An Efficient Sensor Network Architecture Using Open Platform in Vehicle Environment

Hong-bin Yim, Pyung-sun Park, Hee-seok Moon, and Jae-il Jung

Department of Electrical and Computer Engineering Hanyang University, 17 Haengdang-dong, Sungdong-gu, Seoul, 133-791, Korea
{hbyim,xclass}@mnlab.hanyang.ac.kr, hsmoon@katech.re.kr, jijung@hanyang.ac.kr

Abstract. Vehicles have been developed with an objective of safety. A large number of sensors will be required in Advanced Safety Vehicles that provide intelligent and automatic services in future ITS(Intelligent Transport Systems) circumstances. Because current in-vehicle networks must be changed to add new sensors, the number of sensors that can be added is restricted in current in-vehicle networks. To manage the sensors more efficiently and to provide extensibility, we propose a SCSN (Smart Car Sensor Network), which is an in-vehicle architecture based on AMI-C and OSGi standards. In this architecture, Vehicle Interface (VI), defined in the AMI-C standard, performs as a gateway in an AMI-C network. An integrated VI structure has been developed to provide a Vehicle Service (VS) on a standard platform. An interworking structure with a CAN(Controller Area Network) interface is implemented to provide an efficient VI. In current telematics architecture, time delay occurs between the CAN network start-up time and the platform booting time. Message loss occurs during this time delay. In this paper, we propose an efficient gateway architecture to minimize message loss due to this time delay. The efficiency of this platform has been verified using CANoe, which is a vehicle-network simulation tool.

Keywords: SCSN(Smart Car Sensor Network), Telematics, ITS, Sensor Network, Sensor Network Gateway, Sensor Clustering Node.

1 Introduction

At present, the stability of in-vehicle networks is suboptimal due to increases in wire length, difficulty in diagnosing sensor failures, and fault tolerance issues. Because control and sensing data are transmitted in a single network, it is difficult for in-vehicle portable devices to collect sensor data to check a vehicle status. Furthermore, current in-vehicle networks have problems adding new sensors. To solve this problem, we propose an additional in-vehicle sensor network, namely SCSN (Smart Car Sensor Network). The proposed in-vehicle sensor network collects sensor data from distributed smart sensors, using a sensor clustering node,

© Springer-Verlag Berlin Heidelberg 2007

and sends these data to sensor network gateway. The role of the sensor network gateway is to maintain and manage the overall network, as well as to process sensor data. The processing module of a sensor network gateway creates new information using sensor data fusion techniques.

New information created by sensor network gateway is provided to in-vehicle devices, multimedia terminals, and additional control boxes. For this, vehicle middleware is necessary to provide a vehicle status and travel information to in-vehicle devices.

The SCSN platform is based on OSGi[10] and AMI-C[11], which are open standards for telematics. These standards provide extensibility and interoperability for next-generation in-vehicle software and devices.

The rest of this paper is organized, as follows. Section 2 investigates the current in-vehicle network technologies and problems. Section 3 outlines the proposed SCSN platform architecture and components and illustrates the implementation challenges of the gateway in our SCSN. Section 4 explains the simulation environments and evaluates the performance of the SCSN network, compared with current in-vehicle networks.

2 Related Works

2.1 Architecture of Current In-Vehicle Networks and Its Problems

A vehicle consists of about 11,136 electrical devices, 60 electrical control units (ECU) and 3 Controller Area Network (CAN)[1] buses, with 2,500 signals and 250 CAN messages.

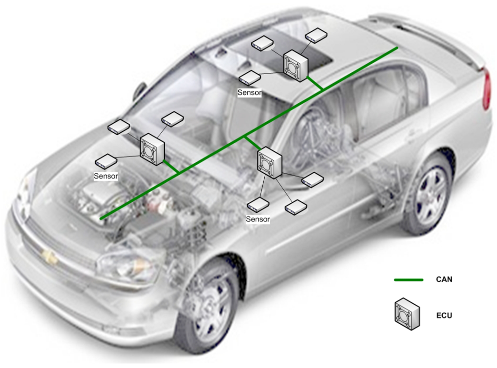

Fig. 1. Current in-vehicle network

Each ECU has multiple sensor nodes and communicates using multi-master communication technology. Sensing data to be collected by ECU are broadcast on the in-vehicle network. Sensing data created by distributed sensors are sent to the actuator in order to perform a job. However, it is difficult to distinguish needed information from all sensing data. Additional overhead is required to acquire proper data from CAN messages collected by ECU. Only proper CAN data needs to be retained from the complete CAN message. Fig 1 shows an example of a current in-vehicle network. Current in-vehicle networks do not support a telematics terminal for travel conditions and location information [2][5].

2.2 Current In-Vehicle Sensor Allocation and Its Problems

A vehicle has many kinds of sensors, such as for safety and airbags. These sensors are connected to each other over a single in-vehicle network. Fig. 2 shows an example of current vehicle sensor allocation. These sensors consist of existing sensors, advanced airbag system sensors, hybrid vehicle sensors, and safety sensors.

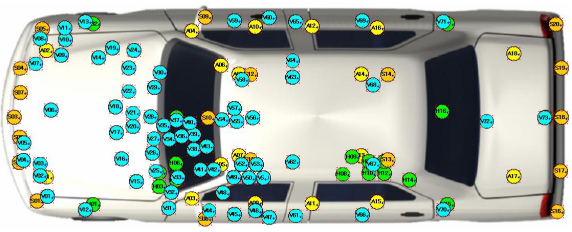

Fig. 2. Vehicle sensor allocation

Current in-vehicle networks are becoming very complex due to increasing numbers of sensors and wire lengths. This network needs to change the existing in-vehicle network to add new sensors. Because all information collected by ECU must be examined to distinguish needed sensing data, technical overhead exists to obtain only the sensing data needed by the ECU.

2.3 Current Telematics Architecture and Its Problems

The AMI-C standard defines a logical vehicle information architecture, namely Vehicle Service (VS) [6][7]. This architecture provides a vehicle information and control service to a platform application by cooperating with the in-vehicle network. The telematics platform based on OSGi operates on the Java Virtual Machine (JVM) [8]. This platform has a problem in that it cannot support VS with the AMI-C standard.

As we see in Fig. 3, this platform uses a vehicle network driver using Java Native Interface (JNI) [8] to provide vehicle information to in-vehicle devices.

JNI on JVM is a method to communicate with other communication technology except TCP/IP [9]. Because a single bundle handles all real-time vehicle messages, using only JNI in this architecture, this communication method is not efficient. A message created by the in-vehicle network may be lost due to platform start-up time delay when the vehicle is started up or when the bundle is restarted. This is further explained in section 3.2.

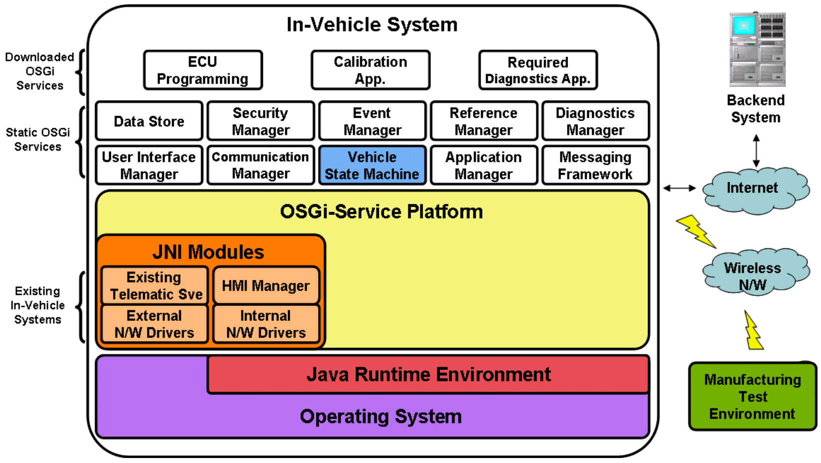

Fig. 3. Current telematics platform architecture[9]

Architecture based on AMI-C provides vehicle travel and status information to in-vehicle devices through middleware [11]. This architecture efficiently provides sensing data management and a processing function for multiple sensors. In this paper, we propose an additional in-vehicle sensor network, using the AMI-C standard, namely a smart car sensor network. The role of this network is to collect and manage sensor data. This network consists of several sensor clustering nodes to collect sensing data from sensors and one sensor network gateway to provide this data to a telematics terminal, multimedia devices, and an additional control box.

3 Smart Car Sensor Network (SCSN)

3.1 Smart Car Sensor Network Architecture

Fig. 4 shows the proposed SCSN architecture. This network architecture provides flexibility to add new sensors and efficient sensor data management using a sensor network gateway and sensor clustering nodes. As we see in Fig. 4, the SCSN consists of one sensor network gateway, several sensor clustering nodes, the current in-vehicle network, in-vehicle devices, and an additional control box.

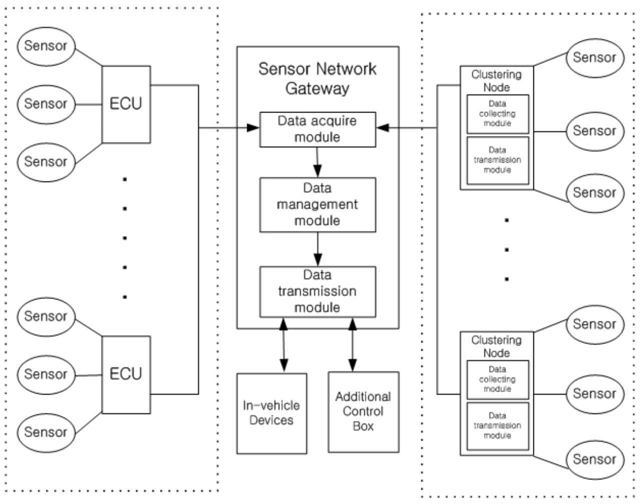

Fig. 4. Smart car sensor network architecture

The sensor network gateway networks among the current in-vehicle network, the sensor clustering node, in-vehicle devices, and an additional control box. The sensor network gateway collects sensing data from the existing in-vehicle network and sensor clustering nodes. This gateway sends this data to a telematics terminal, multimedia devices, and an additional control box. Also, this gateway performs functions, such as data maintenance, processing, and management. The sensor clustering node collects sensing data from distributed sensors. This node transmits sensing data to the sensor network gateway and manages many kinds of sensors connected to this node.

In this architecture, the existing in-vehicle network does not need to be changed to add new sensors so new sensors may be installed on the right side of Fig. 4. A new sensor is connected to a sensor clustering node and is managed by it. This characteristic of SCSN provides flexibility and extensibility for new sensor installation.

3.2 Implementation of the Sensor Network Gateway

Fig. 5 shows the internal architecture of an in-vehicle sensor network gateway. As we mentioned in Section 2.3, message loss occurs during the platform start-up time delay or platform initiation time in current in-vehicle network communication technology. Current in-vehicle communication operates on a JNI driver. This communication technology creates a time delay between the CAN network start-up time and the platform boot time. In current in-vehicle architecture, message loss occurs during this time delay. In our SCSN architecture, the sensor network gateway has components to reduce message loss by decreasing this time delay. These components are as follows.

- CAN gateway
- Vehicle Service Interface (VSI) gateway bundle to provide AMI-C standard vehicle service based on the OSGi platform
- Vehicle Service

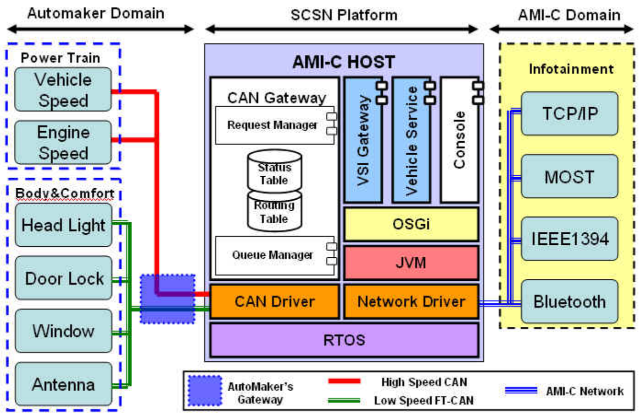

Fig. 5. Internal architecture of sensor network gateway

The CAN gateway cooperates with the VSI gateway using a TCP/IP protocol that is supported by JVM. This communication technology uses mutual communication to request service from the VSI gateway bundle to the CAN gateway. For this communication, we propose a new protocol, namely the GCP (Gateway Communication Protocol), which has the following functions:

1. Receiving/Transmitting the CAN message request
2. ID-based CAN message Registration/Release
3. Inquiry/Modification of routing and status table in CAN gateway
4. Confirm/Reset of processed messages

Mainly, function 1 performs the CAN message receiving/transmitting request and sends the stored message in a status table to the proper application bundle. Function 2 performs ID start/stop for a specific message by message subscription and message transmission request by setting a period. Function 3 and function 4 manage the routing and status table. In other words, the CAN gateway performs processing of vehicle status and sensing data by a message subscription method to control message transmission. This gateway sends the needed message to the VSI gateway so that it performs the role to control CAN and the AMI-C message conversion rate.

3.3 Sequence of Gateway Internal Cooperation

After start-up of the CAN gateway, the VSI on the platform receives stored vehicle data by connecting with the CAN gateway. After this procedure, the VSI on the platform cooperates with the in-vehicle network. This VSI converts each message transmission requirement to a suitable GCP and controls transmission rates by checking the arrival of messages.

Fig. 6 shows sequence diagram of VSI cooperation with the CAN gateway.

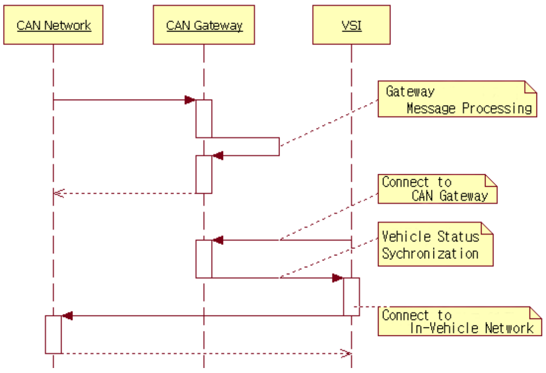
CAN Network
CAN Gateway
VSI
Gateway Message Processing
Connect to CAN Gateway
Vehicle Status Sychronization
Connect to In-Vehicle Network

Fig. 6. Sequence diagram of VSI cooperating with the CAN gateway

4 Simulation

4.1 Simulation Environment

Fig. 7 shows a simulation environment using the CANoe vehicle network simulation tool. The simulation environment consists of a power train network with 500 kbps and a body network with 125 kbps. The power train network consists of the engine, ABS, gear box, and body network, which consists of a door control module, dashboard, and radio channel control console.

These components connect with the sensor network gateway and synchronize the vehicle velocity with the vehicle status information. Window and headlight information for the vehicle are controlled by the VS of the simulation environment.

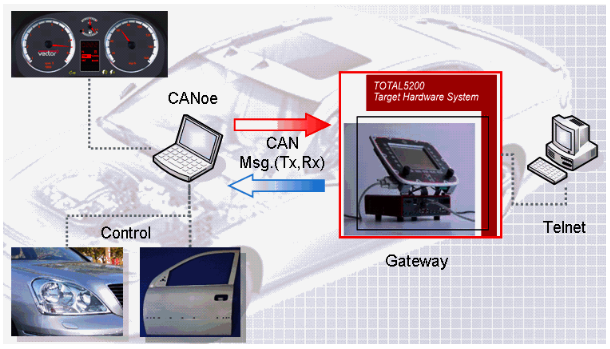

Fig. 7. Simulation environments

4.2 Simulation Results

Vehicle start-up time delay is defined by the OS, CAN driver, VM, and OSGi message processing time, which depends on the bundle initiation time of a standard telematics platform. After this time delay, the sensor network gateway can process CAN messages.

This simulation measures time delay, message loss, and processing efficiency according to two different message processing methods.

1. Message processing method using a CAN module based on JNI
2. Message processing method using a CAN gateway in cooperation with VSI.

The measured items are as follows:

1. Time and message loss until CAN driver initiation
2. Message loss depending on the time from module initiation to message reception completion
3. The number of messages created by the platform application, which is controlled by message processing methods

Fig. 8 shows the overall results for time delay, message loss, and the number of processed messages.

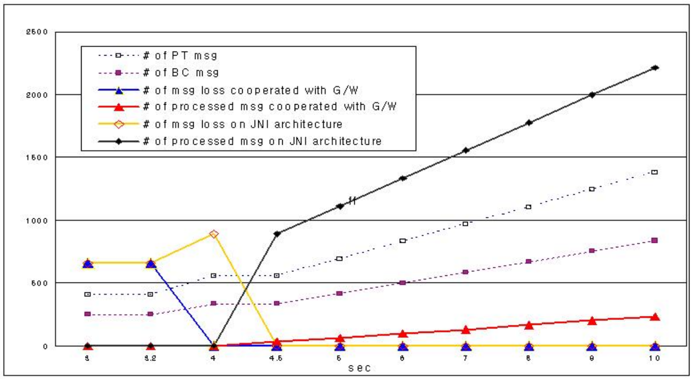

Fig. 8. Overall results for time delay, message loss, and the number of processed messages

The number of message loss cooperated with gateway is less than the number of message loss on JNI architecture, because the start-up time delay of proposed architecture is less than current telematics architecture.

Tables 1 and 2 show the averages of measurement results. PT stands for powertrain and BC stands for body control.

These 2 tables show that message loss is decreased by 27% in the case of powertrain and 25% in the case of body control. These results related to the average start-up time delay. In the case of sensor network gateway, average start-up time delay is decreased by 1.3 seconds in comparison with current telematics architecture.

Table 1. Message loss according to driver start-up delay time

Average of start-up delay(sec)	Average of vehicle message loss			
	Number of PT message	Frame/sec	Number of BC message	Frame/sec
2.711	273	132	168	83

Table 2. Start-up delay time and message loss according to architecture

Architecture	Average of start-up delay(sec)	Average of vehicle message loss	
		Number of PT message	Number of BC message
Currnet telematics	4.5	558	335
Sensor network gateway	3.2	404.6	251

5 Conclusion

In this paper, we have proposed a SCSN platform to solve the problem presented by an increasing numbers of sensors. The SCSN consists of a sensor network gateway and a sensor clustering node, which transmits sensing data to multimedia devices, an additional control box, and a telematics terminal. This network architecture improves sensing data sharing and management efficiency.

According to the simulation results, the SCSN platform is more efficient than current in-vehicle network architecture in terms of message loss and message processing rates. The SCSN platform provides extensibility, using an in-vehicle network standard, and supports easy installation of additional devices or sensors.

The merits of the SCSN platform are as follows:

1. Provides extensibility for an in-vehicle sensor network
2. Shares sensing data using sensor data management
3. Increases interoperability using a vehicle network standard
4. Improves in-vehicle network management by separating control and sensor data
5. Guarantees a predictive start-up time and efficient message processing

In the future, we will refine our proposed network architecture through improvement of the sensor network gateway function. A refined scheme should include an efficient data fusion algorithm and transmission algorithm from the sensor clustering node to the sensor network gateway. We will also study fault-tolerant architecture in an in-vehicle network.

Acknowledgements

This research was supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Assessment) (IITA-2005-(C1090-0502-0020)).

References

1. Peng Shuai, Taeyeon Lee , Ealgoo Kim , Jaehong Park, "A Study on an algorithm of a Network Node for CAN-Based Wiring System Using Polling Structure", IPC-13 1105 - 1108, 2005
2. Choi Chang-hee, "Trends of Telematics Based or Network for Vehicle", Auto Journal, Vol 27, No 6, pp 9-16, 2005
3. http://www.can.bosch.com
4. AMI-C 1003 "AMI-C Release 2 Architectural Overview v1.00"
5. Syed Masud Mahmud, Sheran Alles, "In-Vehicle Network Architecture for the Next-Generation Vehicles", SP-1918, 99-108, 2005 SAE
6. AMI-C 1003, "AMI-C Release 2 Architectural Overview v1.00", 2003
7. Peter Abowd, Gary Rushton, "Extensible and Upgradable Vehicle Electrical, Electronic, and Software Architectures", SAE TRANSACTIONS, VOL 111, pp 495-498, 2002 SAE
8. http://java.sun.com
9. Vivek Kapadia, Veena Bai and G Parthasarathy, "Telematics System Architecture", WHITE PAPER, wipro Ltd., 2005
10. http://www.osgi.org
11. http://www.ami-c.org
12. AMI-C 4002, "AMI-C Requirements and specifications for Human Machine Interfaces v1.00", 2003
13. AMI-C 3001, "AMI-C Requirements and guidelines for software host platforms v1.00", 2003
14. AMI-C 2002, "AMI-C Common Message Set v1.01", 2003
15. AMI-C 2001, "AMI-C Network protocol requirements for vehicle interface access v1.00", 2003