Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

A Reliable Gateway for In-vehicle Networks

S. H. Seo*, J. H. Kim*, T. Y. Moon*
S. H. Hwang, K. H. Kwon*, J. W. Jeon***

**School of Information and Communication Engineering, Sungkyunkwan University, Suwon, Korea*
(e-mail: {imakeit, ssagagy, ninewolf, khkwon, jwjeon}@ece.skku.ac.kr).
***School of Mechanical Engineering, Sungkyunkwan University, Suwon, Korea*
(e-mail:hsh@me.skku.ac.kr)

Abstract: This paper presents a reliable gateway for communication between the LIN, CAN, FlexRay protocols. A gateway is indispensable device for constructing in-vehicle networks. Networks with different protocols have to include an additional gateway in order to exchange information among different networks. The main function of the gateway is translation. However, there is some latency when a message is transferred from one node (source) to another (destination); further, there is a high probability of error due to different protocol specifications such as baudrates, message frame formats, and so on. Therefore the implementation of a reliable gateway is a challenging task. In this paper, we propose a reliable gateway based on OSEK OS and OSEK NM for in-vehicle networks. We develop a gateway embedded system and implement a reliable gateway mechanism. We then examine the developed gateway system and present the results of experiments with several trials.

1. INTRODUCTION

The increase in the number of sensors, actuators and electronic control units (ECUs) in automotive networks over the last few years has increased the complexity of automotive networks. Moreover, several communication protocols such as LIN (Local Interconnect Network), CAN (Controller Area Network), and FlexRay have been established and used for in-vehicle networks (Andrew *et al*., 2004).

Currently, CAN is used in the automotive industry as the primary in-vehicle network (Andrew *et al*., 2004) for enabling any device to communicate and operate with any other device on a network without placing a great burden on the bus. This kind of communication is ideal for powertrain and body electronic applications (Rami *et al*., 2004).

However, the introduction of advanced control systems, which employ a number of sensors, actuators, and ECUs, has begun to place boundary demands on the existing CAN communications bus commonly used in most automobiles. As a result, initiatives by automobile manufacturers and suppliers have led to the creation of FlexRay ─ an open standard for a new deterministic, fault-tolerant, and high-speed bus system. FlexRay is a new network communication system targeted specifically at next generation automotive applications or "by-wire" applications. By-wire applications require high-speed bus systems that are deterministic, fault-tolerant and capable of supporting distributed control systems. Furthermore, LIN is still widely used as it is a cheaper option.

The LIN, CAN, and FlexRay primarily will be used as future in-vehicle networks (Weber *et al*., 2006); Fig. 1 presents a example of in-vehicle network. There is a challenging issue to exchange information between networks with different protocol reliably. Exchanging information between different networks has a high probability of error occurrence due to different protocol specifications such as baudrates, message frame formats, transmission latencies, and so on. If many ECUs can exchange information each other reliably, the performance of vehicles will be improved (Thomas, 2006). A gateway will thus come to be an important and indispensable device for in-vehicle networks (Huang *et al*., 2006).

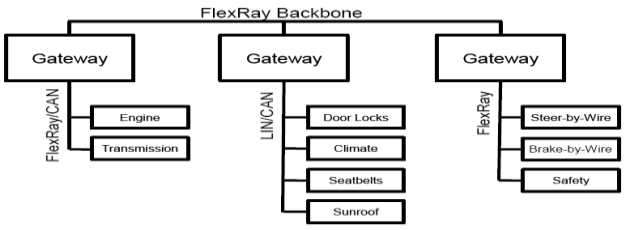

Fig. 1. General Architecture of In-Vehicle Network for Next Generation Vehicles

In this paper, we propose a reliable gateway system (Mohinisudhan *et al*., 2006) based on OSEK operating system (OS) and network management (NM) (Qiao *et al*., 2005). The proposed gateway exchanges information (messages) between the LIN, CAN, and FlexRay networks basically. The operation of the gateway is determined along the status of the destination node. Therefore, any unnecessary overhead of the network is reduced, thereby ensuring reliable operation. In this paper, we developed a gateway embedded system and implemented a reliable gateway function based on OSEK OS and NM. In addition, the developed gateway provides trace information called by a log recorder. A log recorder stores translation information continuously. Thus, we can trace the instant at which a fault occurs just by reviewing the log data.

The remainder of this paper is organized as follows. Section 2 provides an overview of the LIN, CAN, and FlexRay protocols and gateways. Sections 3 and 4 describe the proposed reliable gateway mechanism and the developed gateway embedded system, respectively. Section 5 presents several experimental evaluations and results. Finally, the conclusion is provided in section 6.

2. OVERVIEW OF LIN, CAN, FLEXRAY AND GATEWAY

Today, automobiles come equipped with several in-vehicle networks, each using the same or different communication protocols from LIN, CAN, and FlexRay. A gateway is essential for exchanging information between these networks. In the subsequent section, the specification of each protocol will be provided and the concept of a gateway will be elucidated.

2.1 LIN

The LIN-Bus is a bus-system used in current automotive network architectures. The LIN bus is a small, slow network system that is used as a cheap sub-network of a CAN bus to integrate intelligent sensor devices or actuators into today's automobiles.

As it is described in LIN specification documents, all message traffic on the bus is initiated by the master device. The slave devices respond to commands and requests from the master. Since the master initiates all bus traffic, it follows that the slaves cannot communicate unless requested to do so by the master. However, the slave devices can generate a bus wakeup call, if their inherent functionality requires this feature. The master node uses one or more predefined scheduling tables to start the sending and receiving process on the LIN bus. These scheduling tables contain at least the relative timing pertaining to the initiation of the message sending process. More detailed specifications of LIN are described in (LIN specification, 2003).

2.2 CAN

The CAN is a serial communication protocol that efficiently supports distributed real-time control with a very high level of data integrity. Therefore, CAN is used to construct the distributed control system in many areas of industry such as automobiles, robotics, avionics, and hydraulics.

The CAN supports two message frame formats ― the standard frame and the extended frame formats. The only difference between the two is the length of the identifier; the standard frame format (CAN 2.0A) supports a length of 11bits for the identifier while the extended frame format (CAN 2.0B) supports a length of 29bits. The standard CAN message frame comprises an SOF bit, identifier, RTR, IDE, r0, DLC, payload, CRC, ACK, EOF, and IFS. IDE stands for identifier extension; it is a single bit that indicates that a standard CAN identifier with no extension is being transmitted. ACK comprises 2bits―an acknowledgement bit and a delimiter. When a node receives an accurate message, it overwrites the recessive bit in the original message with a dominant bit, thereby indicating that an error-free message has been transmitted (CAN specification, 2003).

2.3 FlexRay

The FlexRay is expected to be a comprehensive communication system, providing high speed, flexibility and scalability for complex networks as it is based on time division multiple access (TDMA) and flexible time division multiple access (FTDMA) scheme. One of the types of applications the FlexRay is expected to make possible are X-by-wire systems such as brake-by-wire and steer-by-wire systems. Sophisticated electronic systems that are less expensive to construct and easier to maintain are used to connect driver to these systems; therefore, X-by-wire systems eliminate the need for hydraulic and mechanical systems. With a gross data rate of 10 Mbps the net bandwidth delivered by FlexRay is approximately 20 times greater than that delivered by the CAN currently used in advanced automotive control applications.

The FlexRay supports fault-tolerant clock synchronization via a global time base, collision-free bus access, guaranteed message latency, message oriented addressing via identifiers, and scalable system fault-tolerance via the support of either a single or a dual channel.

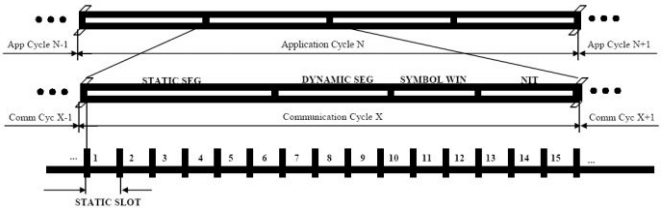

Fig. 2. FlexRay Timing related to the Schedule of the Communication Cycle

Within FlexRay, communication occurs in recurring communication cycles. Each communication cycle is built on a timing hierarchy as shown in Fig. 2. In order to satisfy diverse communication requirements, FlexRay provides static and dynamic communication segments within each communication cycle. In addition, each cycle contains a symbol window used for run-time testing and the network idle time, which is a communication-free period that concludes each communication cycle (FlexRay Specification, 2005).

2.4 The Concept of Gateway

A gateway (Taube, 2005) converts the data frame format of one protocol to that of another protocol. As mentioned previously, several protocols are used in automotive network system; it depends on the feature of applications. Therefore a gateway system is necessary for supporting multiple protocols and satisfying demands that are not currently addressed in the existing system architectures (Li *et al*., 2005).

A gateway has to provide message translation, message routing, message monitoring, and network management, all in real-time. Therefore, dedicated gateways must be developed to reduce the demands on the CPU load and to reduce message latency. In this paper, we implemented a gateway system based on the OSEK OS with NM. In addition, the developed gateway system records information that includes a timestamp, message ID, protocol type, and error during message translation.

The gateway hardware system is based on simple system architecture. It consists of a standard CPU core, an internal RAM or flash, especially, and several communication controllers. Various application-specific modules may be added. All modules are connected via a CPU peripheral bus to a host CPU. The complexity of the communication controller can be increased or decreased depending on the actual requirements of the application field. The required CPU performance will increase in order to guarantee the data integrity and real-time operability in the overall system.

3. RELIABLE GATEWAY MECHANISM USING OSEK OS, NM AND LOG RECORDER

OSEK OS (OSEK/VDX portal, 2007) is a real-time operating system (RTOS) for distributed control units in vehicles. This OS serves as the basis for the controlled real-time execution of concurrent applications. The OSEK OS defines a small, scalable, RTOS, ideal for embedded systems with high memory constraints and dedicated functions. A more detailed description of OSEK OS, NM, log recorder, and the proposed reliable gateway mechanism will be provided in the subsequent sections.

3.1 OSEK/VDX OS

OSEK OS manages real-time tasks, enhanced timer functions, shared resources, task synchronization using events, and inter-process communication. Due to the wide range of scalability, a set of system services, various scheduling mechanisms, and convenient configuration features, the OSEK OS is suitable for a broad spectrum of application and hardware platforms. The OSEK OS provides a pool of services and processing mechanisms for task management and synchronization, data exchange, resource management, and interrupt handling.

3.2 OSEK NM

OSEK NM defines protocols for managing in-vehicle networks. NM ensures the safety and reliability of a communication network. This is done by restricting the access to each node, making the network tolerant to faults, and implementing diagnostic features capable of monitoring the status of the network in a direct or an indirect manner. Moreover, the network management also covers the initialization of network resources, and network configuration, the co-ordination of global operation modes, and support for diagnosis.

As mentioned previously, OSEK NM provides two types of network management mechanisms—direct NM and in-direct NM. Direct NM monitors the network by dedicated NM communication using a token principle. In other words, a dedicated message is used in order to monitor the network. Although it provides more precise network status information, dedicated NM has more overhead than indirect NM. It is suitable for time-critical systems or small network systems. In order to use direct NM it must be implemented in all networked nodes and a node identification (ID) must be assigned to all nodes. On the basis of this ID, the nodes form a logical ring in direct NM. Then, the node transmits a ring message to the next node in the logical ring. The node that receives the ring message transmits the message to the next node (i.e, direct NM cycles a ring message to the logical ring by the relaying the ring message to the next node). If some node cannot relay ring messages to the next node within a predefined time, the other nodes recognize that some nodes cannot send or receive messages. Thus, other nodes form a new logical ring excluding those particular nodes. Direct NM considers nodes to be in the "*present state*" if they participate in a logical ring. In contrast, nodes are considered to be in the "*absent state*" if they cannot participate in the logical ring. The present state implies normal operation while the absent state implies that the nodes cannot send or receive messages.

Indirect NM is another management scheme. It monitors application messages; that is, a periodic application message is used. Indirect NM does not use any additional messages to monitor the network. Therefore, it is simpler than direct NM. Indirect NM is suitable for large network systems that are not time-critical. It regards the source node of a specific message as the present state when a node receives a specific application message successfully. If a node cannot receive a specific message during time-out, indirect NM regards the source node of the specific message as being in the absent state. Unlike direct NM, indirect NM does not assign the ID to the entire node and does not include an indirect mechanism to all nodes in the network. In this paper, we implemented a reliable gateway by considering node status using these NM schemes. The in-direct NM is used for FlexRay and LIN network managements and the direct NM is used for CAN network management.

3.3 Log Recorder

Log recorder is storing and reporting the result of conversion among each different communication protocol. The length of the data frame, possible transmission data length, transmission bit rate, error detection scheme, ID format, data type, and composition field of the frame of each communication protocol are different. For example, the data length of LIN is determined as a specified ID between 0 to 8 bytes. The length of a CAN message also varies from 0 to 8 bytes. The length of the FlexRay message is up to 256 bytes determined by initial conditions. The differences among each protocol can possibly cause problems such as loss of data when one communication protocol is converts into another communication protocol. In addition, the networked node can operate incorrectly due to message loss and latency by

gateway while a node operates correctly independently. Then, if there are conversion reports such as what messages are converted, and what data is lost, the developer can more efficiently debug the errors. If the converted message didn't transmit to the destination node within pre-defined time, the log recorder writes the receive time to the transmission time field equally. Table 1 shows the structure of the log table. A time stamp is written in the Receive Time field when the gateway receives a message with Message ID via input interface. Also, a time stamp is written in the Transmit Time field when gateway transmits the converted message to the destination node with Message ID completely.

Table 1. The Structure of Log Table

Receive Time	Input Interface	Message ID	Output Interface	Message ID	Transmit Time

3.4 Reliable Gateway Mechanism

We consider three factors for improving the reliability of the gateway. First, high priority messages must be processed first. In order to resolve this problem, full-preempt tasks are carried out to convert the protocol of each message and we assign a priority to each task. Then, the tasks having high priority can be processed first.

Second, the gateway cannot transmit messages to erroneous (fault) nodes that are unable to receive messages. In an automotive system, loss of messages can cause hazardous problems if the message is an important message, for example, a brake control message. Moreover, CAN and FlexRay provide a retransmission mechanism for failure messages; this can increase the load of the network. Thus the proposed gateway checks the status (present or absent) of the destination node by using OSEK NM before transmission of the converted messages. If the destination nodes are in the absent state, the gateway searches other paths to reach the destination. Then, the gateway transmits messages over alternative paths. If no alternative paths exist, the gateway holds the transmission of messages until the destination node becomes present.

Third, the gateway has to provide trace information. Trace information is log data that includes the status and result of translation along with a timestamp. Thus, we implemented the log recorder that stores log information. If some problem (missed message or error in message) occurs during the translation process, we can retrace the instant at which the error occurred.

4. GATEWAY EMBEDDED SYSTEM

We developed the proposed gateway embedded system. The following subsequent sections explain the hardware and software architecture of the gateway embedded system, and implementation of the gateway function.

4.1 Hardware Architecture of Gateway

The developed gateway embedded system is based on simple hardware architecture. This system consists of a standard MCU, an internal Flash, RAM, power management devices, and several communication controllers. Fig. 3 shows the block diagram of the gateway embedded system.

In this paper, the MCU of the gateway is a Freescale MPC565 that is suitable for automotive applications. This MCU is the appropriate because all its components satisfy the industrial standard and are able to endure the harsh conditions to which the car may be subjected. In addition, MPC565 has sufficient memory area and includes SCI and CAN communication controllers. A CAN communication controller supports both low-speed CAN and high-speed CAN. In order to supports LIN communication, we implemented LIN communication driver by using SCI modules. We used the MFR4200 communication controller made by Freescale to implement FlexRay. MFR4200 supports FlexRay protocol specification v1.1.

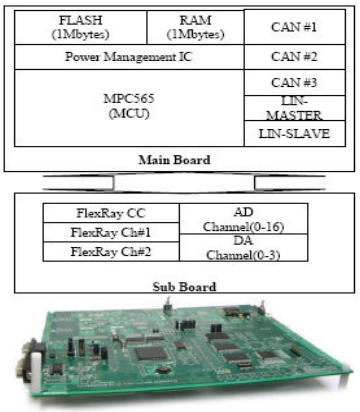

Fig. 3. Hardware Architecture of Gateway Embedded System

4.2 Software Architecture of Gateway

The gateway software architecture is based on OSEK OS (Sun *et al*., 2005). Fig. 4 shows the software architecture of gateway embedded system. We used OSEKturbo made by Metrowerks to implement the OSEK NM mechanism. The implemented OSEK NM specification is compatible with the standard OSEK NM specification v2.5.3. In addition, the communication device drivers are implemented for LIN, CAN, and FlexRay. The gateway function is implemented in the application layer. Log recorder operates with application program and library layer.

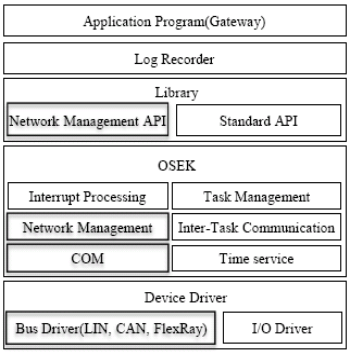

Fig. 4. Software Architecture of Gateway Embedded System

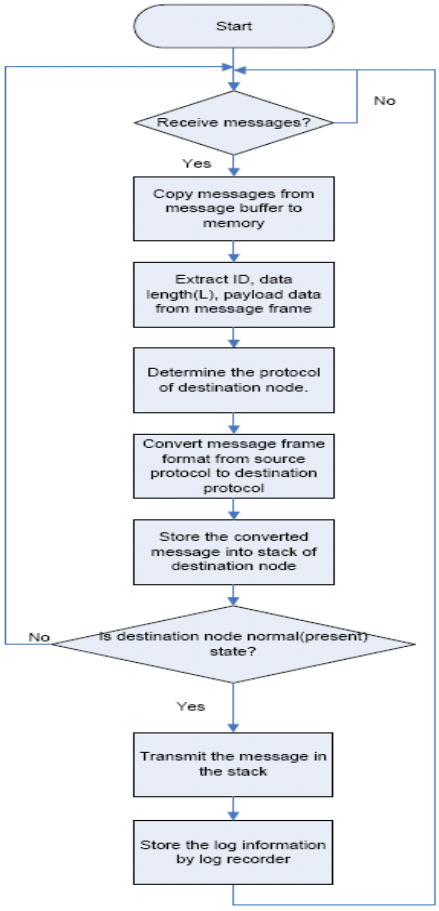

Fig. 5. Flowchart of the Gateway Function

4.3 Processing Flow of the Gateway Function

The gateway function provides primarily the translation between LIN, CAN and FlexRay. The flow of the gateway function is shown in Fig 5. The operation of the proposed gateway is based on OSEK OS.

The converting process among each different communication is as follow. There is an example of conversion between FlexRay and CAN. First, the gateway checks whether message is arrived or not. If there is a received message, the task of protocol dependent message receive interrupt service routine (ISR) is activated. ISR receives the message and copies the message to the memory buffer.

After ISR processed, second, each ISR activates a message-processing task among LIN, CAN and FlexRay message processing tasks. Each message-processing task is operated full-preemptly as priority. Therefore, the proposed gateway system processes received messages based on priority. The message-processing task extracts ID, data length, payload data from the received message. The message-processing task must have knowledge about the output interface and output message ID in order to determine the destination protocol. Therefore, we use two mechanisms to determine the output interface and output message ID — direct forwarding and routing table mechanisms. If the direct forwarding mechanism is used, the output interface and output message ID are defined such that the gateway transfers messages immediately. However, the direct-forwarding mechanism requires processing code for every path, which leads to an increase in the code size.

The routing-table mechanism uses a routing table to determine the output interface and output message ID. The routing table contains the input (source) interface, input message ID, output (destination) interface, and output message ID. The input and output interface fields stores the number of interfaces. For example, if the gateway connects CAN A with CAN B and Channel A for CAN communication and FlexRay, repectively, we assign interface numbers as follows: 0 for CAN A, 1 for CAN B, and 2 for Channel A. The gateway can identify both the interface and the protocol of the source node and the destination node by using the routing table. The lengths of the source ID and the destination ID field conform to the maximum length of the protocol used by the gateway. Table 1 shows the structure of the routing table. The routing table mechanism uses less memory because it requires processing code for each output interface. Moreover, it is possible to modify the routing table by software during runtime so that it is more flexible than the direct forwarding mechanism.

Third, the gateway detects the destination ID and output interface when it receives messages. It uses the direct forwarding or routing table mechanisms selectively for this purpose. The gateway converts the message frame format from source to destination node. And then, gateway stores the converted message in the stack of the destination node.

Fourth, the gateway checks whether the destination node is accessible (present) or not (absent). If the destination node is not accessible, the gateway searches the routing table again. The gateway sends the message via an alternative path if it successfully finds an alternative path. When no alternative path is available, the gateway suspends transmission of messages. The delayed message is still stored in the stack.

Finally log recorder stores log information in the log table. The whole of the process is performed repeatedly every the new message is received.

Table 2. Structure of Routing Table

Input Interface	Message ID	Output Interface	Message ID

5. EXPERIMENTATION AND RESULT

We setup the experimental environment as shown in Fig. 6. It consists of one gateway, three additional node for each protocol connected to the gateway and PC with USB-CAN interface. FlexRay supports dual-channel scheme. We evaluate two situations — normal state, and the error node occurrence state. In the error node occurrence state, we retraced the instant at which an error occurred. In these experimentations, we used the developed gateway embedded system for each node. The gateway only has the proposed gateway functions. Other nodes just transmit or receive

messages periodically. The number of nodes represents the message ID to be received.

In the first experimentation (normal state), node 1 transmits sequential data (65535, …, 0) to node 2. Node 2 transmits the sequential data (0, …, 65535) to node 1. Node 3 transmits sequential data(0, …, 127) to PC. Fig. 7 shows the result of conversion from FlexRay to CAN. We monitored the received data from node 2 via CAN.

In the second experimentation, we disconnect the node #2 with the gateway in the middle of conversion process (during the first experimentation). Fig. 8 shows the result of log report at the instant of error occurrence. The log information includes the receive time, source interface, source ID, destination interface, destination ID, and transmit time.

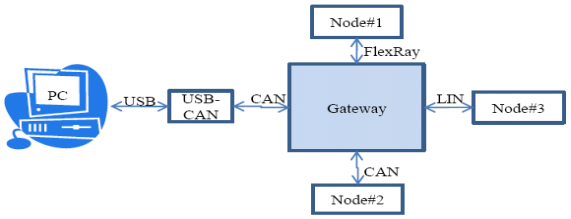

Fig. 6. The Configuration of Experimental Environment

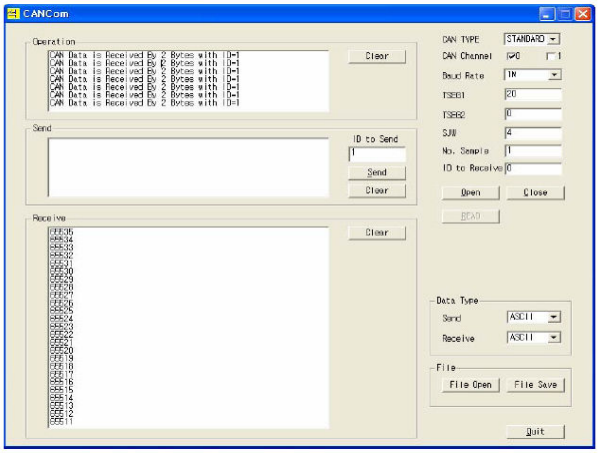

Fig. 7. The Result of Gateway at Normal State

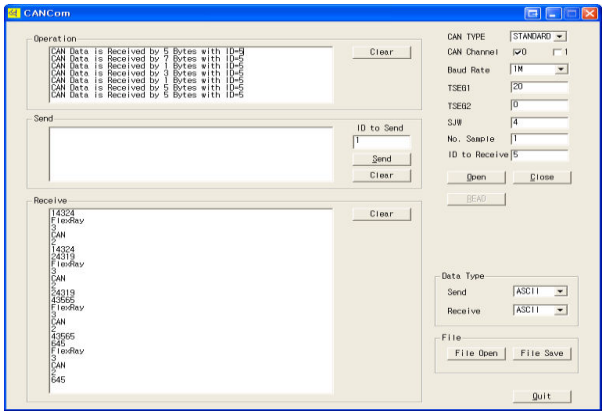

Fig. 8. The Result of Gateway at Fault State

6. CONCLUSION

We proposed a reliable gateway for in-vehicle networks with different protocols. The proposed gateway operates based on OSEK OS and NM. In addition, the gateway provides log information for us in order to analyze the gateway effect within the environment of networked control system. However, the developed gateway doesn't include the fault-tolerant mechanism yet. In the future, we will improve the reliability and safety of the gateway by adding the fault-tolerant mechanism called by OSEK FTCOM.

REFERENCES

B. Andrew, C. Stephen, M. Peter (2004) The Requirement of Future In-Vehicle Networks and An Example Implementation. *SAE 2004 World Congress & Exhibition*, 2004-01-0206.

B. Rami, A. Nizar, R. Asif (2004) Dynamic Discovery Service Protocols for Next Generation Vehicle Network. *SAE 2004 World Congress & Exhibition*, 2004-01-0199.

CAN Specification 2. 0(2003), http://www.can-cia.org/can/

FlexRay Protocol Specification V2.1 Rev. A, 2005, http://www.flexray.com

G. Mohinisudhan, S. K. Bhosale, B. S. Chaudhari (2006) Reliable On-board and Remote Vehicular Network Management for Hybrid Automobiles. *Internation IEEE Conference on Electric and Hybrid Vehicles(ICVHV'06)*, pp.1-4

H. Thomas (2006) In-vehicle Network Design Methodology. *The 2nd IEE Conference on Automotive Electronics*, pp.47-71

J. Taube (2005). Gateway Concepts for Automotive Networks. *Automotive 2005 Special Edition FlexRay*, http://www.hanser-automotive.de

LIN Specification Package 2.0 (2003), http://www.lin-subbus.org/

OSEK/ VDX Portal (2007), http://www.osek-vdx.org

Q. Huang, J. S. Smith, Tuo Li (2006) Web-based Distributed Embedded Gateway System Design. *WI2006*, pp.905-908

T. Weber (2006) Management of Complex Automotive Communication Networks. *Design, Automation and Test in Europe (DATE'06)*, **Vol. 1**, pp.1-2

X. Qiao, Z. Wang, Y. Sun, F. He, F. Y. Wang (2007) A CAN and OSEK NM based Siren for Automobiles. *2007 IEEE international Conference on Networking, Sensing and Control*, pp.868-873

Y. Li, F. Wang, F. He, Z. Li (2005) OSGi-based Service Gateway Architecture for Intelligent Automobiles. *Proceedings IEEE of Intelligent Vehicles Symposium Intelligent Vehicles Symposium*, 861-865

Y. Sun, F. Y. Wang (2005) A design Architecture for OSEK/VDX-based Vehicular Application Specific Embedded Operating Systems. *Proceedings IEEE of Intelligent Vehicles Symposium*, pp.882-887