Fifth International Conference on Internet of Things and Applications (IoT 2021)

MRMC-CAN: A Method to Improve Real-Timeness and Response Time of CAN

Ismail Ghodsollahee
Dependable Distributed Embedded Systems (DDEms) Laboratory,
Department of Computer Engineering,
Ferdowsi University of Mashhad,
Mashhad, Iran.

Yasser Sedaghat
Dependable Distributed Embedded Systems (DDEms) Laboratory,
Department of Computer Engineering,
Ferdowsi University of Mashhad,
Mashhad, Iran.

Abstract— **Although the Industrial Internet of Things (IIoT) has made great improvement in factory automation, there are still many challenges in meeting response time and reliability requirements of IIoT communications. These challenges are because of the need to real-time communications in an industrial environment with high electromagnetic interferences. To meet these challenges, in context of real-time industrial device communications, Controller Area Network (CAN) protocol is commonly employed, which is noise resistance, nevertheless the presence of a faulty node in CAN networks can lead to deadline violation of messages and timing failure. In this paper, to control the behavior of nodes, message retransmission performed based on criticality of message reception (MRMC-CAN). The proposed method in comparison with standard CAN and WCTER-based approaches reduces consumed bandwidth by average 10.5% and 4.4%, respectively. Moreover, the proposed technique improves response time in comparison with standard CAN by average 36.19%.**

Keywords: Controller Area Network (CAN), Industrial Internet of Things (IIoT), Real-Timeness, Reliability, Error Handling.

I. INTRODUCTION

The CAN communication protocol was designed in the 1980s by Robert Bosch for vehicular internal network. This communication protocol is one of the mostly employed communication protocol in vehicular networks due to its low implementation cost and its fault tolerant behaviour against network errors [1]. Today this communication protocol is employed in other industrial fields and IIoT in addition to vehicular internal networks [2, 3].

IIoT systems are safety-critical in nature [4]. These systems require error handling and real-timeness in their communications [5, 6]. However, since in the CAN communication protocol, to deal with communication errors, corrupted message is retransmitted after any type of error detection, and given that this communication protocol employs the carrier sense multiple access with collision detection (CSMA/CD), retransmission of messages is in conflict with real-time constraint required in the safety-critical systems [7].

There are five different type of errors in the CAN communication protocol, including Bit Error, Stuff Error, Cyclic Redundancy Check (CRC) Error, Form Error, and Acknowledgment Error. Among these errors, CRC Error is detected by receiver nodes if there is any differences between received and computed CRC, and the Acknowledgment Error is detected by the sender node if this node does not receive any acknowledgement from the receiver nodes.

In the CAN communication protocol, the correctness of message reception is determined by the content of ACK field. As shown in Figure 1, ACK field consist of ACK slot and ACK diameter. Sender node leaves the ACK slot recessive and waits for acknowledgment. The correct reception of messages will have acknowledged after CRC checking, through changing the ACK slot to dominant by each receiver node. If ACK slot does not changed to dominant, it means that an incorrect message detected by all receiver nodes. In this case, the sender node, detects acknowledgment error, and retransmit unacknowledged frame.

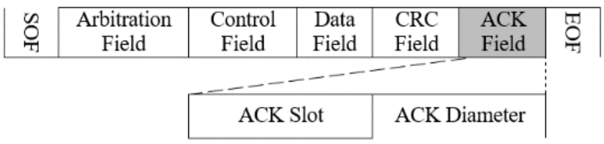

Figure 1. CAN ACK field format

Although the acknowledgement process employed in the CAN communication protocol assures the sender that the message has been received correctly by all nodes, the dominant bit sent in the ACK slot by one of the receiver node which detects the correctness of received message, prevents the sender node from identifying the nodes that received the message incorrectly. In such a situation, as shown in Figure 2, the sender's ignorance of which nodes received the message incorrectly, prevents the sender node from making the right decision about the need to retransmit unacknowledged frame.

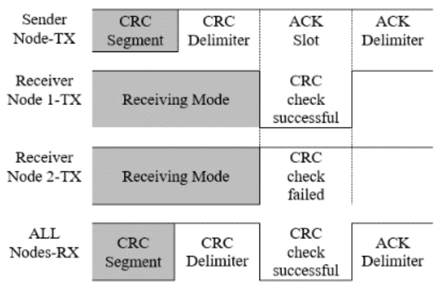

Figure 2. Acknowledgement Process

Any node which detects CRC Error will issue an error flag to notify the sender node about incorrect reception of the message. As shown in Figure 3, although sending an error frame notifies the sender node about incorrect reception, recovery time from detecting an error until the start of the next message is 18 bit times and can be at most 31 bit times [8]. Therefore, to improve the real-timeness of CAN network, in this paper, the MRMC-CAN technique is presented. This technique gives the sender node the knowledge of which nodes did not receive the message correctly. This knowledge allows the sender node to decide whether retransmit the message or not.

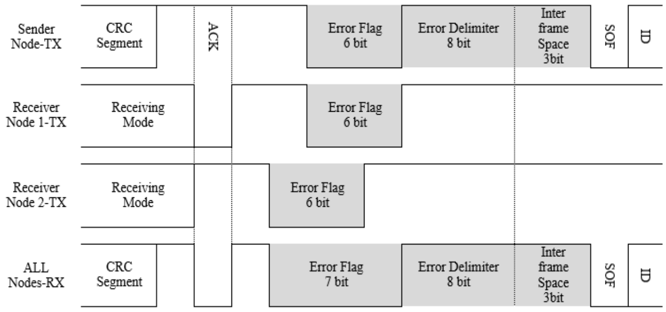

Figure 3. Bit sequence after CRC Error detection

The rest of this paper is organized as follows. The related studies are investigated in Section II, then proposed technique is illustrated in Section III. The response time presented in Section IV. Section V describes the experimental results and evaluation. Finally, conclusion are presented in Section VI.

II. RELATED STUDIES

Since non-violation of message deadlines and timing verification is the key to ensuring vehicle safety during the design phase [9], papers [10, 28] focus on the CAN worst case response-time improvement through consumed bandwidth reduction. CAN worst case response-time defined as the longest time taken for messages to reach their destinations, which measured relative to the arrival time of messages [10].

One way to reduce bandwidth consumption is to prevent offending node from connecting to the CAN network. Network Guardian (NG) is commonly employed to prevent babbling idiot failure which is caused by offending nodes [11], [12]. Although employing of NG prevents the high bandwidth overhead caused by faulty nodes, the level of babbling that NG prevents the node from sending a message to the network is the same for all messages. For this reason, an analysis for the Guardian based approach is presented in [13]. In this analysis, the number of retransmission of a messages is determined based on the criticality level of the messages.

In addition to methods which prevents high traffic consumption due to faulty nodes, others [14,17] prevents fault propagation from one subnet to the others by changing linear topology of the CAN network. In [14] by changing the topology of the CAN network, the RedCAN is presented. In RedCAN, after detecting physical defects in one sector to prevent fault propagation, other nodes disconnect this sector and employ redundant sector. Also Barranco and Proenza [15] propose active star topology, called CANcentrate. In CANcentrate central hub, prevents fault propagation. Although CANcentrate prevents communication network failure due to link faults, the active star topology hub, represents single point of failure. As a result they present replicated active star topology called ReCANcentrate which is based on the hardware redundancy of the hub [16]. Moreover, in [17] a shared clock algorithm named TTC-SC6 proposed which ensures that fault on one link of star network cannot propagate to the rest of the network via port disablement.

In contrast, a series of papers [18, 25], reduce bandwidth consumption through data reduction (DR) techniques. In DR techniques, the compression process is as follow, first a message with the identifier ID’ is sent at t=t’, then the subsequent messages with ID’ sent at t=t’+1 based on signals’ differences [18]. In DR technique presented in [19], the first byte of compressed data frame is assigned to data compression code (DCC). Each bit of DCC indicates whether or not one byte of data frame compressed. In [20] Adaptive DR (ADR) technique is presented. In ADR DCC is based on signals instead of bytes. In addition ADR prevents the current frame from being transmitted if it does not differ from previous one.

Although, ADR reduces the bandwidth consumption, in this technique, if the value of one of the signal differences exceeds the assigned data field, the whole message will be sent uncompressed. Therefore, in [21], the improved ADR (IADR) technique is presented. In this method, it is possible to send a combination of compressed and uncompressed signals in one message. Moreover, [18] proposes Enhanced DR (EDR) technique, considering the overhead caused by DR techniques and its effect on the bit length of the compressed message. In EDR, a signal is sent compressed if it does not increase the length of the compressed message compared to the uncompressed message.

Assign Data field to signals based on the predicted maximum bit length of signal differences (e.g., in boundary of fifteen compression technique (BFC) [22], a signal is compressed if its corresponding signal difference is within the maximum compression rage of ±15 bits.) affect the performance of DR techniques [21, 23]. Therefore, Wu and Chung proposed efficient CAN DR (ECANDC) [23], and improved CAN DR (ICANDR) [24] techniques based on signal rearrangement algorithms (SRA). In these techniques, compression area selection (MAP) is employed to eliminate the prediction of the maximum signal differences bit length. In ICANDR technique, CAN data field, divided into 24, 24, and 16 bit length sub fields. Each combination of signal mapping to these subfields results in different compression efficiency. In [25] a CAN data arrangement algorithm proposed to maximize compression efficiency.

In addition to DR techniques, others reduce bandwidth consumption through minimizing stuffing-bit. In CAN network non return to zero (NRZ) coding employed to ensure synchronization of all nodes. In this coding, an opposite polarity bit is inserted after five consecutive bits with the same polarity. Although, bit stuffing is a fault tolerant mechanism in CAN which synchronize all nodes, Stuffing-bits can cause a 22% overhead in worst case [26]. For this reason Park and

Kang propose a bit stuffing mechanism based on XOR masking to minimize stuffing-bits and prevent priority inversion [26]. In their mechanism, messages are divided into m groups, which each of them contain n identifiers. In this mechanism first XOR mask initialized to “1010…”, then 1 assigned to $1 + [log_2 m]$ most significant bits and 0 is assigned to $[log_2 m]$ bits of XOR mask to prevent priority inversion.

Another category of real-timeness improvement and consumed bandwidth reduction techniques is based on error correction and prevention of message retransmission. In [7], dual CRC error correction (DUCER) technique proposed, which employs redundant communication channel and lightweight error correction software scheme, which can correct 5-bit errors. Classification of real-timeness improvement techniques shows in Figure 4.

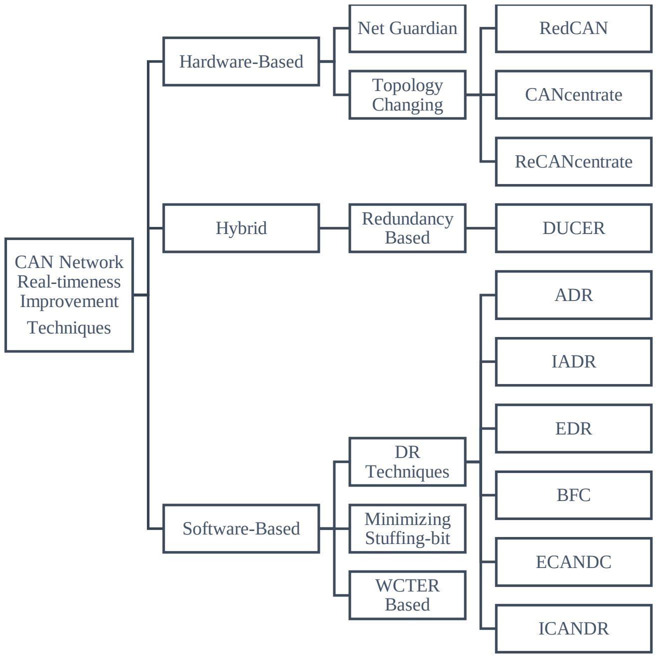

Figure 4. Classification of Methods for Real-Timeness Improvement of CAN Network

III. PROPOSED METHOD

As mentioned earlier, corrupted messages are retransmitted in the CAN communication protocol to deal with communication errors, whereas, message retransmission is in conflict with the real-time requirements of safety-critical systems. For this reason, in this paper the MRMC-CAN is presented, In MRMC-CAN decision to retransmission is made at the receiver nodes. For this purpose, receiver nodes control the message retransmission by controlling error flag propagation based on the criticality of receiving a message. The criticality of receiving a message is determined based on a list of critical IDs defined in each node.

Although message retransmission based on the decision of receiving nodes improve response time and reduce consumed bandwidth, this decision is made based on ID that may be received incorrectly. Therefore in MRMC-CAN, the arbitration field of CAN messages (as shown in Figure 5), is divided into two parts including reduced ID (RID) and ID-CRC. ID-CRC allows the receiver nodes to make sure that the received ID is correct.

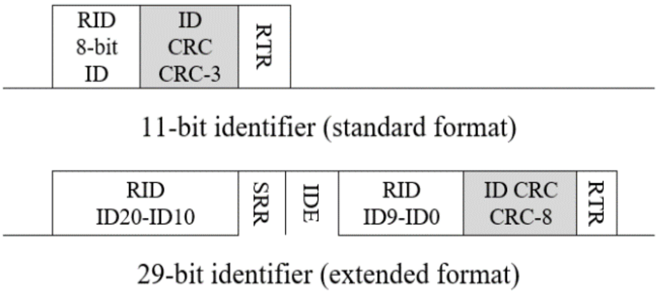

Figure 5. MRMC-CAN Arbitration Field

As shown in Figure 6 MRMC-CAN includes Criticality detection (CD), and Error Flag Transmission Control (EFTC) modules, to give receiver nodes the ability of decision making about the need for message retransmission. The CD module monitors the CAN-RX signal of the standard CAN controller and checks that the received ID matches with list of critical IDs. If the message ID matches with the list of critical IDs, and if received CRC-ID is equal to the calculated CRC-ID, the CD module detects the criticality of receiving this message, and the *criticality* signal goes high. The implementation of CD module is shown in Figure 7.

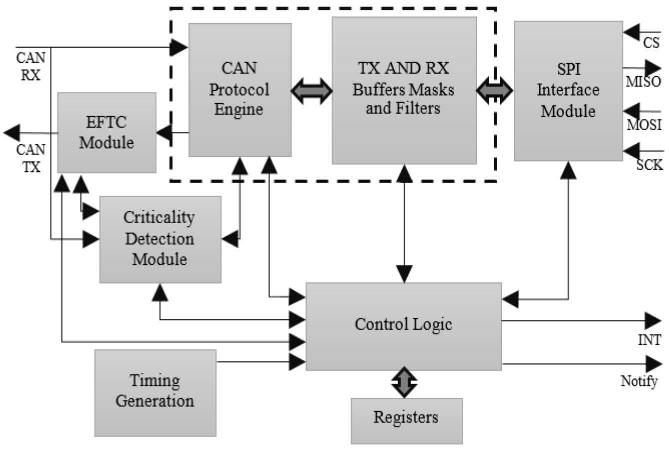

Figure 6. Block Diagram of MRMC-CAN

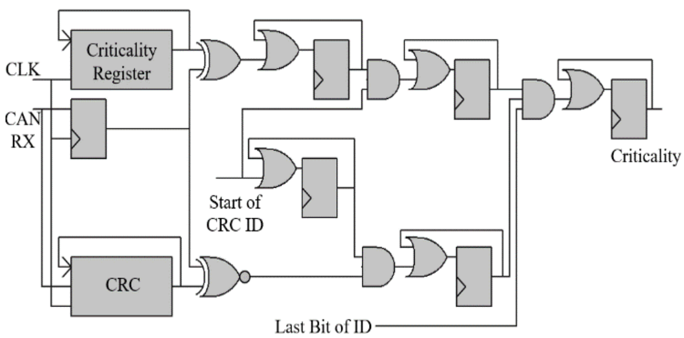

Figure 7. Implementation of Criticality Detection Module

Once the criticality of a message reception was detected by the CD module, the receiver node must control propagation of error flags. For this purpose, if reception of a received message is critical, and an error detected, the receiver node must propagate error flags, otherwise it must be prevented from error flag propagation. Although, preventing error flag propagation due to the incorrect reception of non-critical messages will be reduce bandwidth consumption and increase the real-timeness of CAN network, it causes Bit Error in the receiving node that was prevented from error flag propagation. In CAN nodes, a Bit Error is detected when the value of the monitored bit differs from the transmitted bit. To resolve this issue, if a receiver node, receives an erroneous non-critical message, in addition to preventing it from error flag propagation, CAN-TX signal must be routed to CAN-RX signal, until ongoing message transmitted. In MRMC-CAN, EFTC module, controls error flag propagation. As shown in Figure 8, EFTC module is implemented with one flip-flop two multiplexer, one AND gate and an OR gate.

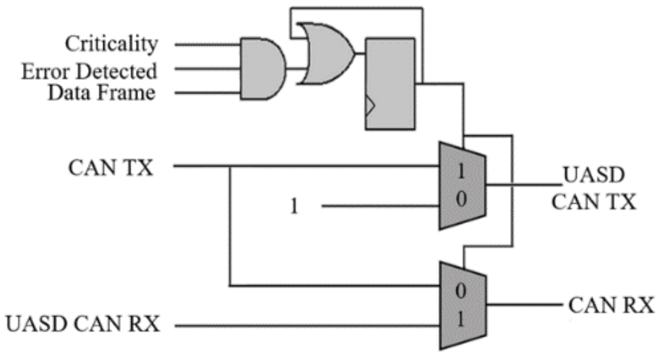

Figure 8. Implementation of EFTC Module

As shown in Figure 9, disconnecting receiver node that received an erroneous non-critical message, creates a silence interval. During this interval, the disconnected node, must be prevented from transmitting its messages. Therefore, the standard CAN transmission procedure must be changed as Figure 10.

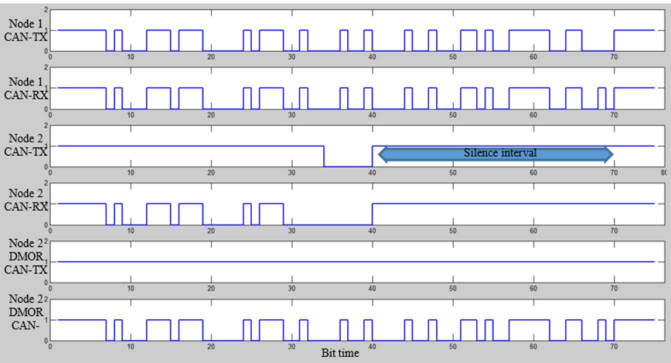

Figure 9. Silence Interval

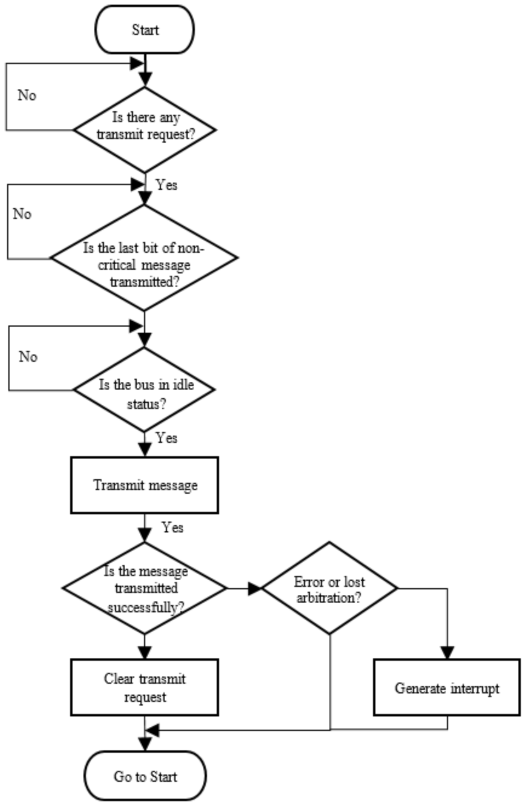

Figure 10. MRMC-CAN Message Transmission Flowchart

IV. Response Time Analysis

In the industrial context, response time has a direct impact on the correctness of operations, therefore determining whether or not a message can be transmitted on its deadline is important, at design time. For this reason, in [10, 27], the first timing analysis of the CAN networks called Tindell's Analysis presented. In Tindell's Analysis the worst case response time of messages is determined by parameters including jitter, worst case queuing delay, and required transmission time. Since the effect of error events was not considered in the Tindell's Analysis, it is modified by Davis and Burns [28], assuming that the maximum number of errors on the bus at time interval t' is given by function F (t). Although in the analysis proposed by Davis and Burns, the effect of errors on the worst case response time is considered, but their analysis is not sufficient to examine the worst case response time of the MRMC-CAN method.

For this reason, in this paper, Tindell's Analysis is modified by considering the probability of erroneous message reception in a node for which this reception is critical. In this analysis it is assumed that the system is composed of periodic and sporadic messages, which are enqueued at periodic or minimum time intervals, and a message M_i is characterized by an 8-tuple: $\langle id_i, m_i, D_i, T_i, J_i, Pe_i, Pea_i, Pecn_i \rangle$. Where in this tuple id_i is identifier, m_i is the payload length, D_i is deadline, T_i is transmission period or minimum time interval, and J_i is jitter of periodic messages. In Tindell's Analysis, the maximum message transmission time C_i is determined by considering the stuffing bit with Equation 1, in which τ_{bit} is the transmission time of one bit.

$$C_i = \left(\left\lfloor \frac{34 + 8m_i}{5} \right\rfloor + 47 + 8m_i \right) \tau_{bit} \quad (1)$$

Equation (1) gives the maximum message transmission time if the probability of error occurrence during message transmission Pe_i is zero, otherwise the maximum transmission time Ce_i^{max} can be found iteratively through (2). Starting value of this recurrence relation is $Ce_i^{(0)} = C_i$ and iterates until m=16, because a node enter to error-passive state after a maximum of 16 times erroneous message transmission.

$$Ce_i^{(n+1)} = (1 - Pe_i) Ce_i^{(n)} + Pe_i (2Ce_i^{(n)} + 23) \quad (2)$$

Equation (2) specifies the maximum message transmission time for Standard-CAN, however for MRMC-CAN, the probability of error occurrence in the arbitration field Pea_i , and the probability of erroneous message reception in a node for which this reception is critical $Pecn_i$ must be considered as in (3).

$$Ce_{e_i}^{(n+1)} = (1 - Pe_i) Ce_i^{(n)} + Pe_i Pea_i ((1 - P_{ecn}) Ce_i^{(n)} - 23Pecn_i + 55) + Pe_i Pecn_i (Ce_i^{(n)} + 23) \quad (3)$$

After obtaining the maximum transmission time, the blocking time B_i, which is caused by messages by higher priority than M_i, is calculated through (4). Then the worst-case queuing delay is obtained through iterative relation of (5). This recurrence relation start with $W_i^{(0)} = B_i$ and iterates until $W_i^{(m+1)} = W_i^{(m)}$. The worst-case response time of message M_i, is given by $W_i^{(m)}$.

$$B_i = \max_{k \in hp(i)} (C_k^{\max}) \quad (4)$$

$$W_i^{(n+1)} = B_i + \sum_{k \in hp(i)} \left\lceil \frac{W_i^{(n)} + J_k + \tau_{bit}}{T_k} \right\rceil C_k^{\max} \quad (5)$$

V. Implementation and Evaluation

In this section MRMC-CAN implemented and evaluated. First, employed fault injector is introduced, then MRMC-CAN is implemented as hardware-based on FPGA and software-based on ARM Cortex M0. Then, to evaluate the real-timeness improvement of proposed method, the response-time and consumed bandwidth is analyzed, and to evaluate the overhead of proposed method parameters including area overhead, hardware utilization, and ROM and RAM usage are evaluated.

A. Prototype Implementation and Simulation

Since in the proposed method receiver nodes make decision about message retransmission, evaluation must be done through individually fault injection to each node. Therefore fault injection performed based on Independent Fault Injector (IFI) [29]. As shown in Figure 11, this fault injector is implemented with one CAN Transceiver, fault injector controller and a multiplexer for each node. Fault injector controller receives fault injection command through RS232 interface.

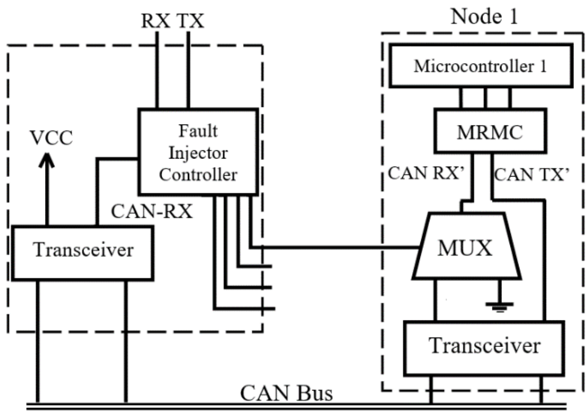

Figure 11. Modified IFI Fault Injector Diagram

To simulate and verify the proposed method, the CAN network implemented as shown in Figure 12 and Figure 13. In this implementation, Node 1 has a software-based MRMC-CAN implemented on ARM Cortex M0 (STM32F030), Node 2 has a hardware-based MRMC-CAN implemented on Xilinx Spartan6 (6SLX9TQG144), and Node 3 has a standard CAN Controller. In this Network IFI fault injector is implemented with a 74HC153 and one Arm Cortex M0 as IFI Controller.

To simulate the implementation, a PC-based logic analyzer is employed. In simulation fragment of Figure 14-a Node1 transmits a message with an identifier (ID'), which matches with critical identifier list of Node 2. During this message transmission, fault injected to Node 2, as a result MRMC-CAN allows Node 2 to transmit error flag. In contrast, in Simulation fragment of Figure 14-b, Node 1 transmits a message with identifier (ID''), which is none-critical for Node2, and during this message transmission fault injected to Node 2. In this situation Node 2 prevented from error flag propagation.

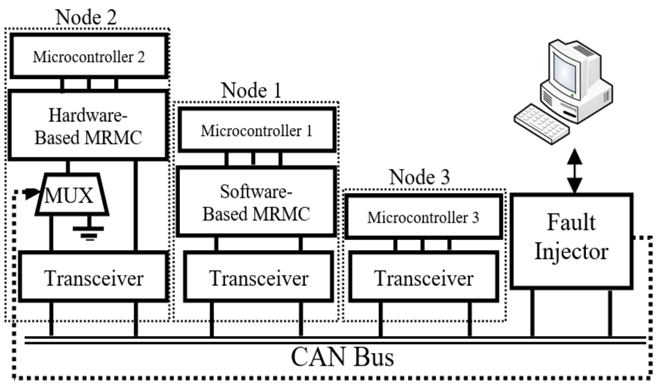

Figure 12. CAN Network Diagram

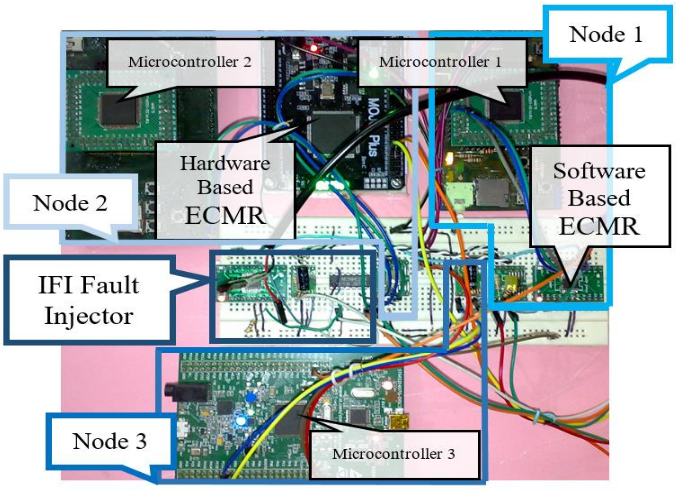

Figure 13. CAN Network Implementation

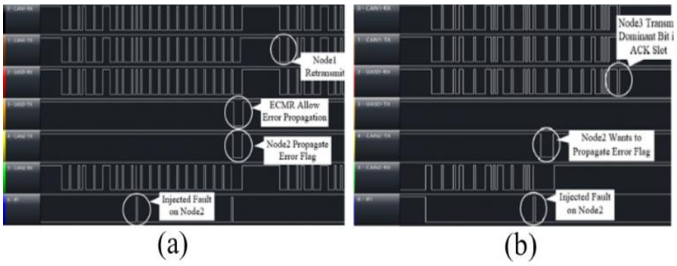

Figure 14. MRMC-CAN error transmission permission simulation

B. *Response Time and Consumed Bandwidth*

In this section, response time and consumed bandwidth of MRMC-CAN evaluated in comparison with standard CAN and WCTER-based approaches [13, 30]. In WCTER-based methods, mixed criticality levels are considered for message set, and the possibility of message retransmission in event of an error is determined based on these levels. The benchmark message set for this evaluation is generated by NetCarBench [31]. However, since this tool does not support the criticality of message reception, it should be modified to generate message sets as Table I. After generating the benchmark message sets, the test board is implemented as shown in Figure 16 and Figure 17. In this test board Electronic Control Units (ECU) and software-based MRMC-CANs implemented on ARM Cortex M3 and ARM Cortex M0 microcontrollers respectively.

The overall response time and consumed bandwidth evaluation process is as follow. Firstly, each ECU employs its internal timer to transmit its message based on the periods defined in the message set benchmark. Secondly, IFI fault injector controller inject faults to the nodes based on received command from PC. Finally, response time is obtained by recording the time it takes for messages to reach their destinations, and consumed bandwidth is obtained based on how long the bus is occupied per unit time. The results of evaluation are presented in Figure 17 and Figure 18. As shown in Figure 17 the MRMC-CAN method in comparison with standard CAN and WCTER-based approaches improves Consumed bandwidth by average 10.5% and 4.4% respectively, and also as shown in Figure 18 proposed method improves response time in comparison with standard CAN by average 36.19%.

TABLE I. MESSAGE SET CREATED BY MODIFIED NETCARBENCH

Node 1							
IDD	***payload***	***Period***	***Deadline***	***Must Receive By Node***			
				1	***2***	***3***	***4***
241	8	20	10	□	■	■	□
738	8	2000	20	□	□	□	■
248	8	20	5	□	□	■	■
Node 2							
IDD	***payload***	***Period***	***Deadline***	***Must Receive By Node***			
				1	***2***	***3***	***4***
787	8	1000	10	■	□	□	□
306	4	50	25	■	□	■	□
455	8	100	30	□	□	□	■
Node 3							
IDD	***payload***	***Period***	***Deadline***	***Must Receive By Node***			
				1	***2***	***3***	***4***
215	8	20	10	■	□	□	■
Node 4							
IDD	***payload***	***Deadline***	***Deadline***	***Must Receive By Node***			
				1	***2***	***3***	***4***
523	8	100	5	□	■	■	□

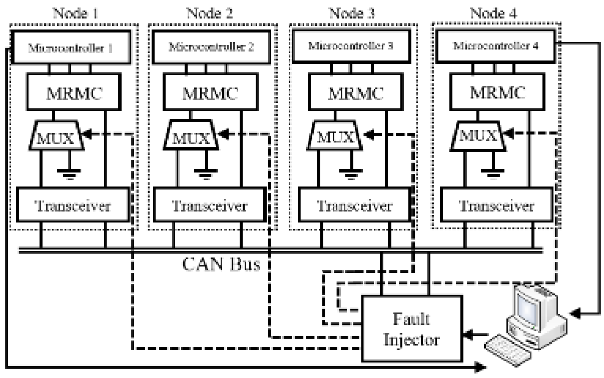

Figure 15. Diagram of Test Board

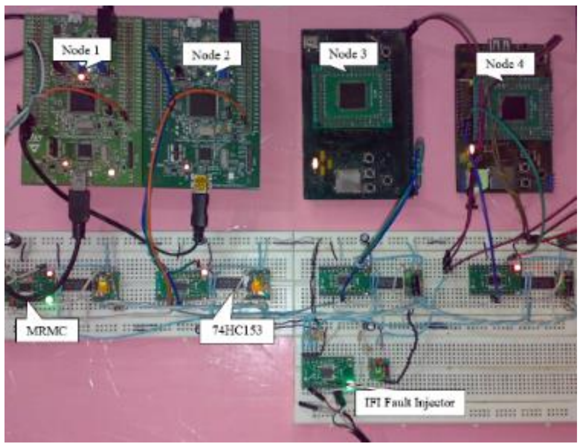

Figure 16. Implementation of Test Board

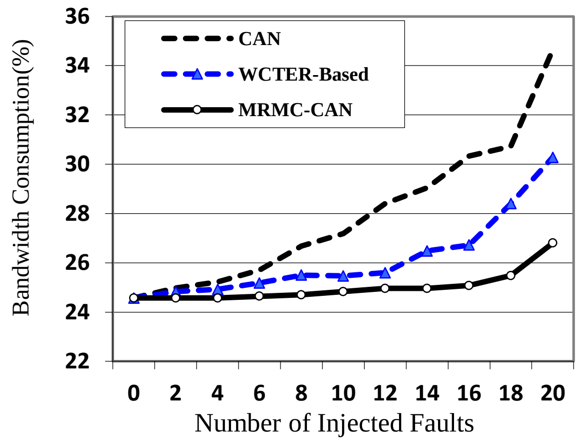

Figure 17. Consumed Bandwidth

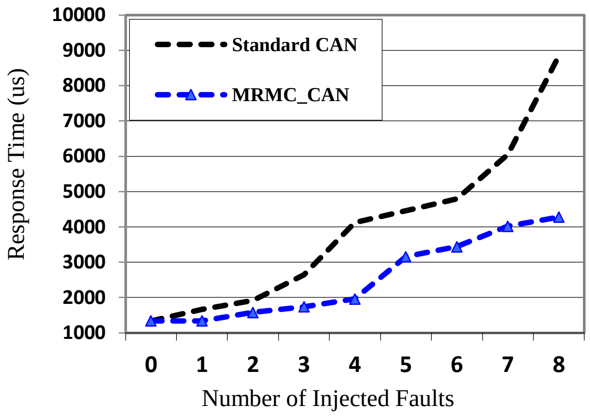

Figure 18. Response Time

C. *Area OverHead and Utilization*

The overall area overhead of hardware-based MRMC - CAN modules is represented in Table II. In this table the hardware size is given in term of two-input NAND gate count. In addition to area overhead, Table III shows the device utilization of hardware-based MRMC-CAN modules in Xilinx Spartan 6. Moreover to show the software implementation capability of the proposed method in microcontroller with limited resources, the ROM, RAM usage of proposed method is shown in Table IV.

TABLE II. AREA OVERHEAD OF MRMC-CAN MODULES

	Number of Gates	Overheads (%)
Basic CAN Controller	20643	-
Active MRMC	111	0.5

TABLE III. HARDWARE BASED MRMC DEVICE UTILIZATION FOR 6SLX9TQG144

	Used	Available	Utilization
Global Buffers	1	16	6.25 %
Function Generators	111	5720	1.94 %
Dffs or Latches	79	11440	0.69 %

TABLE IV. SOFTWARE BASED MRMC ROM, RAM USAGE FOR STM32F030F4PX

	Used	Available	Usage
RAM	29	4096	0.7 %
ROM	5024	16384	30.7 %

VI. CONCLUSION AND FUTURE WORKS

Although, CAN communication protocol, is employed in industrial fields, due to its low-cost implementation, low response time and noise robustness, temporal redundancy feature of this protocol makes it vulnerable against timing-failure. Since real-time capability is an essential requirement for IIoT Communications, this paper presents MRMC-CAN method in which message retransmission is performed based on the criticality of message reception. Message retransmission based on the criticality of message reception, improves the real-timeness of CAN protocol by preventing message retransmission in the event of receiving an incorrect message in nodes that receiving of this message is not critical. The MRMC-CAN method in comparison with standard CAN and WCTER-based approaches reduces consumed bandwidth by average 10.5% and 4.4% respectively and improves response time in comparison with standard CAN by average 36.19%.

REFERENCES

[1] L. Zhang, F. Yang, and Y. Lei, “Tree-based intermittent connection fault diagnosis for controller area network,” IEEE Trans. Veh. Technol., vol. 68, no. 9, pp. 9151–9161, Sep. 2019.

[2] H. Kimm and M. Jarrell, “Controller area network for fault tolerant small satellite system design,” in 2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE). IEEE, 2014, pp. 81–86.

[3] X. Jiang, M. Lora, and S. Chattopadhyay, "An Experimental Analysis of Security Vulnerabilities in Industrial IoT Devices," ACM Transactions on Internet Technology, 2020.

[4] J. Y. Guido Marchetto, Riccardo Sisto and A. Ksentini, “Formally verified latency-aware vnf placement in industrial internet of things,” in 14th IEEE International Workshop on Factory Communication Systems (WFCS), Imperia, Italy, 2018.

[5] S. Saadaoui, A. Khalil, M. Tabaa1, M. Chehaitly, F. Monteiro and A. Dandache, “Improved many to one architecture based on discrete wavelet packet transform for industrial IoT applications using channel coding,” Springer, Journal of Ambient Intelligence and Humanized Computing, vol. 11, no. 12, Dec. 2020.

[6] B. Chen and J. Wan, "Emerging trends of ml-based intelligent services for industrial internet of things (iiot)", In Proc. 2019 IEEE Computing, Communications and IoT Applications (ComComAp), 2019.

[7] H. Kong, J. Cheng, K. Narayanan and J. Hu, "DUCER: a Fast and Lightweight Error Correction Scheme for In-Vehicle Network Communication", 2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES), 2018.

[8] M. B. N. Shah, A. R. Husain, S. Punekkat, and R. Dobrin, “A new error handling algorithm for controller area network in networked control system,” Computer in Industry, vol. 64, no. 8, pp. 984–997, 2013.

[9] D. Kästner, J. Jersak, C. Ferdinand, P. Gliwa and R. Heckmann, "An integrated timing analysis methodology for real-time systems", SAE Technical Paper , pp. 0148-7191, 2011.

[10] K. W. Tindell, H. Hansson and A. J. Wellings, "Analysing real-time communications: Controller area network (CAN)", Proc. 15th RealTime Systems Symp., 1994.

[11] G. Buja, J. R. Pimentel and A. Zuccollo, "Overcoming babbling-idiot failures in CAN networks: A simple and effective bus guardian solution

for the FlexCAN architecture", IEEE Trans. Ind. Informat., vol. 3, no. 3, pp. 225-233, Aug. 2007.

[12] I. Broster and A. Burns, "An analysable bus-guardian for event-triggered communication", Proc. 24th IEEE Real-Time Systems Symp. (RTSS'03), pp. 410-419, 2003.

[13] A. Burns and R.I. Davis, "Mixed criticality on controller area network", In Proc. Euromicro Conference on Real-Time Systems (ECRTS), pp. 125-134, 2013.

[14] H. Sivencrona et al., A Novel Distributed Add-on Concept to Detect and Recover from Bus Failures in Controller Area Networks using REDCAN, Proc. 8th International CAN Conference (ICC-08), Las Vegas, Nevada, USA, March 2002.

[15] M. Barranco, J. Proenza, G. Rodriguez-Navas and L. Almeida, "An active star topology for improving fault confinement in CAN networks", IEEE Trans. Ind. Electron., vol. 2, no. 2, pp. 78-85, May 2006.

[16] M. Barranco, L. Almeida and J. Proenza, "ReCANcentrate: A Replicated Star Topology for CAN Networks", Proc. 10th IEEE Int'l Conf. Emerging Technologies and Factory Automation (ETFA 05), pp. 469-476, 2005.

[17] A. Muhammad, D. Ayavoo and M. J. Pont, "A Novel Shared-Clock Scheduling Protocol for Fault-Confinement in CAN-based Distributed Systems", IEEE 5th International Conference on System of Systems Engineering, 2015.

[18] Radovan Miucic, S. M. Mahmud, Zeljko Popovic, "An Enhanced Data-Reduction Algorithm for Event-Triggered Networks," IEEE Transactions on vehicular Technology, Vol. 58, No.6, pp. 2663-2678, July, 2009.

[19] S. Misbahuddin, S. M. Mahmud and N. Al-Holou, "Development and performance analysis of a data-reduction algorithm for automotive multiplexing", IEEE Trans. Veh. Technol., vol. 50, no. 1, pp. 162-169, Jan. 2001.

[20] P. R. Ramteke, S.M. Mahmud, "An Adaptive Data-Reduction Protocol for the future In-Vehicle Networks," Soc. Automotive Eng., SAE Paper 2005-01-1540, 2005.

[21] R. Miucic and S. M. Mahmud. An improved adaptive data reduction protocol for in-vehicle networks. In SAE, editor, In-Vehicle Software & Hardware Systems, number 2006-01-1327 in Transactions Journal of Passenger Cars: Electronic and Electrical Systems, pages pp. 650-658. SAE, April 2006. SAE 2006 World Congress & Exhibition.

[22] Wu, Z. Piao, J. H. Kim and J. G. Chung, "CAN compression using signal rearrangement", Circuits and Systems (APCCAS) 2014 IEEE Asia Pacific

[23] Y. Wu and J. Chung, "Efficient controller area network data compression for automobile applications", Frontiers of Info. Technol. & Electro. Eng., vol. 16, no. 1, pp. 70-78, Jan. 2015.

[24] Y.-J. Wu and J.-G. Chung, "An improved controller area network data-reduction algorithm for in-vehicle networks", IEICE Trans. Fundamentals, vol. E100-A, no. 2, pp. 346-352, Feb 2017.

[25] Y.-J. Kim, Y Zou,Y.-E. Kim, and J.-G. Chung, " Multi-Level Data Arrangement Algorithm for Can Data Compression", Springer, International Journal of Automotive Technology, vol. 21, no. 6, pp. 1527-1537, 2020.

[26] K. Park, M. Kang, and D. Shin, "Mechanism for Minimizing Stuffing-bit in CAN Messages," The 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON'07), pp. 735-737, Nov. 2007.

[27] K. Tindell and A. Burns, "Guaranteeing message latencies on Controler Area Network", Proceedings of the 1st International CAN Conference, pp. 1, 1994-September.

[28] R. I. Davis, A. Burns, R. J. Bril and J. J. Lukkien, "Controller area network (can) schedulability analysis: Refuted revisited and revised", Real-Time Syst., vol. 35, no. 3, pp. 239-272, 2007.

[29] G. Rodriguez-Navas, J. Jimenez and J. Proenza, "An architecture for physical injection of complex fault scenarios in CAN networks", Proc. IEEE Emerging Technol. Factory Autom., vol. 2, pp. 125-127, 2003.

[30] H. Aysan, A. Thekkilakattil, R. Dobrin and S. Punnekkat, "Efficient Fault Tolerant Scheduling on Controller Area Network", Proceedings of the 2010 IEEE Conference on Emerging Technologies and Factory Automation, pp. 1-8, 2010.

[31] C. Braun, L. Havet, and N. Navet, "NETCARBENCH: a benchmark for techniques and tools used in the design of automotive communication systems", in 7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems, 2007, pp. 321-328, Available at http://www.netcarbench.org.