Proceeding of the 2015 IEEE
International Conference on Information and Automation
Lijiang, China, August 2015

A Parallel Re-programming Method for In-vehicle Gateway to save software update time

Young Seo Lee, Jin Ho Kim, Hoang Van Hung and Jae Wook Jeon

Sungkyunkwan University
The School of Information and communication Engineering
Suwon, Korea

lys673@naver.com, besteng83@gmail.com, hunghv80@hotmail.com, jwjeon@yurim.skku.ac.kr

***Abstract* - In this paper, we propose a parallel re-programming method that can reduce the time to complete software updates of in-vehicle electronic control units. The proposed method has been designed to operate in an in-vehicle gateway that integrates different types of in-vehicle networks including controller area network, FlexRay, and Ethernet to provide seamless communication between heterogeneous networks. The re-programming method provides parallel processing for flash update data that has been received in order to enable a simultaneous software update of multiple electronic control units. We implemented the proposed method in an in-vehicle gateway on an embedded system and designed an experimental environment to evaluate its performance. We also measured and analyzed the performance of the parallel re-programming gateway by conducting an experiment based on the experimental environment.**

Index Terms* - *ECU re-programming; In-vehicle software update; In-vehicle gateway;

I. INTRODUCTION

Since 1970, the number of in-vehicle electronic control units (ECU) used has gradually increased and these days, a luxury car uses more than 70 to 80 ECUs [1-3]. In-vehicle ECUs provide electronic control functions that assist the driver in controlling the vehicle, and these have gradually replaced purely mechanical units [1]. Unfortunately, the increase in the number of ECUs that are used and the growth in the size of the application software resulted in a corresponding increase in the software re-programming time, which leads to economic disadvantages.

The software for the in-vehicle ECUs can be updated during development, manufacturing, and after sales to add new features and to fix bugs. The software is reprogrammed through a network connection using a diagnostic protocol between an external diagnostic device and a vehicle via an on board diagnostics (OBD) connector. As a result, the performance of the ECU re-programming depends on the network configuration and on the diagnostic communication protocols that are used, and enhancement in the diagnostic communication would improve if this performance. However, conventional diagnostic networks are based on the Controller Area Network (CAN) protocol, which has a limited ability to reduce the re-programming time because this type of network offers insufficient bandwidth and maximum data size to handle a large amount of code required of high-performance application software for a large number of ECUs in modern vehicles [3].

To address these bandwidth limitations, the automotive industry has implemented Ethernet in the diagnosis network domain, and various studies related to automotive Ethernet have been carried out. According to a study conducted by BMW in 2008, Ethernet-based diagnostic protocol technologies can be applied with CAN-based diagnostic protocols, it can support simultaneous in-vehicle ECU re-programming and thus reduce the required time for software re-programming [3-5]. When Ethernet is used, external diagnostic devices cannot interface directly with in-vehicle networks (IVN) due to the use of different networking protocols, and therefore, an in-vehicle gateway system is essential part.

In-vehicle gateways have been introduced to use heterogeneous networks with various communication protocols in a vehicle, and various ongoing efforts to develop technologies related to in-vehicle gateways are being carried out [7-12]. A gateway provides seamless communication between the IVNs (CAN, FlexRay and Ethernet) and it has a significant effect on the total re-programming performance of an automotive system with heterogeneous networks. Therefore, the time to update all ECUs depends on the data processing method that the gateway uses. In [12], it is described with respect to the parallel re-programming for reduction in software updating time but it does not provide a description of the detailed process and method for parallel re-programming. Thus, the investigation of the reduction in the reprogramming time has become important to achieve economic benefits. However, previous studies have not provided sufficient information related to re-programming of in-vehicle ECUs.

This paper proposes a parallel re-programming method for multiple ECUs on different IVNs into an in vehicle gateway. This the method includes a routing method and a parallel processing method for a large amount of data, and these are implemented at the gateway system. This method can be used to reduce the total amount of re-programming time that is required by processing the data in parallel. We implemented the gateway using the embedded system with a heterogeneous networks, and we also built an experimental environment that consists of ECUs and IVN systems. We measured the performance with which the ECUs were simultaneously re-programmed through parallel re-programming, we then analyzed the results to evaluate the overall re-programming performance of the proposed method.

The remainder of this paper is organized as follows. Section II discusses the background technologies. Section III describes the parallel re-programming method that is used for each different type of network through the in-vehicle gateway. Section IV describes the implementation of the proposed gateway system for parallel re-programming. Section V presents the experimental results for the parallel re-programming of multiple ECUs. Finally, Section VI concludes this paper.

II. Background

This section describes the general background of IVNs, including traditional IVNs (CAN and FlexRay), automotive Ethernet technologies, and an in-vehicle gateway that integrates different types of IVNs.

A. In-vehicle network

CAN is the dominant network protocol that is used in current IVNs due to its low cost, ease of installation and immunity to noise. Therefore, CAN efficiently supports distributed control. However, automotive systems requires higher-bandwidth than conventional CAN networks can provide, and CAN is not suitable for safety-critical real-time systems, such as x-by-wire systems [1, 13]. FlexRay was introduced to improve on the insufficient bandwidth provided by CAN and to ensure reliability in data communications. FlexRay implements time-deterministic and fault-tolerant communication to provide safety-critical distributed control systems with a bandwidth of 10 Mbps and a maximum data size of 254 bytes. FlexRay operates with time-triggered segments (static segment) based on time division multiple access (TDMA) and an event-triggered segments (dynamic segment) based on flexible TDMA (FTDMA). However, it cannot provide sufficient bandwidth for infotainment and multimedia systems in a modern car. As a result, Ethernet is expected to become the dominant communication protocol for IVNs by 2020 because it can meet the bandwidth requirements of modern automotive systems [6-9]. Ethernet provides several benefits, including a low cost, high bandwidth, large data size and ease of installation. As a result of these advantages, BMW has already implemented Ethernet-based networks for use with in vehicle diagnostics and software updates, and BMW is actively developing Ethernet technologies for use with camera applications and ADAS systems [3-5, 15-16]. Ethernet-based diagnostic technologies have been introduced into automotive systems to solve the problem of using CAN-based diagnostic protocols, which includes the inability to provide the required network capacity due the small data size and low bandwidth, according to the extended diagnostic information. Diagnostics over Internet Protocol (DoIP) is an Ethernet-based diagnostic protocol that was standardized with ISO 13400 to provide a globally standardized interface to develop Ethernet-based vehicle diagnostics communications.

B. In-vehicle gateway

Previously, in-vehicle gateways did not need to control automotive systems because network configurations were simple and ECUs only had simple roles. However, automotive systems have become more complex, so IVNs have begun to require a higher bandwidth because automotive systems now use more than 70-80 ECUs and communicate more than 2,500 signals to provide high-performance applications, including infotainment systems, and ADAS [1, 3]. As a result, automotive systems now require changes in their IVNs to provide improved functionality, such as an increase in bandwidth, as well as to guarantee reliability and stability in the respective functional domains, such as the chassis, body, powertrain or multimedia application [3]. To fulfill these requirements for IVNs, heterogeneous networks that include CAN, FlexRay, and Ethernet, have been applied to automotive systems, and since these applications require different types of networks, the automotive industry has conducted research on in-vehicle gateways to provide seamless communication between different networks [7-12], Figure 1 shows a network architecture based on the use of a central gateway [5]. The use of an automotive gateway allow for automotive systems to use optimum network protocols corresponding to the respective functional domains, and these gateway technologies allow automotive systems to benefit from improvements in application performance.

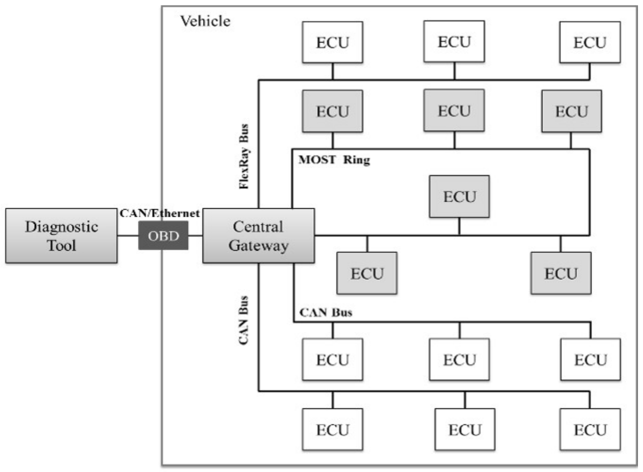


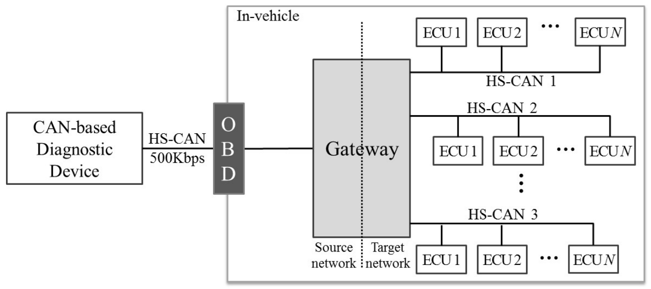
Fig. 1 Central gateway-based network Architecture

III. Parallel re-programming method

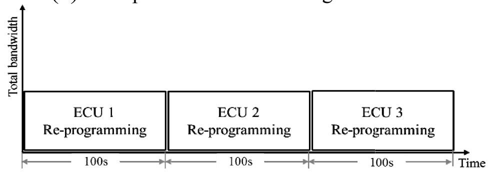
In this Section, we describe the parallel re-programming method for an in-vehicle gateway system. We also describe the concept of parallel re-programming and discuss the routing method and the parallel processing method of the gateway.

A. Concept for parallel re-programming in different networks

Existing software re-programming methods for ECUs using CAN-based diagnostic protocols can be configured as in Figure 2(a). The diagnostic device and the gateway are connected using a CAN network (source network), and the gateway and the ECUs are connected using different CAN networks (destination network). However, re-programming these existing network system in parallels is limited because the bandwidth is the same for the source network and for the target network. Therefore, ECUs can only be updated sequentially, one by one. This sequential processing, extends the total time for the re-programming procedure when the



(a) Simplified Network Configuration



(b) Time of Re-programming data transfer

Fig. 2 Conventional in-vehicle ECU re-programming concept

number of ECUs that require a software update increases, as shown in Figure 2(b).

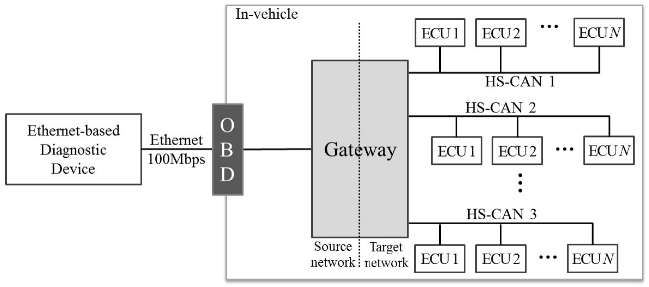
The total re-programming time can be reduced by improving the bandwidth of the network that is used to connect the diagnostic devices and the gateway. These bandwidth requirements for parallel re-programming are defined with Equation (1) as follows.

$$B_S \geq B_{T1} + B_{T2} + \cdots + B_{TN} \tag{1}$$

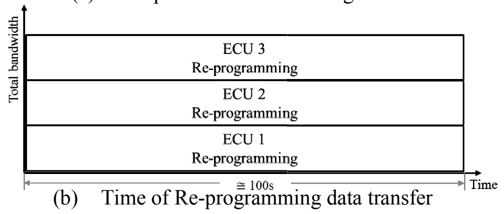
$$B_S \geq \sum_{i=1}^{N} B_{Ti}$$

For example, the sum of all the bandwidth ($B_{T1} + B_{T2} + \cdots + B_{TN}$) of the target networks connected to the gateway and the different ECUs is N Mbps. If the bandwidth (B_S) between the diagnostic device and the gateway is more than N Mbps, the performance of the gateway is sufficient to process the re-programming data, and the proposed parallel re-programming method can be applied. That is, the gateway receives the re-programming data for multiple ECUs from the diagnostic device at a high rate, and then the gateway decentralizes and transmits the re-programming data to each destination network at a lower rate in parallel. As a result, the gateway can achieve parallelism by carrying out a distributed transmission of the re-programming data for each ECU.

In order to meet the bandwidth requirements for parallel re-programming, Ethernet-based diagnostic protocol improves the bandwidth usage between the gateway and the diagnostic device, and provides the basis for efficient parallel processing. When Ethernet-based parallel re-programming is applied, the re-programming time for a single ECU is similar to that of a CAN-based re-programming method because the network bandwidth between the gateway and the ECUs remains the same. In contrast, when the number of target ECUs increase, the re-programming time for multiple ECUs on different networks can be reduced via using parallel processing. The proposed parallel re-programming method uses an Ethernet-based diagnostic protocol, as shown in Figure 3.



(a) Simplified Network Configuration



(b) Time of Re-programming data transfer

Fig. 3 Ethernet-based in-vehicle ECU re-programming concept

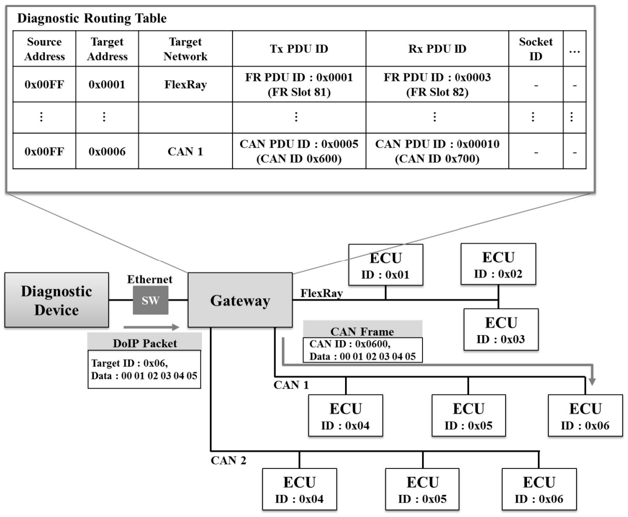
B. Diagnostics and re-programming message routing

The diagnostic device transmits the diagnostics and the re-programming request messages by using the user data field of DoIP frame with the format presented in [6], and the diagnostic frames for CAN and FlexRay are described in [12]. The diagnostic device can transmit request messages to the ECUs by using a unique address that is assigned to each of the ECUs, and the address value of the re-programming target ECU is packed into the target address field of the DoIP frame. When the gateway receives a DoIP frame that packs the re-programming data, the gateway extracts the target address value from the DoIP frame and confirms the information of the target ECU that is connected to the target network by searching for the diagnostic routing table based on the target address. The diagnostic routing table is included in the gateway, and the general routing table to route control data that is separately configured. The re- programming data is routed based on a protocol data unit (PDU), it contains attributes of the network protocol types, and can be used to route to the destination network interface. PDU-based routing is described in further detail in [12]. Figure 4 shows an example of the diagnostic routing table and of the routing operation for diagnostic request messages where each parameter of the routing table is used for the specific purposes, as follows:

1) *Source address*: the logical address of the transmitter (e.g., the diagnostic device address) for the diagnostics and the re-programming messages.

2) *Target address*: a logical address for in-vehicle ECUs that indicates the target ECU address for diagnostics and re-programming services. The gateway can extract the routing information from the diagnostic routing table by reading the target address field that is included in the DoIP frame that is received.

3) *Transmit PDU ID*: can be found by referencing the target address. It is necessary to route the re-programming request data to the destination network interface (e.g., CAN/FlexRay network interface) from the source network interface (e.g., Ethernet network interface).


Diagnostic Routing Table

Source Address	Target Address	Target Network	Tx PDU ID	Rx PDU ID	Socket ID	…
0x00FF	0x0001	FlexRay	FR PDU ID : 0x0001 (FR Slot 81)	FR PDU ID : 0x0003 (FR Slot 82)	-	-
⋮	⋮		⋮	⋮	⋮	⋮
0x00FF	0x0006	CAN 1	CAN PDU ID : 0x0005 (CAN ID 0x600)	CAN PDU ID : 0x00010 (CAN ID 0x700)	-	-

Fig. 4 Diagnostic routing table structure and message routing description

4) Receive PDU ID: used to search for the corresponding socket ID. It is necessary to route the re-programming response data that has been received to the destination network interface (e.g., Ethernet network interface) from the source network interface (e.g., CAN/FlexRay network interface).

5) Socket ID: dynamically assigned according to the socket connection that is used to exchange the TCP-based DoIP frame between the diagnostic device and the gateway. A socket number is stored in the socket ID parameter to map the target address value in the diagnostic routing table. These socket IDs distinguish the socket information that dynamically changes depending on the socket connection and disconnection.

C. Parallel processing of the re-programming data

When the gateway transmits the re-programming data for multiple ECUs to different target networks, it transmits the frame to one-network channel and then to another network channel during a time interval until the next frame is sent to one-network. This transmission process is repeated until all of the re-programming data has been transmitted to multiple ECUs. Therefore, the data transfer can be parallelized for the re-programming process. As a result, the gateway allows to simultaneously transmit re-programming data for multiple ECUs in a similar amount of time as that which would be necessary to transmit data to a single ECU.

The process that is used to transmit re-programming data differs according to the communication protocol that is used by the target network, which can be either CAN or FlexRay. Figure 5 shows the data transfer process according to the type of target network. If the target network is a CAN network, the re-programming data is processed and transmitted through a main re-programming task. The main re-programming task checks whether or not the buffer has been updated by sequentially reading the buffer update flag, and it routes the re-programming data that is contained in the buffers to the destination CAN-based networks immediately, one segmented frame at a time [the transmission of massive re-programming data is segmented by the CAN transport protocol (TP)].

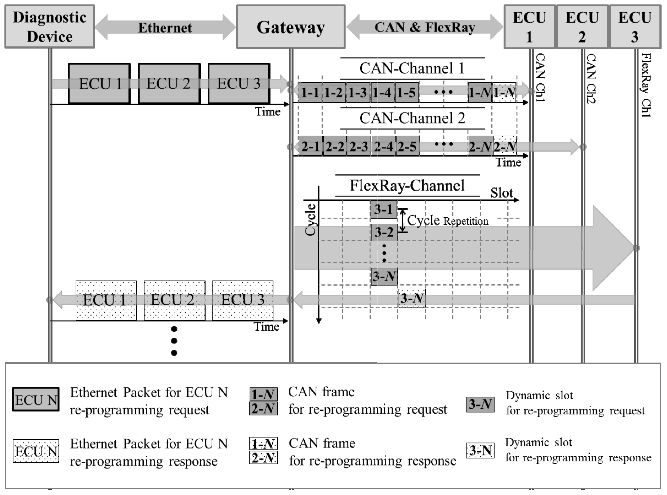

Diagnostic Device
Ethernet
Gateway
CAN & FlexRay
ECU 1
ECU 2
ECU 3
CAN Ch1
CAN Ch2
FlexRay Ch1
ECU 1
ECU 2
ECU 3
Time
CAN-Channel 1
1-1 1-2 1-3 1-4 1-5 … 1-N 1-N
CAN-Channel 2
2-1 2-2 2-3 2-4 2-5 … 2-N 2-N
Time
FlexRay-Channel
Slot
Cycle
3-1
Cycle Repetition
3-2
3-N
ECU 1
ECU 2
ECU 3
3-N
Time
ECU N
Ethernet Packet for ECU N re-programming request
1-N 2-N
CAN frame for re-programming request
3-N
Dynamic slot for re-programming request
ECU N
Ethernet Packet for ECU N re-programming response
1-N 2-N
CAN frame for re-programming response
3-N
Dynamic slot for re-programming response

Fig. 5 Description of the data transmission according to target networks

On the other hand, if the target network is a FlexRay network, the re-programming data cannot be processed directly or transmitted due to the cyclic communication of the TDMA-based system. Therefore, FlexRay diagnostics and re-programming are determined in advance through scheduling, and such messages are generally transmitted through a dynamic slot. A FlexRay task periodically checks the state of the communication cycle by performing a periodic function call. If the turn to transmit a dynamic slot for re-programming comes around, the re-programming data is copied to the dynamic slot and the massive re-programming data is transmitted (which is segmented by the FlexRay TP). Finally, the re-programming data in the dynamic slot is transmitted to the FlexRay network according to the scheduling of the time cycle that has been configured in advance.

IV. Implementation of Parallel Re-programming

In this section, we describe both the in-vehicle gateway system that has been implemented to support parallel re-programming and the experimental environment that has been developed to measure and evaluate a performance of the parallel re-programming method.

A. Hardware for the parallel re-programming gateway

The gateway system was implemented using the Freescale MPC5668G MCU-based evaluation board (EVB). The gateway system supports various communication protocols, including CAN, FlexRay, and Ethernet. It consists of a main micro controller unit (MCU), a CAN transceiver, a FlexRay transceiver, and an Ethernet transceiver. Therefore, the main MCU is an MPC5668G (32bit-MCU based on the PowerPC architecture) with CAN, FlexRay communications controllers, and a fast Ethernet controller for single-channel Ethernet support. Further details of MPC5668G are presented in [17].

B. Software for the parallel re-programming gateway

The software configuration of the overall parallel re-programming gateway was implemented using the AUTOSAR microcontroller abstraction layer (MCAL) 3.0 and AUTOSAR OS, and the overall software architecture is shown in Figure 6. Tresos Studio 2010.a is developed by Elektrobit, which is used

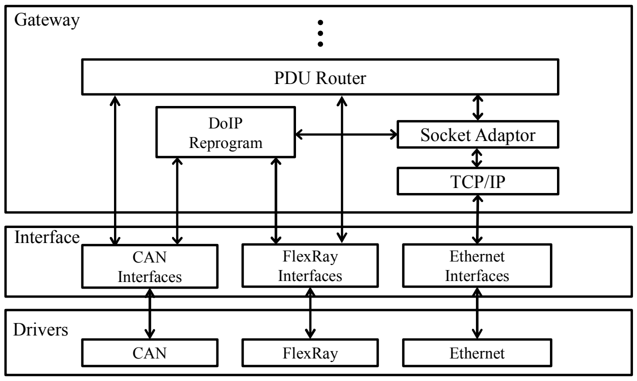


Fig. 6 Software architecture of the parallel-reprogramming gateway

to configure the communication driver and the OS. However, the Ethernet communication driver was developed to contain only the essential elements because it is not provided in AUTOSAR 3.0. A TCP/IP and a Socket Adaptor stack were implemented using a Light Weight IP (LWIP) that is a dedicated embedded TCP/IP stack. In addition, the DoIP and the re-programming module were developed to provide diagnostics and re-programming services, and the DoIP module can manage an active connection between the diagnostic device and the gateway, and it can also route diagnostic messages between the source network interface and the destination network interface. The re-programming module is then called by the DoIP module to process the re-programming data through a TP layer.

C. Experimental environment for parallel re-programming

The experimental environment that was set up to evaluate the parallel re-programming method is shown in Figure 7, and it simply presents an essential network environment to experiment with re-programming. The implemented gateway is connected to CAN-based target ECUs 2 and 3 and to the FlexRay-based target ECU 1 through the corresponding CAN or FlexRay bus, and it is able to extend the CAN network channel to four- HS-CAN and two-LS-CAN. The gateway and diagnostic devices are connected to an Ethernet switch in order to exchange the re-programming data.

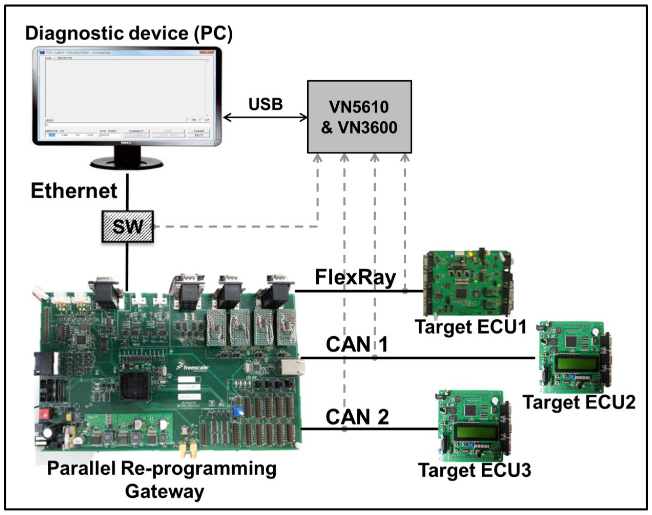


Fig. 7 Experimental environment for parallel re-programming test

TABLE I
THE MEASURED PARALLEL RE-PROGRAMMING TIME

	Re-programming time	
Single channel Re-programming	24 s (CAN 1-channel)	22 s (FlexRay 1-channel)
Two channel parallel re-programming	25 s (CAN 2-channel)	
Three channel parallel re-programming	27 s (CAN 2-channel, FlexRay 1-channel)	

The diagnostic device that was used to transfer the re-programming data was implemented using a GUI-based PC application that opens binary files for multiple ECUs. This application reads the re-programming data and packs the data in the user data field of a DoIP frame that is transmitted to the gateway through Ethernet packets. CANoe is connected to each network through VN3600/5610 interfaces to measure and verify the re-programming performance of the CAN, FlexRay, and Ethernet networks. The VN3600/5610 are a universal serial bus (USB) to IVN interface module developed by Vector, and we use it to measure the re-reprogramming data processing time for the gateway and the total re-programming time for multiple ECUs as well as to verify that the gateway executed the parallel re-programming method.

V. EXPERIMENT AND RESULTS

In this section, we describe the experimental results of the proposed parallel re-programming gateway system. The binary data that is used to re- program each of the target ECUs consists of about 617 Kbyte per the ECU. The binary data is segmented into DoIP frames, and the data is then transmitted to the gateway by the diagnostic device via TCP. Thus, a total of about 1.851 Mbyte of binary data are transmitted to the gateway according to their assignment in the corresponding DoIP frame. The gateway forwards the re-programming data in the DoIP frame that is received to the target ECUs connected to the different networks using TP. We thus measured the total re-programming time for target ECUs.

The result of the parallel re-programming that were obtained through the experiment environment are shown in TABLE I. In the case of a single ECU re-programming on a CAN-based target network, the re-programming time took 24 seconds. In the case of single channel re-programming on a FlexRay-based target network, the re- programming time took 22 seconds.

On the other hand, the parallel re-programming time for two ECUs on different CAN-based target networks, took 25 seconds. This process exhibits a 1 second delay relative to the re-programming time for a single ECU over a single CAN network, and the time was reduced by 48% compared to sequential re-programming of two ECUs, one by one, which took 48 s. The parallel re-programming time of three ECU on three different network takes 27 seconds, which is 61% less than the sequential re-programming time for three ECUs, on by one, of 70 s.

The time to process the re-programming data in the gateway is shown in Figure 8. The processing time for the gateway is measured using CANoe-based the experimental

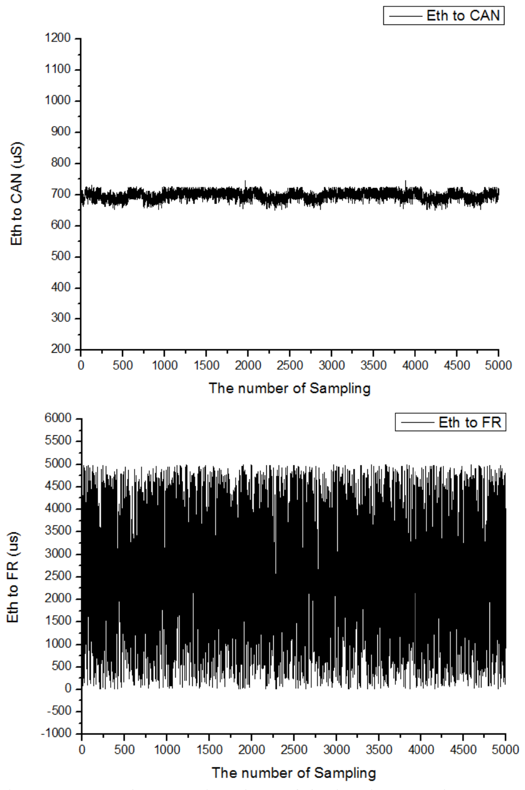


Fig. 8 Measured processing time of the implemented gateway

environment, as shown in Figure 7. The definition for the processing time (T_{proc}) is as follows.

$$T_{proc} = T_{sd} - T_{ss} \quad (2)$$

T_{sd} is the timestamp of when CANoe receives the converted/segmented first re-programming message on the target networks, and T_{ss} is the timestamp of when CANoe receives the re-programming message based-on the DoIP frame on the Ethernet network. As a result, the processing time is the difference between T_{sd} and T_{ss}, which is greater than 0.60 ms and less than 0.75 ms between Ethernet-CAN, and greater than zero less than 5 ms between Ethernet-FlexRay. The Ethernet-FlexRay processing time is affected by the FlexRay scheduling of the time cycle, and the re-programming data received over Ethernet is transmitted to the FlexRay network after waiting for the scheduling of communication cycle.

VI. Conclusion

In the future, the number of ECUs in automotive systems will gradually increase to further develop high-performance multimedia, infotainment, stability, and reliability functionality. Furthermore, the size of the application software will increase proportionally to the increase in the number of ECUs, and as a result, the software re-programming time for IVN ECUs will also increase. Therefore, a re-programming technologies that reduces the time required will be an important issue to be studied, and this paper proposed a parallel re-programming method that reduces such time required to re-program systems that use heterogeneous networks with a gateway. We implemented a gateway that support parallel re-programming by using an embedded system, developed the experimental environment to test the parallel re-programming of multiple ECUs. We measured the total re-programming time and the performance of the parallel re-programming gateway through these experiments, and we also used CANoe to analyze the experimental results of the parallel re-programming and to present the results for the parallel re-programming. In further work, we will optimize the re-programming processing method in the central gateway and will analyze the gateway requirements to re-program systems that use a next-generation IVN architecture (Ethernet-backbone-based architecture).

Acknowledgment

This research was supported by Basic Science Research Program through NRF of Korea, funded by MOE (NRF-2010-0020210).

References

[1] N. Navet, Y. Song, F. SIMONOT-LION, C. WILWERT, “Trends in Automotive communication Systems,” Proceedings of the IEEE, vol. 93, No.6, June 2005, pp. 1204-1223.

[2] M. Broy, I. Krüger, M. Meisinger, “Automotive Software Connected Services in Mobile Networks,” First Automotive Software Workshop, ASWSD 2004.

[3] L.-L. Bello, “The case for Ethernet in Automotive Communication,” RTN2011, Dec 2011, pp. 7-15.

[4] R, Bruckmeier, “Ethernet for Automotive Applications,” Freescale Technology Forum, Orlando June 2010.

[5] T. Thomsen, G. Drenkhahn, “Ethernet for AUTOSAR,” 1st AUTOSAR Open Conference & 8th Premium Member Conference, Detroit, USA, Oct 2008.

[6] ISO 13400, “Diagnostic communication over Internet Protocol (DoIP),”

[7] M. Postolache, G. Neamtu, S. D. Trofin, “CAN-Ethernet Gateway for Automotive Applications,” Control and Computing (ICSTCC), Oct 2013, pp. 422-427.

[8] A. Kern, D. Reinhard, T. Streichert, J. Teich, “Gateway Strategies for Embedding of Automotive CAN-Frames into Ethernet-Packets and Vice Versa,” ARCS 2011, Feb 2011, pp. 259-270.

[9] H. Zinner, J. Seitz, T. Waas, “Application and Realization of Gateways between conventional Automotive and IP/Ethernet-based Networks,” DAC, 2011 48th ACM/EDAC/IEEE, June 2011, pp. 1-6.

[10] Z Rui, Q. Gui-he and L. Jia-qiao, “Gateway System for CAN and FlexRay in Automotive ECU Networks,” ICINA2010, Oct 2010, pp. 49-53.

[11] Freescale, Next generation Automotive Gateways, July 2009, [Online]. Available :https://www.freescale.com/files/training_pdf/VFTF09_AA130.pdf

[12] J.-H. Kim, S.-H, Seo, T. Nguyen, B.-M, Cheon, Y.-S. Lee and J.-W. Jeon, “Gateway Framework for In-Vehicle Networks based on CAN, FlexRay and Ethernet,” IEEE Transactions on Vehicular Technology.

[13] D. Paret, “Multiplexed Networks for Embedded System – CAN, LIN, FlexRay, Safe-by-wire,” 2005.

[14] Freescale, “FlexRay – A communications Networks for Automotive Control System,” WFCS 2006.

[15] Toulouse, “Is Ethernet the rising star for in-vehicle networks,” ETFA2011, Sep 23, 2012.

[16] H.-T. Lim, L. Völker and D. Herrscher, “Challenges in a Future IP/Ethernet-based In-Car Network for Real-Time Applications,” DAC, 2011 48th ACM/EDAC/IEEE, June 2011, pp. 7-12.

[17] Freescale, “MPC5668EVB User’s Manual,” May 2009, [Online]. Available : http://cache.freescale.com/files/32bit/doc/user_guide/MPC5668GKITUM.pdf