Development of Scalable CAN Protocol

Ryo KURACHI*, Masanobu NISHIMURA*, Hiroaki TAKADA, Shigeharu TESHIMA,
Yukihiro MIYASHITA, Satoshi HORIHATA, Hideki YAMAMOTO and Akihiro NATSUME

In today's automotive industry, FlexRay and other next generation protocols for automotive network communications are gaining attention. However, these protocols are unlikely to replace existing applications in the immediate future due to the cost and reliability problems caused by the replacement.
In this paper we propose Scalable CAN, a new automotive network protocol based on the existing CAN (controller area network). Having a new ACK (Acknowledgement) information field, instead of an ACK slot, the Scalable CAN features a transmission speed of 10 Mbps and a new collision resolution algorithm which guarantees the delivery of a message within a given time period. Our simulation analysis indicates that the Scalable CAN protocol is superior to the conventional CAN in throughput performance such as a maximum data transmission speed, scalability, and priority inversion. This paper also includes considerations given for the implementation of the Scalable CAN.

Keywords: Scalable CAN, protocol, controller, delay

1. Introduction

In recent years, there has been a trend in the automotive industry of increasing the number of ECUs (engine control units) installed in a vehicle. Each ECU shares much information through in-vehicle networks and controls various functions. In these networks, the technologies of CAN[(1)] (Controller Area Network) for middle-speed communication and LIN[(2)] (Local Interconnect Network) for low-speed communication are commonly used. However, many ECUs are already connected to these networks, and for further function enhancement, we have to install an additional network using a GW (gateway) and/or unify the functions of ECUs to reduce the number.

Meanwhile, FlexRay[(3)] has been regarded as a fast and highly-reliable network, and some European automakers have already marketed vehicles equipped with the FlexRay. However, there still remain some problems for the replacement of CAN by FlexRay such as software (middleware and application) compatibility and the limit of connectable nodes.

Refurbishing whole automotive systems for the introduction of a new protocol is impractical in terms of costs and quality assurance. Therefore, technology development should be promoted to find a way to utilize existing protocols.

This report introduces the Scalable CAN, a new protocol that enables fast communication with low latency, is adaptable future high speed networks, and exhibits good compatibility with existing software assets.

2. Challenges for CAN

CAN has characteristic mechanisms that mediate collision messages. These mechanisms impose various constraints on the transmission channels and other functions of the CAN, causing transmission delay. Consequently, further improvement in the data processing rate of networks is inhibited. In fact, major automotive CANs use only 500 kbps at most, despite their specification of 1 Mbps. Under some specific conditions, 500 kbps or lower CANs do not satisfy the requirement of applications, either.

The following sections detail the challenges to be overcome for faster CAN.

2-1 Propagation delay

There are two characteristic functions that prevent improvement in the processing performance of the CAN: ① 'bitwise' (non-destructive) arbitration, and ② an ACK (acknowledgement) field.

CAN employs an arbitration mechanism using a CSMA (Carrier Sense Multiple Access) protocol, in which a frame started by the SOF (start of frame) bit may collide with other frames transmitted from multiple nodes. When the frames collide, bit comparison is performed for every bit (synchronized by the synchronization mechanism) in the arbitration field, and a frame with higher priority (dominant bit) is authorized for transmission without being destructed by other frames with lower priority (recessive bit).

Multiple nodes transmit the result of ACK/NAK judgment regarding a reception frame to the same bit position (ACK slot), and thus the ACK mechanism realizes the judgment involving all nodes on the bus.

Though these mechanisms are important to characterize CAN, they also inhibit the improvement of the transmission rate between each node on the bus. To increase the operation speed of CAN, the aforementioned two mechanisms need to be replaced with new delay-insensitive ones.

Although there are other factors that affect CAN's operation speed, including the degradation of signal characteristics, we do not discuss them here because these problems can be solved by using a bus repeater or other general methods in combination with our suggested technique. We will explain the technique in the later section.

2-2 Guarantee of message latency

In an in-vehicle system, a message needs to be transferred from an ECU to another ECU within a given period of time. A message is sent to the bus based on the internal event of a transmission node. When frames are transmitted from several nodes at the same time, a frame with a weak ID (lower priority) is dismissed within the arbitration field. Such dismissals can be continued in the re-transmission process. Consequentially, the traditional CAN is not able to guarantee the worst-case latency for a message to be delivered.

Similar in principle to the task scheduling of operating systems, this problem can be avoided by operational efforts such as avoiding continuous transmission of messages with higher priority. However, this is not a fundamental solution.

3. Proposed Protocol

In this section, we propose the modification of protocol design to guarantee the worst-case time required for message delivery.

3-1 Approach

To minimize the influence caused by propagation delay and predict the worst-case time required for message delivery, the two changes were made to the existing CAN specifications: ① change in the ACK mechanism (message-based ACK operation), and ② change in the arbitration process (slot-based collision avoidance).

(1) Alleviation of propagation delay requirement

As the result of the two aforementioned changes, the bus arbitration and the traditional ACK mechanism are abolished. Accordingly, the requirement of propagation delay is alleviated like existing various serial communication protocols. In this process, a new ACK information field is employed instead of an ACK slot.

(2) Guarantee of the worst-case delay

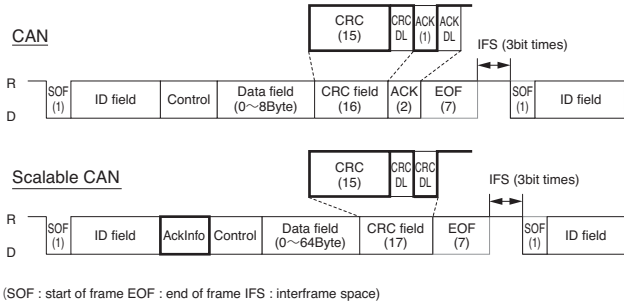
By changing the arbitration process, the bit-wise arbitration mechanism is abolished. Instead of this traditional mechanism, we employed the slot-base timing allocation method that allows only one node to access the bus at a time. This change enables the CAN to estimate the worse-case delay between applications.

We will describe the details of the suggested approach in the following section.

3-2 Frame format

Figure 1 illustrates the comparison of the frame formats between the traditional CAN and Scalable CAN.

In the traditional CAN frame format, the ACK bit plays two important roles. One is an ACK/NAK response for receiving a frame on each node. In the Scalable CAN frame, the ACK bit is replaced by the new additional field (indicated as the AckInfo field in **Fig. 1**), which stores the check results of received frames until it transmits its own frame. Another role is the creation of a dominant-edge with cyclic redundancy check (CRC) delimiters. With this dominant-edge, final bit resynchronization is performed in the frame to minimize the synchronization error occurring in the entire bus. In this study, the CRC field was modified into 17 bit width consisting of 15 bit CRC and a 2 bit CRC delimiter. The new CRC delimiter has a recessive bit and a subsequent dominant bit. Although the ACK bit is removed, the final bit resynchronization is performed by the new CRC delimiter.



(SOF : start of frame EOF : end of frame IFS : interframe space)

Fig. 1. Comparison of frame formats

3-3 Communication sequence

In this section, we explain a two-node configuration as a simple example. **Figure 2** shows a bus topology of a GW and an ECU connected to the Scalable CAN bus and an alternative communication sequence with three messages transmitted between the two nodes.

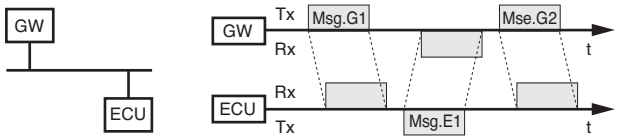


Fig. 2. Bus topology and communication sequence

(1) Basic communication sequence

We explain the basic communication sequence in which no message collision occurs.

The GW sends its message (Msg. G1) to the ECU after confirming that no frame is detected on the bus for a given period of time. The ECU receives Msg. G1 that is detected on the bus.

After sending Msg. G1, the GW waits for a response from the ECU. After receiving Msg. G1, the ECU sends the second message (Msg. E1) with ACK/NAK information of Msg. G1 stored in the AckInfo field. Response data regarding the received message and new notifiable event data can be stored in the data field, if necessary, and sent to the GW.

The GW that has been waiting for a reply message from the ECU receives Msg. E1 and measures it to judge if the first message (Msg. G1) is successfully sent or not based on the information stored in the AckInfo field.

Likewise, the GW sends the third message (Msg. G2) with the ACK/NAK information about the received second message (Msg. E1).

In this way, the alternative communication sequence with the GW and the ECU can prevent the collision of transmitted data and provides each node fairly with a transmission right. Thus, this communication scenario eliminates the possibility of losing arbitration continuously. The latency of a message, which is defined by the time required to start transmission after the transmission request has been made, is also limited. Therefore, at a specific system configuration, it is possible to guarantee the worst-case latency of a message.

(2) Waiting time for transmission

At the collision-free state, the waiting time of a transmission node is defined as the time required to start transmission after receiving the CRC field, and is represented by $t_{backoff}$.

$$t_{backoff} = \left(t_{EOF} + t_{IFS}\right)$$

(3) Waiting time for reception

This section explains the waiting time of a reception node. Once data transmission starts, collision with the next message must be avoided. In **Fig. 3**, t indicates the waiting time for the GW to receive a message from the ECU and detect the SOF (start of frame) bit after sending the CRC field.

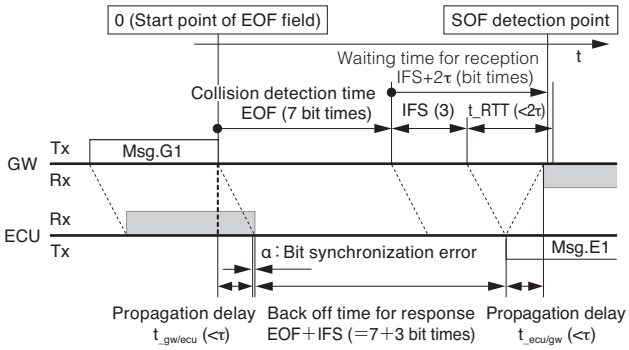


Fig. 3. Communication sequence

The time required to detect the SOF bit after GW sends the message (Msg. G1) is given as t_{response}. Here, we should note that a bit synchronization error α may be a positive or negative number.

$$t_{\text{response}} = \left(t_{EOF} + t_{IFS} + t_{gw/ecu} \pm |\alpha| + t_{ecu/gw}\right)$$

The round trip time between the GW and the ECU is indicated as t_{RTT}, and the time that the GW is waiting for the SOF bit to be detected upon the reception of the response from the ECU is given as (EOF+IFS+$t_{RTT} \pm |\alpha|$). If the condition $t_{RTT} > \alpha$ is not satisfied, there exists the possibility that the SOF bit is detected in the IFS (3 bit times) of the GW node. This is because the value of α depends on the conditions, such as baud rate, sampling resolution and clock accuracy.

For our proposed protocol, the conditional tolerance of t_{RTT} and α is defined as follows.

$$\left(t_{IFS} + t_{RTT} - \alpha\right) > 0$$

(4) Maximum allowable propagation delay

The basic idea of propagation delay occurring between the GW and the ECU has been described in (3). However, when the protocol is actually installed in a vehicle, the propagation delay time may vary depending on the location of the ECU. Moreover, in the case of a multi-node cluster, maximum allowable propagation delay (t_{RTTmax}) also needs to be considered.

Now, a common setting based on the maximum allowable propagation delay is described as τ. Needless to say, τ must be larger than the value of the maximum allowable propagation delay in the following formula.

$$\tau > \left(\frac{t_{RTT_{\max}} + |\alpha|}{2}\right)$$

(5) Collision detection algorithm

Considering (3) and (4), we define the collision detection algorithm as shown in **Table 1.**

Table 1. Judgment conditions

No.	Judgment condition	Result
①	t < 7 [bit times]	collision
②	7 [bit times] ≦ t < *rtout*	no collision
③	*rtout* ≦ t	response timeout

※ *rtout* = EOF + IFS + 2τ

(6) Waiting time for retransmission

When a collision of messages is detected by the judgment conditions shown in **Table 1**, the GW retransmits the broken message. To ensure the retransmission, different length of waiting time needs to be allocated to each node.

Retransmission with randomly allocated waiting time is a common method used to compensate collided transmission. However, there remains the possibility that collision may happen again even though the waiting time is selected at random, and therefore, this method should not be applied to automotive use.

Our proposed protocol selects different length of waiting time (t1 and t2 in **Fig. 4**) using unique IDs assigned to each node and successfully retransmits messages without a collision. The minimum unit of the waiting time should correspond with each node on the bus, and a synchronization error must be considered to decide the unit length.

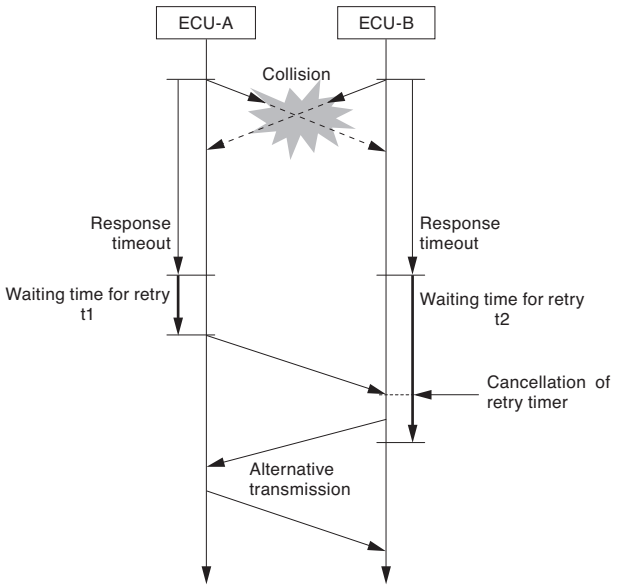


Fig. 4. Asynchronous waiting time for retransmission

4. Evaluation of Proposed Protocol

We adopted the three approaches to evaluate our proposed protocol: ① model checking, ② simulation, and ③ field programmable gate array (FPGA) based trial test.

4-1 Model checking

The behavior of our proposed protocol was analyzed by Ukai et al.[4] using a model checking tool NuSMV[7]. In this analysis, a behavior model was described in the SMV language and a bit error was taken into consideration in designing the communication channel of the model.

The analysis of counter examples resulting from the model checking gave us valuable information including defects in the specification.

4-2 Simulation analysis

Kurachi et al.[8] introduced a network simulation as shown in **Fig. 5**. The analysis revealed that even the worst-case delay time was kept under an acceptable level.

(1) Simulation environment

The simulation test of our proposed protocol was carried out using OPNET Modeler[9]. For this simulation, one multi-port gateway was connected to 20 ECUs. The overview of the simulator and simulation parameters are given in **Fig. 5**.

This test was conducted on the assumption that our proposed protocol was introduced to the existing CAN bus. The simulation model was designed to realize 1 bit resolution and the bus traffic was calculated using an actual message set (ID, data length, and transmission cycle).

(2) Simulation scenario

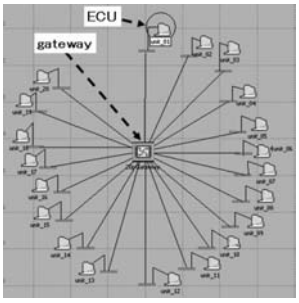
To evaluate delay time per message, the worst delay rate of each message was considered. The worst case was defined as follows: each ECU transmits a message simultaneously at simulation time 0, and all transmitted messages queue up at the gateway to be forwarded to the destination ECU.

(3) Evaluation index

Simulation results were evaluated based on the delay rate. The delay rate is described as follows.

$$\text{Delay rate} = \left(\frac{\text{message transmission time}}{\text{message transmission cycle}} \right) \times 100\%$$

(4) Simulation results



Simulation parameters of 10Mbps CAN.

Parameters	Status
Number of ECUs	20
Number of gateways	1
Size of message queue	200 messages
Data rate	10 Mbps
Simulation time	10 seconds

Fig. 5. Configuration of simulation

Table 2. Simulation results (worst value)

No.	Protocol	Baud rate	Bus traffic	Delay rate
1	Scalable CAN	10 Mbps	×1	1.75%
2	Scalable CAN	10 Mbps	×10	16.76%

The delay rate derived from the simulation was as shown in **Table 2**.

The simulation revealed that the maximum queue reached 181 messages and the worst delay rate was 1.75% at 10 Mbps and 16.76% at 10 Mbps with the bus traffic of 10 times. These results confirmed that the delay rate was kept under an acceptable level for automotive use.

4-3 FPGA based trial test

(1) Overview

Kurachi et al.[8] implemented the proposed protocol in the FPGA and trial tests were conducted at 10 Mbps. **Figure 6** illustrates the overview of the IP configuration. Synthesis report is given in **Table 3**, and the overview of the evaluation environment is shown in **Fig. 7**.

The design structure used in this implementation consists of two major FPGA components: one is Nios II proces-

Table 3. Synthesis report

	ECU	GW
Total logic elements	4,251	5,200
Total pins	81	85
Total memory bits	48,384	48,512

(including 949 [LE] for Scalable CAN Controller)

sor, which serves as a soft core microprocessor, and another is the Scalable CAN Controller, which is the peripheral control unit of the microprocessor to realize our proposed protocol at 10 Mbps. The Scalable CAN Controller consists mainly of a controller core, ACK manager, mailbox, and host IF as shown in Fig.6. The main feature of the controller core is the encode/decode function for frames. The implementation result revealed that the difference between the conventional CAN controller and the Scalable CAN Controller is very small except for the detailed protocol state for ACK management.

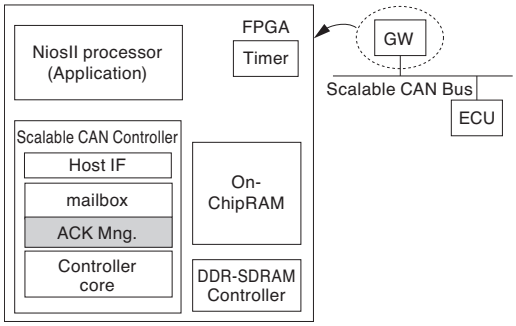


Fig. 6. IP configuration in FPGA

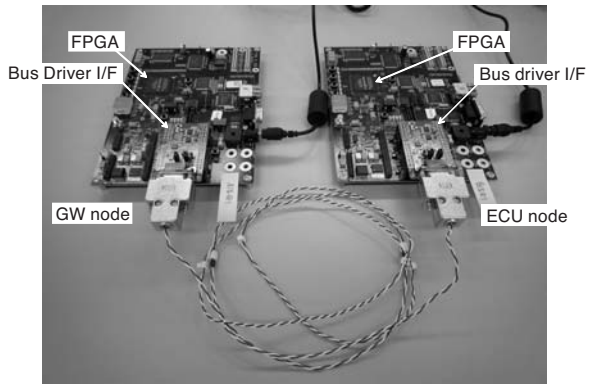


Fig. 7. Overview of evaluation environment

(2) Implementation results

Two analyses were conducted to evaluate the implementation of this protocol.

First, compatibility of communication software was examined by comparing control registers. Although parameter settings were different, Scalable CAN and the traditional CAN performed the same. Thus we confirmed the compatibility after changing the initialization process.

Second, performance of end-to-end communication was evaluated by using an AUTOSAR[10] COM stack and a demonstration application implemented on the stack. The stack consists of a CAN driver, CAN interface, COM, and PDU router (zero cost operation). The result confirmed that the proposed protocol is capable of communicating at approximately 7 Mbps under the 10 Mbps setting.

5. Future Challenge

The future challenge for our proposed protocol is to confirm the feasibility of the bus topology network to which several nodes are connected. For this purpose, further research and development is needed in terms of the modification of frame formats, the definition of transmission-order management, and the ACK/NAK management method for multi-node communication.

6. Conclusion

In this paper, we proposed a new automotive protocol based on the exiting CAN protocol. Our experimental results demonstrated that the proposed protocol is valid for reducing the delay time in message transmission between applications. The FPGA test also confirmed the transmission capability of approximately 7 Mbps. Furthermore, it was confirmed that model checking and network simulation can be effectively used for the quality improvement of the protocol specifications.

We will conduct further specification tests and simulations for the proposed protocol in the operating conditions assuming real in-vehicle environment.

This study was conducted as a joint project of Nagoya University and AutoNetworks Technologies, Ltd.

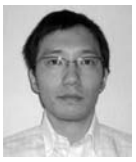
References

(1) International Organization for Standardization, Road vehicles. Controller area network (CAN). Part1: Data link layer and physical signaling, ISO IS11898-1, (2003).
(2) LIN Consortium, http://www.lin-subbus.de/
(3) FlexRay Consortium, http://www.flexray.com/
(4) Kenji Ukai, Toshiki Sakabe, Hiroaki Takada, Ryo Kurachi, Masahiko Sakai, Keiichirou Kusakari, Naoki Nishida, Behavior Analysis of Scalable CAN Protocol on a Bit-Error Channel, Tech. Rep. of IEICE (SS2008-37), Vol.108, No.242, pp.61-66, Oct 2008.
(5) Huth M. and Ryan M: Logic in Computer Science: Modeling and Reasoning about Systems, Cambridge University Press, 2nd ed., 2004, Chapter 3, pp.172–255.
(6) Clarke E.M., Grumberg O and Peled D: Model Checking, MIT Press, 2000.
(7) NuSMV home page, http://nusmv.irst.itc.it/

(8) Ryo Kurachi, Hiroaki Takada, Shigeharu Teshima and Yukihiro Miyashita, Design and Performance Analysis of 10 Mbps CAN, IPSJ Journal, Vol.50, No.11, pp.1234–1245, Nov.2009.
(9) OPNET Modeler, http://www.opnet.com/
(10) AUTOSAR, http://www.autosar.org/

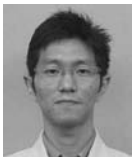
Contributors (The lead author is indicated by an asterisk (*)).

R. KURACHI*



- Researcher of Center for Embedded Computing Systems, Nagoya University. He is engaged in the development and design of in-vehicle network systems.

M. NISHIMURA*



- LAN R&D Department, Network R&D Division, AutoNetworks Technologies, Ltd. He is engaged in the development and design of in-vehicle network systems.

H. TAKADA

- Ph.D.
  Executive Director of Center for Embedded Computing Systems, Nagoya University.
  Professor, Dept. of Information Engineering, Nagoya University.

S. TESHIMA

- Ph.D.
  Director of Center for Embedded Computing Systems, Nagoya University.

H. MIYASHITA

- LAN R&D Department, Network R&D Division, Auto Networks Technologies, Ltd.

S. HORIHATA

- Manager, LAN R&D Department, Network R&D Division, AutoNetworks Technologies, Ltd.

H. YAMAMOTO

- Senior Manager, LAN R&D Department, Network R&D Division, AutoNetworks Technologies, Ltd.

A. NATSUME

- General Manager, Network R&D Division, AutoNetworks Technologies, Ltd.