Leading Our World In Motion

2005-01-1540

SAE TECHNICAL
PAPER SERIES

An Adaptive Data-Reduction Protocol for the Future In-Vehicle Networks

Praveen R. Ramteke and Syed Masud Mahmud
Electrical and Computer Engineering Department, Wayne State University

Reprinted From: In-Vehicle Networks and Software (SP-1918)

ISBN 0-7680-1633-9

9 780768 016338

SAE International™

**2005 SAE World Congress
Detroit, Michigan
April 11-14, 2005**

400 Commonwealth Drive, Warrendale, PA 15096-0001 U.S.A. Tel: (724) 776-4841 Fax: (724) 776-5760 Web: www.sae.org

All SAE papers, standards, and selected books are abstracted and indexed in the Global Mobility Database.

ISSN 0148-7191

2005-01-1540

An Adaptive Data-Reduction Protocol for the Future In-Vehicle Networks

Praveen R. Ramteke and Syed Masud Mahmud
Electrical and Computer Engineering Department, Wayne State University

ABSTRACT

The demand for drive-by-wire, pre-crash warnings, telematics and many new features will require very high bandwidth from future in-vehicle networks. One straightforward solution to satisfy the bandwidth requirements of future vehicle networks would be to use a higher bandwidth bus or to use multiple busses. However, the use of a higher bandwidth bus would increase the cost of the network. Similarly, the use of multiple busses would increase the cost and the complexity of the wiring. Another option would be the development of a higher layer protocol to reduce the amount of data to be transferred. This paper presents an adaptive data-reduction algorithm based on the CAN (Controller Area Network) protocol.

INTRODUCTION

As automakers are incorporating more and more advanced features into vehicles, there is a growing need for enhanced processing power. S. Channon and P. Miller [1] estimate that the number of microprocessors per vehicle will increase exponentially and by the end of year 2010, the number of microprocessors in any high-end vehicle will be 250. Also, the application areas that demand more processing power are active safety systems, infotainment (TV, Video, gaming and navigation systems), and drive-by-wire systems. Additionally, functional integration of individual sensors will be necessary for applications such as collision avoidance. These additional functionalities can be achieved by increasing the number of nodes or Electronic Control Modules (ECM's), sensors and actuators that can exchange data among various other nodes in the network. However, the data traffic over the high-speed communication bus, like CAN, will increase significantly with the increase in the number of ECM's. Another important requirement would be to communicate data between the various ECM's in real time for safety-critical applications. Failure to communicate data within a given period of time may lead to degradation in system performance and may put the occupant's life in danger.

In order to address the bandwidth concerns and eliminate message latencies, data-reduction (DR) techniques have been developed which communicate large amounts of data in a short period of time, consuming very less bandwidth. DR techniques have been applied extensively for the task of image compression (e.g. MPEG, JPEG), data compression and digital data transmission.

Due to the safety-critical nature of automotive multiplexing system, selection of an optimum data-reduction technique is imperative to ensure proper system performance. An optimum DR technique can be characterized as the one that consumes the least network bandwidth and has zero message latency. Many DR techniques have been proposed for automotive multiplexing environment. Kempf *et al.* [2] and Michael *et al.* [3] recognized six data reduction algorithms, which could be applied to automotive multiplexing environment. The major drawback of all the DR techniques was that they could only be applied to text-data classes in automotive body electronics [3].

Misbahuddin *et al.*[4] proposed a generalized data-reduction algorithm that could be applied to all data classes found in automotive multiplexing environment. The algorithm is based on the SAE J1939 protocol [5]. This algorithm uses the fact that some of the bytes in the data field of the J1939 message remain constant over a period of time and that many of them change slowly [6]. For example, car speed, wheel speed, gas level, gear position etc. Also, these messages are sent at periodic intervals of time. For example, wheel speed data may be sent every 5 milliseconds and gas level may be indicated every 500 milliseconds. The bit reserved by SAE for future use or *R* bit in the protocol data unit (PDU) within the J1939 message is used to indicate data compression. The *R* bit is set to "1" and "0" to indicate compression and no compression, respectively. Each intelligent control module (ICM) in the multiplexing system consists of a transmit buffer (T_BUF) and a receive buffer (R_BUF). Each ICM keeps a copy of the most recently transmitted message in the T_BUF and a copy of the most recently received message in the R_BUF. Suppose an ICM transmits a message *ri* every *t* time units. The transmitting ICM keeps a copy of

message *ri* in its T_BUF. The next time the ICM transmits a message *ri* after *t* time units, the ICM compares the data field of the previous *ri*, saved in the T_BUF, with the current message being transmitted. In the event that two or more of the data bytes of the current transmitted message are of same magnitude as of the message in T_BUF, the transmitting ICM realizes that few of the data bytes have been repeated. The transmitter then initiates a series of steps outlined below, to implement data compression.

1. The transmitter sets the *R* bit or data compression bit (DCB) in the PDU to "1".

2. The transmitter prepares a compression code (CC) to indicate repeated bytes in the recently transmitted message.

3. Each bit in the CC indicates a data byte in the message. The bits in the CC are set to "1" or "0" to indicate a repeated byte or non-repeated byte in the message, respectively. For example, if byte 3 in the message is repeated then bit 3 in CC is set to "1" and so on.

4. The non-repeated bytes are concatenated after the CC in the data field of the message being transmitted over the bus.

At the receiver, the ICM keeps a copy of the most recently received message in its R_BUF to perform decompression. The following steps are involved in data decompression process at the receiving end.

1. The receiver checks the DCB of the received message.

2. If DCB is "1", then the receiver treats the first byte in the data field of the received message as the CC.

3. The receiver retrieves the repeated data bytes by indexing through the R_BUF and fetching bytes whose corresponding bits in the CC have a value "1". For example, byte 3 is fetched if bit 3 in CC is "1".

4. The entire message is recreated using the repeated bytes from R_BUF and non-repeated bytes from the received message.

This algorithm works very well when data bytes within the message remain constant with high probabilities. However, in real life situations, one or more parameters in the message may fluctuate at low rates. For example, during braking, car speed may decrease at a constant rate say 8 miles/sec^2, or during acceleration the speed may increase at 4 miles/sec^2. Under such conditions, the above protocol would transmit the entire length of the data bytes used to represent the particular parameter even though the change in value of the parameter is very less. In the worst-case scenario, if all the parameters in the message change by small amounts, the algorithm would transmit all the eight bytes of the message without leading to any DR. This is highly undesirable since it would lead to high bus utilization and consumption of unnecessary bandwidth.

To overcome this drawback of the previous DR algorithm, we present an Adaptive Data-Reduction (ADR) algorithm. The algorithm is designed to be compatible with the CAN protocol since CAN is the standard protocol used in many present-day automobiles [7]. However, the algorithm can be applied to any SAE J1939 protocol with slight changes in the software. The following section presents the details of the adaptive-data reduction algorithm.

PROPOSED DATA REDUCTION ALGORITHM

The proposed ADR algorithm consists of two parts

- Non-Adaptive data-reduction
- Adaptive data-reduction

The ADR algorithm encompasses some of the desirable features of the DR algorithm presented in [4] with some notable changes.

1. The *R* bit or DCB is ignored.

2. The data field of CAN message is divided into signal fields of varying lengths instead of data bytes. This means that individual signal values in the messages can cross byte boundaries or can be contained in less than one byte.

Point 2 above springs from the fact that all signal values in a message cannot be contained in one byte. For example, on the one hand, engine RPM requires 16 bits and on the other hand, gear position can be indicated using just 3 bits. A clear understanding of the data field format of the CAN message can be obtained from Figure 1. The engine control module generates this message. Engine speed or RPM occupies two data bytes within the message and gas level occupies 7 bits. The following sections explain the details of the ADR technique.

DATA REDUCTION TECHNIQUE

An automotive multiplexing system consists of many nodes interconnected between each other using a high-speed bus. The nodes contain sensors, which periodically communicate to send the change in some parameter values to the Electronic Control Unit (ECU), which in turn ensures optimum operation of the entire vehicle by actuating certain precise operations based on the input parameter values. Additionally, each node transmits and receives a fixed set of messages [8]. We have made the following assumptions for the DR algorithm.

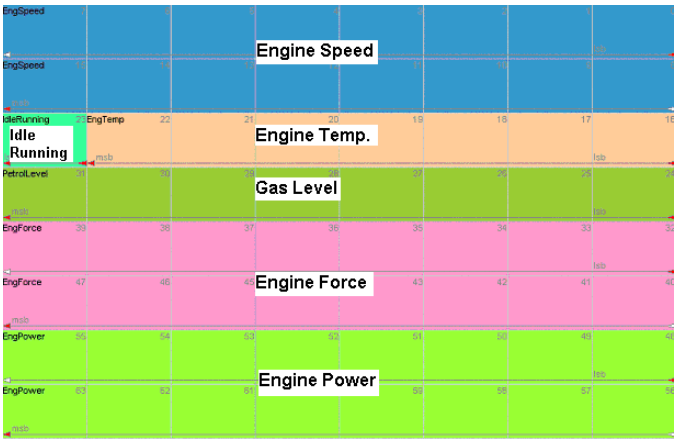

Figure 1. Data field of a CAN message showing various signal fields within the message. Each row indicates a data byte and each column indicates a bit. Some signals within the data field occupy more than a byte where as some others occupy one byte or less.

1. All messages in the system have their compressed version whose individual signal field lengths (in bits) are predefined based on the periodic message transmission rate and rate of change of the signal parameter.

2. The identifiers of the compressed messages differ from the identifiers of their original counterparts by a value of minus one. For example, if the identifier of a normal CAN message is 20, then the identifier of its compressed version would be 19.

3. Each CAN node in the network consists of two buffers called TX_BUF and RX_BUF to store the transmitted and received messages respectively. The TX_BUF and RX_BUF consist of two fields. One field stores the 11-bit (or 29-bit) identifier of the CAN message and another stores its 8-byte data field.

Data Compression

Each CAN node in the network transmits messages at fixed intervals of time depending on the nature of the message being sent. Let us assume that the transmitting unit of each node transmits messages every *t* time units where value of *t* may vary depending on the specific application. A safety critical message will have a very low value for the period *t* when compared to a non-safety critical message.

Assume that a CAN node transmits a message *ci* at time *m.* At time *m,* a copy of the sent message *ci* is stored in the TX_BUF of the transmitting node. When the next *ci* is being transmitted at time *m* + *t* time units, a comparison is made between the data fields of the message stored in TX_BUF at time *m* with that of the current message being transmitted at time *m* + *t.*

Depending on the content of the new message, two types of DR techniques can be applied.

1. Non- adaptive DR

2. Adaptive DR

Let us explore the two techniques further.

Non-Adaptive Data-Reduction

This technique is applied for the case when none of the bytes in the data field of the new message *ci* at time *m* + *t,* change their value since their previous transmission at time *m*. In such a situation, the entire message at time *m* + *t* is not transmitted. The receiving node in the network assumes the most recent value of the message *ci* as the current value of the message. However, this technique has a drawback. In the event that none of the signal fields in the message *ci* change their value over a long period of time, the transmitting node will not transmit the message *ci* until it changes its value. To overcome this shortcoming, a copy of the most recent message *ci* is transmitted to the receiving node at every *y* intervals of time where *y*>>*t.* The value of *y* again depends on the safety-critical nature of the message. In the event of sudden and rapid changes in value of the signals, for example, during hard braking or hard acceleration, normal transmission of CAN messages over the bus resumes till a steady state is reached.

Adaptive Data-Reduction

Adaptive DR is based on the technique of delta modulation or differential pulse code modulation (DPCM) [9] that is widely used in the area digital communications. It is based on the principle that instead of sending the absolute value of a signal at each time instant, only the changes in the signal values from one time instance to another (delta) are transmitted.

To achieve data reduction, the first byte in the data field of the CAN message is used to indicate delta compression. This byte is called the delta compression code (DCC). In the ADR technique, if one or more signal fields in the message *ci* transmitted at time *m* + *t* change their value since their previous transmission at time *m*, the transmitting node executes the following series of steps.

1. The transmitting node computes a delta compression code (DCC). A value of “1” in the DCC indicates the delta change in the value of the signal rather than the absolute value. A value of “0” in DCC indicates no change in the value of the signal.

2. Since a copy of the message transmitted at time *m* is stored in TX_BUF, the transmitting node computes differences (deltas) in the value of the corresponding signals in TX_BUF at time *m* with that at time *m* + *t.*

3. The compressed version of the original CAN message having an identifier whose value is one less than the value of the identifier of the original message is encoded with the delta signal values.

4. The delta-compressed CAN message carrying the delta signals is sent over the CAN bus to the receiving node.

5. In the event that the value of a delta signal exceeds the length of the assigned field, the absolute values of all the signals (i.e. the original CAN message) are transmitted rather than the delta-compressed version of the message. This assumption is based on the fact that a drastic change in the value of a particular signal could reflect a critical condition in which case it would be necessary to communicate absolute values of all signals within the message.

The entire data-compression process is depicted using the flow chart of Figure 3.

Figure 2 shows the format of the data field of the delta-compressed message of Figure 1. The engine module generates this particular message. The first eight bits of the data field are the DCC. The signals within the delta-compressed message occupy half the number of bytes when compared to the original message of Figure 1.

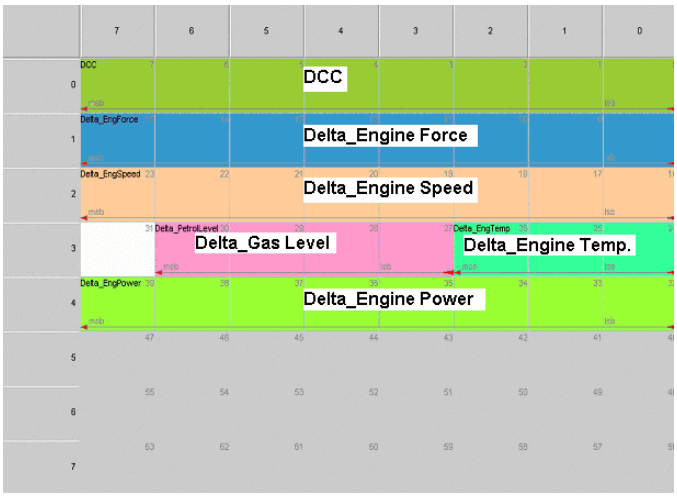

Figure 2. Data field format of delta-compressed message of Figure 1.

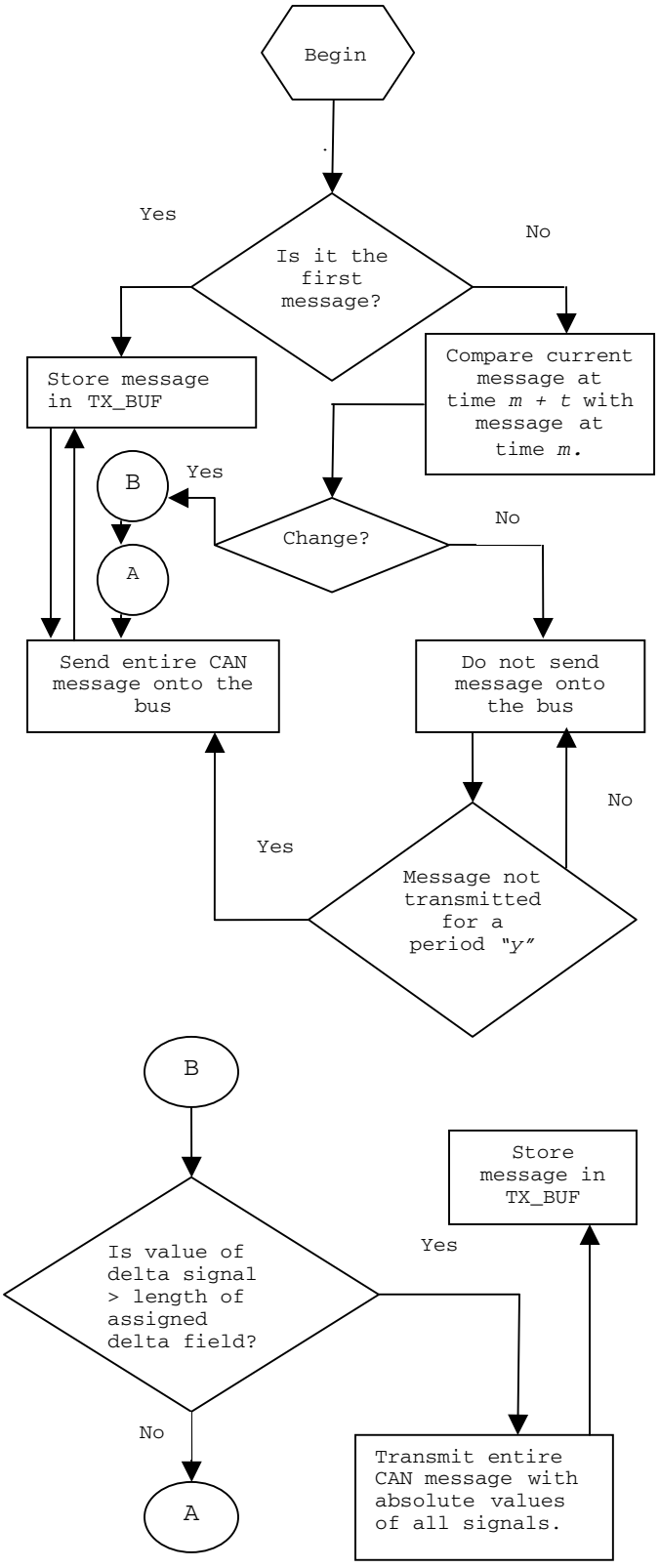

Figure 3. Adaptive data-reduction algorithm

Data-Decompression

The receiving node in the system decompresses the delta-compressed CAN message sent by the transmitter. The following series of steps are executed to perform

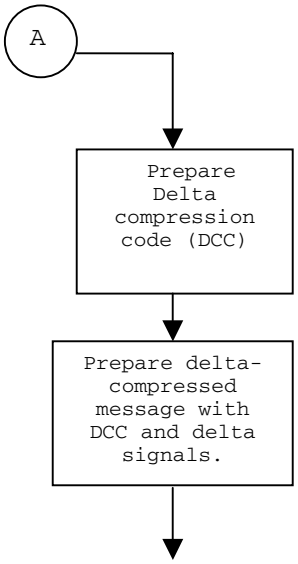

Figure 3. Adaptive data-reduction algorithm continued.

data-decompression. Assuming that a delta-compressed message *Dci* is received,

1. The receiving node checks the identifier of the CAN message *Dci*. If the message is a delta-compressed message, the first byte in the data field of the message is treated as the DCC.

2. The receiver then fetches a copy of the most recently received *ci* from the RX_BUF.

3. The DCC acts as an index to the corresponding signal fields within the delta-compressed message. For example, a value of "1" in the first bit position of the DCC means that the first signal field within the message is a delta signal of the first signal field within the message *ci* from RX_BUF. This delta signal is either added to or subtracted from the corresponding signal field within the message *ci* from RX_BUF to get the new value of the signal.

4. A value of "0" in the DCC means that the corresponding signal has not changed its value since the previous transmission. The absolute value of this signal is fetched from the previous *ci* in RX_BUF.

5. The receiver reconstructs all the signals within the message in a similar fashion.

6. The receiver then updates the RX_BUF with this new message, overwriting the previous *ci*.

The data-decompression process is illustrated graphically below.

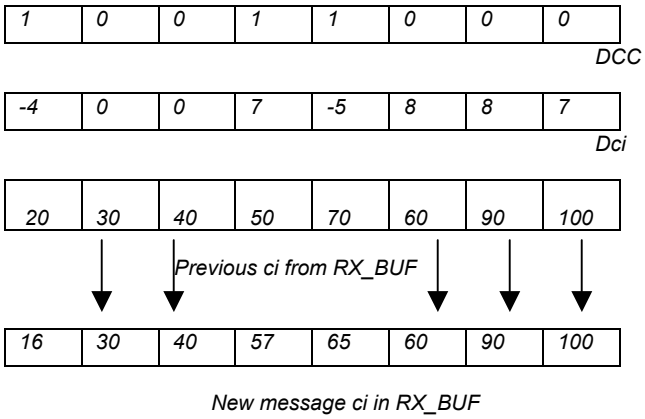

Figure 4. Data-decompression process.

The first row shows the mapping of the bits in the DCC of the received delta-compressed message. The second row is the delta-compressed message *Dci,* which holds delta values of all signal fields within any given message. Third row is the most recent copy of the message c*i* from the RX_BUF. A value of "1" in DCC indicates that the corresponding fields of *Dci* contain the delta values. In this example, delta signals corresponding to a "1" in DCC are -4, 7 and -5. These delta values are then added to the corresponding signal fields of the previous message c*i* in RX_BUF. All other signal fields within the new message retain previous values from the *ci* in RX_BUF. The RX_BUF is then updated with this new message.

The entire data decompression process is depicted in the flow chart of Figure 6. The following section presents simulation results for the presented ADR algorithm.

RESULTS AND DISCUSSIONS

A simulation program has been developed to analyze the performance of the adaptive DR algorithm. The main performance parameters considered are the bus load, message transmission rate (MTR) and the average length of messages (ALM). The simulation model consists of five CAN nodes each of which transmits a predefined set of messages. The distribution of messages among the five nodes is shown in Table I. Their corresponding periodic transmission rates are also included.

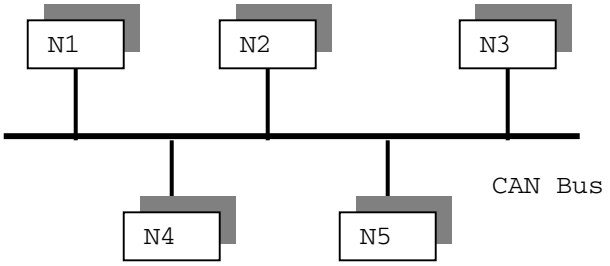

Figure 5. Simulation model consisting of 5 CAN nodes.

The simulation model consisting of 5 CAN nodes is shown in Figure 5. Each of the nodes numbered N1 to N5 consists of the TX_BUF and the RX_BUF to store the transmitted and received messages, respectively.

Table I. Distribution of messages among nodes in the simulation model.

Node Number	Message ID's	Periodic transmission rate (in milliseconds)
1	99 and 100	5
	101 and 102	5
2	200 and 201	5
3	1019 and 1020	50
4	1050 and 1051	500
	1052 and 1053	500
5	272 and 273	500

All message ID's on the left of the message ID column are the ID's of the delta-compressed message. The simulation has been developed to monitor the behavior of the system during three operating conditions.

1. During transient operation of the system i.e. during engine start and acceleration.

2. During steady-state operation i.e. under cruise control, and

3. During hard braking or deceleration.

We will further analyze the performance of the system under these three operating conditions. For the sake of comparison, we will consider the cases when no DR was applied and when non-adaptive and adaptive data-reduction techniques were applied under the three operating conditions. Bus-load, the number of data frames per second or MTR and the average length of messages or ALM have been monitored to evaluate the performance of the ADR algorithm. We will first analyze the system performance under the transient and steady-state conditions.

CASE 1: TRANSIENT AND STEADY-STATE OPERATION

The graphs of Figure 7 through Figure 9 compare the simulation results for the system under transient and steady-state, with no DR and when non-adaptive DR is applied. The variation of different signals within a message (in this case engine data) during transient and steady state is shown in Figure 10. The variation in the MTR is compared in Figure 7.

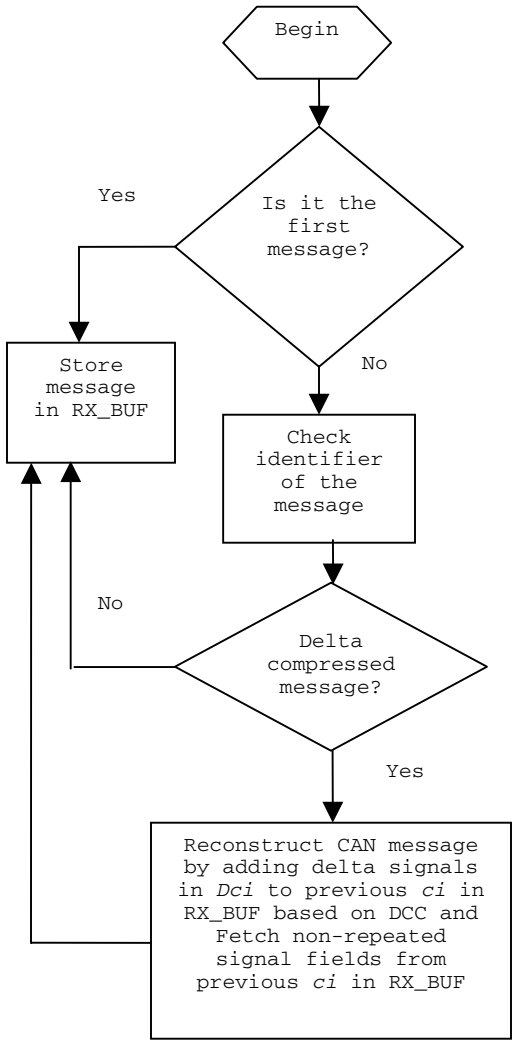

Figure 6. Data-decompression algorithm.

In the transient state, which lasts for sixty seconds after start of simulation, the MTR reaches a peak value of 138 frames/second at time T1 for both the cases, i.e. when no DR is performed and when non-adaptive DR is

performed. After the transient period, the system reaches the steady state where all the signals reach a constant value. During the steady state, there is a considerable decrease in the MTR for the case when non-adaptive DR is applied. The MTR falls to 44 frames/second at time T2 reducing the number of message transmissions by 50 in comparison to the case where no DR is used. This gives a compression ratio of 53% over the case when no DR was applied.

The variation in the bus-load is compared in Figure 8. During the transient state, the variation in bus-load is the same for the cases when no DR is performed and when non-adaptive DR is performed. Notable gain in the parameter is made during the steady-state operation where majority of the signals do not vary rapidly from one instance to another and hence the message is not repeated unless there is significant change in its value since the previous transmission. During the steady-state operation, the peak bus-load at time T2 reduced from 18.55% to 7.98% giving a total bandwidth saving of 56% when compared to the case where no DR was applied.

Figure 9 compares the average lengths of the messages (ALM) transmitted between various nodes during the simulation. The ALM remained constant over majority of the simulation run for the two cases when no DR was applied and when non-adaptive DR was applied. However, as the system approached the steady state, minute reductions in the ALM were observed for the case when non-adaptive DR was applied. This is due to the fact that as the system reached the steady state, the small variations in the signals were transmitted in the delta-compressed messages.

CASE 2: HARD BRAKING OR DECELERATION

The performance of the ADR algorithm during the condition of hard braking or deceleration can be analyzed from the graphs of Figure 11 through Figure 13. The variation in MTR during hard braking with no DR and with ADR is compared in Figure 11. During hard braking, the behavior of the system is similar to the behavior during the transient state. The MTR reaches its peak value of 138 frames/second and 137 frames/second at the instance when the brake is applied at time T2, when no DR and when ADR is applied, respectively. There is no significant reduction in the MTR for the case when ADR is applied since the periodic rate of transmission of the delta-compressed messages is similar to the transmission rates of their uncompressed counterparts. After the period of braking, the system again goes to the steady state. The variation of the various signals within a message (in this case engine data) during the application of the brake is shown in Figure 14.

The variation in bus-load is compared in Figure 12. The application of brake leads to the transmission of the delta-compressed messages and hence, the peak bus-load at time T2 increases slightly to a value of 16.97%. A total bandwidth saving of 43% was gained in comparison to the case where no DR was applied.

Figure 13 compares the ALM for the cases when no DR is applied and when ADR is applied. There is a decrease in ALM when ADR is applied since the delta-compressed messages having a smaller duration are transmitted during the application of brakes. The ALM decreases to a value of 0.2 ms at time T2 when brakes are applied giving a total reduction of ((0.30-0.20)/ 0.30)*100= 33% in the ALM when compared to the case when no DR was applied. Table II summarizes the percentage gain in the values of various performance parameters with the application of non-adaptive and ADR algorithms.

Table II. Gain in the values of performance parameters in comparison to the case when no DR was applied to the system.

	% decrease in bus load	% decrease in MTR	% decrease in ALM
Non-adaptive DR	56%	53%	0%
Adaptive DR	43%	0%	33%

CONCLUSION

The performance of an adaptive data-reduction algorithm for future in-vehicle networking applications has been presented. The algorithm is based on the CAN protocol and uses the fact that some of the signals within a CAN message do not change their value from one transmission time instance to another. Another observation is that some of the signal values change by small amounts i.e. during cruise control or deceleration. The algorithm exploits the above two observations to decrease the amount of data traffic to be transmitted over the CAN bus. In the first case, the algorithm refrains from transmitting the repeated messages during successive transmitting instances and in the second case, the algorithm transmits only the delta-change in the value of signals from one transmitting instance to another. The adaptive data reduction (ADR) algorithm generates a delta compression code (DCC) to indicate signals whose corresponding delta values have been transmitted. A dynamic event-based simulation has been developed to analyze the performance of the presented ADR algorithm. The simulation of the algorithm has shown promising results in terms of the

reduction in the values of important parameters like message transmission rate (MTR), bus-load and average length of messages (ALM). The reduction in the values of these parameters enables us to cater to the bandwidth requirements of new nodes into the network to provide advanced features like infotainment and active safety etc., without increasing the overall cost of the system.

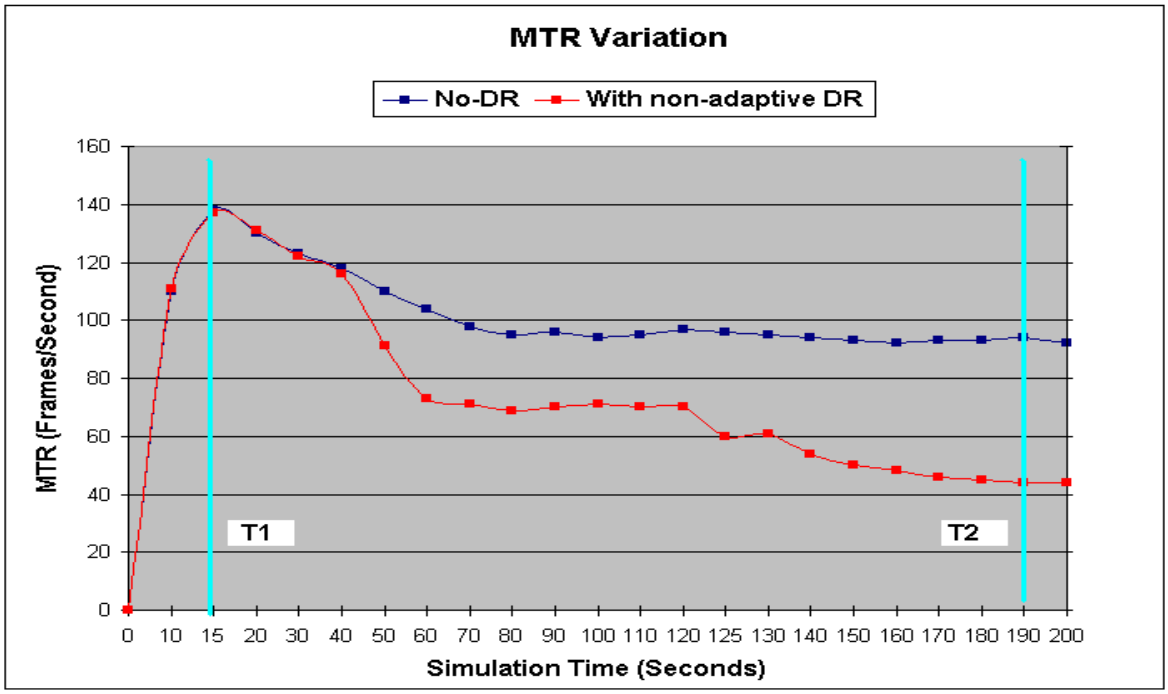

Figure 7. Message transmission rate (MTR) comparison.

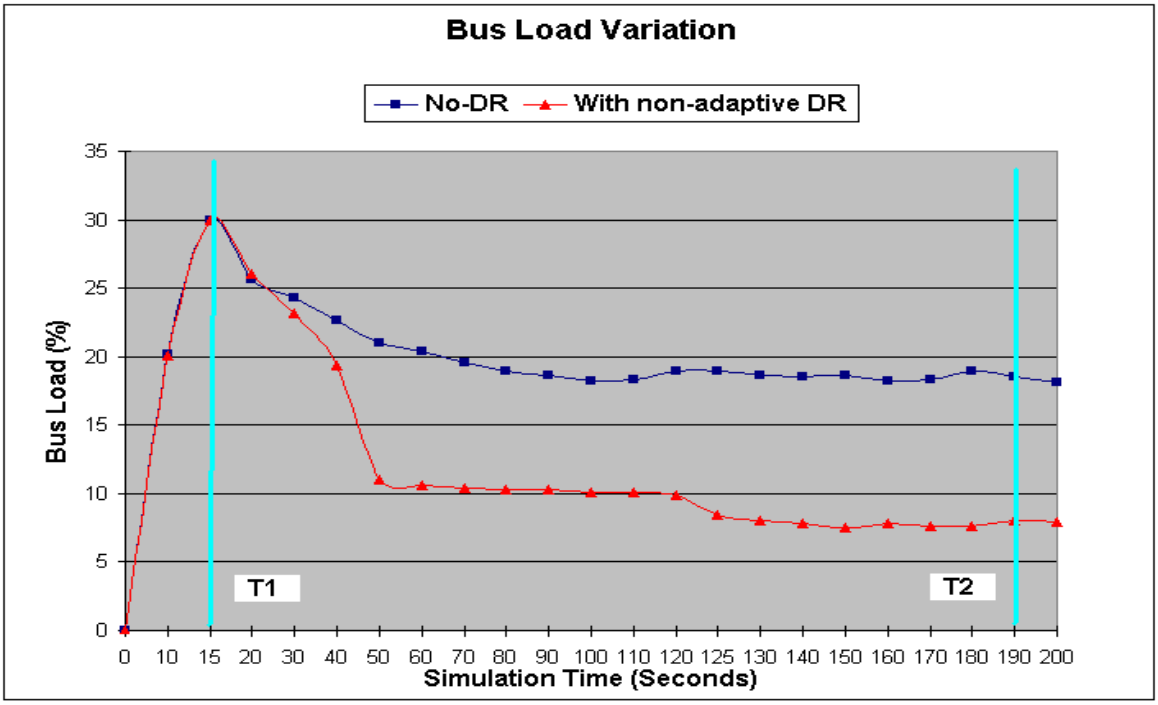

Figure 8. Bus load Comparison.

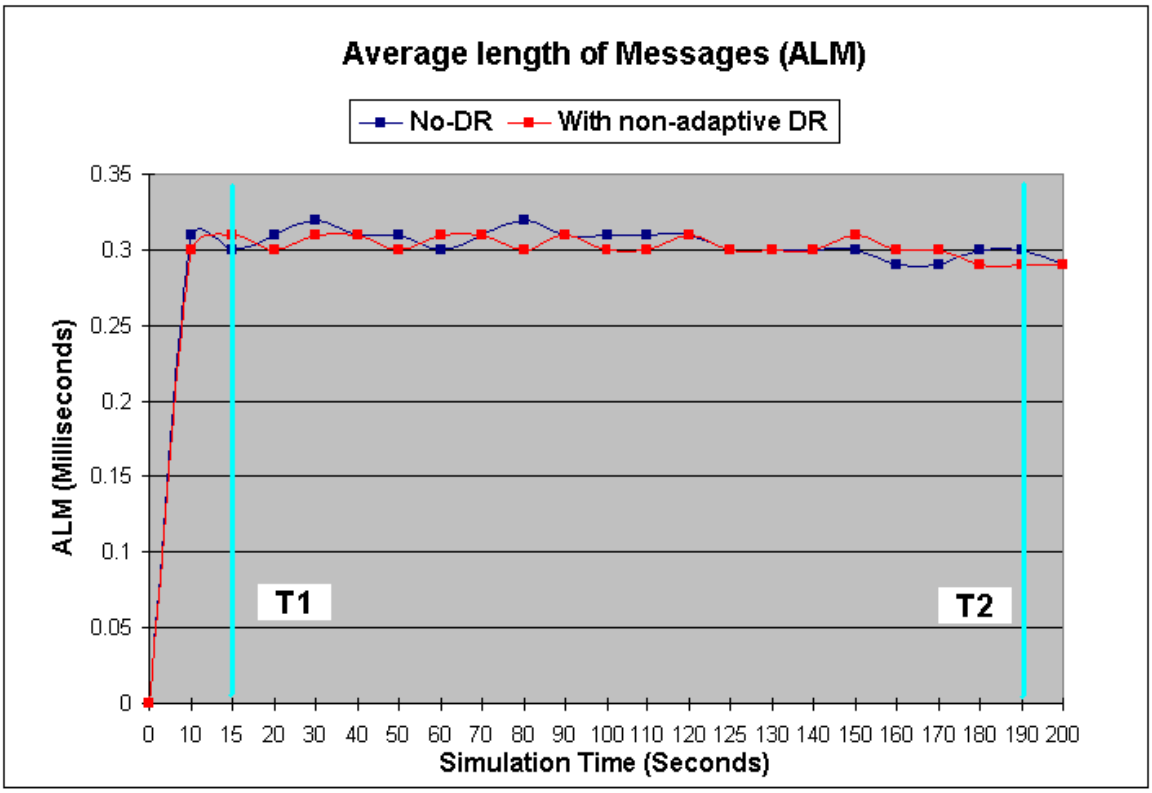

Figure 9. Average message length (ALM) comparison.

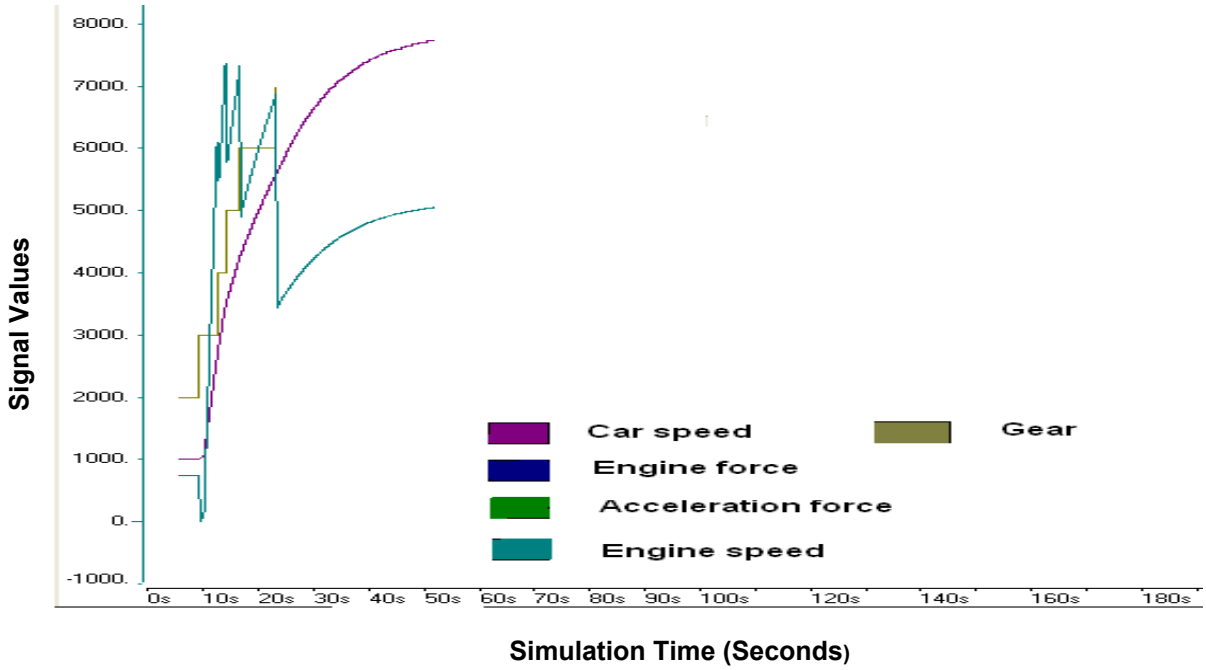

Figure 10. Variation of different signals during transient and steady state.

.

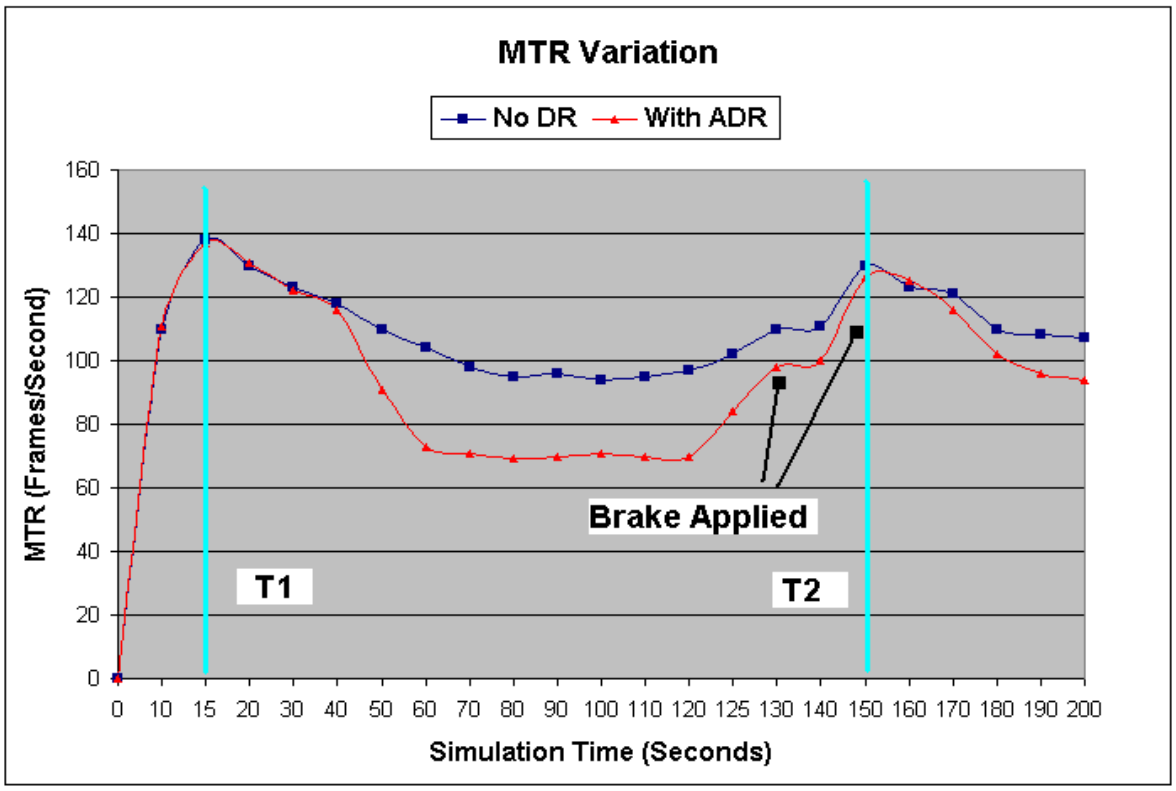

Figure 11. Message transmission rate (MTR) comparison during hard braking.

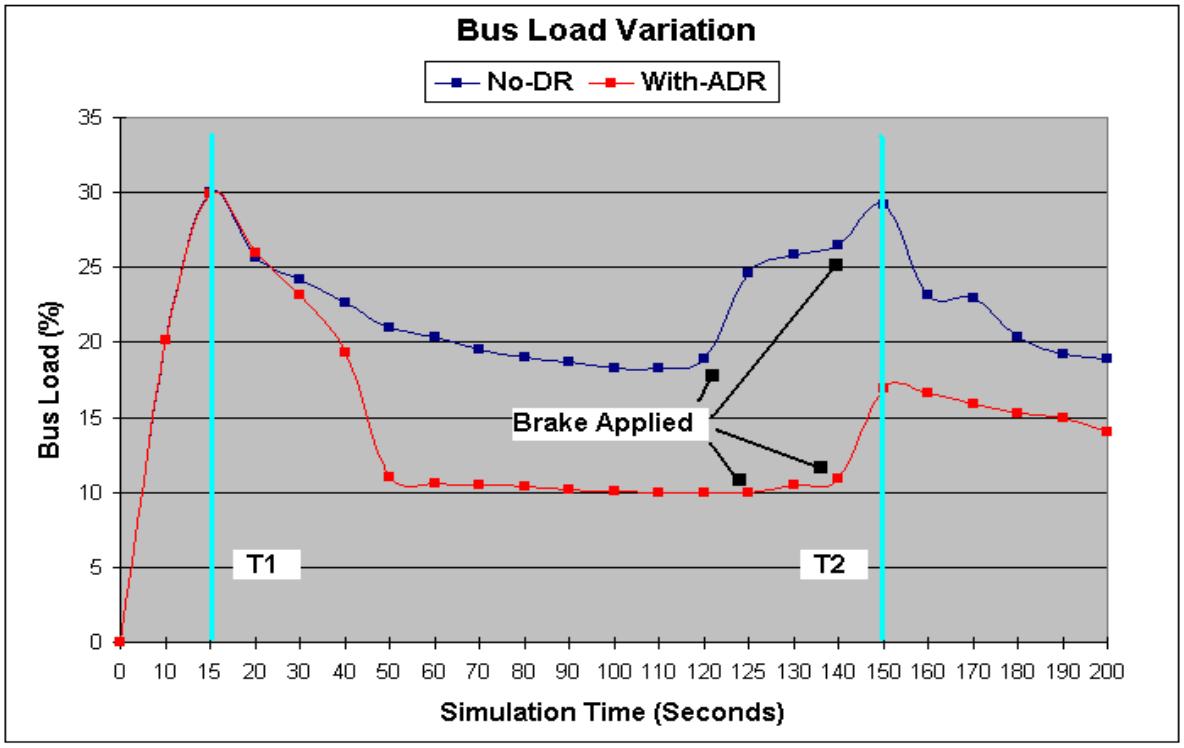

Figure 12. Bus load comparison during hard braking.

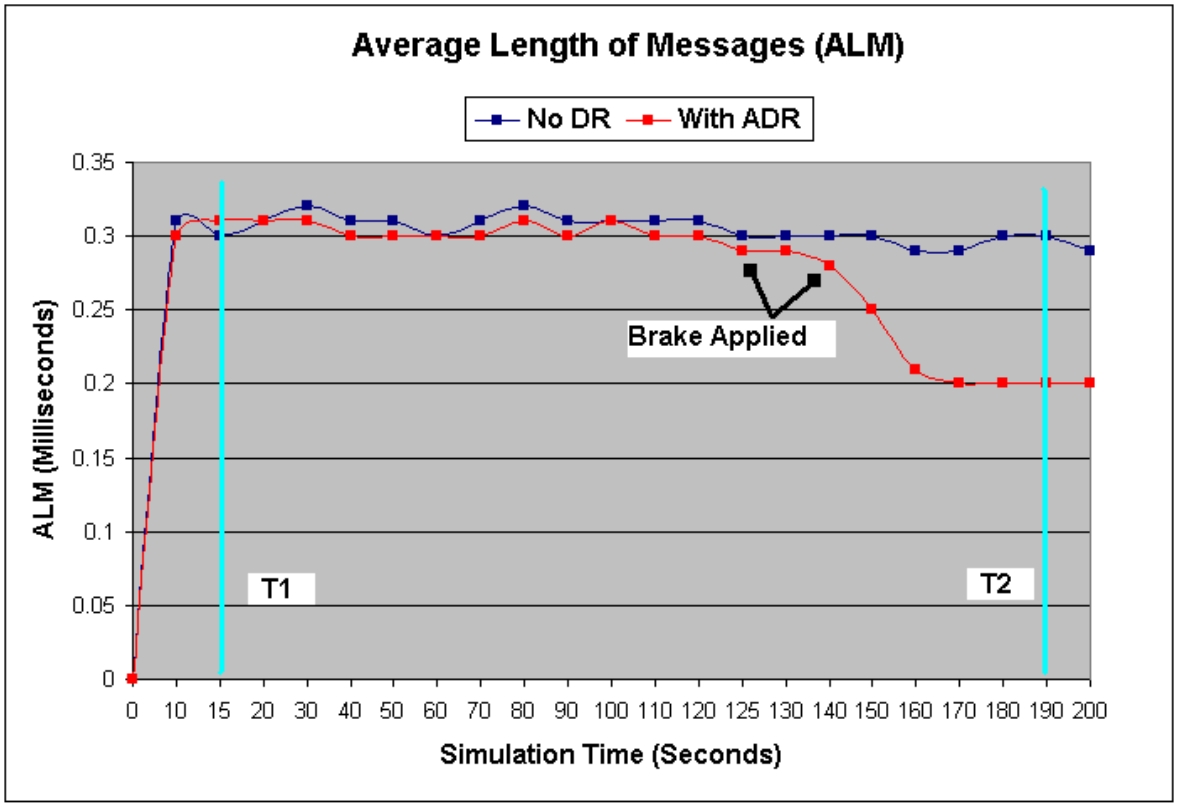

Figure 13. Average message length (ALM) comparison during hard braking.

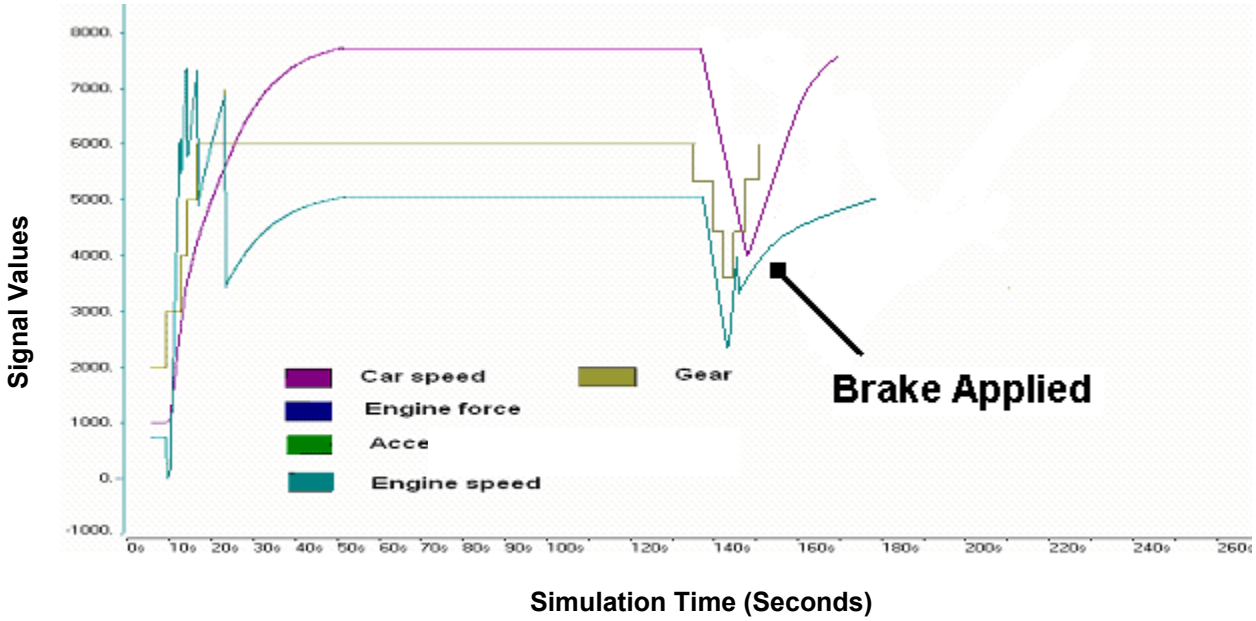

Figure 14. Variation of different signals during transient state, steady state and during hard braking or deceleration.

ACKNOWLEDGEMENTS

I would like to thank Mr. Aakash Arora for his constructive criticism and immense help in the course of this work.

REFERENCES

1. S. Channon and P. Miller, “The requirements of future in-vehicle networks and an example implementation,” SAE paper 2004-01-0206.

2. G. G. Kempf and K. Strenzl, “Robust adaptive data compression for peak-load reduction in automotive multiplexing systems,” SAE paper 941 658, pp. 1-5.

3. G. G. Kempf, M. J. Eckrich, and O. J. Rumpf, ”Data reduction in automotive multiplexing systems,” SAE paper 940 135, pp. 45-50.

4. S. Misbahuddin, S. M. Mahmud and N. Al-Holou, “Development and performance analysis of a Data-Reduction algorithm for automotive multiplexing,”, *IEEE Trans. on Vehicular Technology,* Vol. 50, No.1, pp. 162-169.

5. “Recommended practice for serial control and communications vehicle network (Class C),” SAE draft J1939, 1939.

6. SAE recommended practice, “Vehicle application layer,”, SAE J1939/71, Aug. 1994.

7. Bosch, “CAN specification Ver 2.0,” Robert Bosch GmbH, Stuttgart, Germany, 1991.

8. R. Masaki *et al.,* “Development of class C multiplex control IC,”,SAE paper 930 003, 1993.

9. Bill Waggener, "*Pulse Code Modulation Techniques: With Applications in Communications and Data Recording (Electrical Engineering)*," Kluwer Academic Publishers, 15 January, 1995.

CONTACT

Praveen R. Ramteke
Graduate Student
Department of Electrical and Computer Engineering
Wayne State University, Detroit, MI 48202
Phone: (313) 577-3855
Fax: (313) 577-1101
Email: ramteke@wayne.edu

Syed Masud Mahmud, Ph.D.
Associate Professor
Department of Electrical and Computer Engineering
Wayne State University, Detroit, MI 48202
Phone: (313) 577-3855
Fax: (313) 577-1101
Email: smahmud@eng.wayne.edu