Dimensionality reduction for commercial vehicle fleet monitoring

By

Aliakbar Baldiwala

Thesis submitted

In partial fulfillment of the requirements

For the Master's in Applied Science degree in

Mechanical Engineering

uOttawa

Department of Mechanical Engineering

Faculty of Engineering

University of Ottawa

This work is Dedicated to my loving family. To my parents, Abbasali and Fatema without whose prayers, unmatchable hard work and support, it wouldn't have been possible to pursue my dreams; To my brothers, Hamzaali and Aliasgar, whose continual inspiration and support kept my dreams alive and eventually turn them to reality.

Abstract

A variety of new features have been added in the present-day vehicles like a pre-crash warning, the vehicle to vehicle communication, semi-autonomous driving systems, telematics, drive by wire. They demand very high bandwidth from in-vehicle networks. Various electronic control units present inside the automotive transmit useful information via automotive multiplexing. Automotive multiplexing allows sharing information among various intelligent modules inside an automotive electronic system. Optimum functionality is achieved by transmitting this data in real time. The high bandwidth and high-speed requirement can be achieved either by using multiple buses or by implementing higher bandwidth. But, by doing so the cost of the network and the complexity of the wiring in the vehicle increases.

Another option is to implement higher layer protocol which can reduce the amount of data transferred by using data reduction (DR) techniques, thus reducing the bandwidth usage. The implementation cost is minimal as only the changes are required in the software and not in hardware. In our work, we present a new data reduction algorithm termed as "Comprehensive Data Reduction (CDR)" algorithm. The proposed algorithm is used for minimization of the bus utilization of CAN bus for a future vehicle. The reduction in the busload was efficiently made by compressing the parameters; thus, more number of messages and lower priority messages can be efficiently sent on the CAN bus.

The proposed work also presents a performance analysis of proposed algorithm with the boundary of fifteen compression algorithm, and Compression area selection algorithms (Existing Data Reduction Algorithm). The results of the analysis show that proposed CDR algorithm provides

better data reduction compared to earlier proposed algorithms. The promising results were obtained in terms of reduction in bus utilization, compression efficiency, and percent peak load of CAN bus. This Reduction in the bus utilization permits to utilize a larger number of network nodes (ECU's) in the existing system without increasing the overall cost of the system. The proposed algorithm has been developed for automotive environment, but it can also be utilized in any applications where extensive information transmission among various control units is carried out via a multiplexing bus.

Acknowledgement

First and foremost, I would like to express my deepest grateful and thanks to Allah, the God Almighty, for his continuous blessings. This thesis become a reality with the kind support and help of many individuals. I would like to extend my sincere thanks to all of them.

I submit my heartiest gratitude to my supervisor Dr. Dan Necsulescu, who saw the capability in me and took me under his supervision and guided me through this tough journey. I am deeply indebted to my respected supervisor, you have created the invaluable space for me to do this research and develop myself as a researcher in the best possible way. I greatly appreciate the freedom you have given me to find my own path and the guidance and support you offered when needed. This work would not have been possible without his support. His reviews and inputs always helped me improve different aspects of this thesis.

I would like to express my sincerest thanks and appreciation to Dr. Tet Yeap, for his encouragement, support, and guidance.

Further, I would like to thank my Family, their blessings made me reach here, they provided me all the strength, courage, guidance and the environment to grow which led my interest in the research field.

I am deeply thankful to Vector company for providing me the software, without which this thesis work won't become a reality. I humbly extend my thanks to all my friends who were always there for me and always helped me in my difficult time.

Finally, I am sincerely thankful to Mechanical Engineering Department of university of Ottawa for providing the space, time, and resources required to undertake this project.

Table of Contents

List of figures

List of Tables

List of Abbreviations

ABS	Antilock Braking System.
ADR	Adaptive Data Reduction
BFC	Boundary of Fifteen Compression
CAN	Controller Area Network
CAPL	Communication Application Programming Language
CDR	Comprehensive Data Reduction
DR	Data Reduction
DTC	Diagnostic trouble code
DCB	Data Compression Bit
DCC	Data Compression Code
DPCM	Delta modulation or Differential Pulse Code Modulation
DLC	Data length code
ECU	Electronic Control Unit
EGR	Exhaust Gas Residual
EDR	Enhanced Data-Reduction
ISO	International Standardization Organization
IADR	Improved Adaptive Data Reduction

ID	Identifier
LIN	Local Interconnect Network
MOST	Media Oriented Systems Transport
PDU	Protocol Data Unit
QRC	Quotient remainder compression
ROM	Read Only Memory
RAM	Random Access Memory
TWC	Three Way Catalytic convertor
WSN	Wireless Sensor Network

Chapter 1
Introduction

1.1 Motivation

These days, all the modern vehicle no longer functions on just mechanical part but instead most of the vehicle's functionality are controlled by electronic components. These electronic components are known as Electronic control unit (ECU) or Nodes. Every ECUs present inside the vehicle communicates with each other through multiplex wiring. Multiplexing is a method which combines various digital signals into one message over a shared medium.

With the introduction of automotive multiplexing over the past few decades, not only the cost and the size of the wiring for the automobile are continuously being reduced [1], but it also helped in reducing the number of electrical connections between the various Electronic controls units (ECUs) present inside the vehicle [1]. Figure 1.1 shows the typical onboard network of ECUs present inside a vehicle and figure 1.2 shows various functions of the vehicle controlled by ECUs.

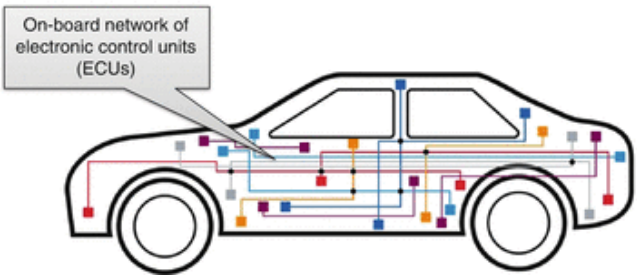

Figure 1.1 On-board network of ECUs

(Source. Available [Online]: http://canacopegdl.com/keyword/can-bus-vehicle.html)

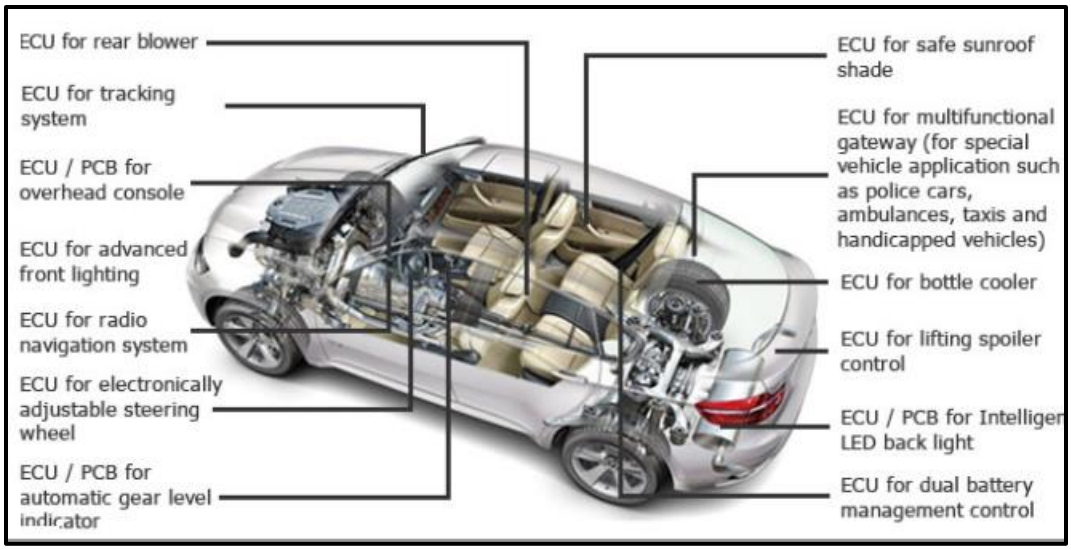

Figure 1.2 Vehicular functions controlled by various ECUs.

(Source. Available [Online]: https://www.pinterest.co.uk/pin/348395721164262328/visual-search/?x=16&y=13&w=530&h=265)

At present, an automotive contains a wide range of Electronic Control Units (ECUs) which are used to control the various functions of the engine such as Antilock braking system (ABS), misfire detection system, Exhaust gas residual (EGR) system, power train, drive by wire, telematics, etc. It has been shown by Kassakian and Perreault [2] that throughout the vehicle there are up to 70 ECUs present inside a car. The Figure 1.3 shows the average increase in the number of ECUs present inside the vehicle over the past two decades. This sudden rise in the number of ECUs in the recent years is because nowadays the factors like safety and comfort are becoming vital governing roles for the success of any Automotive industry[35][36]. To improve and control this factor, more and more number of ECUs are incorporated inside the vehicle.

Present day's vehicular electronic control units are transmitting an escalating amount of information. To transmit this data or information among various ECUs, various serial communication protocols have been proposed. During the past four decades, around forty serial

communication protocols have been proposed for vehicle multiplexing [3]. Among this, for in-vehicle networking, Controller Area Network (CAN) protocol and Local Interconnect Network (LIN) protocol are being widely used [3].

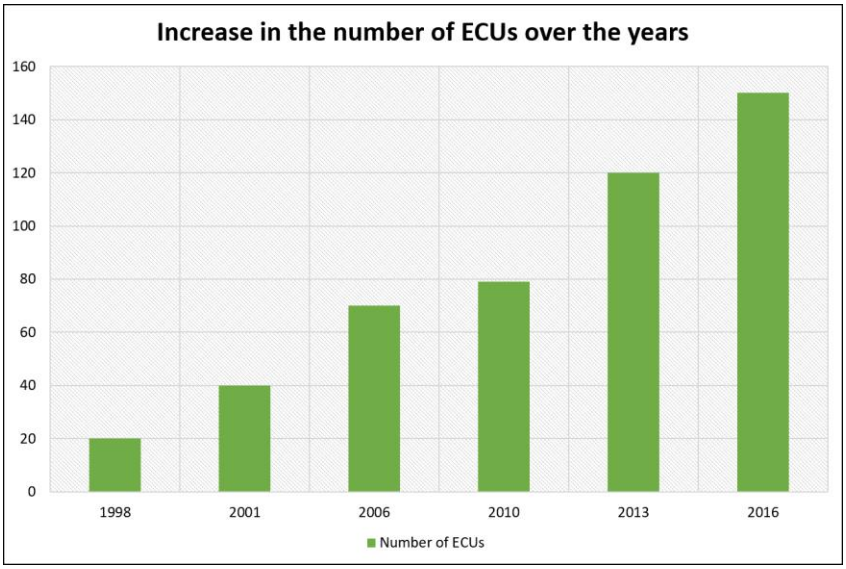

Figure 1.3 Increase in the number of Electronic control units (ECUs) over the years.

[2][34][35][36].

The Society of Automotive Engineers divided these communication protocols into three main categories namely, class A, class B, and class C [3]. At present, compared to all the above-mentioned protocols [3] and many more existing in the automotive area, Controller Area Network (CAN) remains the extensively used network protocol in a vehicle. Lawrenz [4] presented relevant information regarding the CAN protocol and its physical layer, application layer, data link layer, and testing technique.

CAN communication is defined as serial communication bus by International Standardization Organization (ISO) and was initially developed for the automotive industry to replace the complicated and intricate wiring harness with a more simplified two-wire bus. The features like

high immunity against electrical interference and the ability to self-diagnose and repair data errors earned CAN communication popularity in a variety of industries including building, automation, medical, and manufacturing. A vehicle contains as many as three CAN buses with bit rates ranging from 25 to 500 Kbits/sec [7]. The protocol for CAN communication is based on bus topology where two wires are used for the transmission over a bus. Electronic Control Units or nodes (ECUs) which are connected to the bus can either receive or send data. At a time only one node can transmit data while all the remaining nodes hear all transmissions. All nodes will invariably pick up all the traffic. However, the CAN hardware provides local filtering so that each node may react only to the messages if of interest. If more than one node tries to send its message at the same time, then the node with the highest priority is allowed to transmit its message while the other return to receive mode [4][5].

As more and more number of ECUs and sensors are incorporated inside a vehicle, The amount of data transferred on a CAN bus increases which inevitably increases the busload of CAN communication. When a CAN bus is overloaded, it gets difficult to transmit low priority CAN messages because of the increase in the waiting time. Due to this the error probability of the data transmission also increases [4]. The low priority messages can have worst case response time [6].

To transmit messages between various nodes CAN bus uses data frames. These data frames provide eight bytes of data for data transmission [16]. A typical 8-byte data frame is represented in the figure 1.4. Multiple nodes connected on the CAN bus transmits these eight bytes of data consisting of various engine related parameters, for example, engine speed, engine temperature, engine torque, pressure developed inside the engine, etc., Each message transmitted on CAN bus consist of unique message identifier. Figure 1.4 shows a message named ‘power train’ with a unique message identifier (0X786). Based on the message identifier, the priorities are given to the

messages. Where the lower the message identifier, the higher is the priority given to the CAN message [4]. If the data traffic present on the CAN bus is lower or if it is further reduced than the messages having lower priority can still access the bus before the deadline. But if there is higher traffic load, it causes the CAN bus to overload causing critical problems in a CAN bus system like message transmission delay, worst case response time of lower priority messages, etc.

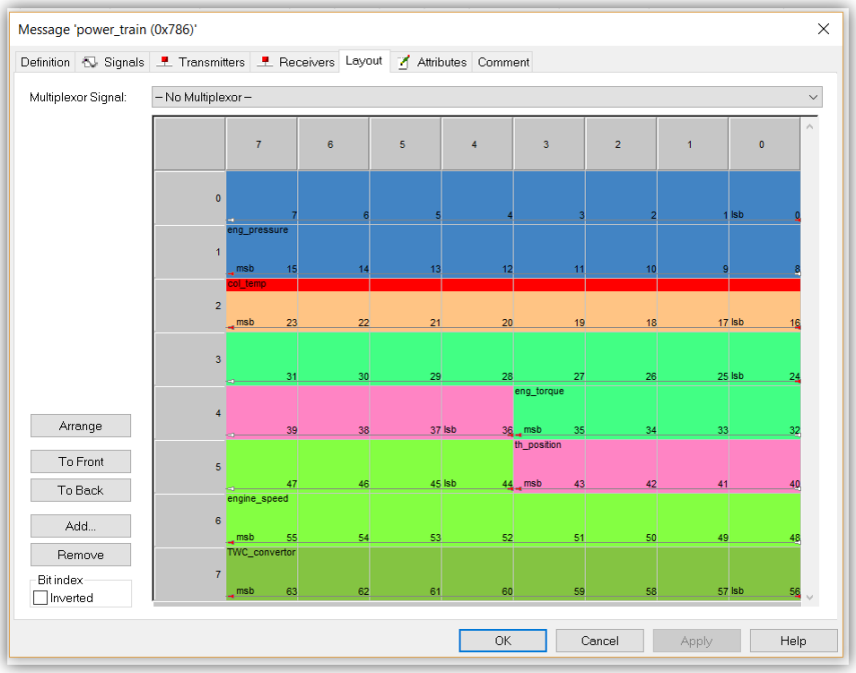

Figure 1.4. An 8-byte data frame of a message generated and transmitted by ECU.

These difficulties can be eliminated by implementing higher bandwidth CAN communication. This high-bandwidth and high-speed requirement can be achieved either by using multiple buses or by implementing higher bandwidth. But, by doing so the cost of the network and the complexity of the wiring in the vehicle increases.

Another option is to implement higher layer protocol which can reduce the amount of data transferred by using data reduction (DR) techniques, thus reducing the bandwidth usage. The data

compression can be achieved by implementing Data Reduction (DR) algorithms. By applying various data reduction algorithm to the CAN data, the field size of the data frame can be effectively reduced. This approach sends the same information in less number of bits.

The cost of incorporating data compression technique is negligible as it is limited to one-time software development. When the DR algorithms are implemented, the electronic control units transmits the same information on the CAN bus in less amount of time, which in turn reduces the overall busload.

1.2 Research Objective

As more number of Electronic Control Units are being incorporated inside the vehicle, it is reasonable to improve the existing data reduction algorithm with regards to the bus-load and compression ratio.

In our thesis work, we propose a new DR algorithm for the data compression technique. The purpose is to evaluate how the proposed algorithm achieves better data reduction compared to the previously available algorithms and for the same, A comparison of the proposed algorithm with the current existing DR algorithms is made based on the performance analysis.

1.3 Thesis Outline

Given the problems associated with the CAN communication and Research objective above, this thesis will examine the following areas of the research work in the upcoming chapters

Chapter 2: - This chapter provides a brief description about the technical background required on the various aspects related to the research work, for example Electronic control units, CAN Communication, data transmission etc. It also presents a literature review of the relevant work Regarding data reduction algorithms. Furthermore, this chapter also explains the various limitations faced by the existing data reduction algorithms.

Chapter 3: - This chapter explains about the new proposed algorithm developed during this research work. It explains in detail about how the data compression and decompression can be achieved by the proposed algorithm inside an ECU.

Chapter 4: - This chapter describes about how the Proposed CDR algorithm was developed using MATLAB / Simulink™ software [31] and CANoe™ software [32]. The impact on message latency by CDR algorithm was also checked. This chapter also uncovers a new aspect of our research work. It represents the synchronization aspect of the proposed algorithm to eliminate any error accumulation taking place inside an ECU of a vehicle.

Chapter 5: - This chapter presents the experimental results obtained by the proposed algorithm. It also provides a Performance analysis test done on the proposed algorithm. Furthermore, the proposed algorithm is compared with the existing data reduction algorithms. The comparison is done based on the parameters like Bus-load, Compression ratio and peak load associated with the CAN bus communication.

Chapter 6: - This chapter concludes the thesis work based on the analysis of the results. Future work that can be done on this approach is also described in this chapter.

1.4 Summary

In this chapter an introductory explanation about the problem associated with the data transmission inside the vehicle was provided. It also provides the objective of the research work and explains in briefly about how the thesis work is formulated in the following upcoming chapters.

Chapter 2
Literature Review

2.1 Technical background.

Over the past few decades, to curb the cost and the size of the wiring harness Automotive multiplexing was introduced inside the vehicle [1]. Multiplexing helped ECUs to communicate with each other inside the vehicle. Every ECU present inside the vehicle communicates its data via a communication channel. Over the years as more number of ECUs and sensor are getting introduced inside the vehicle, they demand high bandwidth and high-speed Communication channel to limit the increase in the busload of the channel. This high bandwidth and high-speed requirement can be achieved either by using multiple buses or by implementing higher bandwidth. But, by doing so the cost of the network, weight of wiring harnesses and the complexity of the wiring increases [8]. Another option is to implement higher layer protocol which can reduce the amount of data transferred by using data reduction (DR) techniques, thus reducing the bandwidth usage. The implementation cost is minimal as only the changes are required in the software and not in hardware.

Before moving to the Data reduction techniques let us first clear the basic concept about how the ECU works, what is CAN bus protocol, and how the data is transferred via a communication channel in the following part of this chapter.

2.1.1 Electronic Control Units (ECUs).

An Electronic control unit also known as ECU is a closed loop embedded electronic device that controls one or more electrical systems or subsystems present inside the vehicle [10]. Sensors positioned at various locations inside the vehicle will transmit data to their respective ECU. ECU will then use this information to control the various operating states of the vehicle [11]. A typical hardware configuration of ECU is shown in figure 2.1.

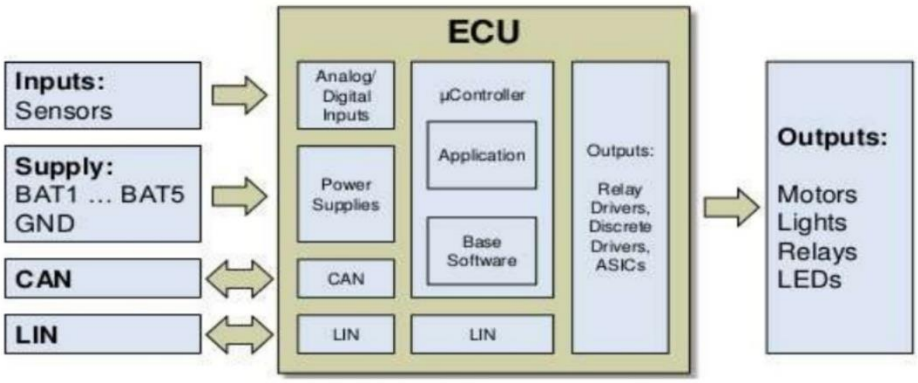

Figure 2.1 Hardware components of an Electronic Control Unit.

(Source. Available [Online]: https://www.slideshare.net/AnkulGupta2/electronic-control-unitecu)

Some of the important component which are present inside the ECUs are processor, Analog-to-digital converter, High-level digital outputs, Digital-to-analog converter, Signal conditioner, Communication chip. Every ECU is programmed in such a way that when any parametric measurement differs from its expected value, an Diagnostic trouble code (DTC) will get generated and stored inside the memory of the ECU. This Diagnostic trouble code come in handy during diagnostic applications [11].

As the number of ECUs present inside the vehicle increases every year, The need for tuning and reprogramming of ECUs software by using external programming applications also arises [12][13].

2.1.2 Communication protocol.

The ECUs of the vehicle prior to 1990s were connected via cables. The sensors whose data are required by the ECUs were directly connected to the ECUs. There are many instances where a particular sensor data is required by more than one ECU, which not only increases the redundancy between them but also adds up the total number of cable wire and weight of wiring harness. To mitigate this issue, a new communication network was introduced. This network uses bus systems for the data transmission between ECUs and Sensors or between various ECUs [14].

Over the years several communication networks have been proposed for vehicle multiplexing [3]. The society of Automotive engineers divided these communication protocols into three main categories namely, class A, class B, class C [3]. Among this, Controller Area Network (CAN) protocol, Media Oriented Systems Transport (MOST) protocol and Local Interconnect Network (LIN) protocol are being widely used for in-vehicle networking. At present, CAN protocol remains the most extensively used network protocol. International standardization organization (ISO) defined CAN communication as a serial communication bus which was initially developed for the automotive industry to replace the complicated and intricate wiring harness with a more simplified two-wire bus [3].

CAN is a multi-master serial bus standard communication used for connecting ECUs also known as nodes. All ECUs are connected with each other through two wire bus also known as a twisted pair wires with 120-ohm impedance.

The CAN bus system is divided into seven layers which are shown in the figure 2.2 below. The detailed information regarding the CAN protocol and its layers is explained by Lawrenz in [4]. Table 2.1 described below gives a brief description about the functionality of each layer.

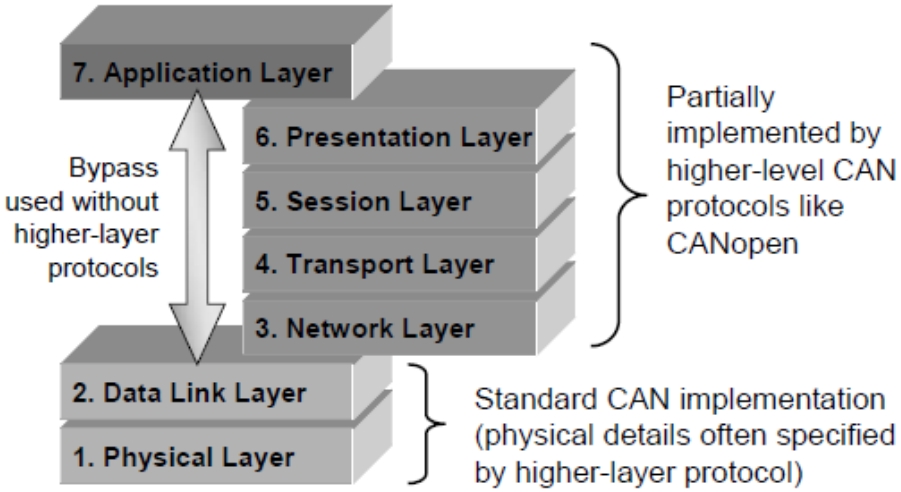

Figure 2.2 The different layers of CAN bus system [37].

Table 2.1 Functionality of different layers of CAN bus system [15].

Layer 7	**Application**	**Defines communications partners, type of service and security**
Layer 6	**Presentation**	**Converts data from one format to another, such as from a text file to a popup window display that text**
Layer 5	**Session**	**Opens, coordinates and ends conversations and exchanges of data between two applications**
Layer 4	**Transport**	**Manages error checking and verifies that packets are delivered**
Layer 3	**Network**	**Routes and forwards data to the proper destination**
Layer 2	**Data Link**	**Builds data packets and synchronizes network traffic**
Layer 1	**Physical**	**Conveys the actual bit stream across the network. Manages the hardware and the mechanical process for sending and receiving data**

In a CAN communication protocol, messages are transferred between various nodes or ECUs via CAN data frame. A frame is a packet of information that contains a complete message from a transmitter to a receiver [15]. The CAN data frame consist of a CAN identifier along with the user data frame between 0 and 8 bytes [14][16]. A typical CAN frame is shown in the figure 2.3. Depending on the size of the Identifier the CAN communication protocol can be divided into three types [17]: -

1. CAN version 1.0
2. CAN version 2.0A also known as Standard CAN. (11-bit CAN Identifier)
3. CAN version 2.0B also known as Extended CAN. (29-bit CAN Identifier)

Each message transmitted on CAN bus consist of unique message identifier. Based on this message identifier, the priorities are given to the messages where lower the message identifier, higher is the priority given to the CAN message [4]. If more than one message is sent simultaneously, then the higher priority message is sent first by an automatic arbitration process.

After completion of the first message the lower priority message then can retransmit the message. Based on the message identifier of the sender ECUs filters the incoming messages [18].

The detailed description of each component of the CAN frame is explained in [15]. Based on the application a CAN system has four different types of frames. They are Data frame, Remote frame, error frame and Overload frame.

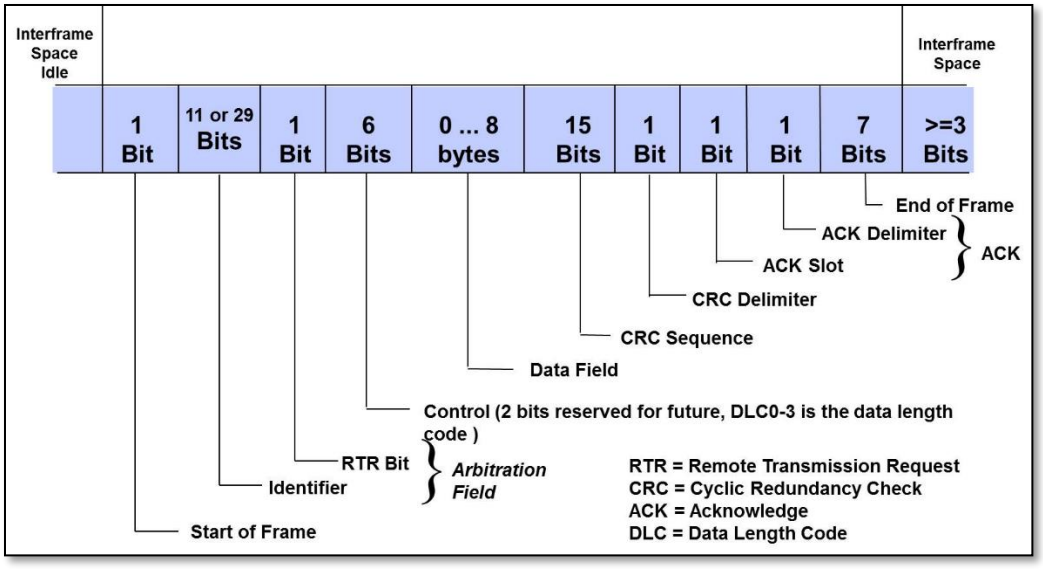

Figure 2.3 A typical CAN frame structure.

(Source. Available [Online]: https://myframe.co/can-data-frame-format/)

A data frame is a frame which transmits the user data over a CAN network. Whenever a request is made by the receiver for the data from the transmitter than the frame used for the request is known as a remote frame. To retransmit the last received message the receiver uses the Error Frame. And last is overload frame which is like an error frame, the only difference is that an Overload frame is sent by the receiver to delay the next message sent by the transmitter. The detailed description about the functionality of each frame is explained in [15][4].

2.1.3 Bit representation and Bit Encoding.

CAN communication protocol is based on bus topology where two wires are used for the transmission over a bus. This two wires are knowns as CAN_L and CAN_H [15][4][17].

A bit in a CAN protocol can be represented into two forms. It defines logic “0” as a dominant bus state and logic “1” as a recessive bus state [15][17]. The bit representation is shown in the figure 2.4.

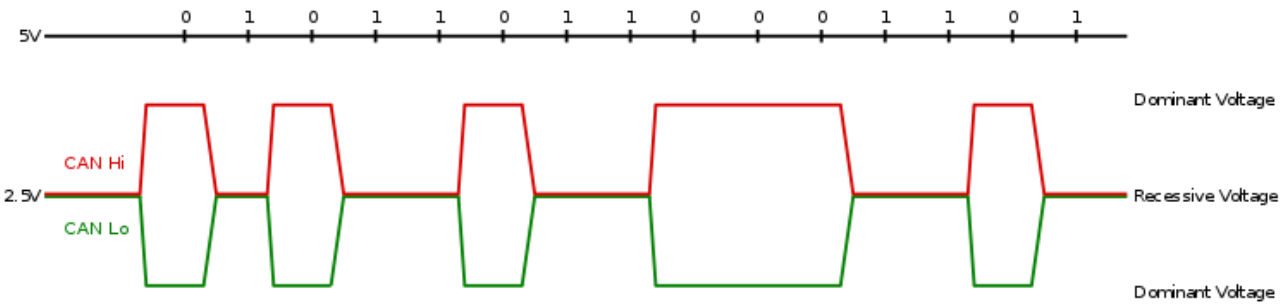

Figure 2.4 CAN bit representation.

(Source. Available [Online]: https://www.iso.org/standard/67244.html)

The Bit encoding inside the CAN communication protocol is done by a method known as “Non-return to zero encoding” [15]. The benefits of using this technique is explained in detailed in [15].

2.2 Data Reduction techniques: -

As discussed earlier that Each message transmitted on CAN bus consist of unique message identifier. Based on the message identifier, the priorities are given to the messages. where lower the message identifier, higher is the priority given to the CAN message [4]. If the data traffic present on the CAN bus is lower or if it is further reduced than the messages having lower priority can still access the bus before the deadline. But if there is higher traffic load, it causes the CAN bus to overload causing critical problems in a CAN bus system like message transmission delay, worst case response time of lower priority messages, etc.

These difficulties can be eliminated by implementing higher bandwidth CAN communication. This high-bandwidth and high-speed requirement can be achieved either by using multiple buses or by implementing higher bandwidth. But, by doing so the cost of the network and the complexity of the wiring in the vehicle increases.

Another option is to implement higher layer protocol which can reduce the amount of data transferred by using data reduction (DR) techniques, thus reducing the bandwidth usage. The data compression can be achieved by implementing Data Reduction (DR) algorithms. By applying various data reduction algorithm to the CAN data, the field size of the data frame can be effectively reduced. This algorithm sends the same information in less number of bits [21] [22].

The cost of incorporating data compression technique is negligible as it is limited to one-time software development. When the DR algorithms are implemented, the electronic control units transmit the same information on the CAN bus in less amount of time, which in turn reduces the overall busload. As more number of Electronic control units are being incorporated inside the vehicle, it is reasonable to improve the existing data reduction algorithm with regards to the bus-load and compression ratio.

There are various data reduction algorithm available for the data compression in the automotive multiplexing. The available data compression techniques can be classified into two groups. The first group is for the data reduction algorithm used for data in the form of text. The second group is for DR algorithms which can be implemented for hexadecimal data.

DR Algorithms such as simple Huffman coding, adaptive Huffman coding, arithmetic coding, higher order arithmetic coding, textual substitution coding and common data stream reference coding [19], belong to the first group and can only be implemented on textual data in the

automotive multiplexing [19]. DR algorithms such as Data Reduction (DR) algorithm [20][21], Adaptive Data Reduction (ADR) algorithm [22], Improved Adaptive Data Reduction (IADR) algorithm [23], Enhanced Data Reduction (EDR) algorithm [24], Quotient Remainder Compression (QRC) Algorithm [25], Boundary of Fifteen Compression (BFC) Algorithm [26], and Compression Area Selection Algorithm [27][28], belong to the second group.

Misbahuddin, Syed Masud Mahmud, and Nizar Al-Holou presented a broad comparison of the first group data reduction algorithms [20][21]. R Kamal and S Kelkar presented a comprehensive comparison of second group data reduction algorithms such as DR, ADR, IADR, and EDR [25]. Excluding the DR algorithm [20], all the remaining algorithms of the second group are based on CAN protocol with 11- bit message identifier. As for DR algorithm, it is based on J1939 protocol with a 29-bit message identifier. A brief description of the second group data reduction algorithms is presented below.

A. Data reduction algorithm [20] [21].

The J1939 protocol contains a reserve bit 'R' which is present in protocol data units (PDUs) [29]. This reserve bit 'R' is used by DR algorithm to indicate whether the message present inside the data field consists of parameters that are in the form of compressed data or not. The individual parameter size considered by this algorithm is assumed to be of 8-bits only. The reserve bit in this algorithm is termed as Data compression bit (DCB). In the data field, the first byte is reserved for data compression code (DCC), and every bit in the DCC is used to indicate whether the individual parameter or signal assigned to the message is compressed or not. A typical CAN frame used in this algorithm is shown in the figure 2.5.

Priority	DCB	DP	PDU Format	Destination Address	Source Address	Data
3	1	1	8	8	8	0-64

DP: Data Page DCB: Data compression bit

Figure 2.5. CAN frame format for data reduction algorithm [20] [21].

When the parameter's value doesn't change, at that time maximum compression takes place, and the data field is reduced from 8 bytes to just 1 byte only. The main drawback of this algorithm is that it considers the data compression process only if the parameter value doesn't change at all, meaning that if there is also a slight change in the value of the parameter (i.e. sensor value) than this algorithm will not implement the data reduction algorithm and will send the original value of the parameter on to the CAN bus.

Figure 2.6 shown below describes how the message is reduced by DR algorithm. As shown the very-first message is sent in its entirety, Whereas the consecutive message is in the compressed form. Only the signal whose values are changed are sent over the CAN bus to the receiving node.

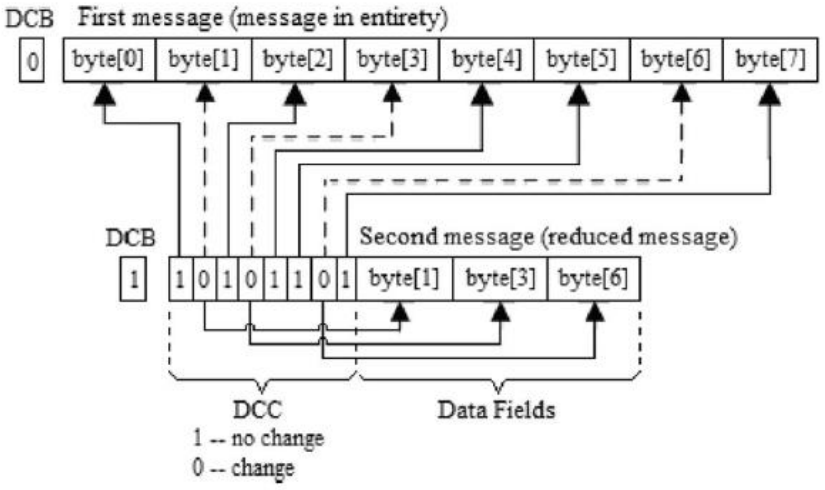

Figure 2.6 First message and the consequent reduced message of the DR algorithm [24].

B. Adaptive data reduction algorithm (ADR) [22].

To show whether the parameters assigned are compressed or not, ADR algorithm utilizes two messages, one for indicating compressed parameters and other for uncompressed parameters. The differentiating factor between these two messages is the message identifiers (ID). The message ID's of a compressed parameter has one value less than the uncompressed parameters message ID's.

The compressed parameters sent by this algorithm are in the form of delta compression. Meaning that the Parameters are first compared with its preceding value and if there is any change in its value than only the delta (i.e. difference between the current and previous value of the parameter) value of the parameter is sent over the CAN bus to the receiving node. Thus, a message can have uncompressed, fully compressed or delta compressed parameters.

This algorithm also uses the first byte of a data field for DCC to indicate whether the parameters are partially or fully compressed. Figure 2.7 shown below describes how the message is reduced by the ADR algorithm.

The main drawback of this algorithm is that if the signal delta value goes beyond the length of the assigned field (Assigned field = 5), then in that case the original values of all the signals are transmitted rather than the delta compressed value of the CAN message.

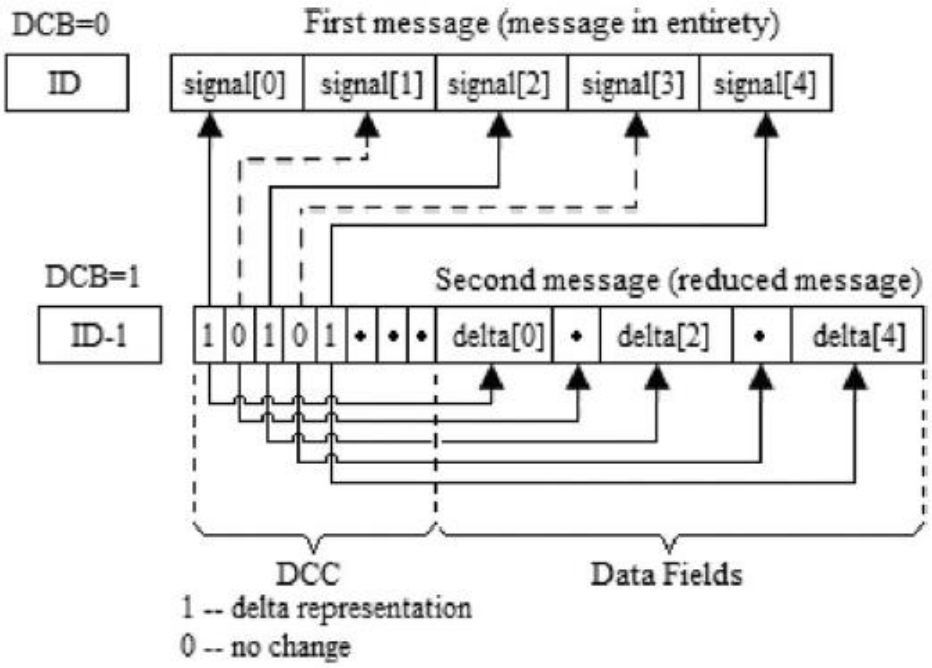

Figure 2.7 First message and the consequent reduced message of the ADR algorithm [24].

C. Improved adaptive data reduction (IADR) algorithm [23].

Improved adaptive data reduction (IADR) algorithm is a further development of the ADR algorithm. Unlike ADR algorithm this algorithm utilizes single message only to send the uncompressed or compressed parameters. The first bit of the data field is used to indicate if the message enclosed is in the compressed or uncompressed form. This bit is Data Compression Bit (DCB). This algorithm also utilizes two extra bits per parameter to indicate whether it is fully or partially compressed. Figure 3 Shown below describes how the message is reduced by IADR Algorithm. The compression in IADR algorithm takes place in the following three stages [23] [24]:

Stage 1: - The first bit of the data field also known as DCB, indicates if there is at least one compressed signal in the message or not.

Stage 2: - After the DCB, comes the DCC byte in the Data field which indicates if the corresponding signal is in some form of compression or the original signal is included instead. Each bit present in the data compression code represent individual signals.

Stage 3: - At this stage, the bit preceding each compressed signal is used to describe whether the signal is in delta compressed form or there is no change in the signal value.

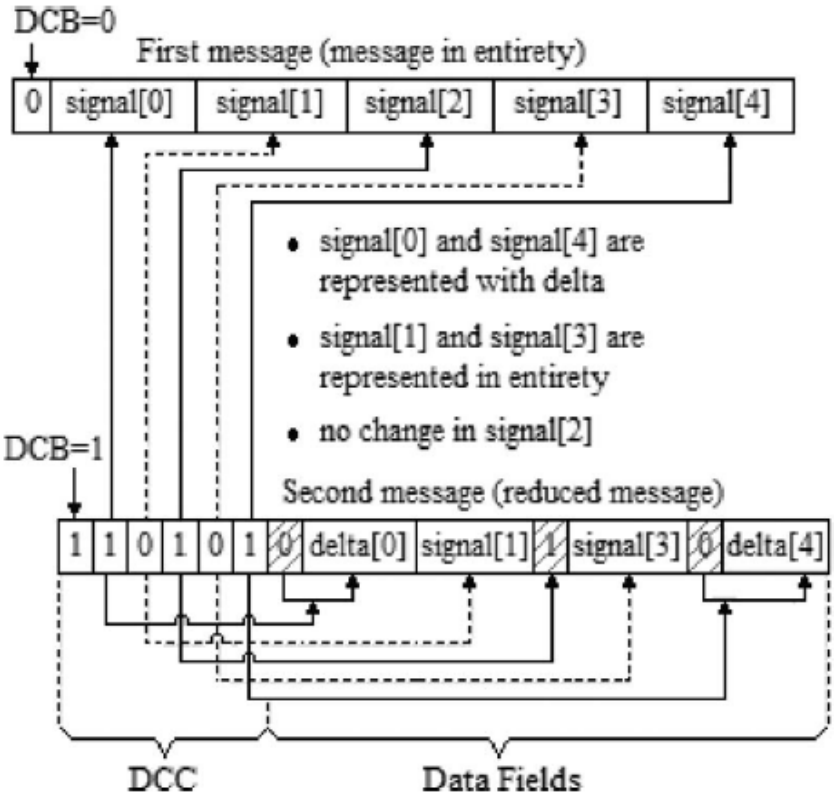

Figure 2.8 First message and the consequent reduced message of the IADR algorithm [24].

D. Enhanced data reduction (EDR) algorithm [24].

IADR algorithm and EDR algorithm have many similarities, but there are two significant differences between them.

1. EDR algorithm eliminates the use of DCB utilized for indicating whether the message is in compressed form or not and utilizes the data length code (DLC) used for indicating the size of the message in the data field. Like IADR it uses two extra bits per parameter to indicate full or partial compression. These two bits have one bit from the DCC and another bit known as reduction type bit placed just before the compressed signal.
2. EDR algorithm implements a protocol which checks whether the signal or parameter is fit for compression (i.e.,>= 5 bits) or not. Signals having bit length greater than 5 is masked in the SDN type group. Whereas signals having bit length less than 5 is masked into SN group. An example of this is shown in the figure 2.9. signals belonging to SDN groups can be either fully compressed or partially compressed or in uncompressed form, whereas signals belonging to the SN group are either fully compressed or not compressed at all.

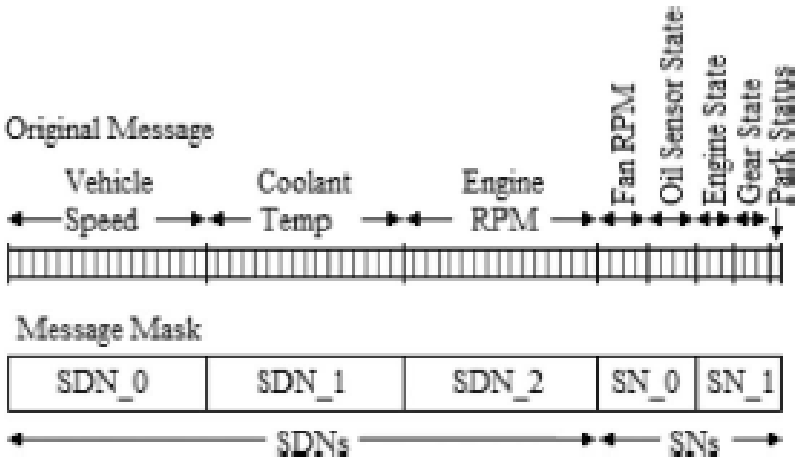

Figure 2.9 An example of the original message and its mask with SDN and SN type groups [24].

E. Quotient Remainder Compression (QRC) algorithm [25].

Similar to the EDR algorithm, QRC algorithm also allocates the first byte of a data field to DCC. There is only one significant difference between QRC and EDR algorithm, i.e., EDR algorithm

implements the delta method for data compression whereas QRC algorithm employ's arithmetic division method for data compression. QRC utilizes extra two bits per parameter to indicate the quotient value of the parameter. The rest of the protocol of QRC algorithm is very much similar to EDR algorithm.

F. Boundary of fifteen compression (BFC) algorithm [26].

BFC algorithm is again similar to EDR algorithm. The main difference is the compression range for the parameters. BFC assumes the compression range of ±15 and any signal value changed (delta value) within this range is considered for data compression. Also, different from EDR algorithm, BFC algorithm does not use the first byte of the data field for DCC and instead the bits indicating the data compression is placed preceding the parameters itself.

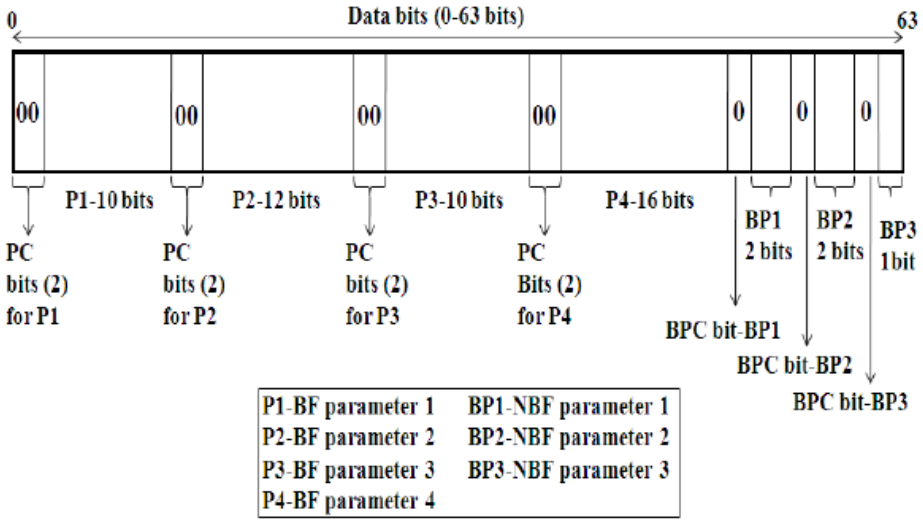

Figure 2.10 Parameters sent without any compression [26].

Figure 2.10 shown above describes the parameters which are sent in uncompressed form. Whereas figure 2.11 shows the compressed form of message generated by BFC algorithm. Here parameter

P1, P2, P3, and P4 are compressed using BFC algorithm. BP1 sent completely, BP2 and BP3 are fully compressed [26].

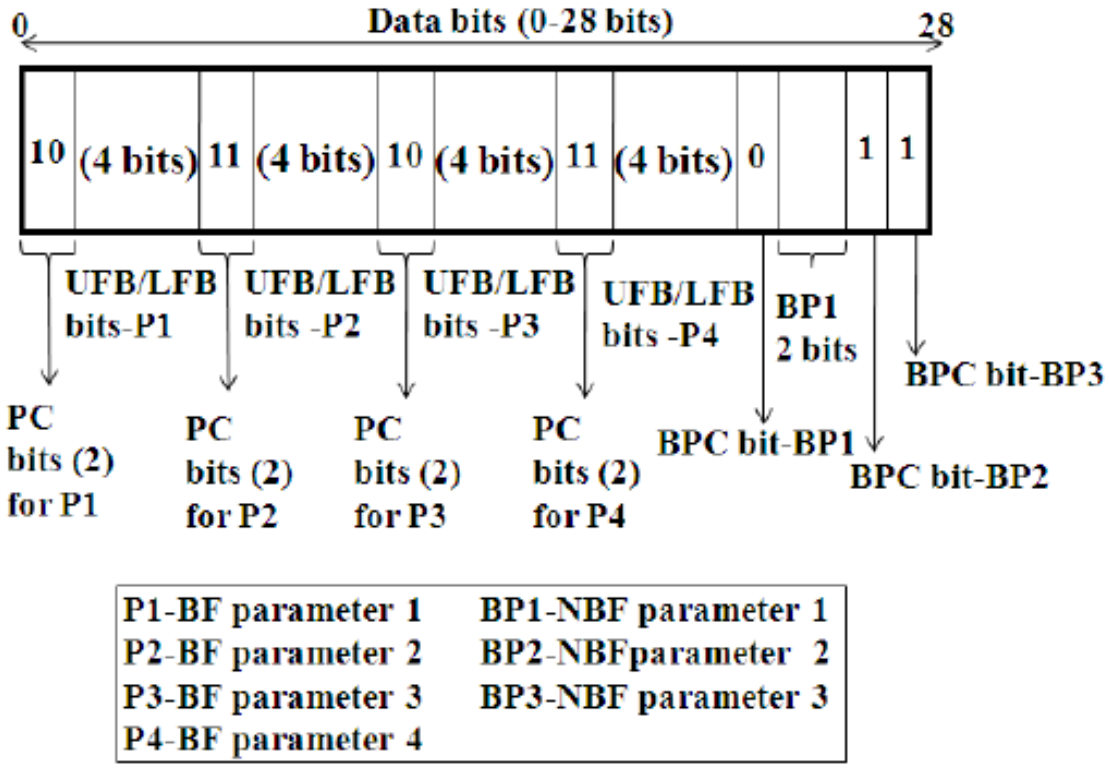

Figure 2.11 compressed message using BFC algorithm [26].

G. Compression area selection algorithm [27].

This algorithm eliminates the need for predicting the maximum value of the difference in the successive parameters. It assumes that the data field always consists of eight signals having 8 bits each. It still needs additionally 8 bits due to the header bits which indicate if the parameters are either in compressed form or uncompressed form.

2.3 Summary

In this chapter, a brief description about the technical background required on the various aspects related to the research work, for example Electronic control units, CAN Communication, data transmission etc. was provided. This chapter also presented the literature review of the relevant work based on the data reduction algorithms. Furthermore, this chapter also explained the various limitations faced by the existing data reduction algorithms.

Chapter 3

Proposed Comprehensive data reduction (CDR) algorithm.

3.1 Limitation faced by the existing Data reduction algorithms.

Every data reduction algorithm discussed earlier utilizes the data compression code (DCC) and header bits or data compression bit (DCB) for indicating whether the message is fully compressed, partially compressed or uncompressed. For every parameter enclosed inside the message there exist an overhead of 2 bits to indicate whether it is compressed or not. Due to the use of these extra bits, some overhead exists in every data reduction algorithm, and as the number of parameters enclosed in the message increases so is the number of the extra bits enclosed along with the message, causing higher overhead.

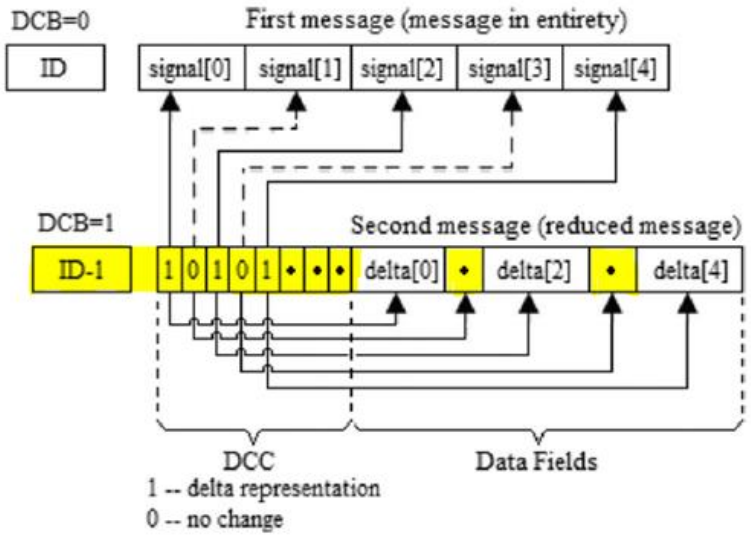

Figure 3.1 Overhead bits associated in a CAN message [24].

For example, a message can consist of as many as 8 parameters enclosed inside it. Thus for 8 parameters there exist 16 bits of overhead in the message which are just used to indicate whether the parameter is fully compressed, partially compressed or in the uncompressed form, taking up to 2 bytes of data from the available 8-byte data field. Due to this overhead, the message sent may be even greater than the original uncompressed message. Figure 3.1 illustrates how the extra overhead bits gets associated into a CAN message. The highlighted region shown in the figure 3.1 is the overhead bits.

Apart from Compression Area Selection algorithm, all other algorithms assume a value of compression range for the parameters, for example, BFC algorithm implements a range of -15 to +15. Any signal value changed (delta) within this range is considered for data compression or else signal in entirety is send, i.e., in uncompressed form. There are many instances where the delta value of the parameter has higher value than the compression range assigned for the parameter, thus the parameter is sent in uncompressed form which inevitably increases the bus load associated with the CAN bus system.

For compression area selection algorithm, there are no assumptions made for the restriction of the maximum compression range. But unlike other algorithms here all signals are assumed to be of 8 bits. Another drawback of Compression Area Selection algorithm is that once the delta values of the signals are obtained, the signals are arranged in such a way that the same number of bits will be assigned to each signal enclosed inside the message. The signal having the highest value determines the number of bits allocated to each parameter, thus highest number of bits is assigned. Therefore, in many cases, the signals having lower values are also sent with the more number of bits.

3.2 Problems addressed by the new proposed algorithm.

To eliminate the above drawback of overhead bits associated inside the CAN message which is sent over the CAN bus system, we propose a new algorithm named as “Comprehensive Data Reduction (CDR) algorithm”.

In CDR algorithm, every message sent will be in the form of delta compression and there will be no limitation for the maximum compression range. Thus, every signal will be in delta compressed form, which eliminates the use of any overhead bits (DCC/DCB/header bits) used to indicate whether the signal is compressed or not.

Another advantage which CDR algorithm offers is that the signal bit length of every parameter will be selected based on the delta value of the signal.

For example, in case of BFC algorithm, it doesn’t matter what is the value of the parameter if its value is less than 15 than the delta value of the parameter is sent and the bit length assigned to the parameter is 4 bits. One of the drawback in this method is that if the parameter value is less than 4 than in that case also the bit length assigned to the signal is 4 bits. Any value of parameter which is less than 4 can also be represented by 2 bits only instead of 4 bits. Thus, in the example described above, 2 unwanted bits (when parameter value ≤ 4) are also sent by transmitter node on the CAN bus system.

While in case of proposed CDR algorithm, the bit length of any parameter is decided based on the delta value of the parameter. thus, no extra bits will be sent on CAN message and the data field of the message will consist of only useful information. This is explained in the figure 3.2 which shows a comparison of the Bitlength assigned to the parameter by the proposed CDR algorithm and the bitlength assigned to the parameter by various existing data reduction algorithm.

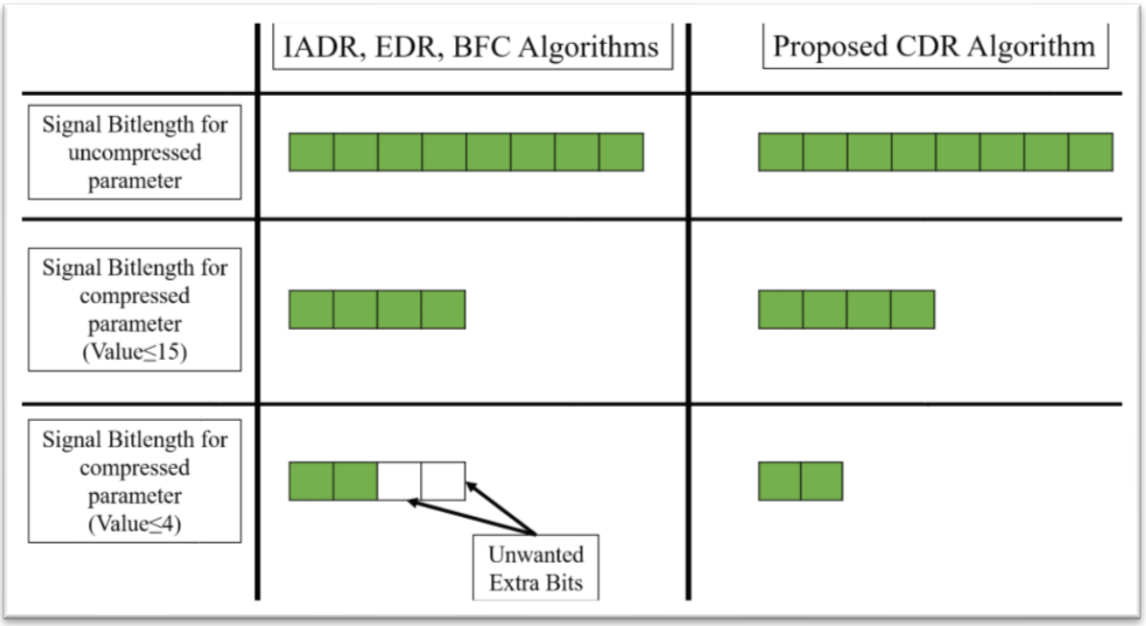

Figure 3.2 Comparison of the bitlength assigned to the parameter by various Data reduction algorithms.

3.3 Comprehensive Data reduction (CDR) algorithm.

The comprehensive data reduction algorithm is explained in the subsequent discussion.

3.3.1 Norms made for this algorithm

1. The data sent by the CDR algorithm over the CAN bus system is always in the delta (compressed) form.
2. The ECU or node which is transmitting the data over the CAN bus is termed as transmitter node whereas the ECU or node which receives the data from the CAN bus is termed as receiver node.
3. Each node can either transmit the data or receive the data at any given time.

4. The data compression takes place on the transmitter node whereas the data decompression takes place at the receiver node.
5. Every node connected to the CAN bus will have two buffers, where the messages can be temporarily stored. These two buffers are termed as transmit buffer and receiver buffer. Before transmitting any messages from the node, the transmit buffer will be utilized whereas after receiving the messages from the CAN bus, receiver buffer will be used by the node or ECU. The transmitter buffer temporarily stores the preceding value of the messages to be transmitted, whereas the receiver buffer will hold the value of the preceding received messages.
6. The sizes of the buffers are equal to the size of transmitted and received messages. The storage space required by the transmit buffer and the receiver buffer is insignificant compared to the space of read only memory (ROM) and Random-access memory (RAM) available in the todays ECUs microcontrollers. For example, a typical ECU which is used for vehicular application transmit very few different types of messages.

   The CAN message consist of 8 bytes of data field. If an ECU can transmit or receives 10 different types of messages then in that case, a node or ECU will need around 160 bytes of random access memory for their receiver and transmit buffers which is insignificant compared to today's microcontrollers.

3.3.2 Data compression and message encoding.

In CDR algorithm, the data compression and the message encoding takes place at the node or ECU which is transmitting the data over the CAN bus system.

When the engine of the vehicle is started, the sensors placed at various locations inside the vehicle will record the data based on their surroundings and will transfer this data to their corresponding ECUs [11]. ECU will then convert this raw data obtained from the sensors into a digitalized form (binary form). This digitalized data of the sensor will then be arranged in the data field of the CAN frame by the Electronic control units, and thus the CAN message will be generated which is then transferred via CAN bus to the receiver node or ECU who requested for this data. Figure 3.3 illustrates how the various ECUs or nodes present inside the vehicle communicates with each other via CAN bus system. As shown in the figure, the ECU-1 transmits the message over the CAN bus system whereas ECU-2 receives the message.

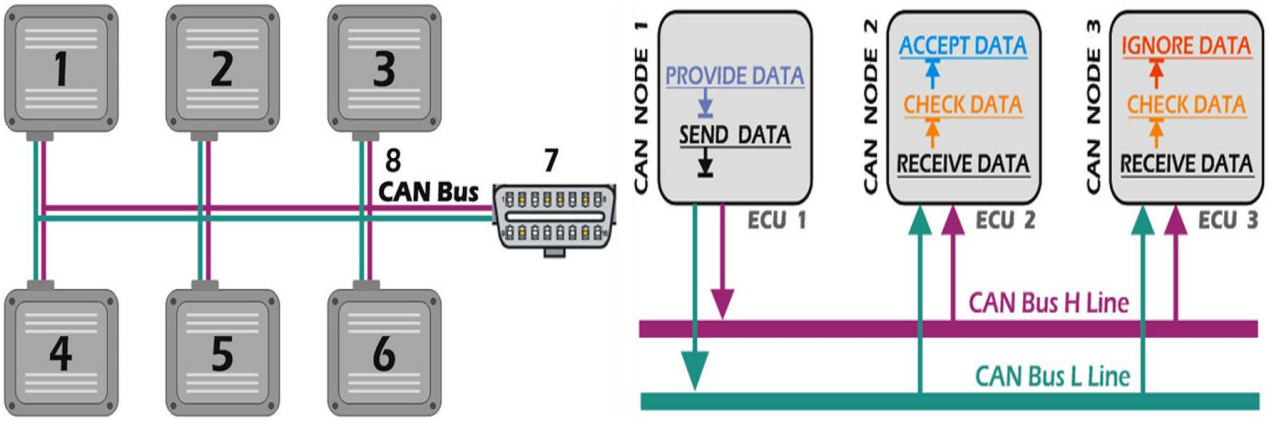

Figure 3.3 ECUs interacting with each other via CAN bus system.

(Source. Available [Online]: https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/language/en)

The first message which is generated or recorded by the ECU is first stored into the transmit buffer and then is transferred over the CAN bus. The next message generated after that is first compared with the preceding message value stored in the transmit buffer. During comparison, the delta value for the signal is obtained by subtracting the current signal value from its previous value which is

based on the method known as “Delta modulation or differential pulse code modulation (DPCM)” [30]. This delta value obtained for the parameter will be the one which is transmitted on the CAN bus. An example illustrating this technique is shown in figure 3.4. As shown in the figure the first message is stored in the transmit buffer and is transmitted over the CAN bus. The following message generated by the ECU is then first compared with the stored message and the delta value is obtained. This delta value is than transmitted over the CAN bus system.

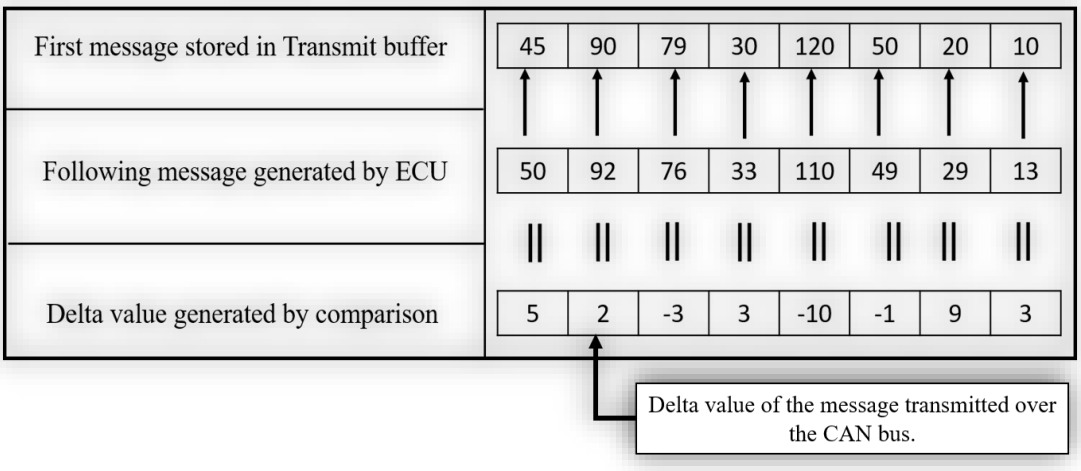

Figure 3.4 Delta comparison technique.

The flowchart shown in figure 3.5 depicts the procedure for Data compression. Here message $M_{(t1+t2)}$ is the current message value whereas M_{t1} is the previous message value stored in the transmitter buffer. Also “t_i” indicates the time at which the message is generated where i = (1,2, 3…. n). The steps shown in the figure 3.5 are the steps followed by the proposed CDR algorithm. After obtaining the delta value for individual parameters, the next step is to determine the signal bit length of the individual parameters.

The flowchart from figure 3.6 depicts the protocol followed by this algorithm for the consideration of the signal bit length. For the discussion purpose, let’s assume that message M_i is to be

transmitted by the node. This message comprises of S_i signals where i = (1, 2, 3, 4.). Thus, message M_i consists of M_i= {S_1, S_2, S_3 S_4}.

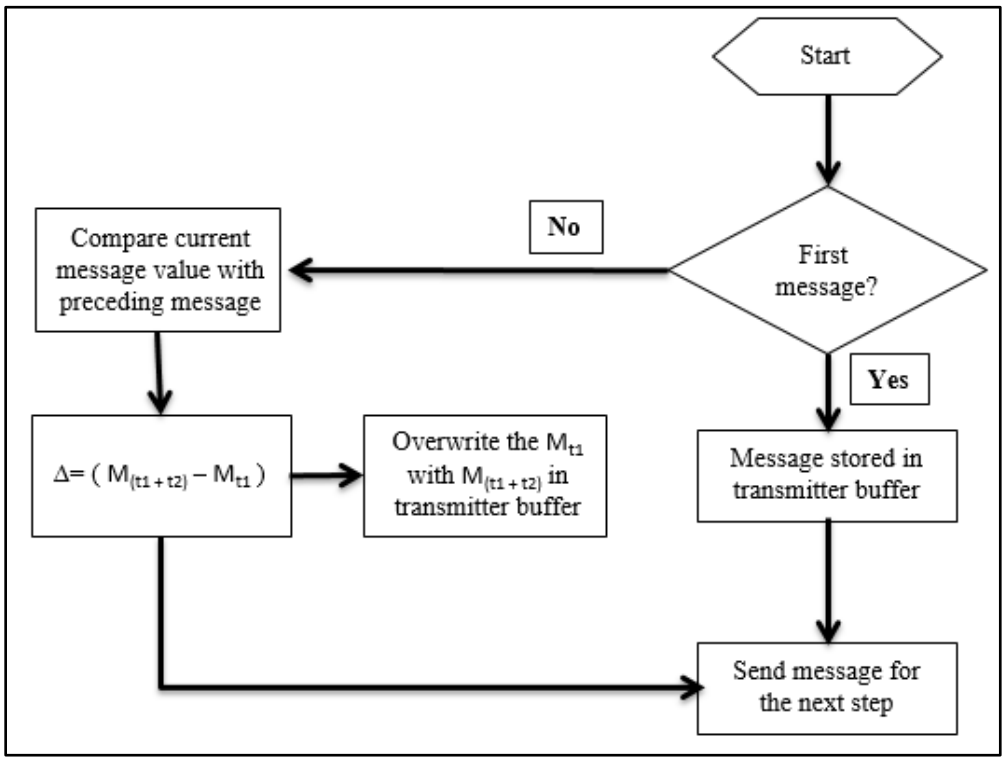

Figure 3.5. Data compression algorithm. [39]

After converting the original values into delta form, the signals follow the protocol shown in figure 3.6. The delta value of the signal S_i obtained by the comparison of current and preceding message is indicated by "Δ (Si)". Equation 1 shown below is used to decide the number of bits (signal bit length) allocated to each signal encoded inside the CAN message.

$$-(2^a - 1) \leq \Delta(Si) \leq 2^a \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

$$n = a+1,$$

Where

- Δ (Si) = Delta value of the signal.

- a = {1,2,3 …. N}
- n = signal Bitlength

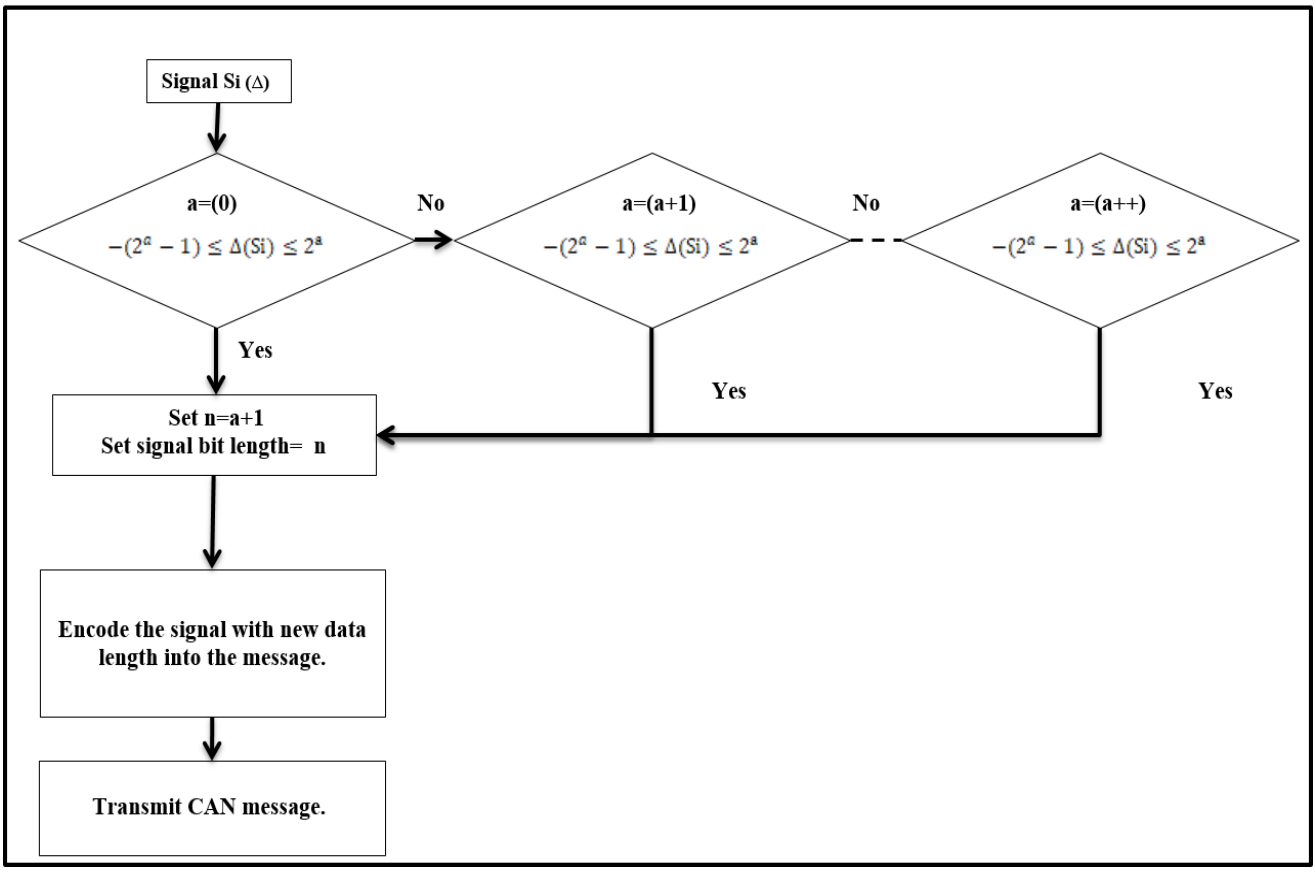

Figure 3.6. Signal encoding into the message frame. [39]

Based on the delta range of the signal, the respective number of bits "n" is allocated to the corresponding signal. To illustrate the process of Signal Bitlength allocation, let us consider an example where let's assume that the signal delta value obtained after the compression is 13. i.e. Δ(Si) = 13. Based on the algorithm explained in the figure 3.6. the value of "a" comes to be equal to 4. Thus, the Bitlength for this signal value will be equal to "n=a+1". i.e. n = 5. Which means that the number of bits required by this signal to represent the delta value is 5.

"n= a+1" is considered because the least significant bit of each parameters is used for indicating the sign of the delta value. After obtaining the bit lengths, these signals are then enclosed in the message and are transmitted on the CAN bus. This is how the Data compression and message encoding takes place at the transmitter node by the proposed CDR algorithm.

3.3.3 Data decompression and message decoding.

In CDR algorithm, the data decompression and the message decoding takes place at the node or ECU which is receiving the data from the CAN bus line. The very first message received by the receiver node is first stored into the receiver buffer and the data is obtained for the further use. The following message which is received by the receiver node is than compared with the preceding message stored in the receiver buffer. For decompression of the message, the receiver node executes the following steps shown in the figure 3.7.

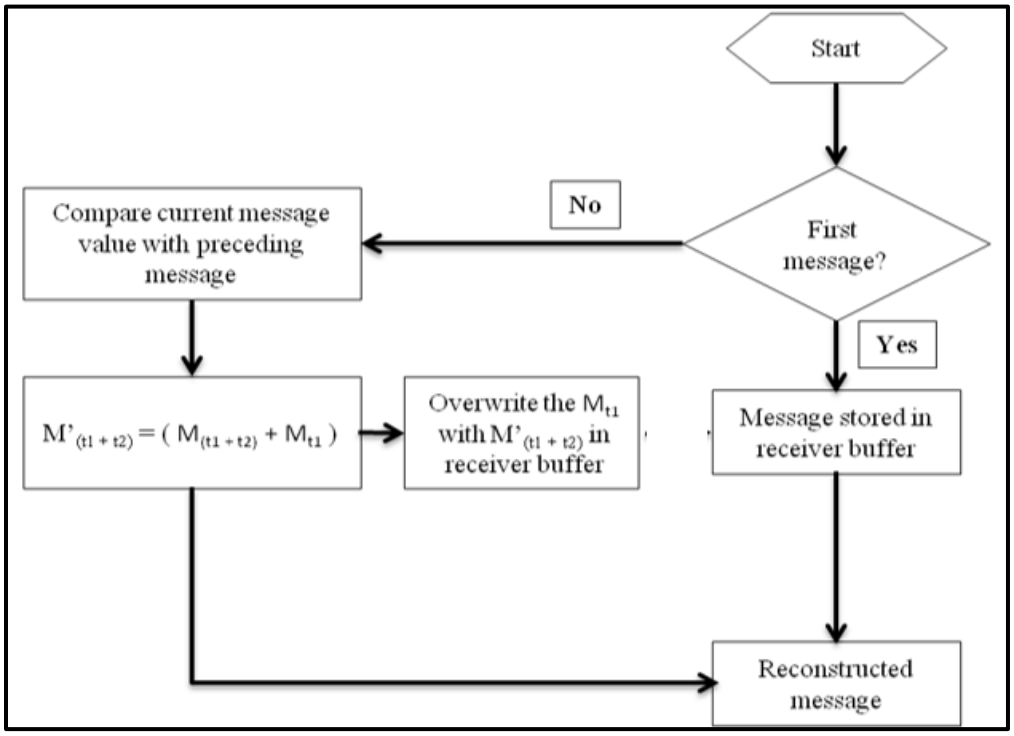

Figure 3.7 Data decompression and message decoding. [39]

As shown in the figure 3.7, let's consider that the first message received by the receiver node is denoted by "M_{t1}". This message is stored in the receiver buffer. The following message which is received by the node is denoted by $M_{(t1+t2)}$. Here "t_i" indicates the time at which the message is received where i = (1,2, 3…. n). This message $M_{(t1+t2)}$ is than compared with the preceding message M_{t1} and the full signal value is constructed back. At the end of the cycle, the receiver node updates the value of receiver buffer with this newly constructed message ($M'_{(t1+t2)}$) by overwriting the previous message. For all the subsequent received messages the same procedure is followed.

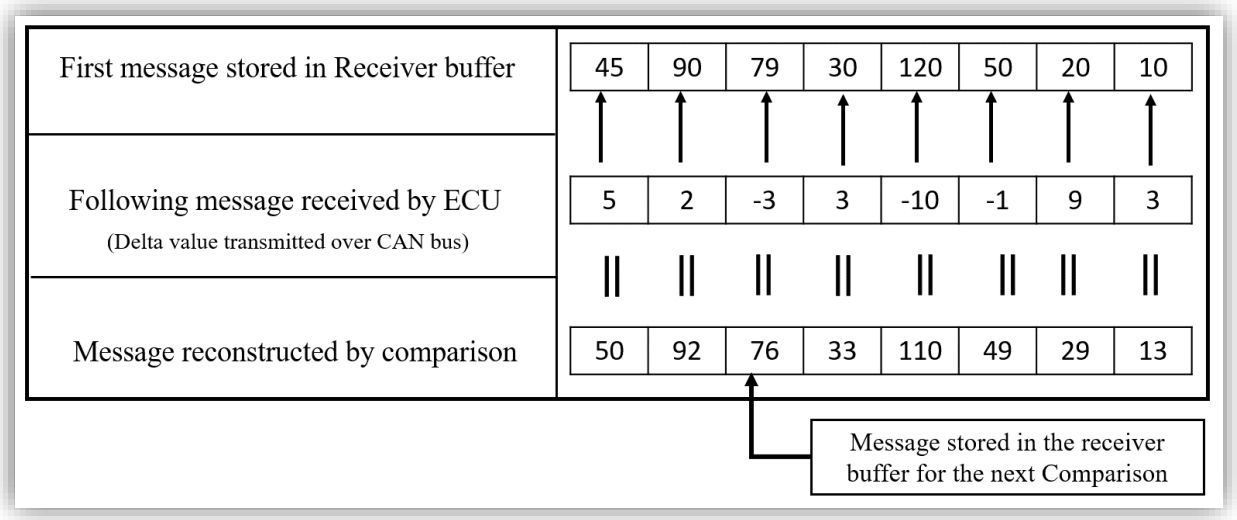

Figure 3.8 Message reconstruction technique.

An example illustrating this technique is shown in figure 3.8. The signal Data considered in this example is same as that of the example considered in figure 3.4. As shown in the figure the first message is stored in the receiver buffer and the data is obtained for the further use. The following message received by the ECU is then first compared with the stored message and the signal value are reconstructed. This reconstructed message is same as the original message which was intended to be transmitted on the CAN bus if data compression technique was not applied.

At the end this reconstructed message is then stored into the receiver buffer and the same steps are followed for the next received message. Thus, in this way the data transmission over the CAN bus is drastically reduced resulting in the reduction of the CAN busload.

3.4 An algorithmic approach for CDR technique.

To clearly understand the concept behind the CDR algorithm, an algorithmic approach is explained in this following article.

3.4.1 Algorithm for Data compression and message encoding.

In CDR algorithm, The data compression and message encoding takes place in two parts as explained in figure 3.9 and figure 3.10.

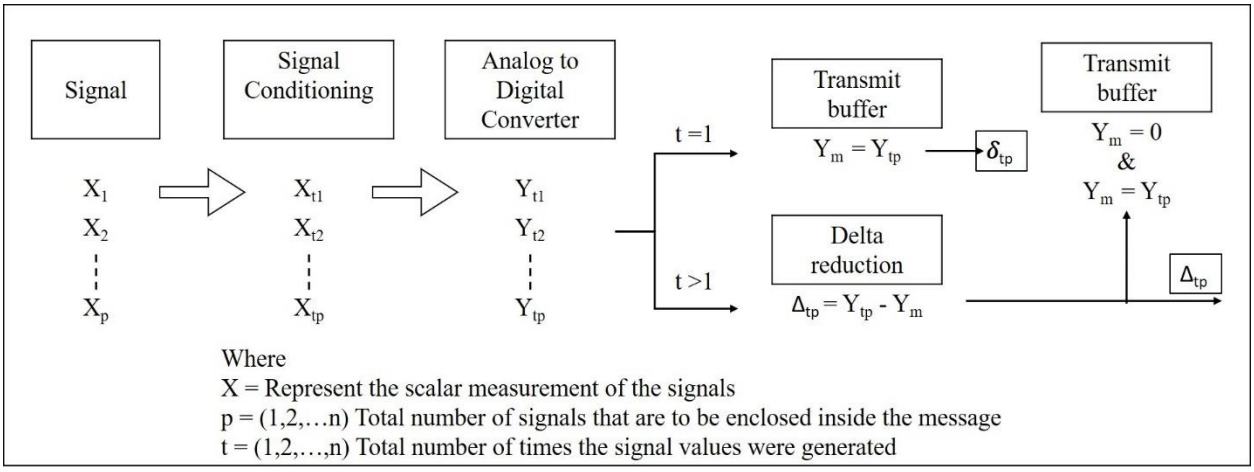

Figure 3.9 an algorithm for data compression

Figure 3.9 explains how the data compression takes place at the transmitter node or ECU. The raw data measured by the sensors are represented by X_p. This raw data is first passed through the Analog signal conditioner and calibration device present inside the ECU. The Signal conditioning

device is basically used to eliminate the noise and other undesired interferences caused in critical and weak signals. Its primary use is for manipulating an analog signal in such a way that it meets the requirements of the next stage as in our case Analog to digital converter. Thus, the signal obtained after the signal conditioning device are now represented by X_{tp}. This analog signal values (X_{tp}) are then converted into digitized form by analog to digital converter and are represented by Y_{tp}.

Once the signal values are converted into digitized form, the algorithm now checks whether the signal is generated for the first time (t=1) or not (t>1). If the signal was generated for the first time than in that case the algorithm will directly store the value of the signal (Y_{1p}) in the transmit buffer and will then send the Signal value as represented by δ_{tp} directly for the next step. The signal value stored in transmit buffer is represented by Y_m where m represents the total number of signals that are to be enclosed inside the message. If the signal value was not generated for the first time (t>1) than in that case the CDR algorithm will compare the current signal value (Y_{tp}) with the value stored (Y_m) in the transmit buffer and carries out the delta reduction ($\Delta_{tp} = Y_{tp} - Y_m$) technique. The new value (Δ_{tp}) obtained will then be transferred for the next step of the algorithm and the current signal value (Y_{tp}) will be stored into transmit buffer (as Y_m) after erasing the preceding signal value.

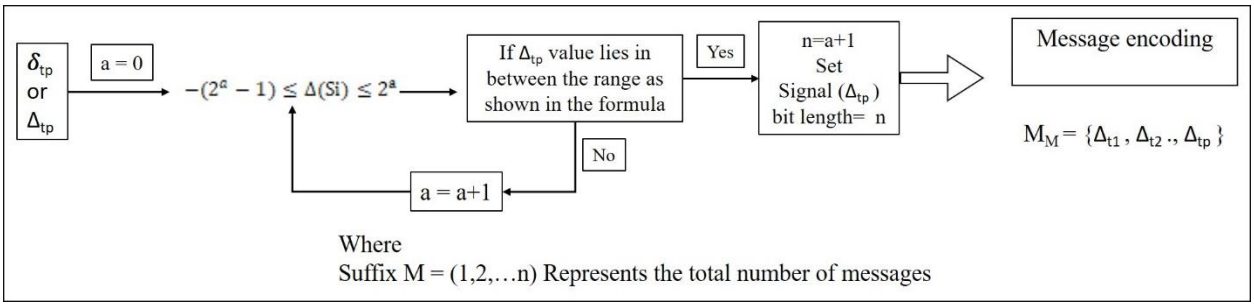

Figure 3.10 Algorithm for calculating signal bit length and message encoding

After obtaining the delta value of the signals the next step which CDR algorithm follows is explained in figure 3.10. Based on the delta value of the signal, the value for parameter "a" is selected using the formula described in equation 1 and the procedure shown in figure 3.6 and figure 3.10. The value for parameter "a" then determines the bit-length (n) of the signal.

Once the bitlength of the signals are obtained based on their delta values, these signals are then enclosed in a CAN message format (M_M) and are then transferred over the CAN bus system to the receiver node or ECU.

3.4.2 Algorithm for Data decompression and message decoding.

The data decompression and message decoding take place at the receiver node. The data decompression is shown in figure 3.11. Firstly, the delta value (Δ_{tp}) of the signals are obtained from the received message (M_M).

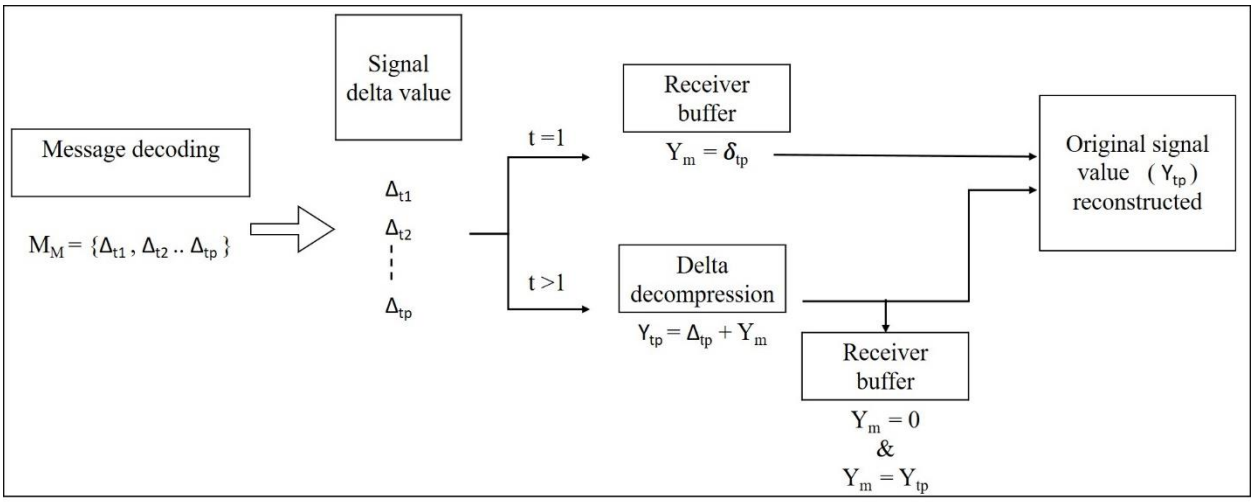

Figure 3.11 Algorithm for message decoding

Once the signal delta values are obtained, the algorithm now checks whether the signal is received for the first time (t=1) or not (t>1). If the signal was received for the first time (t=1) than in that

case the algorithm will directly store the delta value of the signal (δ_{tp}) in the receiver buffer (as Y_m) and first original signal value will be generated directly based on the delta value of the signal.

If the signal value was not received for the first time (t>1) than in that case the CDR algorithm will compare the current signal delta value (Δ_{tp}) with the value stored (Y_m) in the receiver buffer and carries out the decompression ($Y_{tp} = \Delta_{tp} + Y_m$) technique. The signal value (Y_{tp}) obtained by decompression technique will be equal to the original signal value that was intended to be transferred over the CAN bus system. The current signal value (Y_{tp}) will then be stored into receiver buffer (as Y_m) after erasing the preceding signal value.

3.4 Summary

The proposed Comprehensive Data Reduction algorithm developed during this research work was explained in this chapter. This chapter described in detail about how the data compression and message encoding was carried out at the Transmitter node and similarly it also explained about the steps that are followed by the proposed algorithm for the data decompression and message decoding that takes place on the receiver node.

Chapter 4
Development of the CDR algorithm

4.1 Experimental setup of the system.

A simulation model is constructed using Matlab / Simulink™ software [31] and CANoe™ software [32]. The signal values used for the test are obtained from a real-time CAN data log file of a test vehicle which contains the datasets acquired from a driver's driving vehicle under different conditions.

The model that is being tested with the proposed CDR algorithm consists of six nodes and messages are periodically sent from these six nodes on the CAN bus having the baud rate of 500Kbit/sec. The timing for the message transmission from the ECUs are decided based on the real-time data log file obtained from the test vehicle. Figure 4.1 shows the model consisting of six nodes developed using CANoe™ software [32].

The simulation was run for 20 minutes, and the results of the simulation were recorded accordingly. The CDR algorithm is developed using a software termed as CAPL programming which is an integral part of the CANoe™ software [32] and is pre-mounted on each node and is executed continuously by each node.

Table 4.1 lists the name of the signals whose data were considered from the test vehicle data log file for the experimentation work. Table 4.1 also describes the Bitlength values of original signals (i.e. before CDR algorithm was employed).

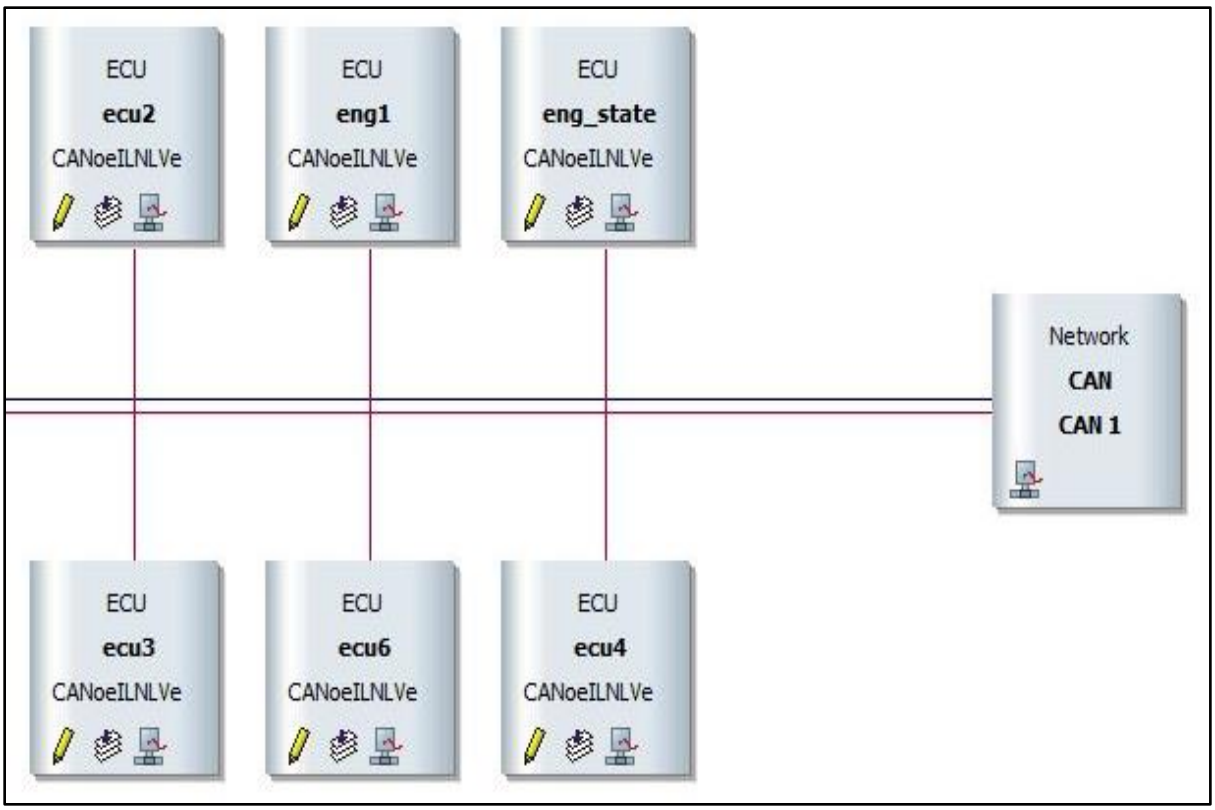

Figure 4.1 Six Engine Control Units (ECU's) connected to the CAN bus having baud rate of 500 Kbits/sec.

Table 4.1 Signals considered for experimentation

Signal Name	Bitlength
Engine speed	12
Gyroscope Data	8
Acceleration data	8
Pitch	8
Engine torque	12

Throttle position	8
Total acceleration	12
Lateral acceleration	8
Rain intensity	4
Coolant temperature	8
Engine pressure	16
TWC converter (oxygen sensor)	8
Engine load	12
Shift number	4
Visibility	4
Vehicle gas level.	12
Air conditioning status	4
Passenger count	4
Window opening	4

4.2 Data compression and Message encoding.

Data compression and message encoding takes place on the node or ECU which is transmitting the data over the CAN bus system as explained in the earlier chapter.

In CDR algorithm, a message that is to be transmitted over the CAN bus is developed in two stages. First stage is the delta compression technique discussed earlier, where the delta value of

the signals is obtained by comparing its current value with its preceding value. This stage was developed using the MATLAB / Simulink™ software [31]. The Simulink model for the first stage is shown in the figure 4.2.

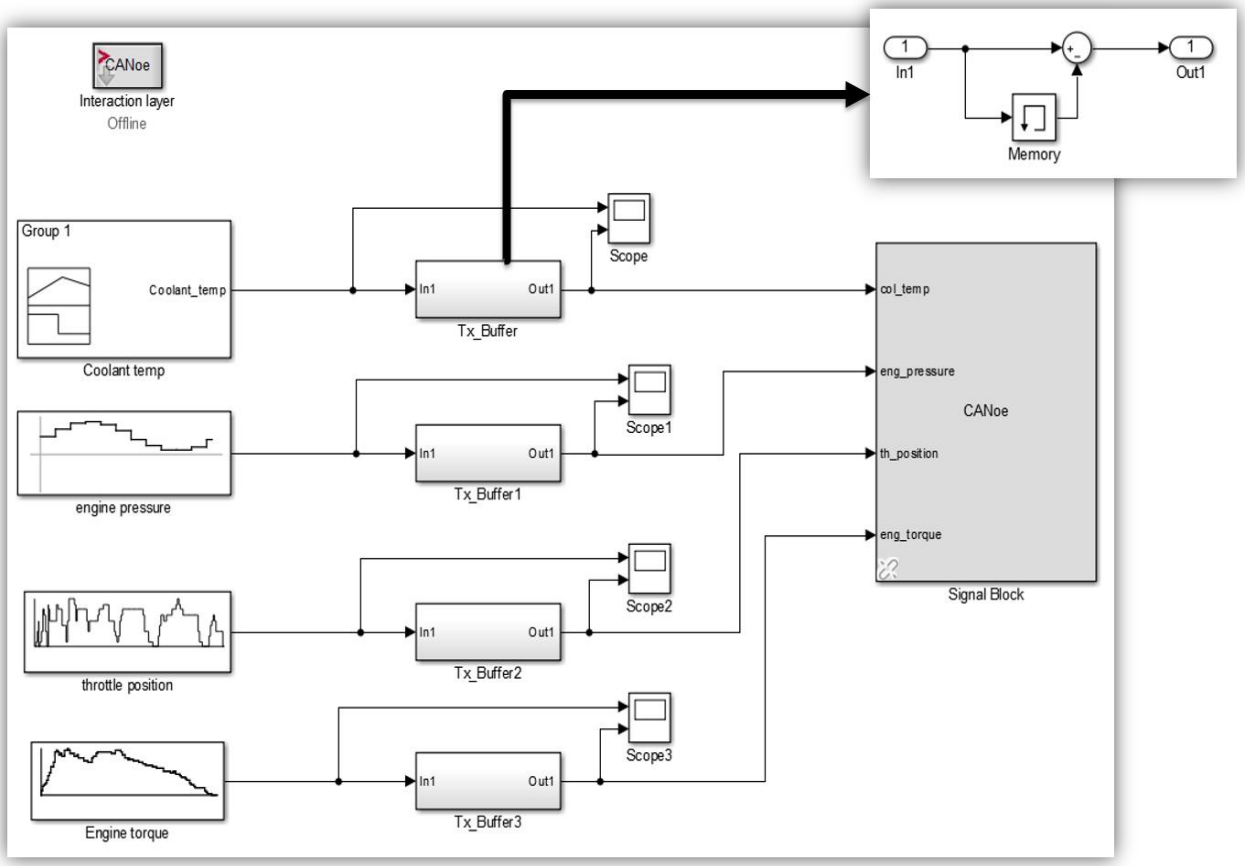

Figure 4.2 Simulink model developed for the delta compression.

The algorithm developed in figure 4.2 is employed for every signal. As shown in figure 4.2, the signal values are first passed through the transmit buffer where the comparison of the signal values takes place. This transmit buffers consist of a memory block which stores the preceding values of the signal. After obtaining the signals delta values from the transmit buffer, this delta values are than send to the signal block. This signal block works as a bridging block between the

MATLAB / Simulink™ software [31] and CANoe™ software [32]. Thus, the delta value of the signals is then transferred to the CANoe™ software [32] where the second stage of the algorithm is executed. Figure 4.3 shows the interaction between the two software's.

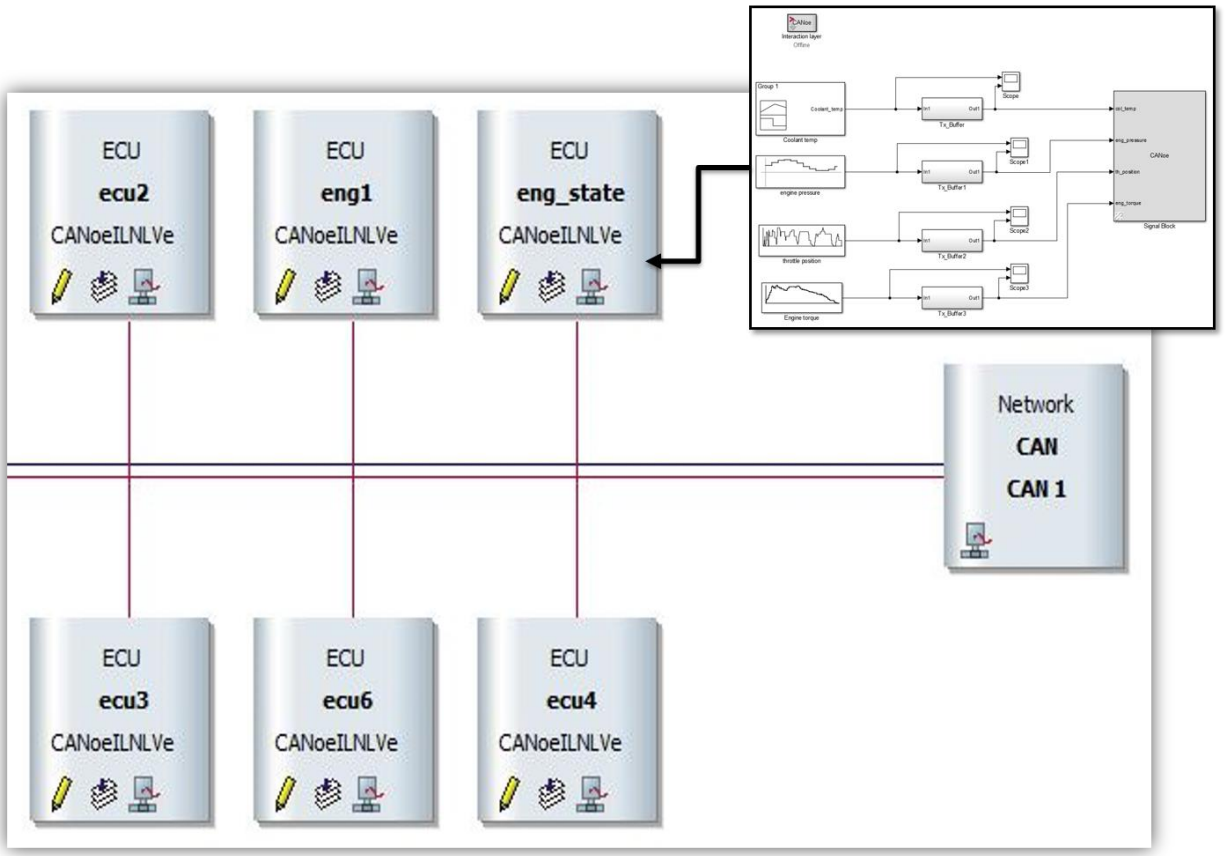

Figure 4.3 Interaction layer between MATLAB / Simulink™ software [31] and CANoe™ software [32].

In second stage, the signals are reassigned to their new Bitlength values (based on their respective delta values obtained from the stage one) and are enclosed into a data frame. This data frame is then enclosed into message and is transmitted over the CAN bus to the nodes or ECU who requested this data. Algorithm for the second stage was developed using CAPL programming which is an integral part of CANoe™ software [32]. A part of the code which was

developed for the signal bitlength assignment is explained in the appendix A. Figure 4.4 shows a typical message comprising of six different signals where 0X786 is the message unique identifier for message “power train”.

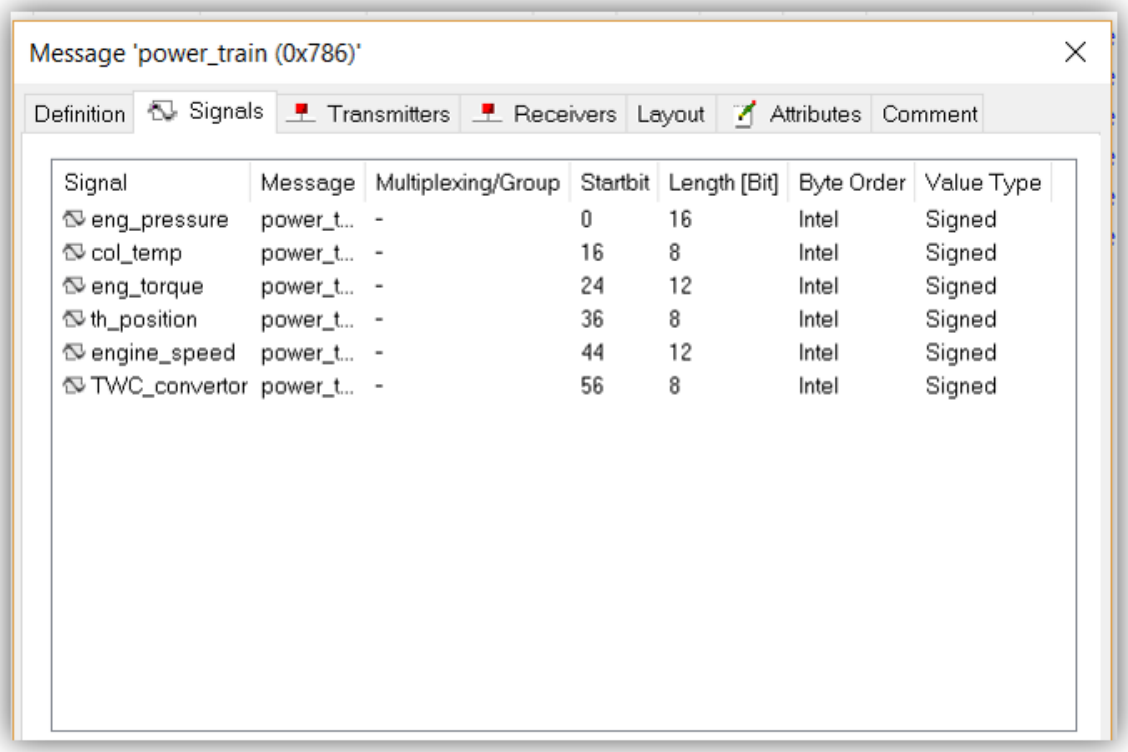

Figure 4.4 message “power train” that is being transmitted over CAN bus system.

Figure 4.5 shows the uncompressed message before implementing CDR algorithm whereas figure 4.6 shows the compressed message which was obtained after implementing the proposed CDR algorithm.

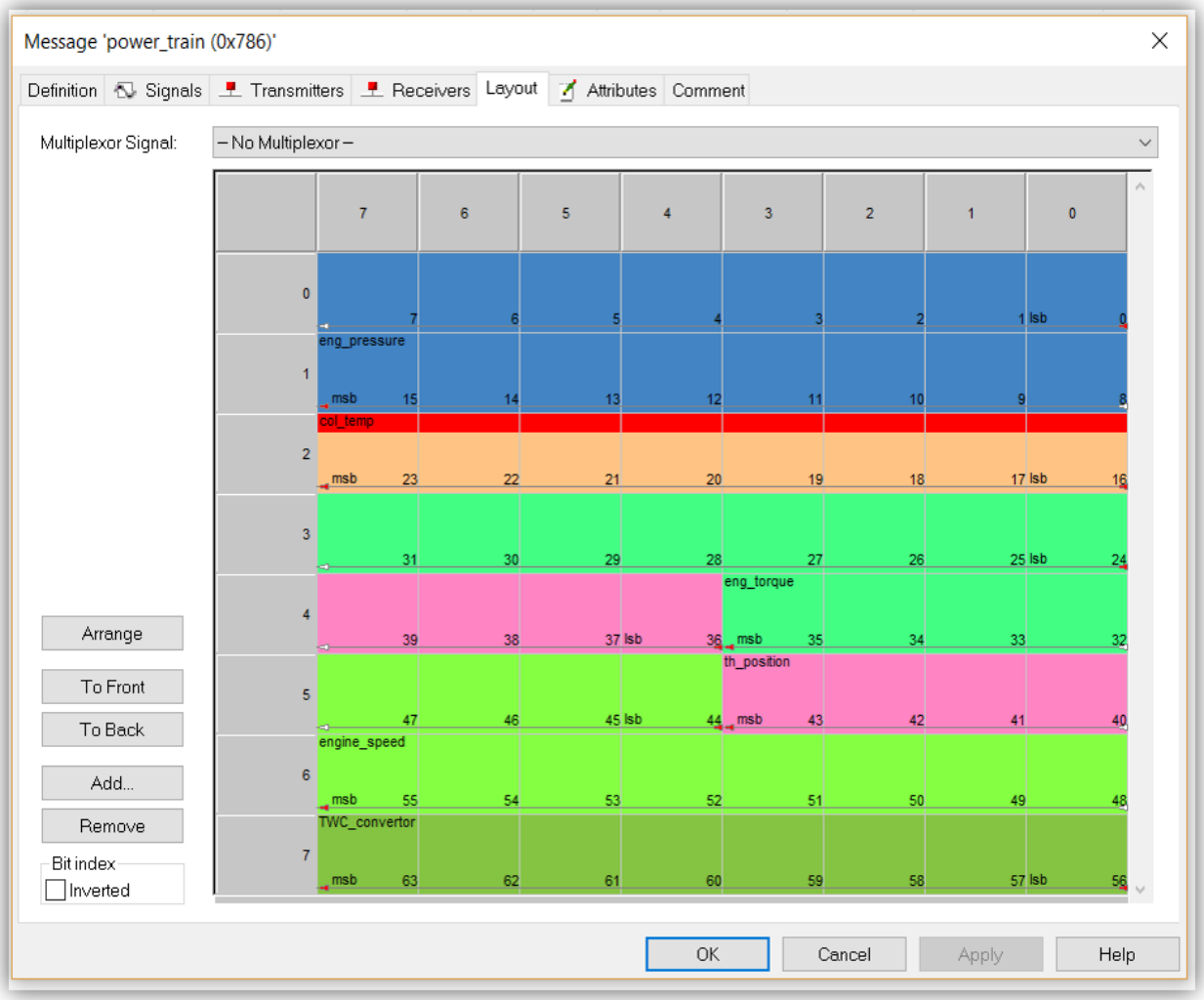

Figure 4.5 Uncompressed message before implementing CDR algorithm.

In figure 4.5 and 4.6, each rows of the data field represent the number of bits whereas the leftmost column shows the number of bytes used to represent the data enclosed inside the data field.

As we can see in figure 4.5 and figure 4.6, the 8-byte data field of message “power train” is reduced to 4-byte data field after implementing the proposed CDR algorithm. Thus, by reducing

the size of data field of the messages we can actually reduce the busload associated with the CAN bus.

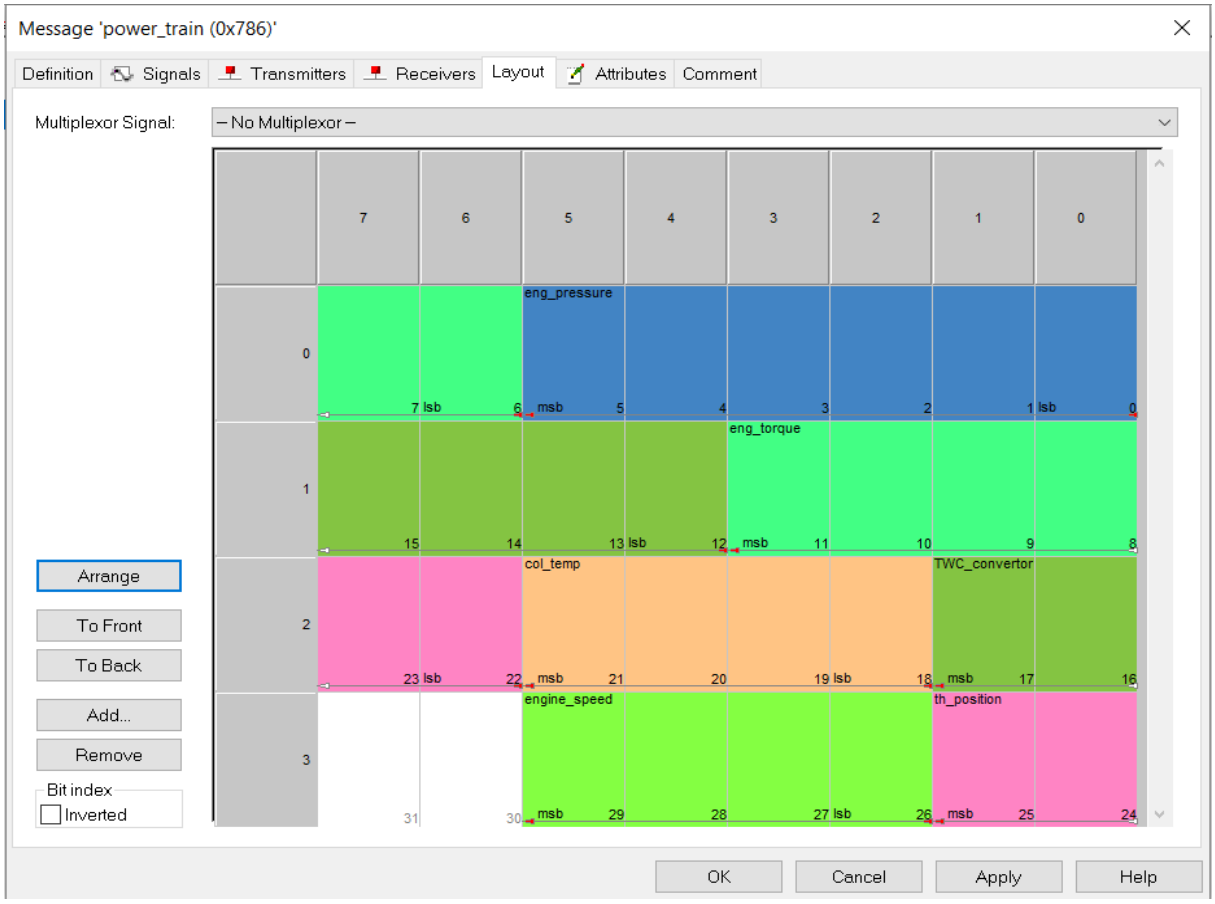

Figure 4.6 compressed message after implementing CDR algorithm.

4.3 Data decompression and Message decoding.

Data decompression and message decoding takes place on the node or ECU which is receiving the data over the CAN bus system as explained in the earlier chapter.

In CDR algorithm, a message that is being received from the CAN bus is regenerated in two stages at the receiving node. First stage is where the message decoding takes place and the signals delta values are obtained from the message. CANoe™ software [32] was used for message decoding.

The delta value of the signals that are obtained after message decoding is then transferred to the Simulink model to execute second stage of the algorithm. Second stage of the proposed algorithm is where the data decompression takes place and the original message is regenerated again on the receiving side. The Simulink model generated for the data decompression is shown in the figure 4.7.

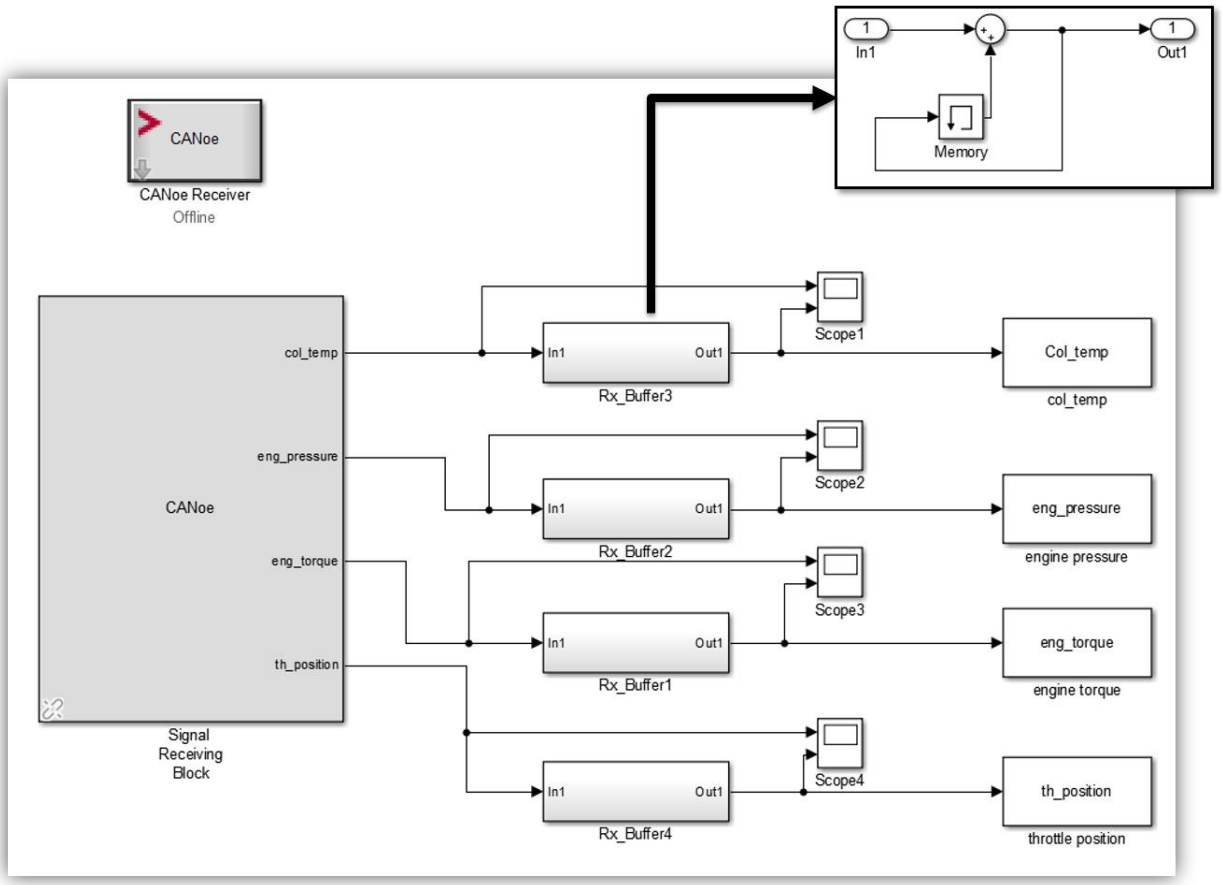

Figure 4.7 Simulink model developed for the data decompression.

As shown in figure 4.7 the signal delta value obtained from the received message are first transferred to the receiver buffer where the delta value of the signal is added to their preceding signal value and then is stored into the memory block present in the receiver buffer. The output of the receiver buffer gives out the original signal value at the receiver node which was intended to be transferred over the CAN bus. Thus, the original message is regenerated at the receiver node which was the primary objective.

4.4 CDR algorithm impact on Message latency.

Message latency is defined as how much time it takes for a CAN message to get from one designated node (transmitter node) to another (Receiver node) [33]. To implement the CDR algorithm, an ECU or node must follow the developed algorithm steps which eventually increases the end to end message latency. Thus, it is vital to investigate the impact of the proposed CDR algorithm on the message latency. Both, compression and decompression process of CDR algorithm will increase the message latency. The impact on message latency is dependent on the processor time required by an ECU to execute the CDR algorithm.

To calculate the Message latency incurred by the CDR algorithm, let us consider the algorithm steps shown in figure 3.5 and figure 3.6 and determine the computational time required to execute the algorithm. Table 4.2 shows the computation time required to encode a signal using the CDR algorithm.

Considering that each assembly or machine-level instruction requires four clock cycles of the processor, the total clock cycle required for encoding any signal into a message is around 32. Similarly, the computational time required to decode a signal is shown in table 4.3. The total clock

cycle required for decoding any signal into a message is around 16. Thus, in total for encoding and decoding a signal by using CDR algorithm, a processor will require in total of 32+16=48 clock cycle.

Table 4.2 computation time for encoding a message.

High level statement	Assembly/machine level instruction	Processor clock cycles
Delta value of signal	Load, Compare, Branch	12
$-(2^a - 1) \leq \Delta(\mathrm{Si}) \leq 2^a$	Load, Compare, Branch	12
n = a+1	Clear bitlength, set bitlength (bitlength=n)	8
Transmit the delta value	Store	4
	Total cycles	32

Table 4.3 computation time for decoding a message

High level statement	Assembly/machine level instruction	Processor clock cycles
Load the delta value from the receiver buffer	Load	4
Compare and add delta value to the preceding signal value	Compare, add	8
Store the new value of the signal into the receiver buffer	Store	4
	Total cycles	16

It has been observed that no more than 14 signals are enclosed inside a single message as the data field of a CAN message is of 8-bytes only. Thus, the processor requires around 14 X 48 = 672

clock cycle to encode and decode for any message. Also, the Data length code of the message is checked once for each message which requires another 12-clock cycle. Thus, in total 672+12=684 clock cycle is required to encode and decode any message.

The processor that are being used in today's commercial vehicle for the ECU have a minimum of 128 MHz of clock speed. That means that the computational time required by the processor to implement CDR algorithm is only 5.35 µs (684 cycles $*$ (1/128) µs/cycle), which is insignificant compared to the message transmission rate. Thus, the implementation of CDR algorithm will have negligible impact on the Message latency.

4.5 Synchronization of CDR algorithm.

In the proposed CDR algorithm, all the communications that occurs between the nodes or ECUs are assumed to be continuous. Sometimes, the errors might gets accumulated in the delta value of the signals. If this happens than all the following values of the signal which are obtained by delta comparison method on the receiving nodes will become invalid. This difficulty may arise the security concern for the vehicle.

To overcome the above difficulty, we propose a new technique termed as "message reprocessing". In this technique, the CAN messages are send in their entirety (without employing delta comparison method) over the CAN bus system after the specified allotted time. This specified allotted time is decided based on the criticality of the message, meaning that the messages having high criticality will be sent in their entirety will have less allotted time (for example: 300 seconds) whereas the messages having low criticality will have more allotted time (for example: 500 seconds).

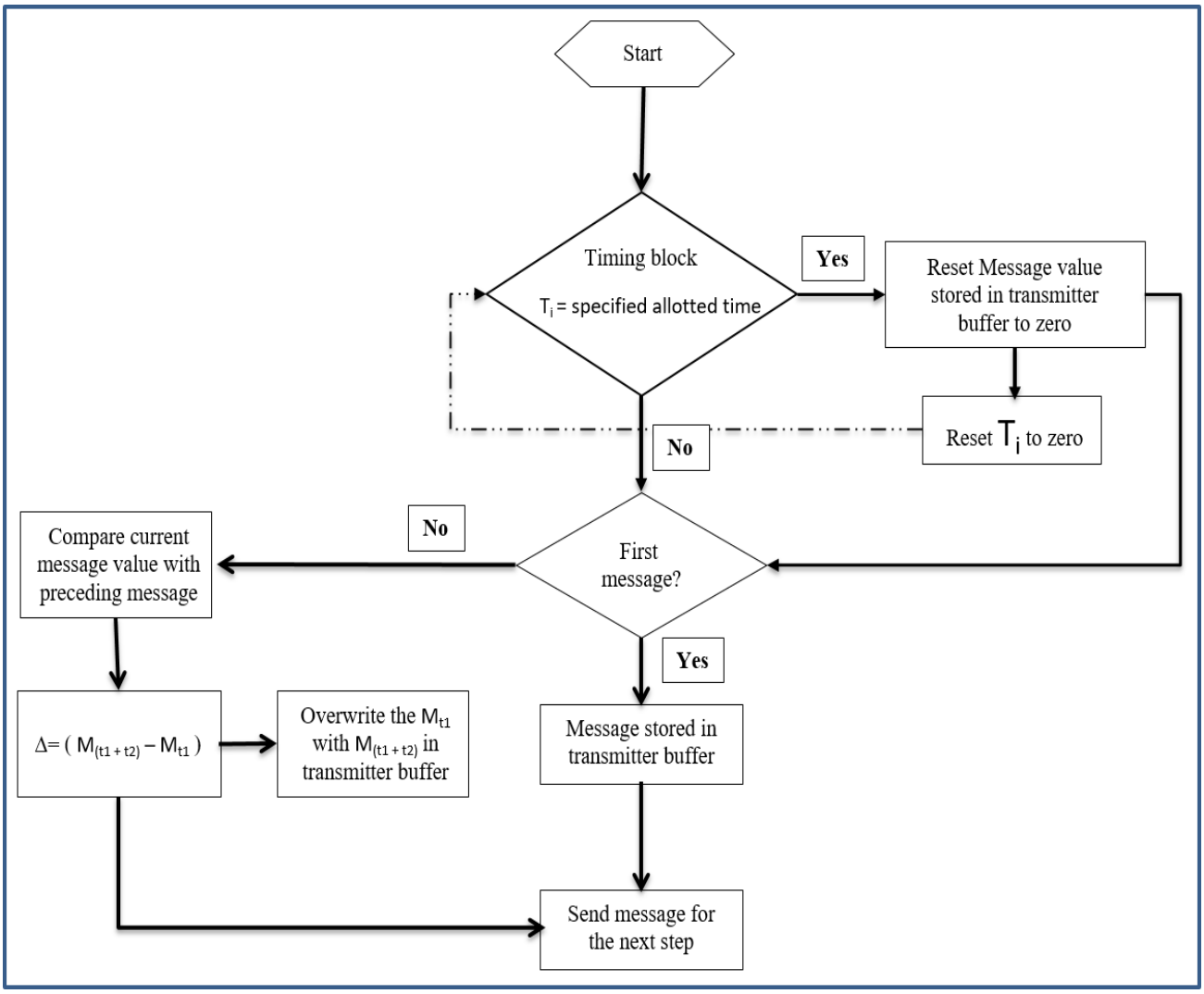

Figure 4.8 Synchronizing of CDR algorithm at transmitter node.

To execute the above technique, an extra block termed as “Timing block” is added into the system. This timing blocks are attached on both side i.e. transmitter nodes and receiver nodes. This block resets the values stored inside the transmit buffer and receiver buffer to zero when the specified allotted time of the message is reached. Thus, the system resets the value of the messages removing any error that might have got accumulated into the signal value of the

messages. Figure 4.8 and figure 4.9 shows the flowchart of the algorithm followed by the transmitter nodes and receiver nodes for the synchronizing of CDR algorithm. Here T_i shown in the figure is the Run time of the system.

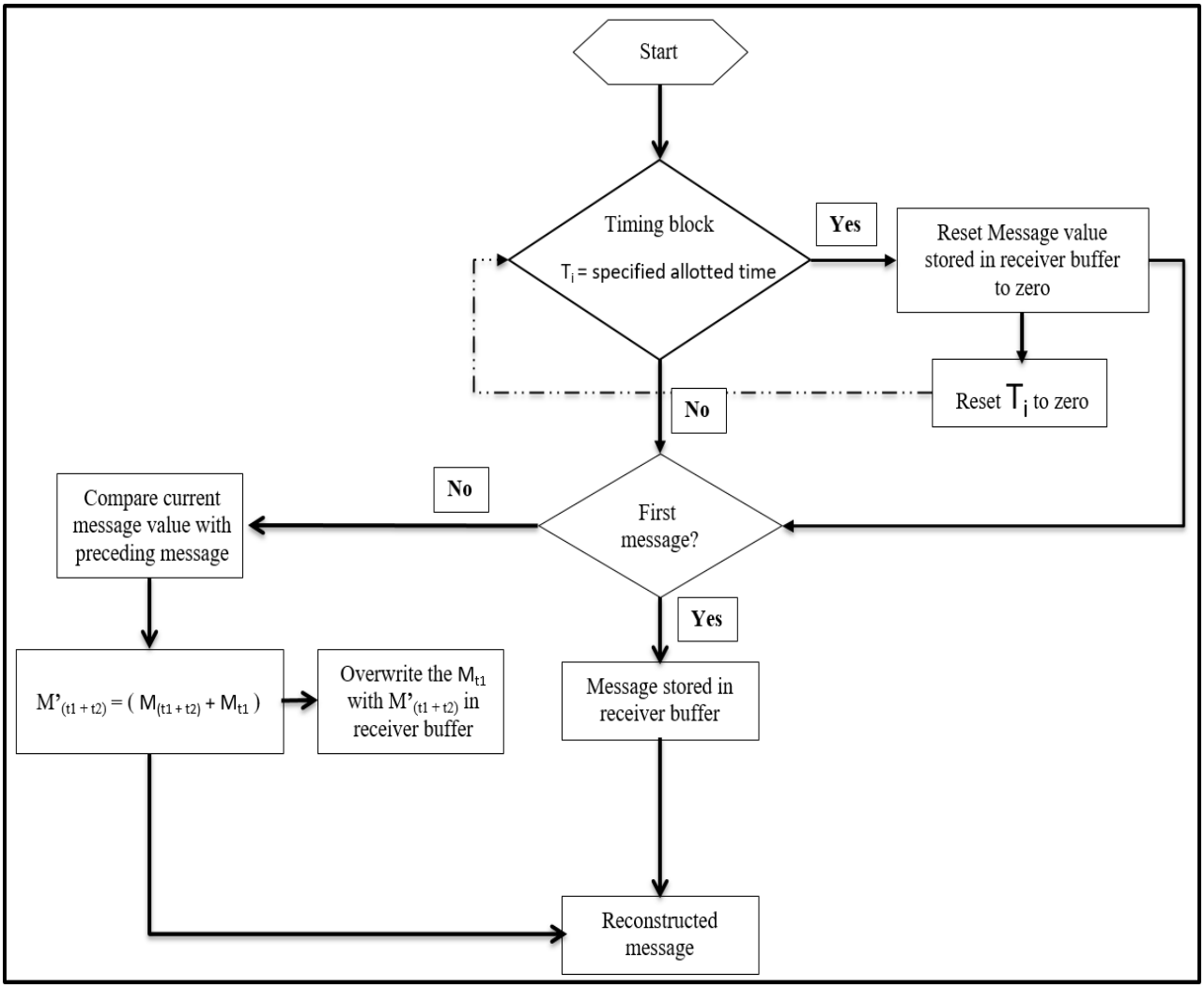

Figure 4.9 Synchronizing of CDR algorithm at receiver node.

4.6 Summary

This chapter describes about how the Proposed CDR algorithm was developed using MATLAB / Simulink™ software [31] and CANoe™ software [32]. The impact on message latency by CDR

algorithm was also checked. This chapter also uncovered a new aspect of our research work. It represented the synchronization aspect of the proposed algorithm to eliminate any error accumulation taking place inside an ECU of a vehicle.

Chapter 5

Performance analysis of CDR algorithm.

5.1 parameters considered for performance Analysis.

After developing the CDR algorithm, the next step is to check whether the proposed algorithm is better than the existing algorithm or not. Thus, for that purpose the performance parameters considered for analysis are CAN busload, CAN peak load for messages sent on CAN bus and compression efficiency obtained by the proposed CDR algorithm.

For the experimentation purpose, the test vehicle trip data was run for 20 minutes and the performance parameters like percent peak load and percent bus load were recorded. The figure 5.1 below shows the compressed message that are being transmitted over the CAN bus system. The message ID's along with their timestamps, data length code (DLC) and the number of data bytes transmitted over the CAN bus system is shown in the figure 5.1. The rightmost column in figure 5.1 shows the amount of data bytes that are being sent by the transmitted message. The first two messages shown in figure 5.1 are the messages that are being sent without implementing the CDR algorithm whereas the rest of the messages are sent after implementing the CDR algorithm. As we can clearly see in the figure 5.1, a significant amount of data byte reduction can be achieved by implementing proposed CDR algorithm.

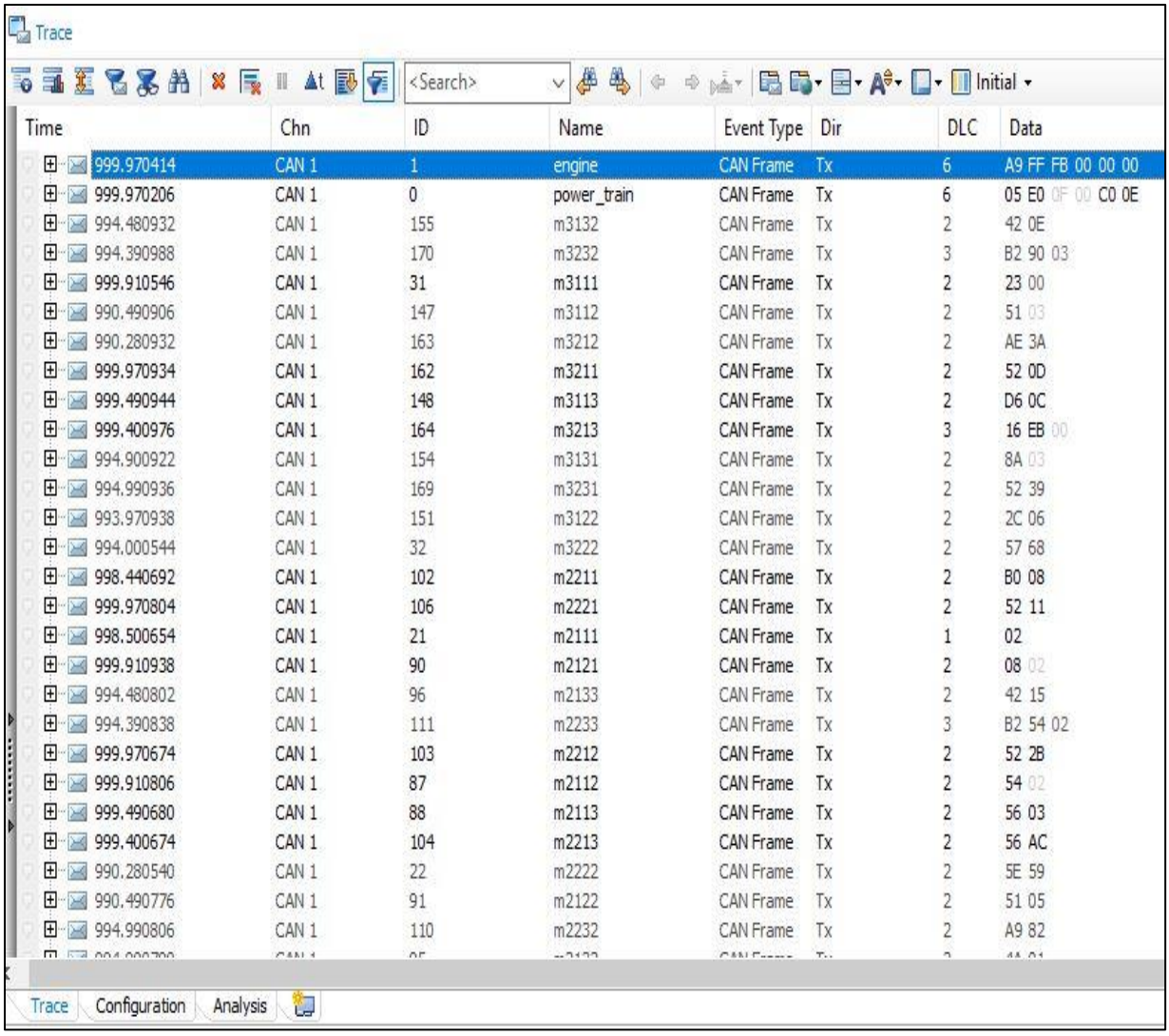

Figure 5.1 Compressed messages sent on CAN bus by nodes implementing CDR algorithm. [39]

5.2 Performance Analysis of CDR algorithm.

This section describes the performance analysis for CDR algorithm. To demonstrate the improvement in data reduction by CDR algorithm, a comparison (based on performance parameters like percent peak load, percent bus load, and compression efficiency) between CDR

algorithm, BFC algorithm [26], Compression area selection [27] and Automotive multiplexing without implementing any data reduction algorithm has been carried out in the subsequent section.

5.2.1 Comparison based on percent busload.

A comparison based on the percent busload has been made between the messages sent by the nodes implementing CDR algorithm and messages sent by the nodes without any implementation of data reduction algorithm. Figure 5.2 shows this busload comparison.

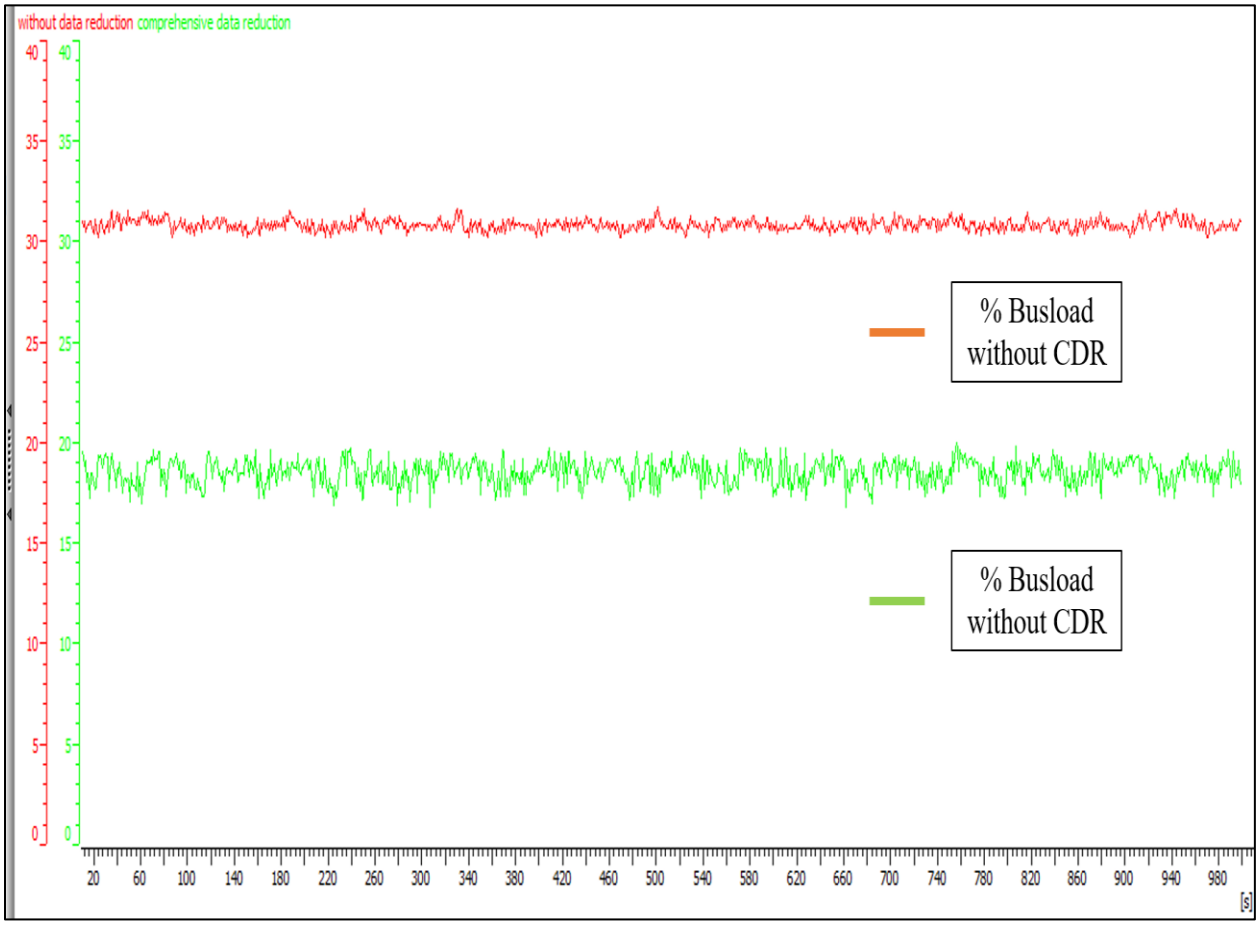

Figure 5.2 Percent busload comparison - with CDR and without CDR. [39]

In figure 5.2 the red line shows the percent busload of CAN bus obtained without implementing CDR algorithm whereas the green line shows the percent busload of CAN bus obtained after implementing the Proposed CDR algorithm. The X-axis in graph shown in figure 5.2 represent the time span whereas the Y-axis represent the percent busload of the CAN bus. The average percent busload obtained for Automotive multiplexing without implementing CDR algorithm was around 30.84% whereas the average percent busload obtained for the messages sent by the nodes implementing CDR algorithm was around 18.56%. Thus, in total an average busload reduction of 39.82% was recorded due to the implementation of the CDR algorithm.

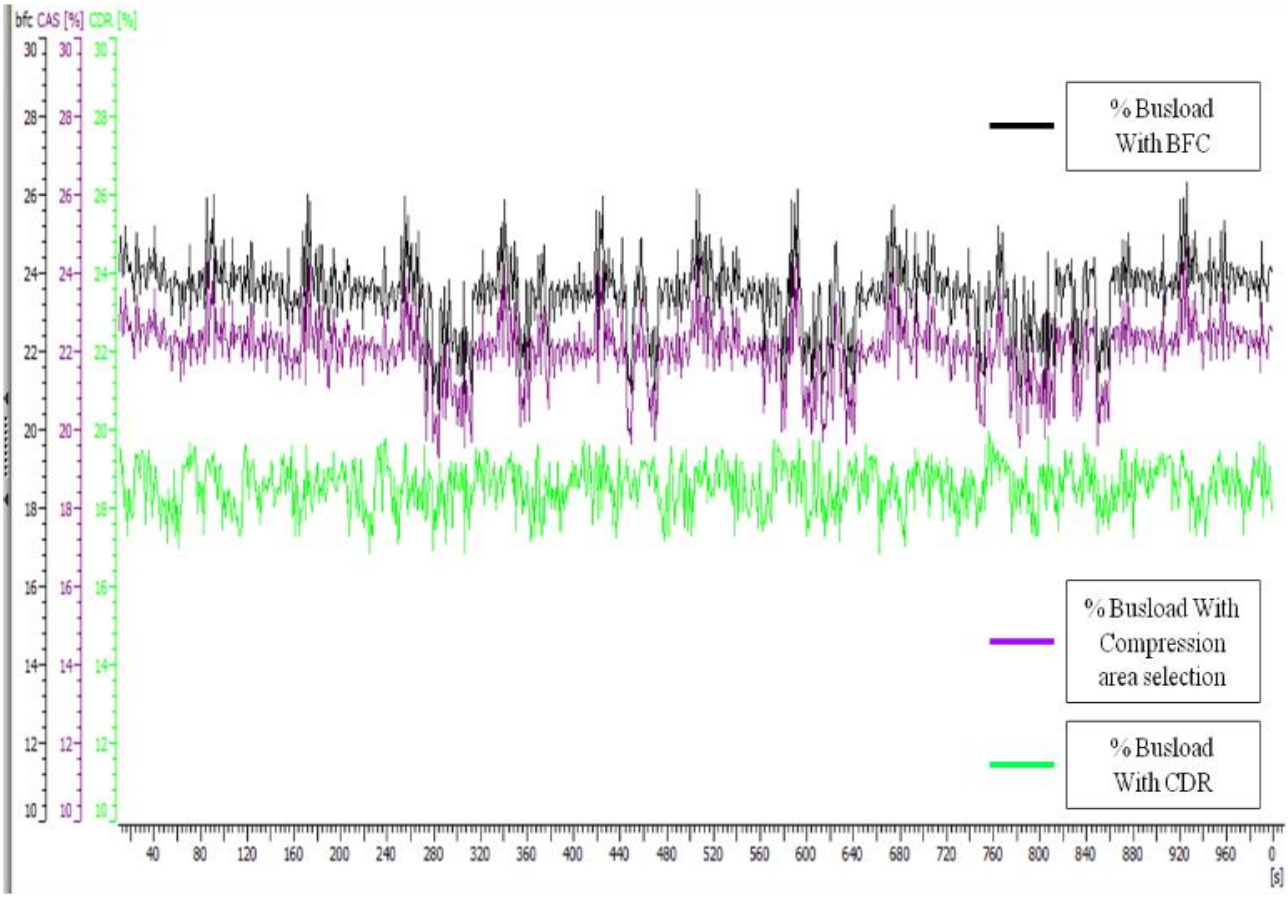

Figure 5.3 Percent busload comparison - with CDR, with BFC and with compression area selection. [39]

Another comparison was made based on the percent busload between the messages sent by the nodes implementing CDR algorithm, nodes with BFC algorithm and nodes with Compression area selection algorithm. Figure 5.3 shows this percent busload comparison.

The values for the percent busload achieved by the BFC, Compression area selection and CDR algorithm are shown in table 5.1 and in chart shown in figure 5.4. The proposed algorithm achieves maximum data reduction of 39.82%. This maximum data reduction achieved by the proposed algorithm is due to the fact that the parameter values are always sent in the form of delta, thus there is no need for any extra byte (DCC, header bits or DCB) which are used by the EDR, BFC, or compression area selection algorithms for indicating whether the parameter was fully compressed, partially compressed or uncompressed.

Table 5.1 Comparison of Average percent busload [39]

	Average percent busload	**Average percent bus load reduction**
Uncompressed message	30.84 %	0%
BFC algorithm	23.53%	23.70%
Compression area selection algorithm	22.06%	28.46%
Comprehensive data reduction (CDR) algorithm	18.56%	39.82%

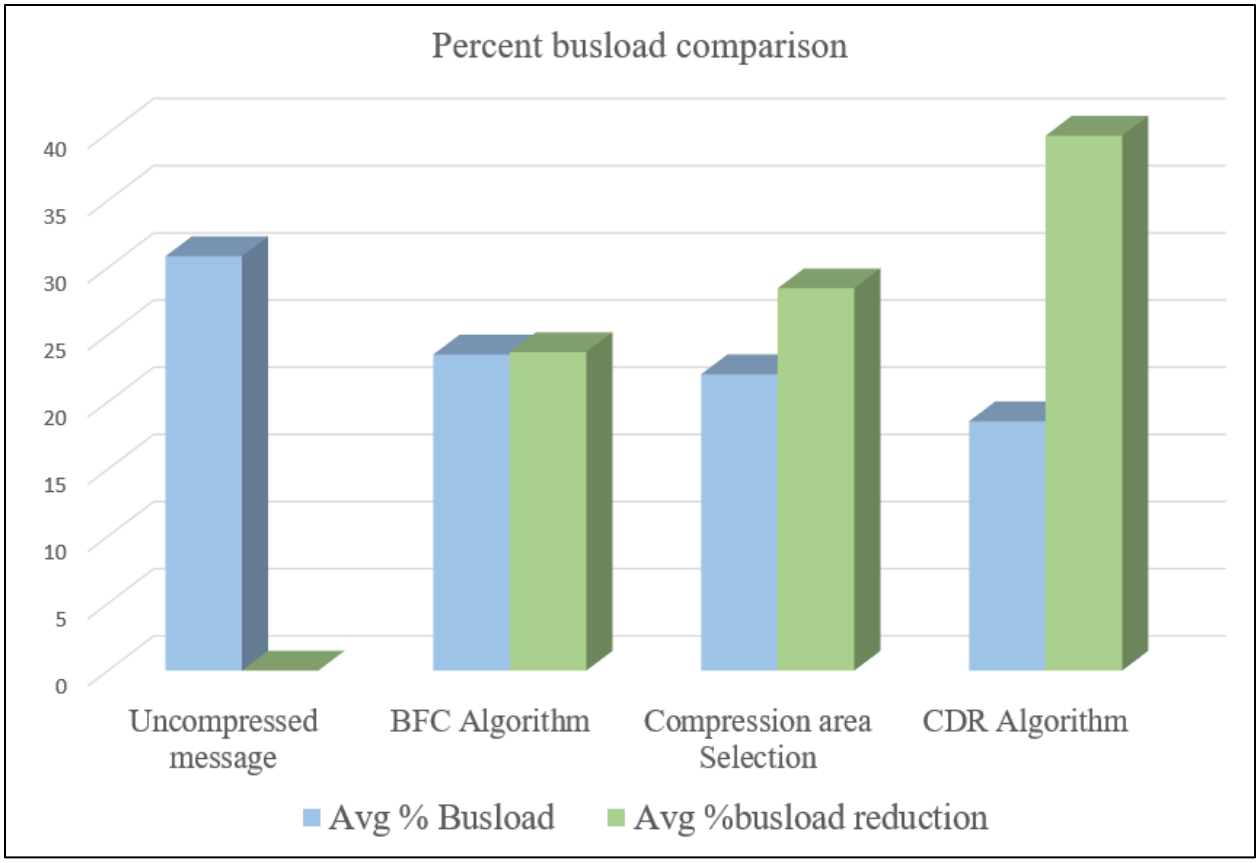

Figure 5.4 graphical representation of Comparison of Average percent busload.

5.2.2 Comparison based on percent peak busload.

A comparison was made based on the percent peak busload between the messages sent by the nodes implementing CDR algorithm, nodes with BFC algorithm and nodes with Compression area selection algorithm.

The values for the percent peak busload achieved by the BFC, Compression area selection and CDR algorithm are shown in table 5.2 and in chart shown in figure 5.5. The proposed CDR algorithm achieved the maximum improvement in peak busload (35.06 %).

Table 5.2 Comparison of Average percent peak busload [39]

% Peak Load	Average percent peak busload	Average percent improvement in peak busload
(uncompressed)	31.71 %	0%
(BFC algorithm)	26.39%	16.77%
(Compression area selection algorithm)	24.72%	22.04%
(Comprehensive data reduction (CDR) algorithm)	20.59%	35.06%

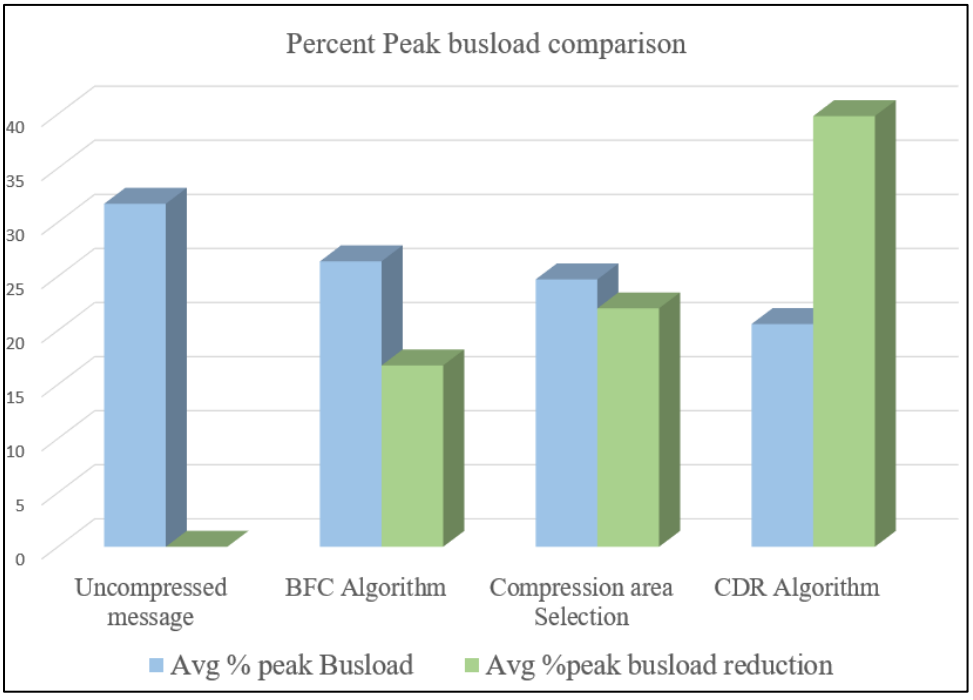

Figure 5.5 graphical representation of Comparison of percent peak busload.

5.2.3 Comparison based on compression efficiency.

The performance of the proposed CDR algorithm was also tested based on the compression efficiency. The comparison of compression efficiencies for CDR algorithm, BFC algorithm, Compression area selection algorithm are shown in table 5.3 and in chart shown in figure 5.6. Compared to other available algorithms, CDR algorithm achieves maximum compression efficiency of 29.99%.

Table 5.3 Comparison based on Compression efficiency [39]

Compression methods	**Data transmitted**	**Compression efficiency (%)**
Uncompressed	4,953,823	0 %
BFC	4,070,060	17.84%
Compression area selection	3,892,218	21.43%
Proposed CDR algorithm	3,468,171	29.99%

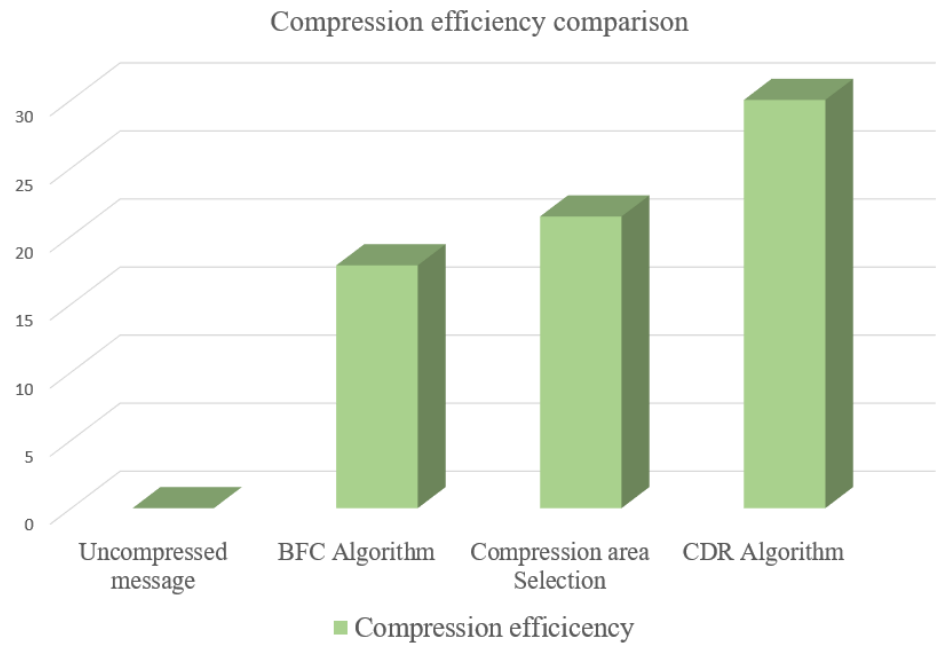

Figure 5.6 graphical representation of Compression efficiency comparison.

5.3 Summary

This chapter presents the experimental results obtained by the proposed algorithm. It also provided a Performance analysis test done on the proposed algorithm. Furthermore, the proposed algorithm is compared with the existing data reduction algorithms. The comparison is done based on the parameters like Bus-load, Compression ratio and peak load associated with the CAN bus communication.

Chapter 6
Conclusion and future work

6.1 Conclusion

In our research we presented a new data reduction algorithm termed as "Comprehensive Data Reduction" Algorithm. The proposed algorithm is used for minimization of the bus utilization of CAN bus for the future vehicle. The reduction in the busload was efficiently done by compressing the parameters; thus, more number of messages and lower priority messages can be sent efficiently on the CAN bus. No hardware changes are required in the existing wired infrastructure for the implementation of the proposed algorithm.

A discrete event simulation has been performed to authenticate the functionality of the proposed algorithm. In the simulation model, the realistic examples of message traffic were considered. The performance analysis of the proposed algorithm showed that significant improvement in the compression and CAN busload can be achieved as compared to existing BFC algorithm and compression area selection algorithm. The promising results were obtained in terms of reduction in bus utilization, compression efficiency, and percent peak load of CAN bus. This Reduction in the bus utilization permits to utilize a larger number of network nodes (ECU's) in the existing system without increasing the overall cost of the system. This algorithm can also be utilized in any applications where extensive information transmission among various control units is carried out via a multiplexing bus or Applications which utilizes message packets to transmit data as in case of Wireless sensor network (WSN) or Space probes communication application.

6.2 Future work.

Present research work was carried out on six ECUs or on six nodes, but in reality, a vehicle can have as many as 150 ECUs in it. Thus, in future work, the practical experimentation on entire vehicular system will be employed. Also, we will look on to the possibility to employ this proposed algorithm on the other applications which utilizes message packets technique to transmit data as in case of Wireless sensor network (WSN) or Space probes communication application.

References

[1] Abdul h. Aleef, "Automotive Multiplexing Wiring," National Semiconductor, Application Notes 454, April, 1997.

[2] J. G. Kassakian and D. J. Perreault, “The future of electronics in automobiles,” in Proc. 13th Int. Symp. Power Semicond. Devices ICs, Osaka, Japan, 2001, pp. 15–19.

[3] Lupini, Christopher A. "In-Vehicle Networking Technology for 2010 and Beyond." SAE Technical Paper Series, 2010. doi:10.4271/2010-01-0687.

[4] Lawrenz, W., & Lawrenz, W. (2013). CAN System Engineering: From Theory to Practical Applications. London: Springer London. doi:10.1007/978-1-4471-5613-0

[5] ISO (International Organization for Standardization), 2015. Road Vehicles—Controller Area Network (CAN)—Part 1: Data Link Layer and Physical Signaling, ISO 11898-1: 2015.

[6] K.Tindell, A. Burns and A. Wellings, "Calculating Controller Area Network(CAN) message response times," In proc. of the IFAC workshop on Distributed Computer Control Systems, 1994, pp. 29-34.

[7] Sangiovanni-Vincentelli and M. D. Natale, "Embedded System Design for Automotive Application," Publication of the IEEE Computer Society, Vol. 40, Issue 10, November, 2007, pp. 42-51.

[8] G. Leen and D. Heffernan, "Expanding automotive electronic systems," Computer, Vol. 35, No. 1, pp. 88-93, 2002.

[9] G. Leen, D. Heffernan and A. Dunne, “Digital Networks in the Automotive Vehicle,” IEEE Computer and Control Eng. Journal, vol. 10, no. 6, pp. 257-266, December 1999.

[10] Shrinath, A., and A. Emadi. "Electronic Control Units for Automotive Electrical Power Systems: Communication and Networks." Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering 218, no. 11 (2004): pp. 1217-230. doi:10.1243/0954407042579996.

[11] A. Karimi, J. Olsson, J. Rydell, (2004). "A Software Architecture Approach to Remote Vehicle Diagnostics", Tech. report., Department of Informatics, Göteborg university and chalmers university of technology, report no. 2004 : 59. ISSN: 1651-4769.

[12] SAE International, (2004) Recommended Practice for Pass-Thru Vehicle Programming. SAE J2534/1_200412. CFR section: 40 CFR 86.096-38(g)(17)(iv). December 2004.

[13] Ebert, Christof, and Capers Jones. "Embedded Software: Facts, Figures, and Future." Computer 42, no. 4 (2009): pp. 42-52. doi:10.1109/mc.2009.118.

[14] Hellburg, J. and Petterson, M. (2013). "Remotely access real-time system in a modern car", Master of Science Thesis in Systems, Chalmers University of Technology.

[15] Tapas R. "Controller area network protocol; Design to tapeout". master’s thesis, national institute of technology, 2011-2012. Retrieved from http://ethesis.nitrkl.ac.in/4057/1/Final_Thesis_Tapas.pdf

[16] Controller Area Network (CAN) Specification Ver. 2.0, Robert Bosch GmbH, 1991. Retrieved from http://esd.cs.ucr.edu/webres/can20.pdf

[17] Corrigan, "Introduction to the Controller Area Network", Application report, Texas instruments incorporated, SLOA101B–August 2002–Revised May 2016

[18] Matthew John Darr. "Development and evaluation of a controller area network based autonomous vehicle". master's thesis, University of Kentucky, 2004. Retrieved from https://uknowledge.uky.edu/cgi/viewcontent.cgi?referer=https://www.google.ca/&httpsredir=1&article=1194&context=gradschool_theses

[19] G. G. Kempf, M. J. Eckrich, O. J. Rumpf, "Data reduction in automotive multiplexing systems," Soc. Automotive Eng., SAE paper 940135, pp. 20-25, March, 1994.

[20] Misbahuddin Syed, Nizar Al-Holou, and S. M. Mahmud. "A Data Reduction Algorithm for Automotive Multiplexing." SAE Technical Paper Series, 1998. doi:10.4271/981104.

[21] Syed Misbuddin, S. M. Mahmood, Nizar AI-Holou, "Development and Performance Analysis of Data-Reduction Algorithm for Automotive Multiplexing," IEEE Transactions on Vehicular Technology, Vol. 50, No. I, pp.162-169, January, 2001.

[22] P. R. Ramteke, S.M. Mahmud, "An Adaptive Data-Reduction Protocol for the future In-Vehicle Networks," Soc. Automotive Eng., SAE Paper 2005-01-1540, 2005. doi:10.4271/2005-01-1540.

[23] Miucic, Radovan, and Syed Masud Mahmud. "An Improved Adaptive Data Reduction Protocol for In-Vehicle Networks." SAE International, 2006. https://doi.org/10.4271/2006-01-1327

[24] Radovan Miucic, S. M. Mahmud, Zeljko Popovic, "An Enhanced Data Reduction Algorithm for Event-Triggered Networks," IEEE Transactions on vehicular Technology, Vol. 58, No.6, pp. 2663-2678, July, 2009.

[25] Kelkar, S., & R. (2012). "Control area network based quotient remainder compression-algorithm for automotive applications". IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society. doi:10.1109/iecon.2012.6389414

[26] Kelkar, S., Kamal, R., 2014. Boundary of fifteen compression algorithm for controller area network based automotive applications. Proc. Int. Conf. on Circuits, Systems, Communication and Information Technology Applications, p.162-167. [doi:10.1109/CSCITA.2014.6839253]

[27] Wu, Y.J., Chung, J.G., Sunwoo, M.H., 2014. Design and implementation of CAN data compression algorithm. Proc. IEEE Int. Symp. on Circuits and Systems, p.582-585. [doi:10.1109/ISCAS.2014.6865202]

[28] Wu, Y., & Chung, J. (2015). Efficient controller area network data compression for automobile applications. Frontiers of Information Technology & Electronic Engineering, 16, pp. 70-78. doi:10.1631/fitee.1400136.

[29] SAE recommended practice, Data link layer, SAE JI939/21, July, 1994. DOI: https://doi.org/10.4271/J1939/21_201012

[30] Waggener, B., W.M. Waggener, and W.N. Waggener. Pulse Code Modulation Techniques. A Solomon Press Book. Springer US, 1995. https://books.google.ca/books?id=8l_o6kI3760C.

[31] MATLAB and Statistics Toolbox Release 2016a, The MathWorks, Inc., Natick, Massachusetts, United States.

[32] CANoe 10.0 SP4, Vector Informatik GmbH. Available online: http://vector.com/vi_canoe_en.html

[33] Mary, Gerardine Immaculate, et al. “Response Time Analysis of Messages in Controller Area Network: A Review.” Journal of Computer Networks and Communications, vol. 2013, 2013, pp. 1–11., doi:10.1155/2013/148015.

[34] Volha Bordyk 2012, "Analysis of software and hardware configuration management for pre-production vehicles." Master of Science Thesis, University of Gothenburg, Retrieved from http://publications.lib.chalmers.se/records/fulltext/156295.pdf.

[35] Embitel."ECU is a Three Letter Answer for all the Innovative Features in Your Car: Know How the Story Unfolded". https://www.embitel.com/blog/embedded-blog/automotive-control-units-development-innovations-mechanical-to-electronics (accessed September 3, 2017)

[36] Technavio. "Global Automotive Electronic Control Unit Market to Witness Growth Through 2021, Owing to Increased Demand for Connected Vehicles: Technavio". https://www.businesswire.com/news/home/20170102005047/en/Global-Automotive-Electronic-Control-Unit-Market-Witness (accessed September 3, 2017).

[37] Tarun Agarwal. "Industrial Automation and Control using CAN Protocol". https://www.elprocus.com/industrial-automation-control-using-can-protocol/ (accessed December 12, 2017).

[38] Bruce Emaus. "Quick Introduction To CAPL". Application Note AN-AND-1-113. https://can-newsletter.org/assets/files/media/raw/a456e3078f907a0482182ce831912427.pdf (accessed October 15, 2017)

[39] Aliakbar B, Necsulescu D. “A Comprehensive Data Reduction Algorithm for automotive multiplexing”. Passengers cars-electronic and electrical system, SAE International journal publications. April 2018. JPCE-2018-0019.

Appendix-A

CAPL Programming.

The Code for Signal bitlength Assignment in Comprehensive Data reduction algorithm was developed using the CAPL programming. CAPL is a scripting language that is used to access the CAN protocol with Logical operations. The Syntax of CAPL programming is based on the programming language C. CANoe software [32] uses CAPL programming to emulate car ECUs for stimulating, simulating, testing and diagnostics [38]. CAPL programs are added in CANoe as CAPL program nodes.

Code for Signal bitlength Assignment

```
//Include function is used to include files that are programmed in CAPL. //

includes
{
#include “trip data.CIN”        // Calling the file having the trip data ( database ) //

}

// Variable function is used to define the global Variables which are being used in the program //

variables
{
```

```
  int a,c,d,e;                // variables with integer data type is defined here //
}
```

// on Signal function is used to call the signal delta value from the database and assign this value to the above defined variables //

```
on signal eng_torque
// the delta value of the engine torque is transferred to the integer variable “d” //
{ d =(this);  }
on signal th_position
// the delta value of the throttle position is transferred to the integer variable “c” //
{  c =(this); }
on signal col_temp
// the delta value of the coolant temperature is transferred to the integer variable “a” //
{  a =(this); }
on signal eng_pressure
// the delta value of the engine pressure is transferred to the integer variable “e” //
{  e =(this);
```

// if and else command are then used to assign the bitlength to the signal based on their delta values.

After assigning the bitlength to the signals, these signals are then enclosed into the CAN message format by using the Message command and are then transmitted over the CAN bus by the output command. //

```
if ( 0 <= a & a <= 1 & -7 <= e & e <= 8 & -10 <= c & c <= 0 & -2 <= d & d <= 0 )
```

// the delta values of the signals are first checked by implementing if and else command. //

```
{
```

// After that, the appropriate CAN message is called from the database by the Message command and the signals are then enclosed inside the message //

```
  message power_train msg;
```

// message power_train is called from the database and is now represented by msg. ("msg" name is used just for the coding purpose) //

```
  msg.ct1= a;
```

// the delta value present in the variable "a" is then assigned to the signal ct1 (having bitlength 1) which is then enclosed in message "power_train". //

```
  msg.ep1= e;
```

// the delta value present in the variable "e" is then assigned to the signal ep1 (having bitlength 4) which is then enclosed in message "power_train". //

```
  msg.tp1= c;
```

// the delta value present in the variable “a” is then assigned to the signal tp1 (having bitlength 2) which is then enclosed in message “power_train”. //

```
  msg.tq1= d;
```

// the delta value present in the variable “a” is then assigned to the signal tq1 (having bitlength 2) which is then enclosed in message “power_train”. //

```
  output (msg);
```

// after enclosing the signal inside the message “power_train” the output command is used to transmit the message over the CAN bus system. //

```
  }
```

else

// The similar steps are repeated up until the right conditions are attained for the delta value of the signals. //

```
{
 if ( 0 <= a & a <= 1 & -7 <= e & e <= 8 & -1 <= c & c <= 0 & -8 <= d & d <= 8 )
{
  message power_train msg;
  msg.ct1= a;
  msg.ep1= e;
  msg.tp1= c;
  msg.tq2= d;
// signal tq2 = 4 bitlength //
  output (msg);
  }
```

```
else
{
 if ( -4 <=a & a<= 2 & -7 <= e & e <= 8 & -1 <= c & c <= 0 & -32 <= d & d <= 30 )
{
  message power_train msg;
  msg.ct2= a;
// signal ct2 = 3 bitlength //
  msg.ep1= e;
  msg.tp1= c;
  msg.tq3= d;
// signal tq3 = 5 bitlength //
  output (msg);
  }
else
{
 if ( -7 <= a & a <= 8 & -16 <=e & e<= 15 & -4 <= c & c <= 3 & -64 <= d & d <= 62 )
{
  message power_train msg;
  msg.ct3= a;                    // signal ct3 = 4 bitlength //
  msg.ep2=e;                     // signal ep2 = 5 bitlength //
  msg.tp2=c;                     // signal tp2 = 3 bitlength //
  msg.tq4=d;                     // signal tq4 = 7 bitlength //
  output (msg);
      }
   }
}
…
```

// This way the code is repeated for all the delta value combinations up until the correct signal bitlength condition are attained, after that the code is terminated and the message is transmitted over the CAN bus system. //

Table A. Signal database stored in the ECU for the message “power_train”.

Message (ID)	Signals	Updated signal (based on bitlength)	Bit length assigned to the signal
Power_train (0X786)	Coolant temperature	Ct1 Ct2 Ct3 Ct4	1 3 4 6
	Engine pressure	Ep1 Ep2 Ep3 Ep4	4 5 6 7
	Throttle position	Tp1 Tp2 Tp3 Tp4	2 3 4 5
	Engine torque	Tq1 Tq2 Tq3 Tq4	2 4 5 7

The table A shows the signal database stored in the ECU for the message “power_train”. The selected signals (by the above program) will be enclosed in the power_train message and will then be transmitted over the CAN bus system.