2011 Fourth International Conference on Intelligent Computation Technology and Automation

Online Fault Diagnosis for Controller Area Networks

Hu Huangshui, Qin Guihe
College of Computer Science and Technology, Jilin University, Changchun 130012, China
huhs08@mails.jlu.edu.cn

Abstract—**The controller area network (CAN) is a field bus that has been widely used in distributed embedded systems due to its inexpensiveness, robustness, guarantee of latency times as well as error detection. However, electromagnetic interference from the operational environment and hardware malfunction may cause short disturbances and permanent failures respectively, inevitably bring on communication errors or even break off the communication. To enhance the dependability of CAN communication, research on the on-line fault diagnosis is carried out in this paper, a Monitor is designed to diagnose faults in CAN nodes and a hybrid method with active and passive mode is presented to diagnose faults among communication links. We analyze the relative works, describe the CAN fault model, and focus on the fault diagnosis mechanisms.**

Keywords- Control Area Network (CAN), fault diagnosis, on-line, active and passive diagnosis

I. INTRODUCTION

The controller area network has been employed in many domains [1]-[8] during the past years due to its advantages, for instance, automobile, security and the other industry control. In particular, its application fields are still significantly increasing.

However, CAN shows drawbacks with respect to reliability and scalability such as babbling idiot failures [9] in despite of powerful measures for error detection, signaling and self-checking are implemented. On the other hand, electromagnetic interference in some applications will result in severe transmission error. In addition, a CAN node consists of a microprocessor, a CAN controller, a CAN transceiver and other peripherals, so hardware and software faults exist without doubt. Once faults occur in the network, the communication continuity is disrupted. Therefore, fault diagnosis tools in CAN systems are strongly required to ensure that all CAN bus devices are able to correctly and reliably interact as specified [10].

To address the aforementioned issues of CAN bus network, we propose a solution to on-line diagnose the faults of CAN nodes and communication links. We analyze the potential hardware and software faults of a CAN node, and design a Monitor to real-time deal with the faults. Furthermore, we apply an innovative method mixed active and passive mode to implement on-line fault diagnosis for the communication links. A Manager is defined and used to passively receive messages from the nodes as well as actively inquire the nodes' state. By analyzing all the information, the Manager can diagnose the faults among the communication links.

The rest of this paper is organized as follows: the relative works are discussed in Section II. We present the CAN fault model in Section III, herein, potential faults types are described in detail. In Section IV, we introduce the fault diagnosis mechanism. Finally, conclusions are presented in section V..

II. RELATIVE WORKS

CAN was initially introduced in 1980s and aimed to apply in automotive applications used to connect electronic control units (ECUs) [11], from then on, CAN rapidly expanded its application domains due to its predominant performance. Whereas, CAN possesses some faults such as shorted medium, babbling idiot failures, communication partition which can significantly impact reliability of CAN communication. Accordingly, many fault diagnosis techniques for CAN has been studied.

In [12] and [13], the method using replicated transmission media was presented, and a method using reconfigurable transmission media was studied in RedCAN [14], *Zhenye Wang*[15] analyzed advantages of analytical redundancy fault diagnosis and hardware redundancy fault diagnosis in detail, and proposed a resistance detection method based on hardware redundancy to detect and locate the fault point of the CAN-bus permanent failure. But it is obviously that the redundant media in these approaches results in high-cost and introduces new fault places. Moreover, replicated media may bring "common-mode" interference to CAN communication because of the proximity of the wires. In addition, a faulty node sending error information to all media can't be prevented. On the other side, RedCAN requires specific hardware which increases risk of failure.

On account of the disadvantages of hardware redundancy approaches, people have worked over non-redundancy methods to diagnose faults in CAN systems. Bus-guardians were proposed in[16]-[17], they monitored the node to response for abnormal behaviors, but it was impossible for bus-guardians to diagnose shorted media fault.

Afterwards, people knew the inherent drawbacks of bus topology, and methods of star topology were proposed in [18]-[19]. *Manuel, B*[18] designed an active star topology called CANcentrate with an active hub, which prevented error propagation from any of its ports to the others. In [20], a CAN router with a star topology was introduced to detect and isolate node failures in the value and time domain, the CAN router could improve the use of the bandwidth, extend

978-0-7695-4353-6/11 $26.00 © 2011 IEEE
DOI 10.1109/ICICTA.2011.126

the possible overall wire length and support multiple namespaces. *Amir Muhammad*[21] explored a star based Shared-Clock algorithm to isolate faults related to data corruption or network hardware malfunction, support a software-based *"Port Guardian"* mechanism to ensure a medium- or node based fault on one link of the star cannot propagate to the rest of the network..

III. CAN FAULT MODEL

The CAN protocol provides elaborate error detection approach to detect five different error types such as bit error, stuff error [22], specified in the data link layer of ISO 11898 [23]. Also, the CAN protocol makes a node signal an error condition by transmitting an active error flag or a passive error flag. Moreover, the CAN protocol enables a node distinguish short disturbance from permanent failures by fault confinement mechanism which determines the unit is error active, error passive or bus-off state. All these make the total residual error probability for undetected corrupted messages less than *message error rate * 4.7 * 10^{-11}*.

In spite of the very low error probability of CAN protocol, it is still provided with some defects as before-mentioned in Section II. Besides, the solutions proposed in the literatures to diagnose the faults in CAN are not completely effective because they only solve one or several sides of the faults in CAN system. In this paper, we develop an all-sided method to manage faults occurring in CAN system. And the CAN fault model is constructed in this section.

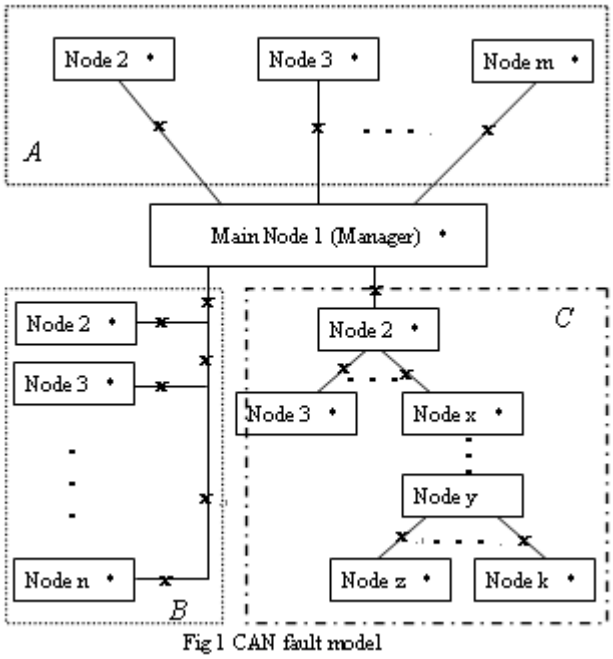

Fig 1 CAN fault model

The fault model shows all kinds of potential faults happening in a CAN network. The fault model is depicted in Fig 1. We can diagnose the faults in CAN system constructed with a star topology in *A*, a bus topology in *B* or a tree topology in *C* combined with the main node. All the possible faults are sorted to node fault denoted with * or link fault labeled by x in fig 1.

It is well known to us that a CAN node usually consists of a microprocessor, a CAN controller, a CAN transceiver, and other integrated circuits like memories, terminal resistances, watchdog and so on. The hardware or software failures will cause the node faults such as stuck-at-dominant, stuck-at-recessive, bit-flipping, or babbling-idiot. Whatever fault it is, the damaged node issues some error information to the network randomly and greedily, finally the communication is completely destroyed. In order to avoid arising of this situation, we design a run-time Monitor in every node to guarantee the normal communication. Detail diagnosis mechanism is described in Section III.

Moreover, all the shorted media, media interruption, attenuation or distortion of the communication signal can result in communication link faults, so we propose a method using an active and passive mode to get information from each node of the CAN system in order to locate the communication link fault such as medium partition fault. We present the diagnosis mechanism in the next section. In order to describe our proposition, we make assumptions that a main node usually close to the operator act as a Manager in the system and every CAN controller meets the CAN specifications and has self-loop function existing in many off-the-shelf CAN controllers.

IV. FAULT DIAGNOSIS MECHANISM

We implement run-time diagnosis on node faults and communication link faults, and now we address the relative method and algorithms as follow.

Node Faults Diagnosis

In order to deal with the hardware and software faults in a CAN node, we design a program called Monitor, at fist the Monitor checks the memories including random access memory (RAM) and read only memory (ROM). It adopt CRC-16 checksum to judge the fault of the ROM. Due to the contents are invariable since the ROM is programmed, so the checksum is unique and stored in one or two ROM unit(s) when programmed. If the computed checksum is identical with data in the specified ROM unit(s), the ROM is normal, otherwise a warning arises and the task is interrupted. On the other hand, the Monitor uses line-by-line scan to diagnose the RAM, when an address line *Ai* is examined, at first, change the value of *Ai* from 0 to 1 with no change occurring on the other address lines *Aj* ($i \neq j$), and two different data are written to the two different addresses, then read them back in turn, if the data are identical, it means the *Ai* address line out of order, the Manage records the fault. After all the address lines are examined, the Monitor judges the state of the RAM according to the records, if the number of the records are more than half of the number address lines, a warning arises and the task is interrupted. Thereafter, it initialize the CAN

controller with properly configuration like bit rate, filter mode etc. Through reading the transmit error count and the receive error count, the Monitor know the current state of the node whether it is in error active, error passive or bus-off state. Once detecting a fault, it sends a fault message to the Manager (defined later). Moreover, the Monitor can test the node whether it sends information self-acting or continuously using the self-loop function with period t1. In addition, the Monitor uses a timer to periodically feed the watchdog in order to prevent error pointer and infinite loop in software. Consequently the babbling idiot fault and bit flipping fault can be withheld. Finally, if the transmit error count or the receive error count reaches the threshold set in advance, the Monitor shuts down or powers off the CAN controller, and restarts the CAN controller after a period t2. The diagnosis mechanism is described in Fig.2.

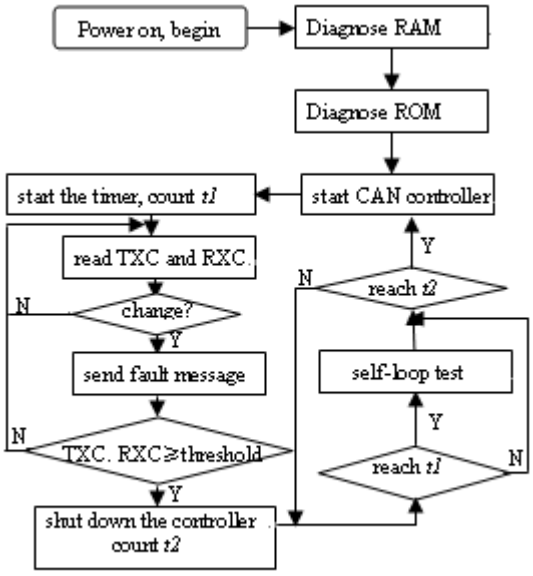

Fig.2 Diagnosis mechanism of the Monitor

Communication Link Faults Diagnosis

In order to diagnose communication link faults, we design a program called Manager in the main node to diagnose faults in active and passive mode. Fist of all, we consider every node has a unique identifier number and can transmit the current message number. The Manager maintains a cache to make a list for the other nodes called source nodes in the network. As illustrated in Fig. 3,

···	source ID	Sequence number	Failure times	···

Fig. 3 The cache content of the Manager

the source node ID and the sequence number of the recently received message from a node are put in the list. Besides, the failure times are attached. As shown in algorithm 1, upon receiving a message, the Manager first checks its own cache, if there is no entry for the source node ID of this message, then creates a new entry for this source node in its cache and records the sequence number for the message. If

Algorithm 1 Passive_Diagnosis (message n)

1: check cache
2: if no entry for source node of n
3: create entry with source node ID and sequence number in n,
4: Failure times=0
5: else if entry exists and sequence number continuous
6: update entry with new sequence number
7: else if entry exists and sequence number not continuous
8: Failure times= new sequence number – old sequence number,
9: update entry with new sequence number
10: end if
11: return

there exists an entry in the cache for the source node and the sequence number in the message is consistent with the cache entry, the Manager updates the cache entry with the new sequence number. If the sequence number of the packet is not consistent with that recorded in the cache entry, it might be due to the message loss. The Manager then updates its cache entry with the new sequence number of this message and adds failure times with difference between the new sequence number and the old sequence number. According to the entries in the cache, the Manager can make an elementary decision on the quality of the link between the nodes and the main node. For example, more Failure times between the source node and the main node means worse link quality between them.

At the same time, the Manager takes active steps to judge the links be in order or disorder. Here, we specially emphasize that the query are made only in bus idle state and the query period is much larger than the transmission time of all the messages produced in the process of inquiring so as not take much extra loads on the bus. The Manager allocates another cache to store the results of the active and periodical query. The data structure is depicted in Fig. 4, destination node ID and the total query times and

···	Destination ID	Ack times	Total times	···

Fig. 4 The data structure of Qlist

acknowledgment times of every node are added in the list called Qlist. As described in Algorithm 2, once starting the query, the Manager cyclically sends query message with the destination node ID increased by one every time, and waits a short time *(tw)* between sending to receive the acknowledgment from the destination node. During *tw*, once acknowledgment is received, the Manager sends message for next destination node immediately. Finally, the Manager can verdict the link quality based on the probability of the acknowledgment.

Therefore, the Manager can easily locate the communication link faults by analyzing the contents of the

two caches and the error notice messages from each node.

```
Algorithm 2   Active_Diagnosis ( )
1:  initialize tp=0; i=1;
2:  send a message to node i; start timer;
3:  if no entry for node i
4:  create entry with node i; and Ack times=0; Total times=1;
5:  else
6:  Total times=Total times+1;
7:  end if
8:  while ( tp<tw)
9:  if receive the acknowledgement form node
10:  Ack times=Ack times+1; break;
11:  end if
12:  end while
13:  tp=0;
14:  i=i+1;
15:  if( i==n)        (n: number of nodes)
16:  return;
17:  else
18:  go to 2;
19:  end if
```

V. CONCLUSIONS

Although there have been many approaches proposed for fault diagnosis of CAN systems, few works have been devoted towards on-line integrated diagnosis tool in the field. In this paper, we propose a comprehensive method to diagnose CAN nodes and communication links faults, especially, we present a passive and active diagnosis approach which can be efficiently implemented and applied to a CAN system providing on-line fault diagnosis. Moreover, the proposed approach only injects a little overhead to the original system. We implement our approach and verify its effectiveness in a field test in our joint-defense alarm project.

REFERENCES

[1] Sun Ning, Chen Nan, Zhang Bingjun, Pi Dawei, Zhong Guohua, "Effects of CAN Communication Delay on ABS Fuzzy Control," Jiangsu Daxue Xuebao, Vol.31, No.4, pp.397-402, July 2010.

[2] Mazran Esro, Amat Amir Basari, Siva Kumar, A. Sadhiqin M I, Zulkifli Syariff, "Controller Area Network(CAN) Application in Security System," Proceedings of World Academy of Science, Engineering and Technology, Vol.59, pp.299-302,Nov. 2009

[3] P. Marino, F. Poza, M. Dominguez, and S. Otero, "Electronics in automotive engineering: A top-down approach for implementing industrial fieldbus technologies in city buses and coaches," IEEE Trans. Ind. Electron., vol. 56, no. 2, pp. 589–600, Feb. 2009.

[4] J. M. Giron-Sierra, C. Insaurralde, M. Seminario, J. F. Jimenez, and P. Klose, "CAN bus-based distributed fuel system with smart components", IEEE Transactions on Aerospace and Electronic Systems, Vol. 44, No. 3, pp. 897-912, July 2008.

[5] A. Bergnoli, A. Bertolin, R. Brugnera, E. Carrara, A. Cazes, F. Dal Corso, S. Dusini, G. Felici, A. Garfagnini, A. Longhin, U. Mantello, A. Mengucci, A. Paoloni, L. Stanco, V. Sugonyaev, F. Terranova, and M. Ventura, "The OPERA Spectrometer Slow Control System," IEEE Transactions on Nuclear Science, Vol. 55, No.1, pp. 349-355, Feb. 2008.

[6] F. Gil-Castineira, F. Gonzalez-Castano, and L. Franck, "Extending vehicular can field buses with delay-tolerant networks," IEEE Trans. Ind. Electron., vol. 55, no. 9, pp. 3307–3314, Sep. 2008.

[7] E. Desa, P. K. Maurya, A. Pereira, A. M. Pascoal, R. G. Prabhudesai, A. Mascarenhas, R. Madhan,; S. G. P. Matondkar, G. Navelkar, S. Prabhudesai, and S. Afzulpurkar, "A Small Autonomous Surface Vehicle for Ocean Color Remote Sensing," IEEE Journal of Oceanic Engineering, Vol. 32, No. 2, pp. 353-364, April 2007.

[8] D. Lim and A. Anbuky, "A distributed industrial battery management network," IEEE Trans. Ind. Electron., vol. 51, no. 6, pp. 1181–1193, Dec. 2004.

[9] K. Tindell and H. Hansson. Babbling idiots, the dual-priority protocol, and smart can controllers. In Proceedings of the 1st Int. CAN Conference, 1994.

[10] William Prodanov, Maurizio Valle, Roman Buzas, "A Controller Area Network Bus Transceiver Behavioral Model for Network Design and Simulation", IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 56, NO. 9, SEPTEMBER 2009

[11] M. Farsi, K. Ratcliff, and M. Barbosa, "An overview of controller area network," Comput. Control Eng. J., vol. 10, no. 3, pp. 113–120, Aug. 1999.

[12] J. Rufino, P. Veríssimo, and G. Arroz, "A Columbus' Egg idea for CAN media redundancy," in Proc. 29th Int. Symp. Fault-Tolerant Computing, Madison, WI, Jun. 1999.

[13] M. J. Short, and M. J. Pont, "Fault-Tolerant Time-Triggered Communication Using CAN," IEEE transactions on Industrial Informatics, vol. 3, No. 2, May 2007, Page(s): 131-142.

[14] L.B. Fredriksson, "CAN for critical embedded automotive networks," IEEE Micro, vol. 22, no. 4, pp. 28–35, Jul.–Aug. 2002.

[15] Zhenye Wang, Xiaosong Guo, Chuangqiang Yu, "Research of Fault-tolerant Redundancy and Fault Diagnosis Technology Based on CAN", 2010 2nd international conference on advanced computer control, pp. 287–291, March 2010.

[16] I. Broster. and A. Burns., "An analyzable bus-guardian for event triggered communication," Proceedings of 24th IEEE Real-Time Systems Symposium, Dec. 2003, Page(s): 410–419

[17] Giuseppe, B.; Juan, P.; Alberto, Z., "Overcoming Babbling-Idiot failures in CAN networks: A simple and effective Bus Guardian solution for the FlexCAN architecture," IEEE transactions on industrial informatics, vol. 3, No 3, August 2007, Page(s): 225-233.

[18] Manuel, B.; Julián, P.; Guillermo, N.; Luís. A., "An active star topology for improving fault confinement in CAN networks," IEEE transactions on industrial informatics, Vol 2, No 2, May 2006, Page(s): 78-85.

[19] P. Sathish, P. Vanaja Ranjan and S. Solai Manohar, "A New Approach for Fault Confinement in CAN-Network", IEEE - ICSCN 2007, MIT Campus, Anna University, Chennai, India. Feb. 22-24, 2007. pp. 551-554.

[20] R. Obermaisser, R. Kammerer," A Router for Improved Fault Isolation, Scalabilityand Diagnosis in CAN"

[21] Amir Muhammad, Devaraj Ayavoo, Michael J. Pont, "A Novel Shared-Clock Scheduling Protocol for Fault- Confinement in CAN-based Distributed Systems", 2010 5th International Conference on System of Systems Engineering

[22] J. Charzinski, "Performance of the Error Detection Mechanisms in CAN," Proceedings of the 1st International CAN Conference, Mainz, September 1994.

[23] ISO-11898, "Road Vehicles - Interchange of digital information controller area network (CAN) for high speed communication," 1993.