Fault detection and diagnosis for in-vehicle networks

Jittiwut Suwatthikul, PhD.
National Electronics and Computer Technology Center (NECTEC)
Thailand

1. Introduction

Automotive vehicles are nowadays equipped with a significant number of networked electronic systems by which advanced vehicle control, elimination of bulky wiring, and sophisticated features can be achieved. Most of the features are enabled by the use of distributed electronic systems including sensors, switches, actuators and electronic control units (ECUs). In today's premium automobiles, there can be fifty or more individual ECUs communicating over multiplexed data networks such as Controller Area Network (CAN), Local Interconnect Network (LIN), FlexRay for X-by-wire applications (Kopetz & Bauer, 2003; Leen et al., 1999; Leen & Heffernan, 2002; Shrinath & Emadi, 2004).
However, as more features and ECUs are introduced, overall system complexity increases, in turn raising the likelihood for unpredictable or emergent behaviour that could not have been anticipated during ever shrinking vehicle development cycles. These cycles reduced from 48 months in 1985 to 24 months in 2005, and are expected to be 12 months in 2010 (Ortega et al., 2006). The consequences of the unpredictable behaviour or implementation errors would discourage brand loyalty and bring a manufacturer into disrepute. In addition, rising feature levels have resulted in the embedded software and electronic components becoming an increasing proportion of the total value of the vehicle. The average cost for in-vehicle electronic content increased from 2% of the total car price in 1974 to 23% in 2004, and is forecasted to reach 40% by 2010 (Ortega et al., 2006).
Against this background, vehicle manufacturers are striving to reduce costs and at the same time to improve levels of customer satisfaction. Work to improve test and validation of large distributed electronic systems has been ongoing for years (Athanasas & Dear, 2004; Ehret, 2003; Simonot-Lion, 2003). This has provided manufacturers with approaches to test and validation, with some degree of coverage. It is, however, still impossible to use an ideal test scheme that provides complete input test coverage to perform exhaustive testing and validation because of the large number of possible system states (Storey, 1997), possibly resulting in vehicles not working properly due to some design flaws and errors. This is compounded by general wear out of mechanical, electrical and electronic components. To date, on-board diagnostic systems (OBD) have come into play to cope with faults when vehicles are used by customers. OBD are integrated in ECUs to detect and diagnose vehicle faults such that diagnostic trouble codes (DTCs) relevant to the faults are set and logged in

the ECUs' memory for later off-board, return-to-dealership-based fault analysis and rectification.
Despite the available OBD, diagnostic techniques have largely been focused on individual or defined areas of a vehicle, e.g. engine management, brakes and steering. In a vehicle where only a few ECUs and communication messages are deployed, traditional off-board diagnostics can be adequate. As system complexity continues to increase, off-board diagnostic approaches have become more costly and sometimes ineffective, resulting in high levels of "no-fault-found", incorrect component replacement and increased warranty costs. Recent years have seen research work on a paradigm shift from off-board dealership-based diagnosis and repair to on-board remotely-assisted diagnosis and in-vehicle repair (Amor-Segan et al., 2007). It is anticipated that the new on-board vehicle diagnosis scheme will improve customer expectations and satisfaction for vehicle reliability.
This chapter concerns fault detection and diagnosis (FDD) techniques applied to automotive electronic systems, especially focusing on faults in in-vehicle networks. FDD on a CAN network is demonstrated. Readers will be provided with knowledge on how vehicle faults are generally managed, and the trend of intelligent FDD in future vehicles. The rest of this chapter is divided into three main sections: (i) a brief introduction to in-vehicle electronic systems; (ii) FDD for in-vehicle electronic systems including those in component, feature and network levels; (iii) recent research on FDD techniques for in-vehicle networks.

2. In-vehicle electronic systems

Over the past two decades, the rapid growth in performance and reliability of electronic embedded systems has enabled vehicle manufacturers to implement complicated automotive control systems through the use of sophisticatedly integrated mechanical and electronic devices, so called mechatronics (Isermann, 2008). Vehicles are equipped with a variety of electronic devices performing different functions, and mostly transferring signals via electrical wiring.

2.1 In-vehicle electronics

State-of-the-art control algorithms and rapidly-improved semiconductor technology make it possible for modern vehicles to meet the demands for driver-assisted functions, safety, comfort and environment protection. A number of sensors are used to measure controlled variables as input signals for ECUs, e.g. engine speed, temperature. The input signals can be analogue, e.g. voltage signals from sensors, digital such as switch positions, or modulated e.g. Pulse Width Modulation (PWM) signals. With these input signals, ECUs calculate required parameters to adjust controlling devices such as actuators. Improved performance and additional functions are obtained by synchronising processes controlled by individual control units and by adapting their respective parameters to each other in real-time. An example of this type of function is a traction control system (TCS) which reduces the driving torque when the drive wheels spin (Robert Bosch GmbH, 2004a).
In 1902, a magneto-ignition system, the first on-board electrical system, was installed in a vehicle. It consisted of the magneto itself, an ignition distributor, ignition coils, spark plugs and cables. More than a decade later, Bosch had the first complete automotive electrical system ready for installation. The system comprised the magneto-ignition system with spark plugs, a starter, a generator, headlamps, a battery, and a regulator switch. This was

the starting point of a progress towards a genuine on-board electrical and electronics system (Robert Bosch GmbH, 2004).
Today's in-vehicle electronic systems have become extensive and much more complicated. A large number of electronic devices and software contents are integrated in modern vehicles. Applications range from simple door or window control to remotely wireless data transfer between vehicles or between a vehicle and the infrastructure. Fig. 1 illustrates the exponentially soaring complexity of today's in-vehicle electronics.

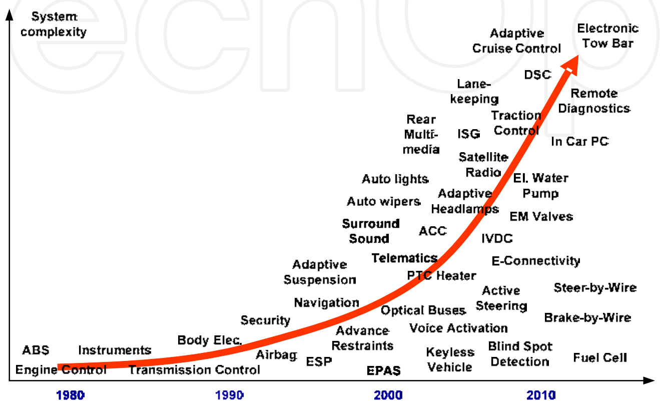


Fig. 1. Rapid increase in in-vehicle electronic systems (adapted from McMurran et al., 2006).

In-vehicle electronics can be broadly classified into four main categories corresponding to different functionalities, constraints and models: "Powertrain", "Chassis", "Body", and "Infotainment" including telematics, multimedia, various types of information and entertainment (Zurawski, 2006).
"Powertrain" refers to components that generate power and deliver it to the road surface. This includes the engine, transmission, clutches, driveshaft, etc. "Powertrain" sometimes simply refers to the engine and transmission, and the other components in the transmission. "Chassis" involves the systems that control the interaction of the vehicle with the road and the chassis. It is related to steering, braking and ride quality. This category includes systems such as ABS (Anti-lock Braking System), ESP (Electronic Stability Program) and the new forthcoming technology so-called X-by-Wire. "Body" refers to the electronic systems not directly involved in the movement of the vehicle. These are, for instance, the systems that control doors, windows, seats, boots, etc. "Infotainment" provides drivers with information and entertainment through HMI (Human Machine Interface). This enables drivers to see or exchange not only in-vehicle information and entertainment such as vehicle conditions and movies from an in-car video player, but also information from remote sources via telematics (Zurawski, 2006).

2.2 In-vehicle networks

Historically, the communication between simple electronic devices was mostly achieved by using point-to-point links, as shown in Fig. 2 a). Signals in the vehicle were transmitted and received among ECUs over non-multiplexed and hard-wired cables. This resulted in bulky, expensive and complicated wiring when dealing with the increasing use of ECUs, because the number of required communication channels grows exponentially with the number of ECUs (Navet et al., 2005). The attempt to eliminate wiring difficulties and to improve

automotive distributed control systems became a challenge for automotive manufacturers. A wiring harness of a middle-class vehicle was roughly 1 mile long and included approximately 300 connectors with 2,000 pins (Robert Bosch GmbH, 2004a). The early 1980s saw the emerging solutions for vehicle networking, many of which were based on simple data transfer e.g. point-to-point or master-slave UART (Universal Asynchronous Receiver/Transmitter) (Leen et al., 1999).
As requirements in vehicle control grew quickly, and were mostly based on real-time control strategies, such simple network configurations and protocols became unwieldy. In the mid 1980s, Robert Bosch GmbH invented a robust automotive control network known as CAN. CAN is based on a bus configuration, as shown in Fig. 2 b), which allows ECUs connected on the bus to receive in-vehicle signals digitally encoded in CAN messages at almost the same time. This significantly enhances real-time applications in the vehicle.

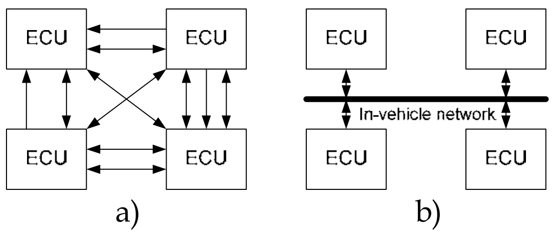


Fig. 2. Evolution of in-vehicle networks.

Whilst CAN is concerned in this chapter, more recently, there have also been several other communication protocols and network topologies used in different categories, depending on requirements and cost constraints of applications. These are, for example, Local Interconnect Network (LIN) for Body, Time-Triggered CAN (TTCAN) for Powertrain, FlexRay for X-by-wire applications, MOST (Media Oriented System Transport) for infotainment, and the short-range wireless communication Bluetooth. Fig. 3 shows a cost and speed comparison of different networks.

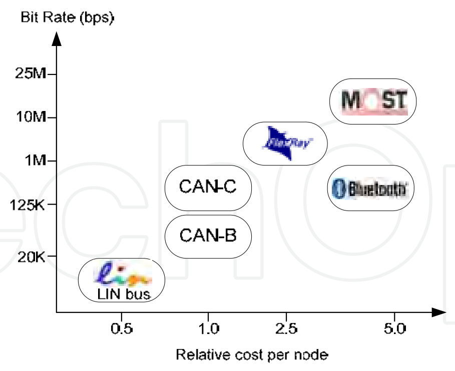


Fig. 3. Speed and cost comparison of the well-known in-vehicle networks.

CAN is currently one of the serial communication protocols widely used in automotive and industrial automation applications. Fig. 4 illustrates a CAN bus topology in a typical automotive application where a number of ECUs can be connected on a high-speed CAN (CAN-HS) and a low-speed CAN (CAN-LS) buses which are connected via a gateway.

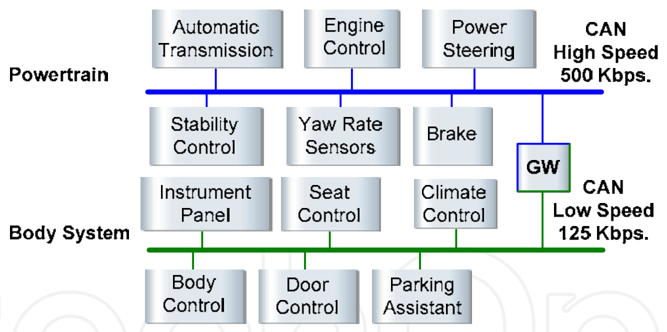


Fig. 4. CAN in automotive application.

A CAN communication controller is used in each ECU to control message transmission between ECUs connected on the same bus. CAN uses an arbitration feature, illustrated in Fig. 5, to control bus access in order to avoid transmission collision which causes communication errors. Messages which are assigned lower message identifications (IDs) have higher priorities to access the bus. For instance, in Fig. 5, node B loses the arbitration to node A because the third bit of node B (logic 1, so called "recessive") is replaced by the third bit of node A (logic 0, so called "dominant").

If errors occur during communication, an error management feature in the communication controller detects, handles and confines such errors. An error frame is transmitted to notify ECUs that errors have been detected so that the ECUs ignore the message recently present on the bus. The errors involve Bit Error, Bit Stuffing Error, CRC Error, Form Error and Acknowledgement Error. Effect of the errors is also limited by CAN controllers to prevent further communication failure. More details on CAN and these errors can be found in Lawrenz, 1997.

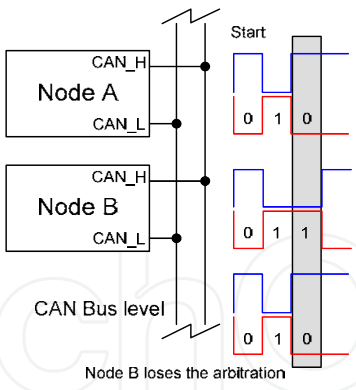
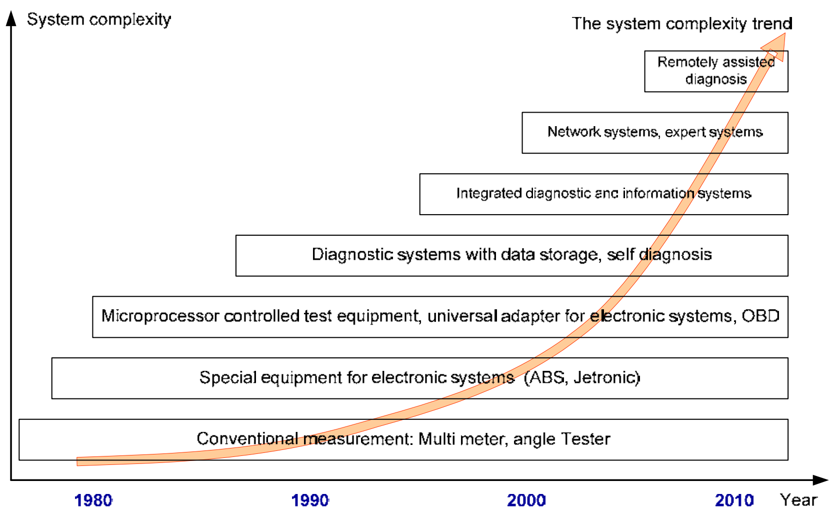


Fig. 5. CAN arbitration.

In case of communication loss, network-relevant DTCs are set and logged in the ECUs that expect the lost messages. Applications of CAN embrace not only vehicle control but also diagnostic services. Despite the common diagnosis standards such as ISO 9141 K-line, ISO 14230 KWP2000 (Keyword Protocol 2000), ISO 15765, etc., the trend of diagnostic services is towards diagnostics over CAN such as the standard defined in ISO 14229.

3. Fault detection and diagnosis for in-vehicle electronics

As mentioned earlier, the rapid growth in hardware and software content in today's vehicles results in increased overall system complexity. Fig. 6 illustrates the evolution of vehicle diagnostics together with the system complexity trend. Vehicle diagnostics with only conventional instruments, which has been used for simple measurement of normal or faulty electrical signals, is no longer effective. Enhanced in-vehicle diagnostic methods have been introduced in order that root causes of faults occurring during vehicle operation are efficiently detected and identified. For instance, OBD, which was initially intended for emission monitoring, is now able to provide logged DTCs and signal measurement for off-board diagnostics of other non-emission related functions.


System complexity
The system complexity trend
Remotely assisted diagnosis
Network systems, expert systems
Integrated diagnostic and information systems
Diagnostic systems with data storage, self diagnosis
Microprocessor controlled test equipment, universal adapter for electronic systems, OBD
Special equipment for electronic systems (ABS, Jetronic)
Conventional measurement: Multi meter, angle Tester
1980
1990
2000
2010
Year
Fig. 6. Evolution of diagnostic test equipment (adapted from Robert Bosch GmbH, 2004b).

In-vehicle electronic and software control systems can operate on a number of different levels: (i) component level—individual ECUs, sensors, actuators and components; (ii) functional or feature level—such as braking, cruise control, stability control; (iii) network level—such as system-wide coordination and configuration, network wake-up and sleep, inter-network gateways and the network itself. As illustrated in Fig. 7, a diagnostic application in each level monitors different parameters. For instance, the diagnostic application in the component level monitors input and output signals and the battery voltage. ECU's internal operation and communication are monitored by the ones in the feature and the network levels respectively. Once faults are detected, tested and confirmed, some default or alternative signal values are used by the ECU in a "limp-home" mode where the vehicle can be driven to the nearest service centre. The diagnostic processor then provides a warning to the driver as well as logging DTCs and additional environment parameters, e.g. speed, temperature and timestamp. It should be noted that there are currently no actual processors in deployment. The diagnostic processor illustrated in Fig. 7 is a sub-function in each ECU.

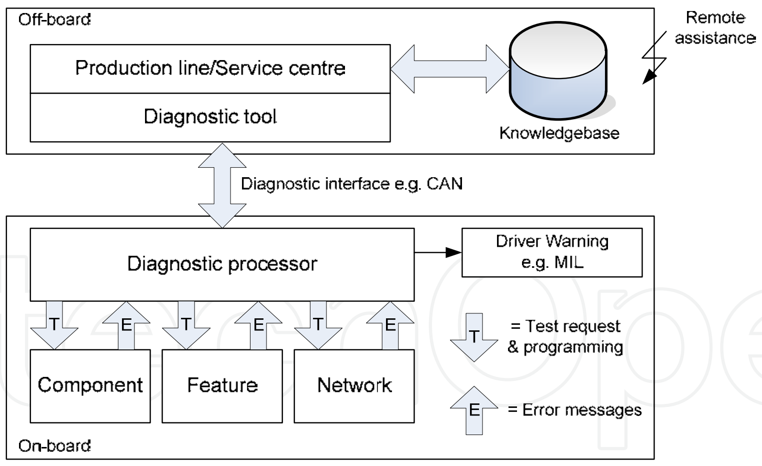


Fig. 7. Diagnostic scheme for in-vehicle electronic systems.

A flow chart of vehicle diagnostics is shown in Fig. 8. The logged DTCs are accessed off-board by a diagnostic tool at a service centre, and can be appropriately deleted once the faults are found and rectified. Fig. 9 illustrates an example of an off-board diagnostic tool used to read DTCs. If any further information about the vehicle repair is needed, the tool can be connected to another service centre or the manufacturer's central diagnostic knowledge base in order to request remote assistance.

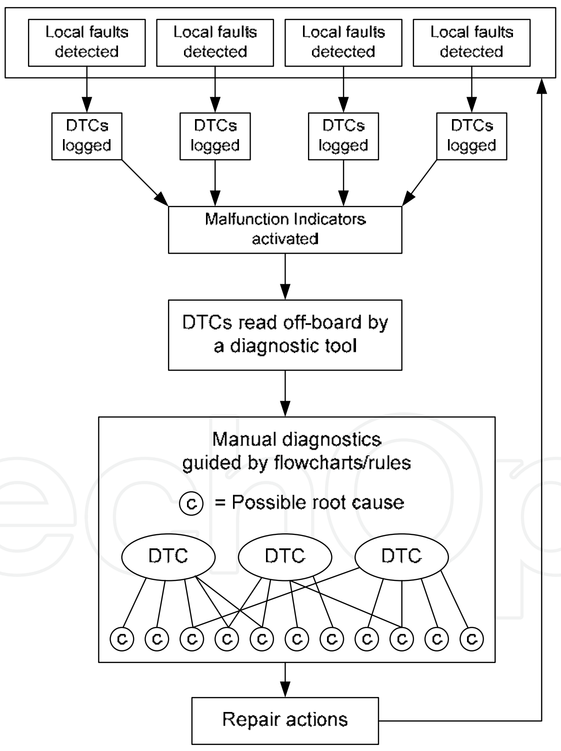


Fig. 8. Flow chart of conventional diagnostics.

Fig. 9. Diagnostic tool for reading DTCs.

Since this chapter is intended to focus on FDD for in-vehicle networks, this section next discusses FDD for the component and feature levels briefly, and then puts emphasis on the network level.

3.1 Component and feature levels

In the component and feature levels, FDD applications employ different methods for detecting functional software and physical hardware faults. Detection approaches range from simple threshold and out-of-range comparisons to learning algorithms for non-linear signal processing.

Input and output signals from sensors, connecting lines (signal paths), and actuators are checked. Short-circuits to steady-state voltage e.g. battery voltage and the device ground, and line interruptions are monitored. Measurements from sensors are examined whether they are in permissible ranges. If additional information is available, a plausibility check is performed by cross-checking two signals e.g. comparison of crankshaft and camshaft speeds. Internal ECU's operations are checked to ensure that the ECU works as specified. These checks are performed immediately after the vehicle being switched on, when the ECU operating or controlling the vehicle, and after the ECU finishing its operation. After faults are detected, some alternative values are used so that the ECU can continue its operation. Additionally, limp-home measures, e.g. limitation of engine speed, may be initiated to maintain driving safety, prevent consequence damage, and minimise emissions (Robert Bosch GmbH, 2004c).

Recent research work has proposed new methods for diagnosing root causes of faults in distributed electronic systems in vehicles. For example, firstly, Murphey et al., 2004 proposed the distributed diagnostic agent system (DDAS), a novel diagnostic architecture developed for automotive fault diagnosis. The DDAS, implemented for fault detection of ECU signals, consists of a vehicle diagnostic agent (VDA) and a number of signal diagnostic agents (SDAs), each of which is responsible for diagnosing one particular automotive signal using either a single or multiple signals. A novel fuzzy learning algorithm was implemented to learn to classify good and bad signal segments by supervised learning from good examples only in the training data set. This work used a case-based reasoning (CBR) approach in the VDA agent to find the cause of vehicle faults, by which effective diagnostic results have been obtained. Secondly, an intelligent diagnostic system (IDS) was developed for diagnosis of component faults in a multi-ECU environment by using model-based reasoning (MBR), and information taken from failure modes and effects analysis worksheet

(FMEA) (Foran & Jackman, 2005). This work tried to isolate a core faulty component instead of simply returning a series of faults. The system was tested by a particular list of test cases and was capable of identifying all scenarios in the list with 100% success rate.

3.2 Network level

A number of research experiments on network design, verification and test to prevent network level faults and to improve reliability have been reported (Tindell & Hansson, 1995; Temple, 1998; Navet & Song, 1999; Navet & Song, 2001; Gaujal & Navet, 2005; Buja et al., 2007). Hardware-in-the-loop facilities are playing an important role in the testing of networked electronic systems (Isermann et al., 1999; Kendall & Jones, 1999; Short & Pont, 2008). Methods more specific to network level testing have been discussed (Armengaud et al., 2004; Armengaud et al., 2005a; Armengaud et al., 2005b). This has provided device suppliers and manufacturers with testing approaches with some degree of test coverage and system validation. Despite this, it is inevitable that unexpected faults can still occur due to untested conditions or worn components.

3.2.1 Existing Network Management

It is well understood that generally automotive manufacturers do not develop ECUs themselves; rather they mostly outsource this task to different suppliers to do so in accordance with the manufacturers' specifications and existing standards. ECUs from different suppliers are then integrated and networked into vehicles by the manufacturers. One such well-known standard related to in-vehicle networks is Open System and Corresponding Interfaces for Automative Electronics Network Management (OSEK NM).
Due to the increasing number of ECUs deployed in vehicles, the Open Systems and the Corresponding Interfaces for Automotive Electronics (OSEX/VDX) working group has defined a standard for communication of ECUs in automotive applications, called Network Management or NM (OSEK/VDX, 2004), illustrated in Fig. 10. NM resides in the environment where an Operating System (OS) controls system interactions among different software layers. NM consists of algorithms defined by OSEK and protocol specific algorithms. NM can communicate with the application software in the application layer, and vice versa, by using an Application Program Interface (API) in order to enable (or disable) OSEK algorithms.

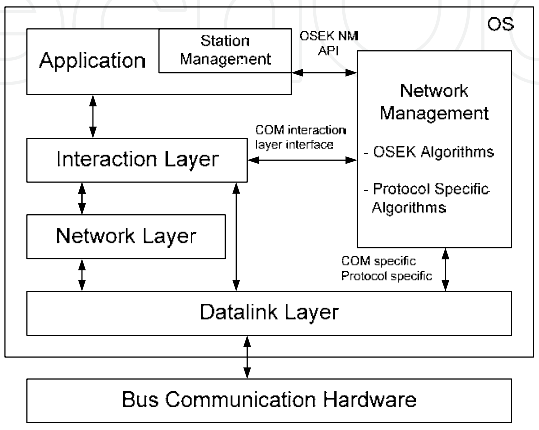


Fig. 10. OSEK Network Management.

The protocol specific algorithms are responsible for handling data transfer on the network by communicating with the datalink layer. OSEK NM is now a published ISO standard: ISO 17356-5: 2006. It provides standardised features which ensure the functionality of inter-networking by using standardised interfaces. NM implementations are incorporated in all networked nodes, e.g. ECUs. This means that a solution for NM can be implemented throughout the varieties of available ECU hardware. The status of the network is recorded and evaluated in all ECUs and thus each node features a determined behaviour regarding the network and the application concerned. NM supports diagnostic applications by providing acquired network status.

NM features two mechanisms for network status and fault monitoring—direct network management (DNM) by using dedicated messages with a token principle and indirect network management (IDNM) by monitoring application messages. The use of these mechanisms is dependent on a particular system. Processing of information collected by these mechanisms must be in accordance with requirements of the entire networked system.

Dedicated NM messages are used in DNM: Ring, Alive and Limp Home messages. Each node is monitored by other nodes on the network. Nodes transmit and receive NM messages via a logical ring in which a communication sequence for synchronisation is defined. Each node on the network has a unique ID normally available from 1 to 255. To set up a logical ring, a Ring message is sequentially passed from the lowest to the highest ID node which then passes the Ring back to the lowest ID node, as illustrated in Fig. 11. DNM also requires a broadcast type of network implementation so that every node can hear the messages that are being sent (Lemieux, 2001).

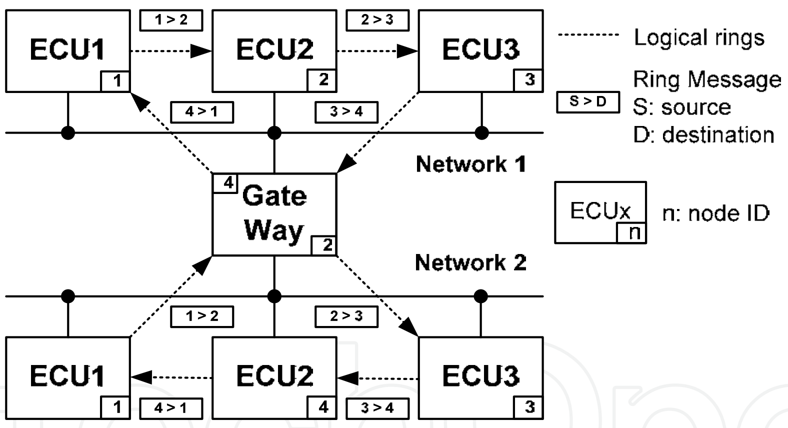


Fig. 11. DNM logical ring.

An Alive message identifies a new node present on the network and puts the system into a transient state. When the new node has identified all nodes on the network, the system changes into a stable state where the NM component is fully aware of the status of all nodes on the network. A faulty node transmits a Limp Home message cyclically until: (i) it is able to receive NM messages from other nodes correctly; or (ii) NM component stops; or (iii) the bus goes to sleep mode. Then the node enters a reset state and performs an NM initialisation. Other nodes that have received a Limp Home message update their configurations to identify the malfunction node being absent from the bus.

Although DNM provides useful information on network and node status, it requires a somewhat amount of computational resources. Thus, DNM may not be suited for some systems in which simple software algorithms and computational resource consumption are of concern. IDNM is therefore defined as an alternative mechanism for network management.

For IDNM, instead of dedicated NM messages circulating in a logical ring, application messages are monitored by nodes that receive the messages to determine the status of the transmitting node. In addition, two message monitoring methods are defined: a single timeout for observation (TOB) for all messages and an individual timeout per message. Single TOB is used by all nodes for monitoring monitored application messages to identify the node states. For instance, if a node is able to successfully transmit its monitored application message within TOB, the node is considered as "Not Mute". If monitored messages are successfully received before TOB elapses, the monitored node (the node that sent the messages) is "Present", shown in Fig. 12.

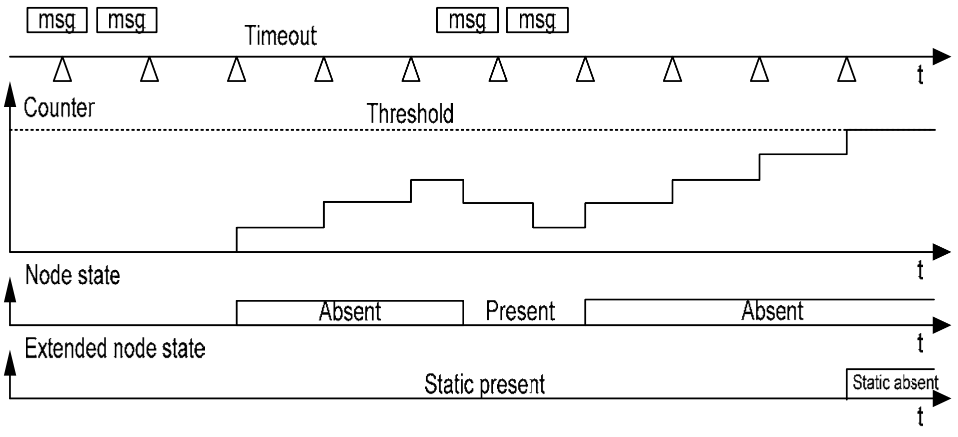


Fig. 12. Application message monitoring for IDNM.

As for an individual timeout per message, IDNM uses COM Deadline Monitoring, a feature of OSEK Communication Component, to monitor individual message timeouts. If a monitored message has not arrived within predefined time, COM component will signal NM to update its configuration to indicate that the monitored node is "Absent". Similarly, the COM component also signals NM if the local node has failed to transmit a message within a predefined time. As a result, that local node is classified as "Mute".

To identify the static states (extended states), a specific counter and a threshold level are used. A counter is incremented and decremented in accordance with a node state. For instance, if the node is "Absent", the counter will be incremented; if the node resumes being "Present" again, the counter will be decremented. When the counter reaches the threshold, NM identifies this state as "Node Absent Statically". NM modes, state transitions and timer definitions are also defined in the OSEK NM standard. Further details of OSEK NM can be found in OSEK/VDX, 2004 and Lemieux, 2001.

3.2.2 Message rate fault detection

Examples of CAN application message faults which are most likely to occur on in-vehicle networks are shown in Fig. 13 and are defined in detail in Table 1. In general, details of periodic messages are specified in a message database during a design process. For instance, a message 'M' is transmitted by an ECU 'E' at every 10 ms. This transmission period can be guaranteed by the use of special network software tools. However, message faults can still happen when the bus traffic quality is poor due to a large number of error frames, or in a system where no such special software tools are implemented in ECUs. It is critical when faults occur on messages which are used in real-time control loops. Therefore, network level fault detection becomes necessary to monitor the number of messages and their timings, and the number of error frames actually present on the network.

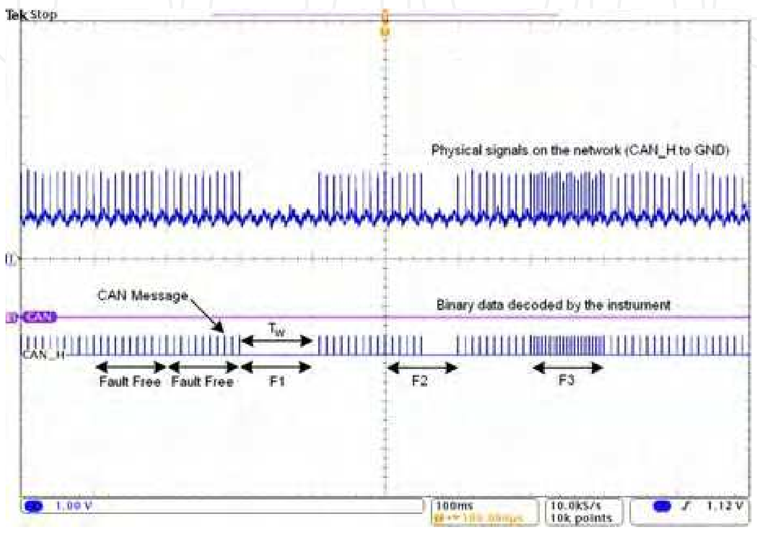


Fig. 13. CAN message faults.

An application message on the network is periodically monitored by the fault detection in a time window T_w (e.g. 100 ms as shown in Fig. 13). The intermediate message rate N_{mi} (in msg/s) is compared with the rate specified in a message database N_{si} and a threshold value N_{ti}, where i is a message identification number. T_w is chosen to satisfy the inequality:

$$T_w \geq 1/\text{MIN}[N_{s0}, N_{s1}, \dots N_{sn}]$$

where n = 0 to the last message ID, e.g. 255.

Faults	Details
Fault free	Message is sent at the specified rate ($N_{si} - N_{ti} \leq N_{mi} \leq N_{si} + N_{ti}$)
F1: Timeout	Message is not sent within predefined time ($Nm = 0$)
F2: Missing-Message	Message is missing in a particular period ($Nm < Ns - Nt$)
F3: Too-Many-Message	Too many messages are transmitted ($Nm > Ns + Nt$)

Table 1. Details of application message faults.

The measurement of the message rates can be seen in the statistics window of CANoe, a CAN bus simulation and network design tool. The horizontal axis of the window displays hexadecimal message IDs and the vertical axis represents the message rates in msg/s. Fig. 14 shows the normal condition of the simulated network, where application messages with different message IDs are transmitted within the normal operating bands defined by:

$$N_{si} - N_{ti} \leq N_{mi} \leq N_{si} + N_{ti}$$

The fault detection is responsible for detecting any message rates that are out of the bands such as F1, F2 and F3 shown in Fig. 15. After faults are detected, DTCs and relevant snapshot data are logged in accordance with the manufacturer's definitions.

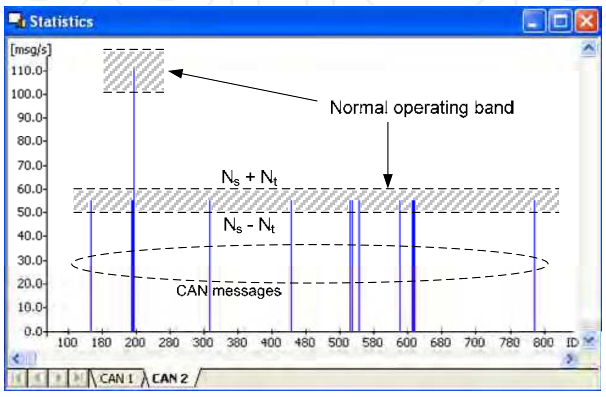


Fig. 14. Normal condition.

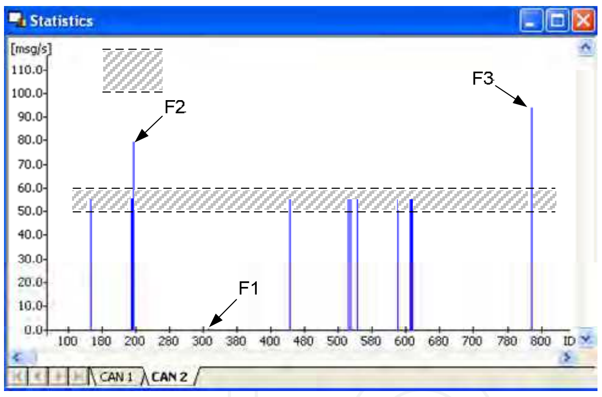


Fig. 15. Faulty condition.

3.2.3 Fault isolation with multiple discrete events

It is possible that any of the faulty electronic components on the communication link contributes to F1 and F2 presented above. This subsection demonstrates a method for isolating possible faulty components by inferring from multiple discrete events in the networked system.

In engineering systems where some basic knowledge of relationships between faults and symptoms is available, the knowledge can be expressed in rules based on fault-tree analysis (FTA) or event-tree analysis (ETA). A simple example a rule of '*FAULT1*' caused by two symptoms—'*SYMPTOM1*' and '*SYMPTOM2*' is as follows:

IF (*SYMPTOM1* AND *SYMPTOM2*) THEN *ERROR1*

IF (*ERROR1* OR *ERROR2*) THEN *FAULT1*

FTA is a graphical method that hierarchically represents a binary relationship between a fault ('top event') and symptom(s). Intermediate events relevant to the 'top event' are combined using logical operations. FTA is commonly used in hazard analysis in safety-critical systems as it can represent dependencies in the systems (Storey, 1997). Fig. 16 illustrates a simple fault tree of a headlamp failure in a vehicle, which involves several root causes that can be traced back. FTA is also widely used in fault diagnosis with inference methods. Fault diagnosis can be preformed by inferring Boolean values of symptoms from sensor measurements such as pressure and position switches, and fault-symptom relationships.

For in-vehicle network diagnosis, this subsection addresses how to utilise local message monitoring data in ECUs for isolating faulty components. Fault isolation with multiple discrete events by interpreting symptoms or intermediate events of the presence and correctness of message communication is discussed.

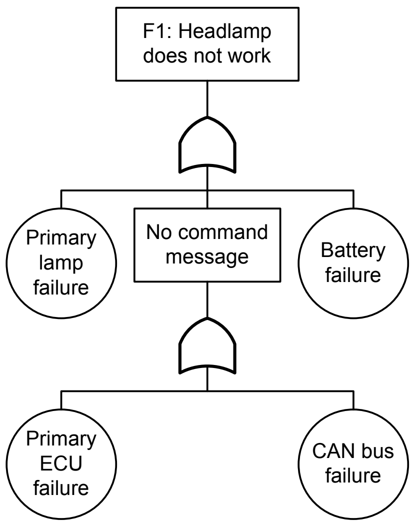


Fig. 16. Simple fault tree of a headlamp failure in a vehicle.

Conventionally, each ECU monitors particular messages it expects to receive. Faulty event data such as "timeout" events of the expected messages is locally stored in the ECU. When faults occur in a particular component which is not in a communication path between a monitoring ECU and a monitored message, the monitoring ECU will not recognise the faults and will not store any fault data. For instance, in the ECU diagram shown in Fig. 17, suppose that E5 expects M1 from E1 and the other ECUs do not monitor any messages from E5. If a "timeout" fault of M1 is detected and stored in E5, the root cause can be several components in the communication path e.g. E1, W1, C1, W6, W7, C5, W5 and E5. This list of possible faulty component(s) can be narrowed down if every ECU (i) locally monitors particular messages from the other ECUs even if the messages are not used in its control functions, and (ii) can provide local fault events to be analysed by diagnostic applications in a dedicated ECU, provided that there is a redundant communication channel available if the main channel is permanently unavailable.

A communication path is shown in Fig. 18 where a monitoring ECU receives a monitored message from a monitored ECU through several peripheral components. The peripheral components can be classified into three major groups: (i) components at the monitoring ECU, (ii) those at the monitored ECU and (iii) those shared among ECUs.
Consider the networked system in Fig. 17 where M5 is monitored by all ECUs, fault symptoms in a situation where M5 is not received by E1 - E4 and M1 - M4 are not received by E5 can be summarised in Table 2. Possible faulty components can be identified in relation to an individual symptom and the peripherals involved in the communication path.

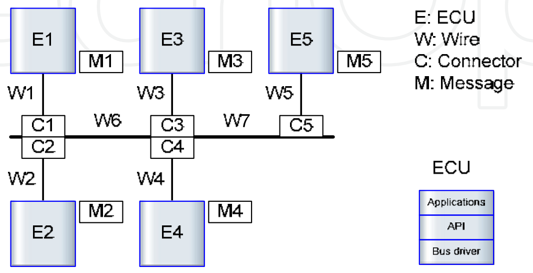
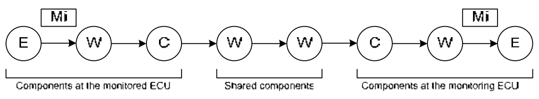


Fig. 17. In-vehicle network with ECUs and peripheral components.


Mi
Mi
E
W
C
W
W
C
W
E
Components at the monitored ECU
Shared components
Components at the monitoring ECU
Fig. 18. Communication path.

		M1	M2	M3	M4	M5
S1	E1	1	1	1	1	0
S2	E2	1	1	1	1	0
S3	E3	1	1	1	1	0
S4	E4	1	1	1	1	0
S5	E5	0	0	0	0	1

Table 2. Fault symptom table: M5 not received by all ECUs except E5 itself.

where 1: M*i* is received by E*i*; 0: M*i* is not received by E*i*; S*i*: Symptom i.
As shown in Table 3, for instance, S1 can be caused by E1, W1, E1, E5, W5, C5, W6 and W7. After considering all of the discrete fault symptoms on the entire network (S1 – S5), W7 is involved in every symptom; therefore it is isolated as a faulty component.
Another example of fault isolation for multiple faults is provided in Table 4. In this example, M5 is not received by all ECUs except E5 itself; M1 - M2 are not received by E3 - E5; M3 - M5 are not received by E1 - E2; M3 - M4 are not received by E5. Similarly, after considering multiple discrete fault symptoms on the entire network, W6 and W7 can be isolated as faulty components.

	E1	W1	C1	E2	W2	C2	E3	W3	C3	E4	W4	C4	E5	W5	C5	W6	W7
S1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1
S2	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	1	1
S3	0	0	0	0	0	0	1	1	1	0	0	0	1	1	1	0	1
S4	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	1
S5	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1
P	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	x

Table 3. Fault isolation table for the symptoms in Table 2.

where 1: component is in the communication path and can cause S*i*;
0: component is not in the communication path;
P: possible faulty components.

		M1	M2	M3	M4	M5
S1	E1	1	1	0	0	0
S2	E2	1	1	0	0	0
S3	E3	0	0	1	1	0
S4	E4	0	0	1	1	0
S5	E5	0	0	0	0	1

Table 4. Fault symptom table: M5 not received by all ECUs except E5 itself; M1 - M2 not received by E3 - E5; M3 - M5 not received by E1 - E2; M3 - M4 not received by E5.

	E1	W1	C1	E2	W2	C2	E3	W3	C3	E4	W4	C4	E5	W5	C5	W6	W7
S1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1
S2	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
S3	1	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1
S4	1	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1
S5	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1
P	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	x	x

Table 5. Fault isolation table for the symptoms in Table 4.

In practice, however, the presented fault isolation technique can be implemented only if a redundant channel is available. Multiple fault events from all ECUs of the entire network are gathered by a dedicated ECU through either the existing network itself or the redundant channel when the main network is permanently unavailable, e.g. permanently broken wires. The redundant channel can be a wired or wireless link among the ECUs, e.g. FlexRay dual-channel topology or Bluetooth. Intermittent fault data could be used to represent a

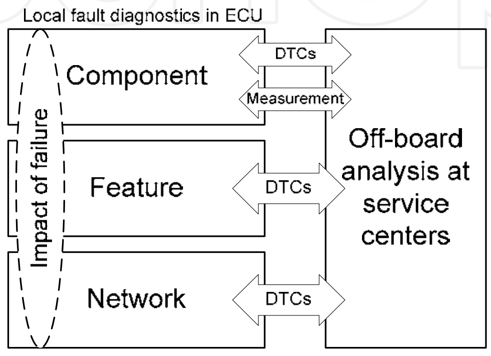
deterioration statistics by using a counter to count the number of detected faults in the component.
This section has discussed in-vehicle network fault detection which ranges from simple application message monitoring to more robust and complicated network and ECU status monitoring e.g. OSEK DNM. After faults are detected, on-board diagnostic applications in ECUs can obtain current network or status by requesting NM, if implemented, to provide the status such as an ECU in "Limp-Home", "Present", or "Mute" mode. After gathering the status and information, and performing fault isolation, the diagnostic applications log network-relevant DTCs according to a description specified by the manufacturer. Network-relevant DTCs are commonly standardised to have a prefix 'U'. Some examples are shown below.

U0003: "High speed CAN communication bus (+) Open"
U0100: "Lost communication with ECU A"
U0301: "Software incompatibility with ECU B"
U0401: "Invalid data received from ECU C"

These DTCs are read by a specific tool to be guidance to engineers to find out root causes. In general, there can be several DTCs that relate to a single fault logged by all diagnostic levels. To correctly diagnose the real root causes, interpretation from experienced engineers or assistance from an advanced guiding method such as the work of Huang et al., 2008 may be required.

4. Recent research on FDD for in-vehicle networks

The previous section discussed how faults in automotive electronic systems are conventionally handled. However, network level faults can result from a number of causes. As illustrated in Fig. 19, failures within an in-vehicle electronic system can manifest themselves in many different ways and on a number of different levels. The causes can be directly from the network level itself or indirectly from the other two levels. For instance, if a message is not transmitted from an ECU as specified, this problem can be caused by an internal function in the ECU itself (feature level), the CAN controller or the electrical wiring (component level). It is noted that physical hardware faults of in-vehicle networks such as CAN cables short-circuit-to-ground or the battery are considered as component level faults.


Local fault diagnostics in ECU
Component
DTCs
Measurement
Impact of failure
Feature
DTCs
Network
DTCs
Off-board analysis at service centers
Fig. 19. Impact of failure across different diagnostic levels.

Vehicle diagnosis using DTCs intended for later off-board root cause analysis and repair actions can be unwieldy when it comes to dealing with a system comprising a large number of ECUs and messages. For instance, several components in the system or communication path can cause a DTC U0401 to be set. The data might have been sent correctly by the "ECU C" but it was probably corrupted on the bus by intermittent faults in the cable wires. When performing off-board diagnostics, it is almost impossible to simulate the same operating condition as when the faults were present. This subsection discusses recent research on fully on-board FDD techniques that are being implemented to manage faults in real-time and in a system-wide and network-wide perspective.
Recent years have seen a number of research projects on various areas related to intelligent vehicles such as vehicles with collision avoidance and self diagnosis capabilities. One of such projects is known as the Self-Healing Vehicle concept which outlines a required paradigm shift of fault diagnostics and management of in-vehicle electronic systems (Amor-Segan et al., 2007). The Self-Healing Vehicle concept, illustrated in Fig. 20, is intended to mitigate failures within the vehicle's embedded software and electronic control systems in order to "safely keep the vehicle on the road". The concept envisages a vehicle equipped with a standardised, general purpose, networked computing architecture that facilitates software and application mobility. This mechanism will be managed by an intelligent fault management system with access to remote support services via a telematics link. The Intelligent Black Box (iB^2) will efficiently use available information in a vehicle that is currently under-utilised. This is the large volume of real-time component-, feature- or network-related information flowing through the vehicle's networks. Intelligent use of this information will be a key factor in the accurate operation of whole-vehicle diagnostics and prognostics, and the overall effectiveness of the fault management system.

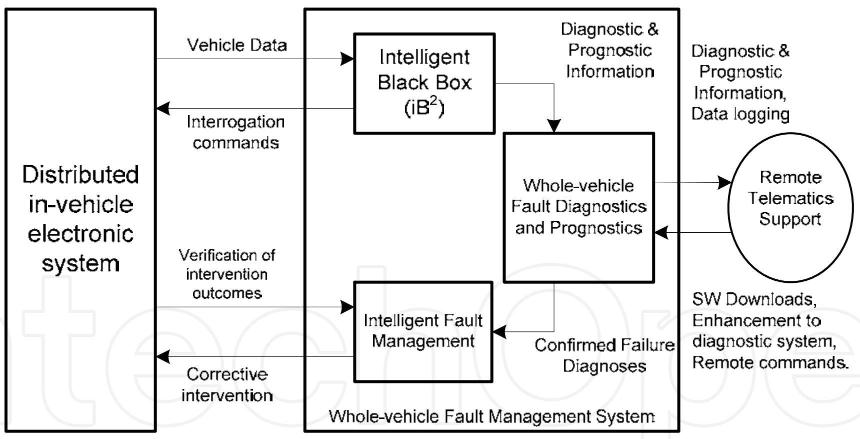

Distributed in-vehicle electronic system
Vehicle Data
Interrogation commands
Verification of intervention outcomes
Corrective intervention
Intelligent Black Box (iB2)
Diagnostic & Prognostic Information
Whole-vehicle Fault Diagnostics and Prognostics
Intelligent Fault Management
Confirmed Failure Diagnoses
Whole-vehicle Fault Management System
Diagnostic & Prognostic Information, Data logging
Remote Telematics Support
SW Downloads, Enhancement to diagnostic system, Remote commands.

Fig. 20. The Self-Healing Vehicle concept (adapted from Amor-Segan et al., 2007).

The iB^2 is a system capable of performing the following important functions:

- Monitoring of a variety of data sources such as network conditions, system and component resources, internal ECU data, etc.;
- Intelligent data mining, data compression, filtering and analysis;
- Adapting its behaviour under the instruction from the whole-vehicle diagnostic and prognostic system;
- Proactive system observations for fault diagnosis and prognosis.

These functions will support the on-board whole-vehicle fault management in obtaining accurate diagnoses. Clearly, a new framework for fully on-board monitoring and utilising the system information—component-, feature-, and network-related information—becomes necessary.
As for the network level, network information can be analysed by an intelligent network diagnostic and prognostic process as illustrated in Fig. 21. As a result, accurate diagnoses can be generated, and an alarm is triggered when predicted healthiness of the network falls below a threshold value (T). Certainly, to achieve this, robustness, accuracy, and continuity of network level fault detection will need to be improved.

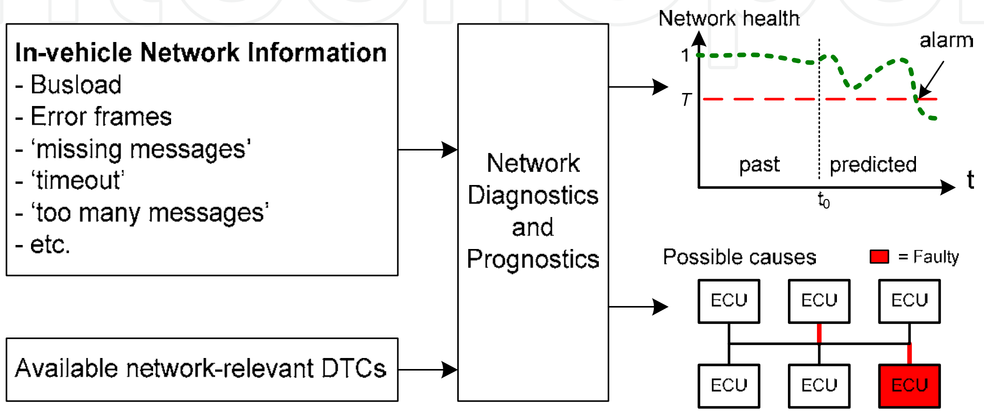


Fig. 21. In-vehicle network diagnostics and prognostics.

4.1 Evaluating network health

The network health could be indicated by the number of error frames. An error frame is transmitted when any of the communication error types is detected by an error management unit fitted in a CAN controller. Error detection embraces Cyclic Redundancy Check (CRC), bit monitoring, bit stuffing and frame checking. Errors can be caused by a number of reasons one of which is the presence of physical disturbances. When CAN buses are installed in a vehicle, there can be a situation where the physical wiring or connectors are damaged or gradually deteriorated by vibration or corrosion. This would cause the electrical signal to vary from specifications, thereby resulting in communication failures and error frames being transmitted.
Fig. 22 illustrates a captured CAN message from an oscilloscope displaying a signal distortion effected by analogue disturbances such as low resistance between CAN cables. The data can still be decoded by the oscilloscope and there are no error frames transmitted. Fig. 23 depicts a zoomed-in capture of the distorted signal. Fig. 24 shows a situation where there are more disturbances injected such that error frames are transmitted and the data cannot be decoded by the oscilloscope.

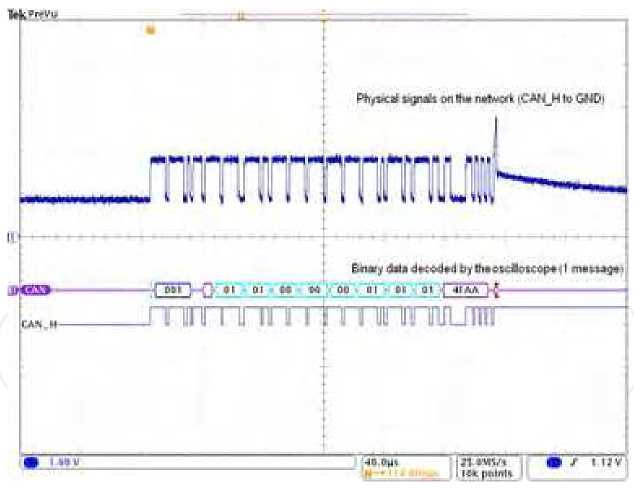

Tek PreVu
Physical signals on the network (CAN_H to GND)
Binary data decoded by the oscilloscope (1 message)
CAN
001
01
01
00
00
00
01
01
01
4FAA
CAN_H
1.00 V
40.0µs
25.0MS/s
10k points
1.12 V

Fig. 22. Physical CAN signal disturbed by an analogue disturbance.

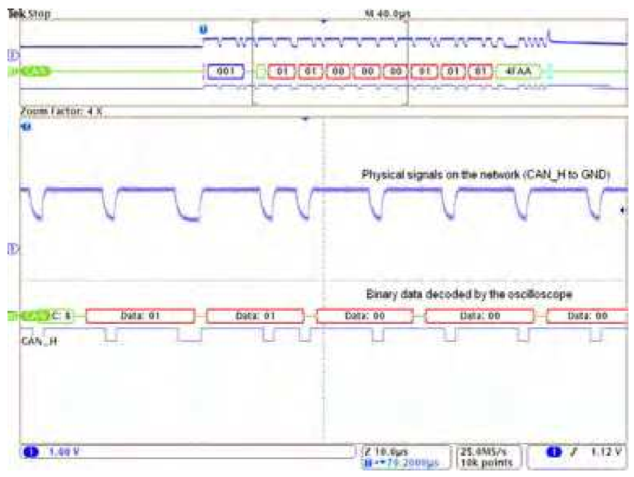


Fig. 23. Zoomed-in CAN signal disturbed by an analogue disturbance.

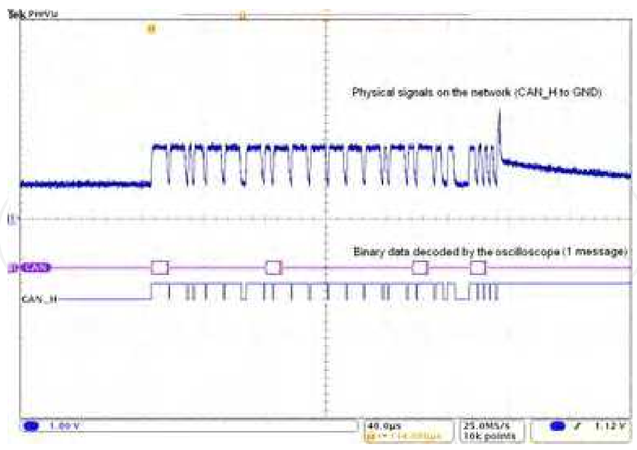

Tek PreVu
Physical signals on the network (CAN_H to GND)
Binary data decoded by the oscilloscope (1 message)
CAN
CAN_H
1.00 V
40.0µs
25.0MS/s
10k points
1.12 V

Fig. 24. Physical CAN signal distorted by more analogue disturbances.

The relationship between the number of error frames and the fault types remains under investigation at this stage. However, inference systems may be used to analyse the collected data from preliminary experiments. For instance, as shown in Fig. 25, an adaptive-network-based fuzzy inference system (ANFIS) can classify the number of error frames into such network health conditions as "unhealthy", "fair" and "healthy".

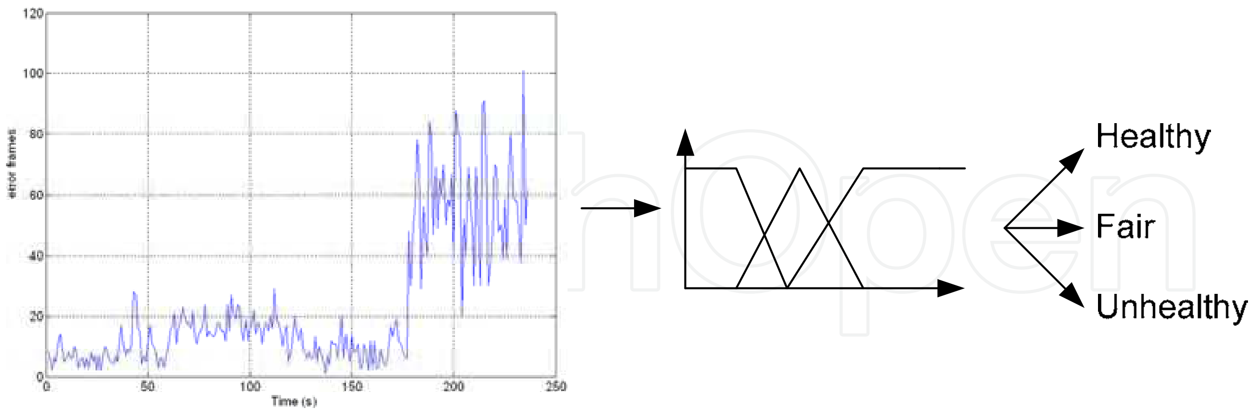
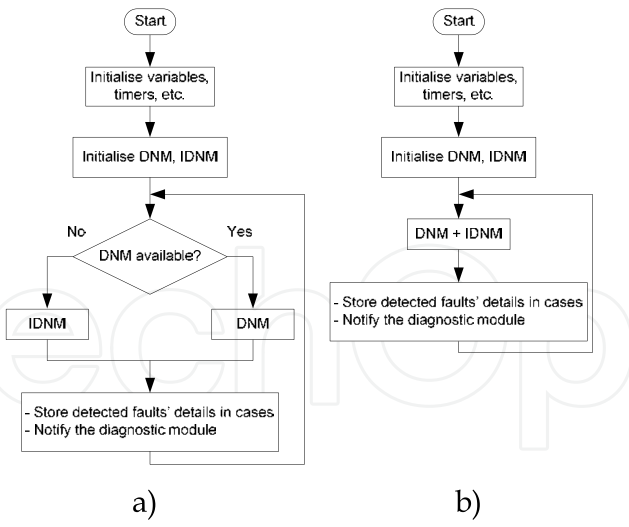


Fig. 25. Network health classification using ANFIS.

4.2 Adaptive network management

As discussed in the previous section, DNM provides network status to an application which is running in a networked ECU environment. With DNM, there is less overhead to the application and a high level of portability across multiple applications can be achieved. IDNM, on the other hand, is much simpler to implement. Despite being less robust and limited availability of node and network status, IDNM is an alternative mechanism for systems where rapid response from NM is not necessary, and simple software algorithms and computational resource consumption are of concern.
Traditionally, network diagnostic applications in ECUs only use either DNM or IDNM to provide them with node and network status. If DNM is implemented, some application message faults such as intermittent application message missing cannot be detectable because application messages are not monitored. Moreover, if NM messages are not available from a particular node as a result of its embedded software faults while application messages can still be transmitted, diagnostic applications in other nodes will no longer be able to detect faults from that faulty node. Conversely, if IDNM is implemented, as mentioned earlier, availability of node and network status is limited. This therefore would result in less fault coverage in a networked ECU environment. To enhance coverage, robustness and continuity of network level fault detection the Adaptive OSEK NM technique has been proposed (Suwatthikul, 2007). This technique is divided into two approaches: (i) switching DNM to IDNM, called Switched NM; and (ii) combining DNM and IDNM, called Combined NM.
The concept of combining both approaches is based on the fact that if the use of NM by applications is not fixed to DNM or IDNM, then the applications will benefit from using the advantages of both types of traditional OSEK NM as fault monitoring continues. As illustrated in Fig. 26, both approaches begin with variable and timer initialisation. For Switched NM, DNM is initially used as the main NM of the system. If DNM is not available for some reason such as NM API failure, the IDNM mode will be entered, i.e. application messages are monitored by a diagnostic application rather than using information from dedicated NM messages. For Combined NM, an application uses DNM and IDNM together such that application messages and NM messages are monitored at all time.


Start
Initialise variables, timers, etc.
Initialise DNM, IDNM
No
Yes
DNM available?
IDNM
DNM
- Store detected faults' details in cases
- Notify the diagnostic module
a)
Start
Initialise variables, timers, etc.
Initialise DNM, IDNM
DNM + IDNM
- Store detected faults' details in cases
- Notify the diagnostic module
b)
Fig. 26. Adaptive OSEK NM: a) Switched NM; b) Combined NM.

Although Combined NM will inherently consume more computational resources than when traditionally using separate NM, the simulation results show that Combined NM can cover more fault scenarios, and provide more accurate network fault detection and improved robustness (Suwatthikul, 2007).

In conclusion, this chapter has provided readers with an idea of how FDD can be applied in practice, especially current and future applications in the automotive industry. Modern in-vehicle electronic systems have become much more complicated due to additional sophisticated features. This certainly requires more intelligent and robust FDD. Despite available FDD approaches from a number of non-automotive domains, the approaches cannot be simply applied to the automotive domain due to obvious constraints such as costs and different end users. The aerospace industry has employed advanced FDD for decades, involving model-based FDD, bespoke components and redundant systems, in particular for safety-critical applications. Inevitably, such approaches are extremely costly and therefore not well suited for applications in the automotive industry.

The direction of future research on FDD in automobiles tends to focus on cost-effective and intelligent approaches to whole-vehicle fault management, and new component and system architectures. For instance, the use of generic processing components to implement different fault management strategies in a vehicle that will achieve the required levels of resilience or fault tolerance, yet in a way that is cost-effective and realisable for the automotive domain.

5. Acknowledgements

The author would like to thank the International Automotive Research Centre, University of Warwick, for the support of equipment and tools used in the experiments.

6. References

Amor-Segan, M. L.; McMurran, R.; Dhadyalla, G. & Jones, R. P. (2007). Towards the Self-Healing Vehicle, *Proceeding of the 3rd IET Automotive Electronics Conference*, Coventry, June 2007.

Armengaud, E.; Steininger, A.; Horauer, M.; Pallierer, R. & Friedl, H. (2004). A monitoring concept for an automotive distributed network: the FlexRay example, *Proceeding of the 7th IEEE Workshop on Design and Diagnostics of Electronic Circuit and Systems*, Stara Lesna, April 2004. pp. 173-178.

Armengaud, E.; Steininger, A. & Horauer, M. (2005). Efficient stimulus generation for testing embedded distributed systems: the FlexRay example, *Proceeding of the 10th IEEE Int. Conference on Emerging Technology and Factory Automation*, Catania, September 2005, pp. 763-770.

Armengaud, E.; Rothensteiner, F.; Steininger, A.; Pallierer, R.; Horauer, M. & Zauner, M. (2005). A structured approach for the systematic test of embedded automotive communication systems, *Proceeding of International Test Conference 2005*, Texas, Nov. 2005, pp. 1-8.

Athanasas, K. & Dear, I. (2004). Validation of complex vehicle systems of prototype vehicles. *IEEE Transactions on Vehicular Technology*, Vol. 53, No. 6, pp.1835-1846.

Buja, G.; Pimentel, J. R. & Zuccollo, A. (2007). 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, pp. 225-233.

Ehret, J. (2003). Validation of safety-critical distributed real-time systems. Dr.-Ing. thesis, Technical University of Munich, Munich.

Foran, T. & Jackman, B. (2005). An intelligent diagnostic system for distributed multi-ECU automotive control systems, SAE paper: 2005-01-1444, *SAE World Congress*, Michigan, April 2005.

Gaujal, B. & Navet, N. (2005). Fault confinement mechanism on CAN: Analysis and Improvements. *IEEE Transactions on Vehicular Technology*, Vol. 54, No. 3, pp.1103-1113.

Isermann, R.; Schaffnit, J. & Sinsel, S. (1999). Hardware-in-the-loop simulation for the design and testing of engine-control systems. *Control Engineering Practice*, Vol. 7, No. 5, pp. 643-653.

Isermann, R. (2008). Mechatronic systems—Innovative products with embedded control. *Control Engineering Practice*, Vol. 16, No. 1, pp. 14-29.

Kendall I. R. & Jones, R. P. (1999). An investigation into the use of hardware-in-the-loop simulation testing for automotive electronic control systems. *Control Engineering Practice*, Vol. 7, No. 11, pp. 1343-1356.

Kopetz, H. & Bauer, G. (2003). The time-triggered architecture. *Proceedings of IEEE*, Vol. 91, No. 1, pp.112-126.

Lawrenz, W. (1997). *CAN System Engineering: From Theory to Practical Applications*, Springer-Verlag, ISBN 0387949399, New York.

Leen, G.; Heffernan, D. & Dunne, A. (1999). Digital networks in automotive vehicle. *Computing and Control Engineering Journal*, Vol. 10, No. 6, pp. 257-266.

Leen, G. & Heffernan, D. (2002). TTCAN: A new time-triggered controller area network. *Microprocessors and Microsystems*, Vol. 26, No. 2, pp. 77-94.

Lemieux, J. (2001). *Programming in the OSEK/VDX Environment*, CMP Books, ISBN 1578200814, Kansas.

McMurran, R.; McKinney, F.; Tudor, N. J. & Milam, M. (2006). Dependable Systems of Systems, SAE paper: 2006-01-0597, *SAE World Congress*, Michigan, April 2006.

Murphey, Y. L.; Crossman, J. A.; Chen, Z. & Cardillo, J. (2003). Automotive fault diagnosis—Part II: A distributed agent diagnostic system. *IEEE Transactions on Vehicular Technology*, Vol. 52, No. 4, pp.1076-1098.

Navet, N. & Song, Y.-Q. (1999). Reliability improvement of the dual-priority protocol under unreliable transmission. *Control Engineering Practice*, Vol. 7, No. 8, pp. 975-981.

Navet, N. & Song, Y.-Q. (2001). Validation of real-time in-vehicle applications. *Computers in Industry*, Vol. 46, No. 2, pp. 107-122.

Navet, N.; Song, Y. Q.; Simonot-Lion, F. & Wilwert, C. (2005). Trend in automotive communication systems. *Proceedings of IEEE Special Issue Industrial Communication Systems*, Vol. 93, pp.1024-1223.

Ortega, E.; Heurung, T. & Swanson. R. (2006). System design from wires to warranty. *Automotive Electronics Magazine*, February, pp. 14-18.

OSEK/VDX (2004). Network Management: Concept and Application Programming Interface Version 2.5.3. [Online]. Available: http://www.osek-vdx.org.

Robert Bosch GmbH. (2004). *Automotive Electrics Automotive Electronics*, Professional Engineering Publishing Limited, ISBN 0470519371, Suffolk.

Robert Bosch GmbH. (2004). *Gasoline-Engine Management*, Robert Bosch GmbH, ISBN 0837610524, Bury St. Edmunds.

Robert Bosch GmbH. (2004). *Automotive Handbook*, Robert Bosch GmbH, ISBN 0768015138, Plochingen.

Short M. & Pont, M. J. (2008). Assessment of high-integrity embedded automotive control systems using hardware in the loop simulation. *Journal of Systems and Software*, Vol. 81, No. 7, pp. 1163-1183.

Shrinath, A. & Emadi, A. (2004). Electronic control units for automotive electrical power systems: Communication and networks. *Proceedings of IMechE Part D: Journal of Automobile Engineering*, Vol. 218, pp. 1217-1230.

Simonot-Lion, F. (2003). In-car embedded electronic architectures: how to ensure their safety, *Proceeding of the 5th IFAC International Conference on Fieldbus Systems and Their Applications*, Aveiro, July 2003.

Storey, N. (1997). *Safety-Critical Computer Systems*, Addison Wesley Longman, ISBN 0201427877, New York.

Suwatthikul, J; McMurran, R. & Jones, R. P. (2007). Adaptive OSEK Network Management for In-vehicle Network Fault Detection, *Proceeding of the 2007 IEEE International Conference on Vehicular Electronics and Safety*, Beijing, Dec. 2007.

Temple, C. (1998). Avoiding the babbling-idiot failure in a time-triggered communication system, *Proceeding of the 28th Annual International Symposium on Fault-Tolerant Computing*, pp. 218–225.

Tindell, K. & Hansson, H. (1995). Babbling idiots, the dual priority protocol, and smart CAN controllers, *Proceeding of the 2nd International CAN Conference*, pp. 7.22–7.28.

Zurawski, R. (2006). *Embedded Systems Handbook*, CRC Press, ISBN 0849328241, Florida.

Fault Detection
INTECH