13th IMEKO TC10 Workshop on Technical Diagnostics
Advanced measurement tools in technical diagnostics for systems' reliability and safety
June 26-27, 2014, Warsaw, Poland

Validation of CAN Nodes Health Monitoring Method

Tomáš Pospíšil[1], Jiří Novák[2]

[1] *Faculty of Electrical Engineering, Czech Technical University in Prague, pospito7@fel.cvut.cz*
[2] *Faculty of Electrical Engineering, Czech Technical University in Prague, jnovak@ fel.cvut.cz*

Abstract-Controller area network (CAN) is now used in wide range of distributed applications. The quality of communication in such applications is usually an important factor. This paper describes validation of new developed online diagnostic method which aim is to recognize progressively degrading nodes on the CAN bus. The method uses existing network equipment and requires no additional/specialized network nodes. Standard components for CAN bus faults management like node error counters are used instead. Validation of diagnostic model compares results of the simulations with the results achieved on actual CAN bus implementation with progressively degrading node.

I. Introduction

CAN bus is currently used in many distributed industrial applications running with a long maintenance period (several months is not an exception). Progressive degeneration of a node parameters appearing over time may have negative influence on the communication. Some of CAN applications are safety-related, for example in vehicles, trains or industrial automation. Detection of the node degradation could be beneficial for improvement of functional safety of such systems. In paper [1] we introduced a new diagnostic method for early detection of the degradation of network nodes communication on the CAN bus. Development and testing of the diagnostic method were performed by software simulations only. This paper brings more results and provides the validation of the diagnostic method by means of actual CAN network with communication unit equipped with a fault injection circuitry.

The diagnostic method has been built on two basic requirements. The first requirement is the absence of dedicated diagnostic equipment; the diagnostics should be possible using standard CAN controllers and therefore it should easily be integrated into existing CAN nodes and systems. As the CAN network nodes often lack the memory capacity for storing a large amount of diagnostic data for a routine maintenance period (that can be really long), the second requirement is that the method works in real time and amount of data stored in particular network nodes is limited.

Absence of a dedicated diagnostic hardware requires the use of existing tools for faults processing. In case of CAN there are error counters in individual nodes, which values are used to control an error state of particular node. CAN node error states are intended to reduce the influence of faulty node on the rest of the system. This mechanism is activated when the frequency of communication errors is rather high and results in partial or even complete disconnection of the CAN node from the network. The goal of our method is to generate warning far before the degrading node can seriously influence communication within the distributed application. In this paper, after the brief introduction of the diagnostic method, we focus on comparison of the simulated results to the results measured on the actual CAN network.

A. CAN error handling

The CAN error handling mechanism is described in [2] in detail. For our diagnostic purpose there are two baseline scenarios (other variants are also possible, but these two are the most common). In both scenarios the node which first detects an error is marked as leader and other nodes detecting leader's error flag are marked as followers. In the first scenario, the error is initially detected by transmitter and after that the receiving nodes detect the transmitter generated error flag. In this case the transmitter increases its transmit error counter (TEC) value by 8, and all other nodes (receivers) increase their receive error counters (REC) values by 1. In the second scenario, at least one (but possibly more) receiving nodes detect error simultaneously. These receivers increase REC value by 8 and generate an error flag on the bus. All remaining nodes detect the error flag; the transmitter

increases its TEC value by 8, and remaining receivers increase REC value by 1. The first scenario corresponds to a bit error; the second scenario corresponds to the stuff, form, CRC or acknowledgement errors. This mechanism ensures that under the normal conditions data are delivered either to all nodes (no error detected) or to no node (error detected, retransmission follows). In case of a faultless operation the respective error counter is decremented by 1 (REC in case of successful reception, TEC in case of successful transmission) in all nodes.
If a node detects errors too often (because of its local failure); it could reduce the communication throughput of the whole bus (up to zero). To solve this situation, three so called node error states are introduced in the CAN standard. The first state is 'error active' – nodes behave as described above in this chapter. In case the error is detected an active error flag is generated, error information is passed to all other nodes, current transmission is broken and retransmission follows. The second state is 'error passive' – a node detecting an error is not able to break communication by transmission of the active error flag. Instead it sends a passive error flag, but communication continues and all nodes except this one receive the data. The last error state is 'bus off' state; it is reached if the node encounters high error rate during the transmission. In this state the node is disconnected from the bus. Transitions among the error states are controlled by the REC and TEC counter values for each node individually. If both counters are below 128, the node is in 'error active' state. If any of them gets over the limit, node moves to the 'error passive' state. If TEC value is higher than 255 the node moves to the 'bus off' state.
For many CAN bus applications the error states mechanism is not sufficient. Above described approach is based on the reaction on relatively high error rate. The serious communication problem appears first and then it is solved by removing the faulty node from the communication (fully or partially). The aim of our diagnostic method is to avoid switching a node in 'error passive' or even 'bus off' state by using early prediction of potential serious communication problems.

II. Related work

Several methods providing some level of CAN bus health monitoring were developed in past. Some methods for improving safety parameters use a modification of network topology ([3], [4]), or enhanced protocols ([5]). In [3] authors describe an active star topology with a diagnostic algorithm, which aim is to find permanent error nodes (stuck-at-dominant, stuck-at-recessive and bit flipping faults), and to disconnect these nodes from the rest of the bus. Authors of [4] also use a star topology, but they add a sophisticated router to control traffic on the bus. Two mechanisms dealing with a missing message due to transient faults using the properties of FTT-CAN protocol (Flexible Time-Triggered communication on CAN) are presented in [5]. However, this approach is not compatible with conventional units and their application would be rather expensive.
Other approach uses dedicated bus monitoring equipment ([6], [7]). Basically, it is a special bus node that listens communication and searches for faults; for example so-called CANsniffer from [6]. Method described in [8] is also working with the external device for communication error detection. The device uses information about errors and detects communication error rate between specific pairs of nodes, through which it calculates the resulting health index of each node and the bus.
In contrast, the method described in this article does use neither modified protocol or topology nor dedicated monitoring devices. It focuses on the use of error counters (TEC, REC) of individual nodes to obtain diagnostic information about the state of nodes and the whole bus.

III. Diagnostic method

This chapter describes the diagnostic method in short; the full text can be found in the [1]. As mentioned above, the aim of the method is to detect the degrading communication nodes before the communication throughput of the bus is seriously reduced. The diagnostic method consists of two parts. First, the incidence of errors on the bus and the intervals between them are monitored. Roles of leaders and followers are assigned for each communication error. This assignment helps with decision whether the error was forced by a global disturbance or degradation of one local node. In the second step the method utilizes the information described above and evaluates the degradation of specific nodes.
The method was designed in a centralized form, where one diagnostic node evaluates the degradation of all nodes in the network. This approach may be advantageous in terms of (very slightly) increased computing demands in only one unit, or for example in terms of network topology reasons (diagnostic algorithm is implemented in a gateway that interconnects multiple subnets). Moreover the diagnostic algorithm can be implemented in multiple units providing redundancy.
Due to the TEC and REC counters are local variables of each node, a special message about their values is sent to the diagnostic node in case the error is detected. This Fault message is sent by each node that is a leader for each error event (TEC or REC counter was increased by 8). The message contains a unique identifier of the sending node, the error type (TEC or REC change) and the actual TEC and REC values.

High priority identifiers are assigned to the Fault messages, but they are not expected to have the highest priority. Therefore, it can be assumed that they are transmitted within a short period of time after the occurrence of the error event. The error event is reported by one or more Fault messages that arrive during a predetermined time window. There is always just one of the received Fault messages carrying information about TEC increment and zero or more messages containing information about the REC increment (only the leaders transmit the diagnostic message). This mechanism operates well only under the assumption of low error rate on the bus (this is exactly what we want, for high error rates standard CAN error states handling is used).

A. Diagnostic mechanism

The basic idea of the diagnostic mechanism is a comparison of the time intervals between occurrences of errors with their median. A step of the diagnostic process is started by reception of the Fault messages within the time window after the error event. Using the identifiers of the Fault messages a set of transmitter – receiver pairs can be determined. These pairs provide information about which node detected a fault during the transmission of another node. All transmitter-receiver pairs form an N × N matrix, where N is the number of the nodes on the bus. One row and one column of a matrix are assigned to each node. Columns represent the nodes transmitting during error whilst rows represent the nodes receiving during error. Diagonal of the matrix corresponds to errors observed only by transmitting node itself.
Diagnostic node contains several similar N × N matrices. The first such matrix is called Last Fault time Matrix and particular cells contain the timestamp of the last corresponding fault. Next one is Interval matrix, it is only temporal matrix intended for next step. Its particular cells contain time delay (interval) between the actual and the last recorded error event (1).

$$\begin{aligned} &Interval(x,y) = current\_time - LFM(x,y) \\ &LFM(x,y) = current\_time \end{aligned} \qquad (1)$$

where x identifies the transceiver, y identifies receiver, LFM – Last Fault time Matrix.
In the second step, cumulative moving average of intervals from previous step is calculated. The average is saved to Cumulative Moving Average matrix (CMA).

$$\begin{aligned} &CMA(x,y) = \frac{Interval(x,y) + CMA(x,y)}{index(x,y)+1} \\ &Index(x,y) = Index(x,y)+1 \end{aligned} \qquad (2)$$

Both the CMA and Index are matrices of the N × N type described above.

In the third step the last N × N matrix is used; it is Diagnostic Matrix (DM). It contains information about the degradation of communication. In general case it can be expected that the length of the interval between two random events has an exponential probability density. Then we can write

$$\begin{aligned} &median(x,y) = \ln 2 * CMA(x,y) \\ &if\,(Interval(x,y) < median(x,y)) \\ &\;DM(x,y) = DM(x,y) + comm\_Coef(x) \\ &else\; DM(x,y) = DM(x,y) - comm\_Coef(x) \end{aligned} \qquad (3)$$

The DM cell value is modified by communication coefficient in any case. The communication coefficient of node is inversely proportional to the frequency of message transmission for each particular node. The error occurring during the transmission of the node with lower transmission rate has more significant impact. In case of the length of the interval between two error events has not an exponential probability density, an alternative method should be used; for example the P2 algorithm for dynamic calculation of quantiles without storing observation [9].

B. Evaluation

Following equations are used for the evaluation of the diagnostic results

$$\begin{aligned} &Tx\_Degradation\_index(y) = median(DM(x,y)); \forall x \neq y \\ &Rx\_Degradation\_index(x) = median(DM(x,y)); \forall y \neq x \end{aligned} \qquad (4)$$

where x and y define the investigated node. The median is used to reduce the impact of one degrading node on the results of other nodes. The resulting values of degradation indices are compared against predetermined threshold.
The second part of the diagnostic method uses the values of the TEC and REC counters that are received in Fault messages. The Diagnostic node records maximum values reached by the TEC and REC counters for each monitored node. Warning of possible future communication problem (e.g. moving the node into error passive state) is issued in case when any of the degradation indices exceeds the threshold and exceeding the specified limits for relevant error counter is observed at the same time.

C. Reduction of median estimation error

At the beginning of the diagnostic process an insufficient number of samples causes an error of median estimation. This error manifests itself as a gradual growth of the degradation indices. Forgetting mechanism (5) is used to reduce this influence (applied only for indices that have been changed).

$$DM(x,y) = DM(x,y) * K_{forget}; 0 < K_{forget} < 1 \qquad (5)$$

IV. Simulation and validation

The simulation model for development and testing of the diagnostic method was designed at a system level. This means, the model contains all the essential elements of the CAN bus and a tools for external injection of all error types into the communication. The smallest simulated communication entity is a CAN message. In the simulation some details of bus operation can be simplified or neglected for the purposes of the diagnostic method verification. For the simulations multi-agent simulation environment Repast Symphony tool ([10]) was chosen. Multi-agent approach has led to relatively simple implementation of model and simulation. The results of the simulated diagnostic algorithm were visualized using Matlab.

To validate the diagnostic method the fault injection technique ([11]) was chosen. This technique artificially inserts errors into the monitored system communication. The system designed for the validation of the diagnostic method consists of four standard nodes and a diagnostic node. The diagnostic node functionality could be included in the standard node; the diagnostic functionality was moved to the specific node for easier implementation and evaluation. Evaluation kits with STR710 microprocessor and PCA82C250T CAN bus driver were used as communication nodes. Firmware sending the CAN messages in regular intervals was implemented for each communication node. The nodes have also sent the Fault messages in case of error according to the algorithm described above. The diagnostic node was realized using a personal computer with CAN bus interface. PC application program has implemented the diagnostic method and the MATLAB was used for visualization of a measured data.

A. Fault injection

The fault injection is provided by purpose built circuits. This circuit is placed between a CAN bus driver and the MCU at just one network node.

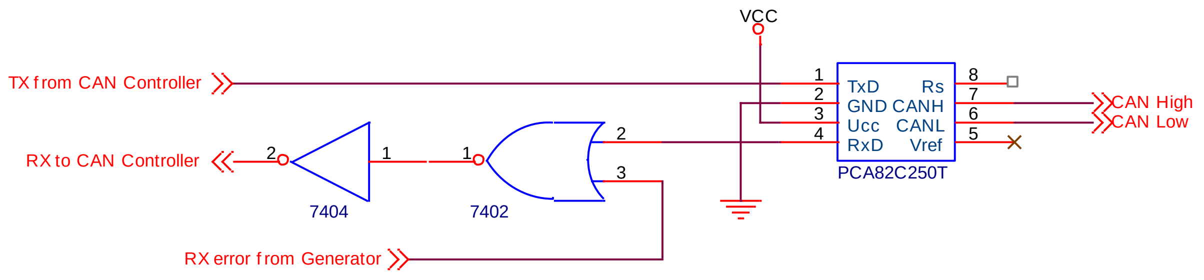

Figure 1. The receive fault injection circuit

In case of the receive fault injection, the logical function of an error insertion circuit can be seen from the diagram above. If the error input is in log.0, RX output to the CAN controller copies the output from the CAN transceiver. If the error input is in log.1, RX output to the CAN controller is in log.1 irrespective of the CAN transceiver RX output. The circuit is therefore able to generate log.1 value instead of log.0 in case of error. Other

error type (log.0 instead of log.1) is not implemented as it would generate false start of frame bits within an idle periods. For the transmission fault injection a circuit was assembled working on a similar principle.
Mechanism described here ensures that the errors are generated only during the message transmission time periods and not during the bus idle periods. The error type used for validation has no influence on the method itself. The external generator is configured to sweep the rate of the inserted errors, and thereby it simulates the gradual degradation of the communication.

B. Experiment

The bus consisted of four network nodes; the communication speed was set to 500 kb/s. Transmission rate of 100 messages per second was chosen for three nodes; the transmission rate of the last one was 33 messages per second. The receive fault injection circuit was implemented at one of higher transmission rate nodes. Error generator output pulse length was set to 2 μs (it represents one bit length at 500 kb/s) with initial repetition rate of 10 Hz. After the initial phase (5 minutes) the repetition rate was increased up to 80 Hz with speed 1 Hz per minute.
The same system was simulated within the above mentioned simulator. To get comparable result the same conditions was set, especially the message error probability. It was expressed using the equation (6).

$$P = \frac{message\ length * K}{communication\ speed} * error\ rate \qquad (6)$$

where *message_length* specifies length of message in bits (110 in our case study), *K* is the ratio between counts of log.0 and log.1 bits within the message (0.63 in our case study; only bits of log.0 level can be inverted to fault value), *communication_speed* specifies the CAN bitrate (500 kbit/s in our case study) and *error_rate* specifies the frequency of error pulses (10 to 80 Hz in our case study).
For the simplification of the experiment, all messages had a predefined uniform content. Experimental as well as simulation results are shown in the following chart and table. Simulation was run four times to show the dispersion of particular values.

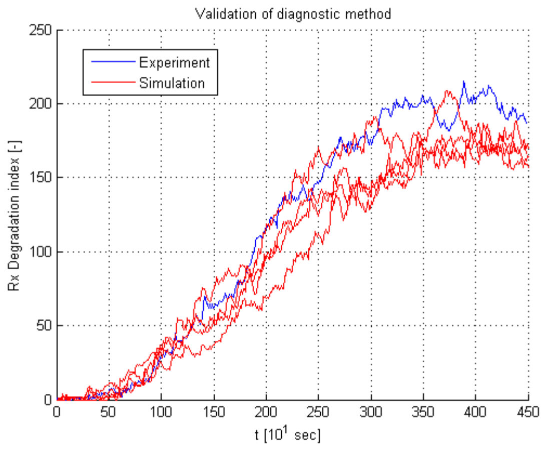

Figure 2. Simulation and validation results

The results of the experiment confirm the simulated values in case of the receive fault injection circuit. The difference in the measured waveforms can be explained by the random nature of the experiment.
In case of the fault injection into the transmission path the results do not match as well (Fig. 3). The reason is that the simulation doesn’t cover such a case. After inserting errors into the data or CRC segment, the receiving nodes do not confirm a message with acknowledge. In addition, the receiving nodes do not report CRC errors via an error flag immediately, but the acknowledge error is generated first by the transmitter. This scenario has not been added to the simulation, because it is unclear whether this is a specific feature of our CAN controllers (STR710 microcontrollers with on chip CAN controller were used), or a general rule. Fig. 3 also contains the intrinsic transmit degradation index (the diagnostic matrix’s diagonal cell) as a dashed green line. From the waveforms of the two indices, it is clear that the method is able to detect gradual degradation of the transmitter despite the unexpected behavior of nodes.

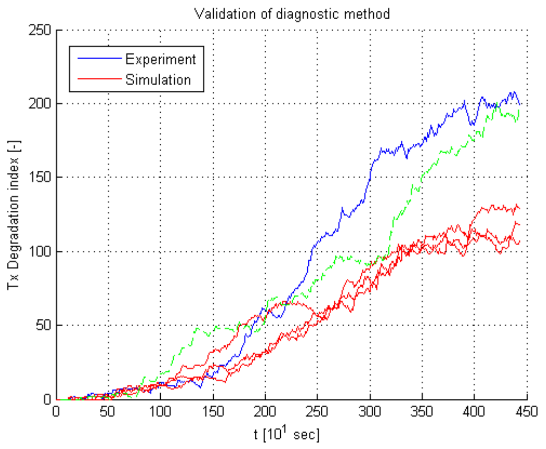

Figure 3. Simulation and validation results

V. Conclusions

The paper describes the diagnostic method detecting gradually degrading CAN nodes far before they get into error passive or even bus off state. The method uses the error counters information of each node and this information is processed in diagnostic node. Diagnostic method runs online and requires relatively small resources in terms of data storage and processing power. Thanks to these features, the method can be easily integrated into existing systems without additional hardware costs. Simulation and experiment were used to verify the diagnostic method under the same conditions. Errors were inserted into the simulated communication as well into real node in validation experiment. Both results match quite well and it is obvious that the method is capable of detecting the degradation of communication capability of a CAN node in real-time.

Acknowledgment

This work was supported by the Grant Agency of the Czech Technical University in Prague, project Systems of Data Processing and Fusion for Diagnostics and Measurement Applications (# SGS14/153/OHK3/2T/13) and by the support of Technological Agency, Czech Republic, programme Centres of Competence, project # TE01020020 Josef Božek Competence Centre for Automotive Industry.
This support is gratefully acknowledged.

References

[1] Pospíšil T.; Novák J., “New Method of CAN Nodes Health Monitoring”, *19th International Conference Applied Electronics*, in review process, 2014.
[2] Bosch GmbH, *Controller Area Network (CAN) specification - version 2.0.*, 1991
[3] Barranco M.; Proenza J.; Rodriguez-Navas G.; Almeida L., "An active star topology for improving fault confinement in CAN networks," *Industrial Informatics, IEEE Transactions*, vol.2, no.2, pp.78-85, 2006.
[4] Obermaisser R.; Kammerer R., "A router for improved fault isolation, scalability and diagnosis in CAN," *Industrial Informatics (INDIN), 2010 8th IEEE International Conference on*, pp.123-129, 13-16 July 2010.
[5] Marques L.; Vasconcelos V.; Pedreiras P.; Almeida L., "Tolerating transient communication faults with online traffic scheduling," Industrial Technology (ICIT), *2012 IEEE International Conference*, vol. 396, no. 402, pp. 19-21, 2012.
[6] Reorda M.S.; Violante M., "On-line analysis and perturbation of CAN networks," Defect and Fault Tolerance in VLSI Systems, 2004. *DFT 2004. Proceedings of 19th IEEE International Symposium*, vol. 424, no. 432, pp. 10-13, 2004.
[7] Braescu F.C.; Ferariu L.; Franciuc A., "Monitoring CAN performances in distributed embedded systems," System Theory, Control, and Computing (ICSTCC), *2011 15th International Conference on*, 2011.
[8] Butts N.L., “Controller Area Network condition monitoring and bus health on in-vehicle communications networks ”, *US patent 8 213 321*, 2007.
[9] Jain R.; Chlamtac I., “The P2 algorithm for dynamic calculation of quantiles and histograms without storing observations”, *Commun. ACM 28*, pp. 1076-1085, October 1985.
[10] Repast – The Repast Suite: http://repast.sourceforge.net
[11] Mei-Chen Hsueh; Tsai, T.K.; Iyer, R.K., "Fault injection techniques and tools," *Computer*, vol.30, no.4, pp.75-82, 1997.