A Frame Packing Mechanism Using PDO Communication Service within CANopen

Minkoo Kang† and Kiejin Park††

Division of Industrial & Information Systems Engineering, Ajou University, Suwon, Gyeonggi-do, South Korea

Summary
The utilization of the CANopen-based network needs to be minimized in order to improve the communication performance such as worst-case response time of CANopen messages. To achieve this end, the messages should be packed together as many as possible so that the message frame overhead will be decreased. This paper suggests a frame packing mechanism using an object dictionary (OD) and a process data object (PDO) communication service within CANopen. The performance of this mechanism is evaluated using SAE benchmark data, and from this it is seen that the network utilization of CANopen decreased by about 10% in comparison to the result of the previous research, called 'piggyback' mechanism. Furthermore, the worst-case response times of the CANopen messages are similar to or less than the previous research.

Key words:
CANopen, Controller Area Network (CAN), Network Utilization, Process Data Object (PDO), Worst-Case Response Time (WCRT)

1. Introduction

To support development of an embedded control system such as factory automation and automotive electronics at the application layer with data link layer and physical layer that are used in controller area network (CAN, refer to Fig. 1) [1], [2], several new protocols were suggested such as CANopen [3], CAN application layer (CAL) [4], and DeviceNet [5]. Especially, CANopen protocol is developed to provide a solution for the hard-ware dependency problems that occur during the implementation of CAN-based applications. CANopen protocol allows different types of devices such as electronic control units (ECUs), sensors, and actuators, all using the CAN and CAL services, to communicate and to interoperate with each other using a concept called profiling, which leads to a reduction in the development time of CAN-based applications. The Object Dictionary (OD), the Service Data Object (SDO) and the Process Data Object (PDO) were defined for the profiling concept in CANopen [6], [7], as is seen in Fig. 1.
Both CANopen and CAN are event-triggered protocols, which provide a high degree of flexibility in their communication in comparison to time-triggered protocols [8], but they have the disadvantage of having a nondeterministic response time for their messages [9], [10]. In order to resolve this drawback, there has been previous research where the worst-case response time of the CAN message is calculated [11], [12]. These researches have been applied to most of studies on the timing analysis of the CAN-based application system.

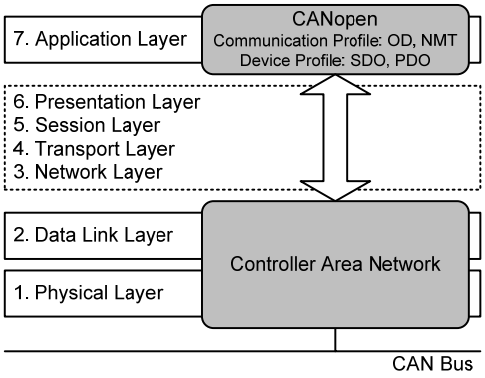

Fig. 1 CAN and CANopen protocols in the ISO/OSI reference model.

In order to improve the network utilization of the CAN bus and to reduce the worst-case response time of the CAN message, the transmission time of CAN message should be minimized, which in turn minimizes the overhead for transmission of CAN messages. The overhead, as used here, refers to additional bits from the bit stuffing mechanism and all those parts of the CAN message frame except for the data field, e.g. the arbitration field, control field, etc (see Fig. 2).

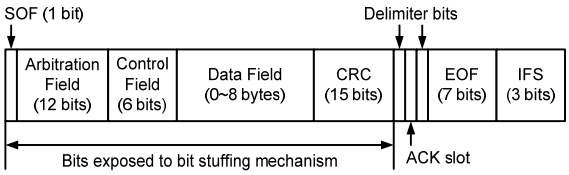

Fig. 2 Standard format of CAN message frame.

Nolte et al. [13] proposed a mechanism to minimize the number of stuffing bits of the CAN message frame by performing XOR operations on the bit mask, such as

Manuscript received August 5, 2012
Manuscript revised August 20, 2012

'010101…' and on the CAN message frame before the bit stuffing. However, the problem of priority inversion occurs in the case of the XOR operations on the bit mask and on the arbitration field in the CAN message frame. In our previous work [14], advanced bit stuffing (ABS) has been presented, which prevents this priority inversion and minimizes the number of stuffing bits. However, this mechanism only operates with special CAN controllers that support the ABS mechanism.

Another way to minimize the overhead of the CAN message frame is to pack as many signal data as possible into one frame, commonly called frame packing. Several studies of frame packing problems have been proposed for CAN communication systems. Sandström et al. [15] suggested frame packing heuristics for CAN network, but the schedulability of the solution is not considered. Bandwidth-Best-Fit decreasing (BBFd) and Semi-Exhausive (SE) heuristics have been proposed in [16]. However, SE heuristic cannot be applied realistic problem size.

Packing the CAN messages directly on to the data link layer is inefficient because the receiver nodes must update their ID list of the messages to be received and an additional mechanism for splitting the received data is needed. Alternatively, the data to be transmitted within CANopen can be packed into the PDO efficiently by using the operations of inserting, deleting, and modifying the text in the OD. As another advantage, the PDO can also automatically be converted into the CAN message frame on the data link layer [7].

This paper presents a frame packing mechanism that uses the OD and the PDO communication service within CANopen. Section II describes the several communication services in CANopen. Next, the PDO packing mechanism is presented with the system model in Section III. In Section IV, the performance of the mechanism is analyzed, and finally, Section V presents the conclusion.

2. The Communication Services in CANopen

The OD is similar to a lookup table that contains all of the parameters for the network configuration and its operation. As can be shown in Table 1, the 32-bit index range is divided up into sections that structure the OD and these OD entries in each section are stored in the CANopen nodes as an electronically readable file format, namely an electronic data sheet (EDS) or a device configuration file (DCF). The index range from 0001h to 0FFFh is used to define the data types, and the range from 1000h to 1FFFh contains the parameters associated with the CANopen communication. The index range from 2000h to 5FFFh is used for the storage of data for a particular device type that exists outside of any CANopen standard.

Table 1: The Object Dictionary Index Range

Index Range	Description
0000h	Reserved
0001h – 0FFFh	Data Types
1000h – 1FFFh	Communication Entries
2000h – 5FFFh	Manufacturer Specific
6000h – 9FFFh	Device Profile Parameters
A000h – FFFFh	Reserved

The SDO allows for read/write access to all of the OD entries of all the ECUs that are connected to the network, so it is useful for the network initialization. However, the SDO has many communication overhead fields, making the exchange of real process data inefficient. Thus, the PDO is defined in CANopen in order to minimize the communication overhead and to allow for the best usage of the available network bandwidth. Before CANopen messages on the application layer such as the SDO and the PDO are processed, they turn into the CAN messages on the data link layer. The CAN message ID is used in the CAN when determining whether an ECU receives a sent message or not. CANopen, on the other hand, can determine if an ECU receives a sent message by the OD index, which can be referred to using the PDO ID [7].

Table 2: The OD Index Range of Parameters for the PDO Communication Service

Index Range	Description
1400h – 15FFh	RPDO Communication Parameters
1600h – 17FFh	RPDO Mapping Parameters
1800h – 19FFh	TPDO Communication Parameters
1A00h – 1BFFh	TPDO Mapping Parameters

The PDO communication service allows several data to be packed into one PDO, and this PDO is further classified as either a Transmit PDO (TPDO) or a Receive PDO (RPDO). CANopen supports up to 127 ECUs in the system and each ECU can use up to four TPDOs and four RPDOs. A problem exists when one node requires more than four TPDOs or four RPDOs, but this problem can be resolved by reducing the maximum number of ECUs that the system can support and by assigning a CAN ID to each PDO manually. There are communication parameters and mapping parameters for the PDO communication service, which are placed in the OD index range from 1400h to 1BFFh (see Table 2). For example, the TPDO mapping parameter of ECU 5 is located at the index 1A04h and the RPDO communication parameter of ECU 7 is located at the index 1406h. The communication parameter has values for the communication configuration, including the PDO ID, and the mapping parameter has the information of the

data packed into the PDO. Thus, carrying out the PDO packing means inserting, deleting, and modifying the PDO mapping parameter within the OD.

Fig. 3 shows an example of the PDO packing that packs five data into one PDO. The five PDO mapping parameters are stored at the index 1A04h in the OD. At the sub-index 0, the number of data that exist as mapping information (i.e. 5) is stored, and the 5 packs of data of the mapping information are stored at the sub-index from 1 to 5. The mapping information has a 32-bit length. The upper 16 bits and the following 8 bits are the OD index and the sub-index where the data is stored, respectively. The remaining 8 bits represent the length of the data. Consequently, each 1 byte data of 'Hour', 'Minutes' and 'Seconds' at the OD index 2013h and each 2 bytes data of 'Temperature' and 'Pressure' information at the OD index 5010h are packed into the TPDO 5, which has 7 bytes of data.

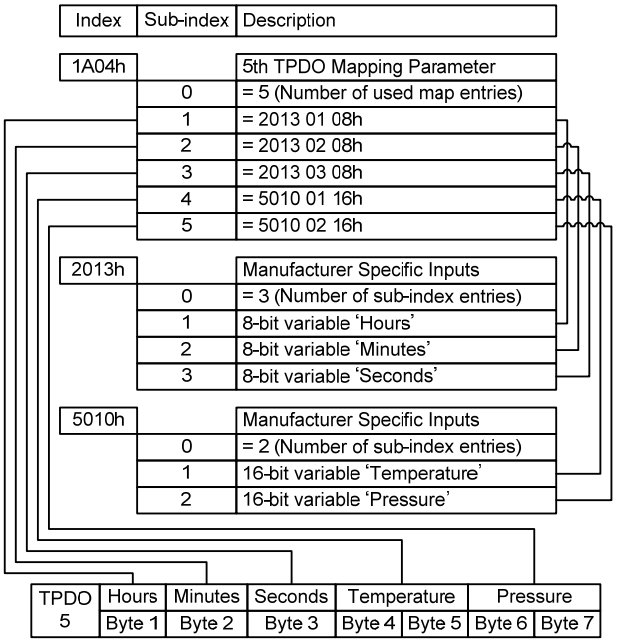

Fig. 3 The Configuration of the Mapping Parameter for the PDO Packing.

3. The PDO Packing Mechanism

The object of the PDO packing mechanism is to improve the network utilization of the CANopen communication system when the set of PDO is schedulable. Subsection 3.1 describes the schedulability analysis of CANopen messages. Subsection 3.2 defines the system model and describes the PDO packing mechanism. Finally, Subsection 3.3 presents the procedure of the on/off-line implementation of the PDO packing mechanism.

3.1 The Schedulability Analysis of CANopen Messages

CANopen message m_i can be classified as schedulable only if the worst-case response time R_i is less than or equal to the deadline of the message D_i. The worst-case response time R_i can be calculated by the sum of the jitter J_i, the queuing delay t_i, and the transmission delay C_i as follows [12]:

$$R_i = J_i + t_i + C_i \tag{1}$$

The schedulability of the message m_i, S_i is defined with the value of either 0 or 1, where $S_i = 1$ means that the message m_i is schedulable (i.e., $R_i \leq D_i$), and otherwise $S_i = 0$. Finally, the schedulability of a whole system, S can be derived from the product of S_i over all of the messages.

3.2 The System Model

The network system model for the PDO packing mechanism consists of p CANopen ECUs and one CAN bus (see Fig. 4). In order to develop the frame packing mechanism, some assumptions are made: 1) there are no hardware failures and transmission errors in the CANopen communication system; 2) there is more than one unit of signal data to be sent within each node; 3) all PDO transmit periodically.

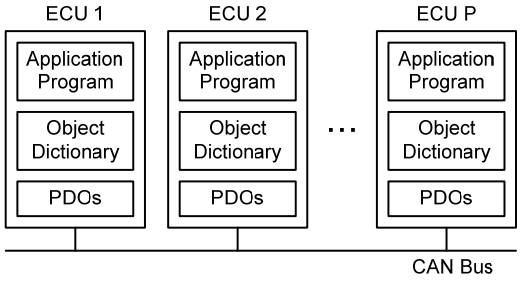

Fig. 4 The CANopen Network System Model.

The combination of PDO packing that allows the net-work utilization to be a minimum should be found under the condition of the system being schedulable. To find this combination, analysis of the schedulability and the network utilization for every combination of the PDO packing is needed. The frame packing problem is similar to the 'bin packing' problem and it was to be NP-hard in [15]. Therefore, in this paper, we have suggested a heuristic method for PDO packing mechanism.

The PDO packing mechanism begins by initializing the system and then repeats the following operations: 1)

selecting 2 PDOs that satisfy the PDO packing conditions which are explained below and 2) packing them into one PDO until the PDO packing mechanism cannot operate any longer (see Fig. 5). It is also defined that every PDO can be in one of three states (i.e., the packable state, the non-packable state, and the temporarily non-packable state) during the operation of the PDO packing mechanism, and if every PDO is in the non-packable state, then the PDO packing mechanism has finished.

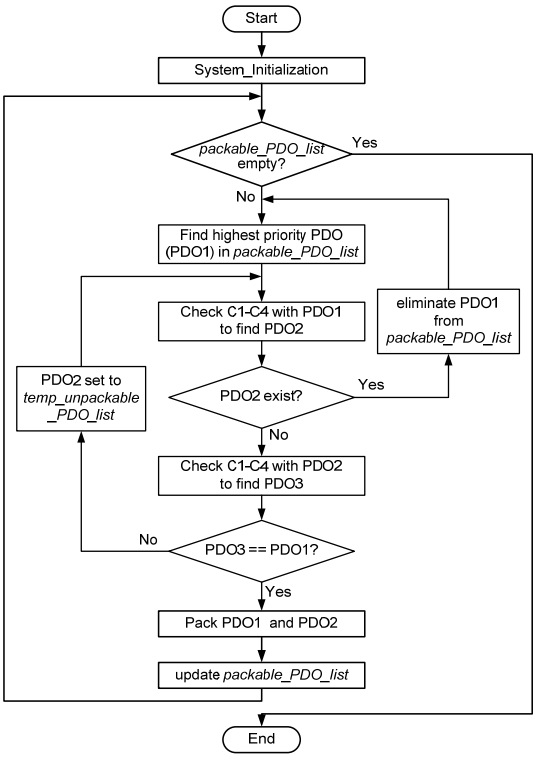

Fig. 5 The Flowchart of the PDO Packing Mechanism.

At the initial state of the system, there are as many PDOs as there are the number of data to be transmitted by each node. Each PDO is defined as m_i regardless of the number of nodes, where i represents the priority of the PDO; a small number of has a high priority, meaning m_1 has the highest priority. The (Deadline–Jitter)-monotonic algorithm can be applied to assign the appropriate priority to the PDO. This works because the (D-J)-monotonic algorithm is an optimal priority assignment algorithm when the deadlines of all the messages are less than or equal to their periods [17]. Here, the period and the deadline of m_i are denoted by T_i and D_i, respectively. The number of data bytes is also represented by b_i.

The number of PDOs at the initial state of the system can be decreased by repeating the PDO packing operation. The PDO packing only operates under following conditions:

1. (C1) The PDO packing is operated only for the PDO that is in the identical node.
2. (C2) The whole length of the data packed into the PDO should be less than or equal to 8 bytes.
3. (C3) The worst-case response time of all the PDOs should be less than or equal to their deadlines.
4. (C4) The network utilization of CANopen should be decreased as compared to before the PDO packing.

It is very inefficient to calculate the worst-case response times of all the PDOs to check whether C3 is satisfied or not. It is more efficient not to consider the PDO with a higher priority than m_i when calculating the worst-case response time of the PDO because the way this calculation is done has not changed, even after the PDO packing operations of m_i and m_j having priorities i and j $(i < j)$, respectively.

The network utilization before and after the packing should be compared in order to check C4. The network utilization of a PDO can be calculated from the duration that the CAN bus is occupied during the PDO transmission, per unit time. For example, in the case of the packing of m_x and m_y ($T_x \le T_y$), when the number of bytes of each PDO is and the transmission time is C_x, C_y, we can obtain the network utilization before the packing (U_{before}) as the following equation:

$$U_{before} = \frac{C_x}{T_x} + \frac{C_y}{T_y} = \frac{\{55(T_x + T_y) + 10(b_x T_y + b_y T_x)\}\tau_{bit}}{T_x T_y} \quad (2)$$

where τ_{bit} is the transmission time for a single bit.

After the PDO packing of m_x and m_y, the number of bytes and the period become $b_x + b_y$ and T_x, respectively. Consequently, the network utilization after packing (U_{after}) can be calculated from the following:

$$U_{after} = \frac{\{55 + 10(b_x + b_y)\}\tau_{bit}}{T_x} \quad (3)$$

The procedure of the selection of two packable PDOs, PDO1 and PDO2, is described in Fig. 5. The PDO1 is the packable PDO having the highest priority and another PDO, that is adequate for packing along with the PDO1, becomes the PDO2. The criteria for selecting the PDO2 is based on the four conditions from C1 to C4.

The PDO1 goes into the un-packable state if there is no PDO2 to be selected, and then another PDO1 can be selected. In the case that multiple PDOs can be selected, the PDO that minimizes the network utilization is selected as the PDO2. Here, the PDO that has the same period as that of PDO1 has higher priority than the other PDOs whose period is different from PDO1's because PDOs whose period is the same as the PDO1's, regardless of the number of bytes they hold, minimize network utilization.

Another reason for this packing procedure is to pack first the data used together, such as 'Hours', 'Minutes' and 'Seconds', as is seen in Fig. 3. If there are multiple PDOs that have the same priority as PDO1's, the PDO that has the highest priority is chosen, and if there are no such PDOs, the PDO that decreases network utilization the most is chosen.

The PDO2 could be the most adequate for packing along with the PDO1, but not necessarily vice versa. Therefore, a new PDO that can be packed with the PDO2 is needed and it is denoted as PDO3. If the PDO3 is identical to the PDO1, the PDO packing operation occurs. Otherwise, another PDO2 is searched for, after the selected PDO2 goes into a temporary non-packable state. Here, the temporary non-packable state was defined in order to find another PDO2 that did not include the PDO2 already selected.

3.3 The Implementation of the PDO Packing Mechanism

The PDO packing mechanism can be implemented for both off-line and on-line operations. The off-line PDO packing mechanism is easy to implement because it operates during the design phase. The off-line PDO packing mechanism is operated based on the data transmission property required in the system (e.g. jitter, period and deadline). As is described in Section 2, this mechanism is implemented by inserting, deleting and modifying the PDO mapping parameters in the OD of each node, which means creating, deleting, and modifying the EDS or the DCF automatically.

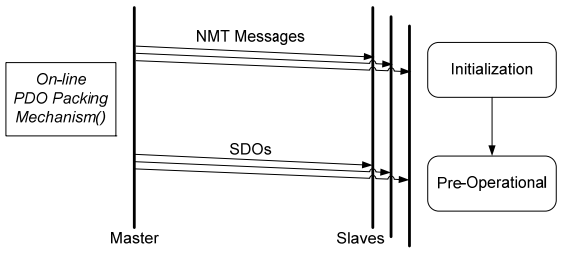

Fig. 6 The Operation of the On-line PDO Packing Mechanism.

The on-line packing mechanism, alternatively, provides network flexibility during the system operation phase. When a new node or data is added to the CANopen network system, the on-line PDO packing mechanism is automatically operated within the master node, and every node except for the master node is changed into the initialization state by the network management (NMT) message (see Fig. 6). Continually, each node is turned into a pre-operational state and receives the result of the PDO packing mechanism in the form of EDS or DCF through the SDO received from the master node. This subsection only provides the guideline for implementing the on-line packing mechanism, and the feasibility of the on-line packing mechanism is not discussed.

4. Performance Evaluation

We use the benchmark data reported by the Society of Automotive Engineers (SAE) [18] to evaluate the PDO packing mechanism. The data set includes the lengths, jitters, periods, and deadlines of the 53 signals that are transmitted from the 7 ECUs.

The 53 signals are packed into 17 CAN messages by using the 'piggyback' [11,12]. The worst-case response times of the signals are calculated at a bus speed of 125 Kbit/s. The worst-case response time of every CAN message meets the deadlines at this bus speed, and the bus utilization here is 84.44% (see Table 3).

Table 3: Analysis of the Worst-case Response Time of CAN Messages Using 'piggyback'

#	Size (bit)	Jitter (ms)	Period (ms)	Deadline (ms)	WCRT (ms)
1	1	0.1	50	5	1.544
2	2	0.1	5	5	2.128
3	1	0.1	5	5	2.632
4	2	0.1	5	5	3.216
5	1	0.1	5	5	3.720
6	2	0.1	5	5	4.304
7	6	0.2	10	10	5.192
8	1	0.2	10	10	8.456
9	2	0.2	10	10	9.040
10	3	0.2	10	10	9.696
11	1	0.2	50	20	10.128
12	4	0.3	100	100	19.088
13	1	0.3	100	100	19.592
14	1	0.2	100	100	20.096
15	3	0.4	1000	1000	28.904
16	1	0.3	1000	1000	29.408
17	1	0.3	1000	1000	29.912

A total of 17 PDOs are packed using the PDO packing mechanism, and at a bus speed of 125 Kbit/s the worst-case response times of all the PDOs meet their deadlines (see Table 4). The CAN bus utilization is 74.54% when the PDO packing mechanism is applied, which results in a decrease by about 10% when compared to the previous work that used the 'piggyback'. Fig. 7 compares the worst-case response times of the PDOs packed in Table 4 and the CAN messages in Table 3. This graph shows that

the worst-case response times of the PDOs are similar to or less than the previous research.

Table 4: Analysis of the Worst-case Response Time Using the PDO Packing

#	Size (bit)	Jitter (ms)	Period (ms)	Deadline (ms)	WCRT (ms)
1	1	0.1	50	5	1.563
2	2	0.1	5	5	1.641
3	1	0.1	5	5	2.070
4	2	0.1	5	5	2.734
5	1	0.1	5	5	3.164
6	4	0.1	5	5	3.984
7	8	0.8	50	20	4.805
8	6	0.8	50	20	5.391
9	3	0.5	50	20	9.141
10	1	0.3	20	20	9.883
11	8	0.2	50	20	14.023
12	8	0.2	50	20	14.141
13	1	0.4	100	100	14.648
14	1	0.2	100	100	18.633
15	2	1.6	1000	1000	19.219
16	3	1.0	1000	1000	19.648
17	1	0.3	1000	1000	20.002

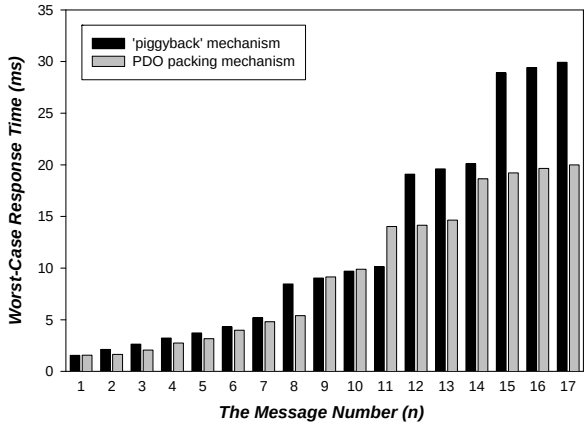

Fig. 7 Comparison of the Worst-Case Response Time of the 'Piggyback' and PDO Packing Mechanism.

5. Conclusion

Minimizing the worst-case response time and improving the network utilization of the CAN messages is necessary to guarantee real-time performance. This paper presented the PDO packing mechanism that can reduce the overhead for data transmission by effectively using the OD and the PDO communication service supported in the CANopen protocol on the application layer of the CAN. SAE benchmark data was used for the performance evaluation of the mechanism, and this study demonstrated that the network utilization of CANopen decreased by about 10%. The PDO packing mechanism can minimize the network utilization of CANopen without using another mechanism on the data link layer for packing the CAN messages. Our future studies will research the dynamic ID assignment mechanism using the OD and the SDO of CANopen for guaranteeing the response time of CAN messages.

Acknowledgments

This study was supported by the R&D Center for Valuable Recycling (Global-Top Environmental Technology Development Program) funded by the Ministry of Environment (Project No.:12-A32-MD).

References

[1] M. Farsi, K. Ratcliff, and M. Barbosa, "An Overview of Controller Area Network," Computing & Control Engineering Journal, Vol. 10, pp. 113-120, 1999.
[2] International Standards Organisation (ISO). Road Vehicles - Interchange of Digital Information – Controller Area Network (CAN) for High-Speed Communication. ISO Standard-11898, 1993.
[3] CAN in Automation (CiA), CAL-Based Communication Profile for Industrial Systems-CANopen. Version 3.0, 1996.
[4] CAN in Automation (CiA), CAN Application Layer for Industrial Applications: CAN in the OSI Reference Model, 1996.
[5] Cenelec, Industrial Communications Subsystem Based on ISO 11898 (CAN) for Controller-Device Interfaces, Part 2: DeviceNet, 2001.
[6] M. Farsi, K. Ratcliff, and M. Barbosa, "An Introduction to CANopen," Computing & Control Engineering Journal, Vol. 10, pp. 161-168, 1999.
[7] O. Pfeiffer, A. Ayre, and C. Keydel, Embedded Networking with CAN and CANopen, RTC Books, 2003.
[8] R. Obermaisser, Event-Triggered and Time-Triggered Control Paradigms, Sptinger-Verlag, Dec. 2004.
[9] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, "Trends in Automotive Communication Systems," Proceeding of the IEEE, Vol. 93, pp. 1204-1223, 2005.
[10] K. Anwar and Z. A. Khan, "Dynamic Priority Based Message Scheduling on Controller Area Netowk," International Confer-ence on Electrical Engineering (IECC'07), pp. 1-6, 2007.
[11] K. W. Tindell and A. K. Burns, "Guaranteed Message Latencies for Distributed Safety-critical Hard Real-time Networks," Technical Report YCS 229, Department of Computer Science, University of York, 1994.
[12] K. W. Tindell, A. Burns, and A. J. Wellings, "Calculating Controller Area Network (CAN) Message Response Times," Control Engineering Practice, Vol. 3, pp. 1163-1169, 1995.

[13] T. Nolte, H. Hansson, and C. Norstrom, "Minimizing CAN Response-Time Jitter by Message Manipulation," Eighth IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'02), pp. 197-206, 2002.
[14] K. Park, M. Kang, and D. Shin, "Mechanism for Minimizing Stuffing-bit in CAN Messages," The 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON'07), pp. 735-737, 2007.
[15] K. Sandström et al., "Frame packing in real-time communication", Proceedings of the International Conference on Real-Time Computing Systems and Applications, pp. 399-403, 2000.
[16] R. Marques et al., "Frame packing under real-time constraints," 5th IFAC International Conference on Fieldbus Systems and their Applications (FeT2003), 2003.
[17] A. Zuhily, "Optimality of (D-J)-monotonic Priority Assignment," Technical Report YCS404, University of York, 2006.
[18] SAE. Class C Application Requirement Considerations. SAE J2056/1, 1993.

Minkoo Kang received the B.S and M.S. degrees in industrial and information systems engineering from Ajou University, Suwon, Korea, in 2007 and 2009, respectively, where he is currently working toward the Ph.D. degree in the Department of Industrial Engineering.
His research interests include frame packing and message scheduling for in-vehicle network and task scheduling for MapReduce software framework in cloud computing.

Kiejin Park received the B.S. degree in industrial engineering from Hanyang University, Seoul, Korea, in 1989, the M.S. degree in industrial engineering from Pohang University of Science and Technology, Pohang, Korea, in 1991, and the Ph.D. degree from the Department of Computer Engineering, Graduate School, Ajou University, Suwon, Korea, in 2001.
From 1991 to 1997, he was with the Software Research and Development Center, Samsung Electronics Company Ltd., Suwon, as a Senior Researcher. From 2001 to 2002, he was with the Network Equipment Test Center, Electronics and Telecommunications Research Institute, Daejeon, Korea, as a Senior Researcher. From 2002 to 2004, he was with the Department of Computer Engineering, Anyang University, Anyang, Korea, as a Professor. Since 2004, he has been an Associate Professor with the Division of Industrial and Information Systems Engineering, Ajou University. From 2010 to 2011, he was with Rutgers, The State University of New Jersey, Piscataway, as a Visiting Professor. His research interests include in-vehicle network, fault-tolerant computing, and cloud computing.