(a)



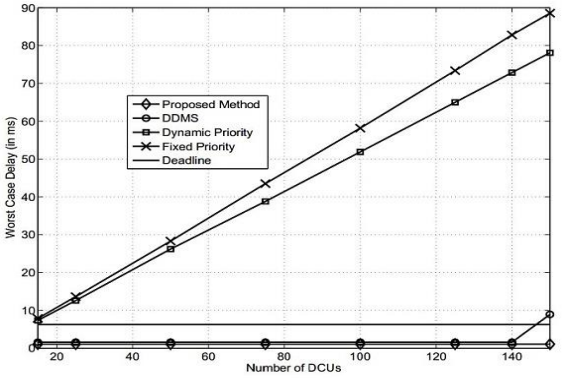
(b)



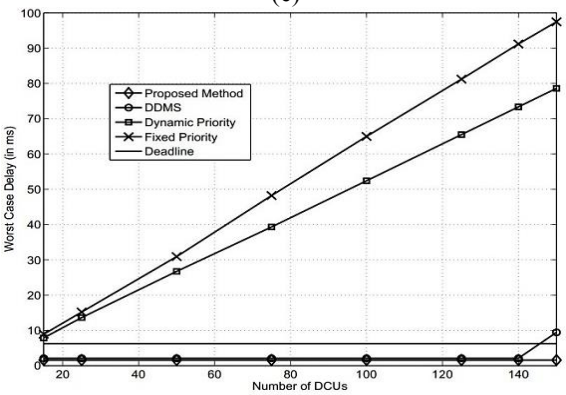
(c)



(d)



(e)



(f)

Figure 8. Result of Worst Case Delay Observation on Several DCUs That Generate PRT Message

The second place, after the proposed algorithm, DDMS can maintain the delay under the deadline before reaching 150 DCUs for DCU ACC-1, ACC-2, ACC-4, ACC-5, and ACC-6. However, for DCU ACC-3, the result is out of expectation which the delay is bigger than the deadline at the point number of DCU equal 50. These results can happen because DDMS cannot handle many DCUs that attempt to occupy CAN bus.

Similar results are shown by dynamic priority and fixed priority algorithm. The worst case delay of both algorithms tends to increase as the number of DCU increase. Even though in DCU ACC-1, the worst case delay can be maintained under the deadline, but the worst case delay of remaining DCUs are always bigger than the deadline. This results happen because the dynamic priority and fixed priority give last real ID DCU least priority so that the worst case delay tend to increase as the number of DCU increase.

5. Measurement Result

In this section, the implementation result related to feasibility of proposed algorithm will be reported and analyzed in detail. Prior to explain the implementation result, some terminologies are introduced and listed below to get better

understanding. The illustration of those terminologies is shown in the figure 9.

- U^{cb}_{bw} is the bandwidth utility of the CAN bus. This value is derived from comparison of t^{cb}_{comp}, t^{cb}_{mfr}, and t^{cb}_{ifs} spacing as shown in equation below.

$$U^{cb}_{bw} = \frac{t^{cb}_{mfr} + t^{cb}_{ifs}}{t^{cb}_{mfr} + t^{cb}_{ifs} + t^{cb}_{comp}} \quad (5)$$

As shown in the previous section, the implementation system is realized using AT90CAN128 with 16 MHz clock. Those new terminologies that are introduced in the beginning of this section is investigated by changing the transmission bit rate of the DCU from 125 kbps to 1 Mbps based on the Atmel datasheet [2]. Figure 11 shows the test bed of the implementation part.

The implementation result is shown in the Figure 10 and each of sub-figures report the result of each transmission bit rate. To give better understanding, Table 3 summarizes the implementation result.

Based on the Table 3, the bandwidth utility is decreased whenever the transmission bit rate is increased. This phenomenon happens because the message frame value tends to decrease as the transmission bit rate increase. However, the computation delay remains same even though the transmission bit rate changes.

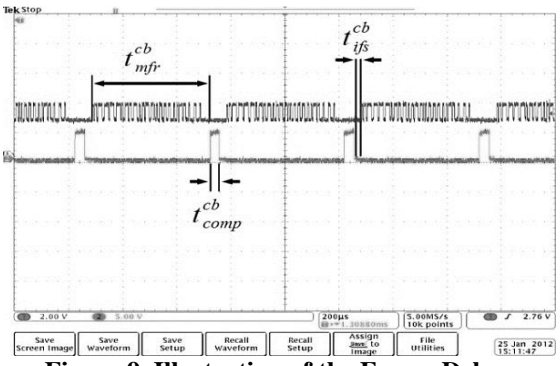
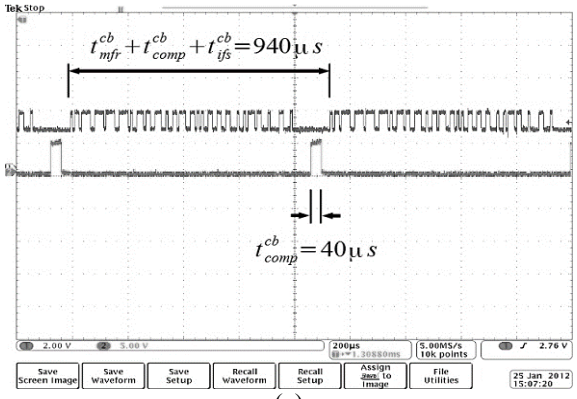
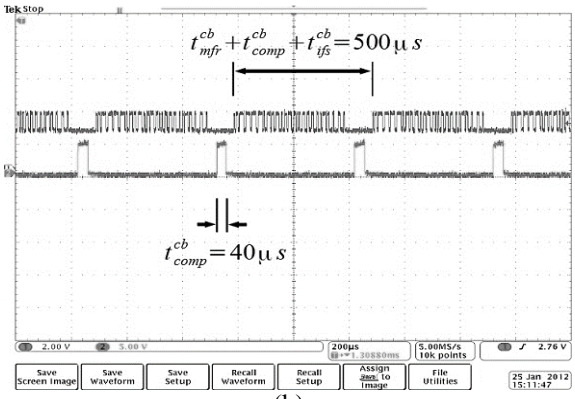


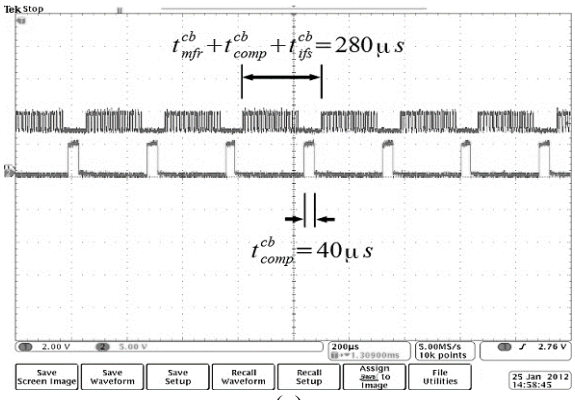
Figure 9. Illustration of the Frame Delay, Computation Delay, and Inter-frame Spacing Delay in the Measurement Result Figure



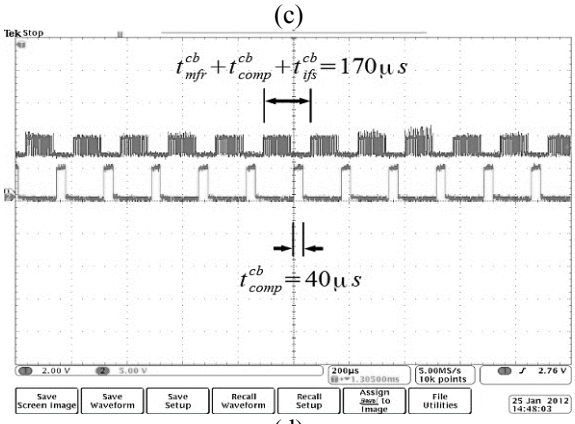
(a)



(b)



(c)



(d)

Figure 10. Measurement Result of Several Transmission Bit Rate

Table 3. Summarize of Measurement Result

Bit rate (kbps)	$t^{cb}_{mfr} + t^{cb}_{ifs} + t^{cb}_{comp}$ (µs)	t^{cb}_{comp} (µs)	U^{cb}_{bw} (%)
125	940	40	95.745
250	500	40	92.000
500	280	40	85.714
1000	170	40	76.471

Figure 11. Screenshot of the Test Bed

6. Conclusion

This paper proposes a new dynamic priority scheduling for handling the distributed control unit using a CAN bus. This algorithm accommodates buffer condition and message type (alarm, periodic real time, and non-real time message). All parameters are calculated to assign priority and put into the identifier field of the CAN frame which is used for CAN bus arbitration.

Simulation results show that the proposed algorithm is robust even under fully loaded traffic, can provide fairness, reduce the data loss due to buffer overflow, and can guarantee delay bounds for PRT message under the deadline. The proposed algorithm can be useful for making the DCU stable and for guaranteeing the reliability of message scheduling in the CAN bus.

In the implementation part, the use of proposed algorithm introduces fixed computation delay (40 μs). By changing the transmission bit rate, there is a tendency that the bigger the value of transmission bit rate, the smaller the utility of the CAN bus bandwidth.

References

[1] Anwar, K. and Z. Khan, "*Dynamic Priority Based Message Scheduling on Controller Area Network*", in International Conference on Electrical Engineering (ICEE '07), pp. 1 – 6, April 2007.

[2] Atmel, "*At90can128*", [Online], May 2006. www.atmel.com/dyn/resources/prod_documents/doc4250.pdf

[3] C. Lin, S. Tai, C. Li, H. Lin, and T. Chen, "*Small Aircraft Avionics Using Hybrid Data Bus Technology*", Aerospace and Electronic Systems Magazine, IEEE, vol. 21, no. 7, pp. 17 – 21, July 2006.

[4] Cena, G., A. Valenzano, and S. Vitturi, "*Introducing Intelligent Sensors in Presses for Plastic Material Injection*", IEEE Transactions on Industrial Informatics, vol. 1, no. 2, pp. 136 – 148, May 2005.

[5] F-L. Lian, J. Moyne, and D. Tilbury, "*Performance Evaluation of Control Networks*: *Ethernet, Controlnet, and Devicenet*", Control Systems, IEEE, vol. 21, no. 1, pp. 66 – 83, February 2001.

[6] Goller, G., "*Atmel Wireless & Microcontrollers Can Tutorial*", [Online], March 2004. http://www.grifo.com/PRESS/DOC/Temic/CAN_TUT.PDF

[7] H. Zeng, M. Di Natale, P. Giusto, and A. Sangiovanni-Vincentelli, "*Stochastic Analysis of Can-Based Real-Time Automotive Systems*", IEEE Transactions on Industrial Informatics, vol. 5, no. 4, pp. 388 – 401, November 2009.

[8] Nolte, T., H. Hansson, and C. Norstrom, "*Minimizing Can Response-Time Jitter by Message Manipulation*", in Proceedings of 8th IEEE Real-Time and Embedded Technology and Applications Symposium, pp. 197 – 206, 2002.

[9] Pedreiras, P. and L. Almeida, "*Edf Message Scheduling on Controller Area Network*", Computing Control Engineering Journal, vol. 13, no. 4, pp. 163 – 170, August 2002.

[10] Piorno, J., S. San Roman, J. Giron-Sierra, and J. de la Cruz Garcia, "*Fast Ship Electronic System for Seakeeping Experimental Studies*", IEEE Transactions on Instrumentation and Measurement, vol. 58, no. 10, pp. 3427 – 3433, October 2009.

[11] Q. Wu, J. Liu, Y. Guo, and Y. Li, "*A Dynamic Distributed Message Scheduling Method for Can-Based Networked Control Systems*", in International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), vol. 1, pp. 85 –89, March 2010.

[12] Schliecker, S., M. Negrean, and R. Ernst, "*Response Time Analysis on Multicore Ecus with Shared Resources*," IEEE Transactions on Industrial Informatics,, vol. 5, no. 4, pp. 402 – 413, November 2009.

[13] T. Xiao-Peng, L. Xiao-Bing, and X. Ti-Liang, "*Real-Time Analysis of Dynamic Priority of Can Bus Protocol*," in International Conference On Electronics and Information Engineering (ICEIE), vol. 1, pp. 219 – 222, August 2010.