Modeling of Vehicle Controller Area Network for Control Systems Simulation

THESIS

Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in the Graduate School of The Ohio State University

By

Bharatkumar Hegde, B.Tech.

Graduate Program in Mechanical Engineering

The Ohio State University

2014

Thesis Committee:

Dr. Shawn Midlam-Mohler, Advisor

Dr. Giorgio Rizzoni
Networked control systems are being used extensively because of their reliability, cost effectiveness and manageability. The automotive industry uses Controller Area Network (CAN) for exchanging data among multiple Electronic Control Units (ECU). Each ECU has a set of dedicated control tasks such as engine control, transmission control, brake systems management, traction control, etc. Generally, every control task needs information from multiple ECUs and sensors to perform its duties. And CAN helps in minimizing the wiring requirements while maintaining a reliable pathway for data exchange among ECUs. The use of CAN as a part of control loop introduces certain effects on the performance of the control which are significant and non-linear. Therefore it is important to include the CAN behavior in model based control design and simulations. The work presented in this thesis develops a CAN simulator on MATLAB/SimEvents platform that can be integrated into existing vehicle simulators for control systems simulations. The CAN simulator is built for shorter simulation time and modularity. CSMA/CD+AMP protocol with CAN ID based arbitration is implemented on the simulator along with packet loss mechanism and online network delay estimation. The simulator performance is compared and validated against experimental data.
The effects of CAN on control performance are studied by simulating different systems in multiple configurations using the CAN simulator. Based on the results of such simulations, design guidelines are developed. Network induced delay analysis and certain control solutions for improving the performance of the CAN based NCS are discussed as well.
DEDICATION

Dedicated to my parents, Subraya and Nirmala for their selfless love, patience and support.
ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Shawn Midlam-Mohler for guiding me through my Master’s degree and for providing the opportunity to work with him. I would also like to thank Dr. Giorgio Rizzoni and Center for Automotive Research for enabling my research.

I would like to thank the entire Ohio State University EcoCAR2 team for all their support in performing my research and for providing a great atmosphere to work in. I would like to thank Matthew Yard, Matthew Organiscak, Katherine Bovee, Amanda Hyde, Eric Gallo, Jason Ward, Teng Ma, Andrew Garcia and Luke DeBruin for helping me out on various occasions to solve problems and learn new things. I would like to extend my gratitude to Eric Schacht, Jim Shively and Dr. Lisa Fiorentini for their technical advice.

I would like to acknowledge the support and help from Rajashree Maiya and Apoorva Rama in completing this thesis. I would like to thank my family and my friends for their love and for being there for me at every step of the way.

I would also like to thank the universe for choosing to be in its current state, making everything we do possible.
VITA

May 2011 ....................................................... B.Tech. Mechatronics Engineering,

S.R.M. University, Chennai. India.

May 2011 to March 2012 ......................... Graduate Engineering Trainee,

Engine R&D – Greaves Cotton Ltd. India.

August 2012 to Present ......................... Graduate Research Associate, Department

of Mechanical Engineering,

The Ohio State University

FIELDS OF STUDY

Major Field: Mechanical Engineering
TABLE OF CONTENTS

ABSTRACT .................................................................................................................. II

DEDICATION ........................................................................................................ IV

ACKNOWLEDGEMENTS ........................................................................................ V

VITA ........................................................................................................................ VI

TABLE OF CONTENTS .......................................................................................... VII

LIST OF TABLES ................................................................................................ XI

LIST OF FIGURES ................................................................................................ XII

CHAPTER 1: INTRODUCTION ......................................................................... 1

1.1 Motivation ........................................................................................................... 2

CHAPTER 2: LITERATURE REVIEW .............................................................. 4

2.1 Introduction ...................................................................................................... 4

2.2 Networked Control System ........................................................................... 4

2.3 Controller Area Network Modeling ............................................................... 6

CHAPTER 3: CAN NETWORK MODELING TOOLS AND METHODS .......... 9
3.1 Computer Networks ........................................................................................................... 9
  3.1.1 Open System Interconnection Model ........................................................................... 9
  3.1.2 Classification ............................................................................................................. 11
3.2 Controller Area Networks ................................................................................................. 14
  3.2.1 CAN Basics ............................................................................................................... 14
3.3 Delays Approximations in Control Systems ..................................................................... 19
  3.3.1 Padé Approximant .................................................................................................... 19
  3.3.2 Thiran Filter ............................................................................................................. 20
3.4 System Identification ......................................................................................................... 21
  3.4.1 MATLAB – System Identification Toolbox ............................................................ 22
3.5 Model Description ............................................................................................................. 23
  3.5.1 Transmission Actuator ............................................................................................. 23
  3.5.2 CP1 Clutch Actuator ................................................................................................. 27

CHAPTER 4: CAN NETWORK SIMULATOR ................................................................. 33

4.1 Introduction ...................................................................................................................... 33
4.2 SimEvents Platform ......................................................................................................... 34
  4.2.1 Entities ..................................................................................................................... 35
  4.2.2 Attributes ............................................................................................................... 35
  4.2.3 Gateways ................................................................................................................ 36
  4.2.4 Queues ................................................................................................................... 36
  4.2.5 Routing .................................................................................................................. 37
  4.2.6 Servers ................................................................................................................... 38
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.7 Generators</td>
<td>39</td>
</tr>
<tr>
<td>4.3 CAN Network Modeling</td>
<td>40</td>
</tr>
<tr>
<td>4.3.1 Fidelities and Assumptions</td>
<td>40</td>
</tr>
<tr>
<td>4.3.2 CAN Simulation Parameters</td>
<td>42</td>
</tr>
<tr>
<td>4.4 Model Calibration and Validation</td>
<td>55</td>
</tr>
<tr>
<td>4.4.1 Intergeneration Time - Mean and Distribution</td>
<td>56</td>
</tr>
<tr>
<td>4.4.2 CAN ID and Priority</td>
<td>60</td>
</tr>
<tr>
<td>4.4.3 Bit Length</td>
<td>60</td>
</tr>
<tr>
<td>4.4.4 Baud Rate</td>
<td>61</td>
</tr>
<tr>
<td>4.4.5 Length of Message and Transmission Time</td>
<td>62</td>
</tr>
<tr>
<td>4.4.6 Packet Loss Probability</td>
<td>65</td>
</tr>
<tr>
<td>CHAPTER 5: EFFECTS OF CAN ON CONTROL PERFORMANCE</td>
<td>69</td>
</tr>
<tr>
<td>5.1 Structure of the Simulator</td>
<td>69</td>
</tr>
<tr>
<td>5.2 Architectures</td>
<td>72</td>
</tr>
<tr>
<td>5.2.1 CAN for Closing the Control Loop</td>
<td>72</td>
</tr>
<tr>
<td>5.2.2 CAN for Supervisory Control</td>
<td>74</td>
</tr>
<tr>
<td>5.3 Effects of CAN network</td>
<td>75</td>
</tr>
<tr>
<td>5.3.1 Performance Evaluation Metrics</td>
<td>76</td>
</tr>
<tr>
<td>5.3.2 Sampling Interval</td>
<td>79</td>
</tr>
<tr>
<td>5.3.3 Quantization</td>
<td>81</td>
</tr>
<tr>
<td>5.3.4 Packet Loss Probability</td>
<td>85</td>
</tr>
<tr>
<td>5.3.5 Bus Speed</td>
<td>88</td>
</tr>
</tbody>
</table>
5.3.6 Priority of Messages.................................................................90

5.3.7 Network Utilization ..................................................................92

CHAPTER 6: DESIGN GUIDELINES AND CONTROL SOLUTION

EXPLORATION ..................................................................................94

6.1 Analysis techniques ....................................................................94

6.2 Design Guidelines ......................................................................101

6.3 Control Solutions ......................................................................105

CHAPTER 7: CONCLUSIONS AND FUTURE WORK .........................112

7.1 Conclusions ...............................................................................112

7.2 Future Work..............................................................................113

BIBLIOGRAPHY ..............................................................................115

APPENDIX: LIST OF SYMBOLS AND ABBREVIATIONS .....................118
LIST OF TABLES

Table 1: Network classification by scale ................................................................. 12

Table 2: Network classification by topology ............................................................. 13

Table 3: Transmission actuator parameters ............................................................ 26

Table 4: CP1 communication CAN messages ......................................................... 28

Table 5: Load test details on CP1 ........................................................................... 30

Table 6: CAN message composition ........................................................................ 47

Table 7: Packet loss simulation ................................................................................ 67

Table 8: Nominal parameters of the CAN simulator ................................................ 76
LIST OF FIGURES

Figure 1: CAN message data frame.................................................................17

Figure 2 MATLAB system identification GUI..................................................23

Figure 3: Transmission actuator schematics......................................................24

Figure 4: Transmission actuator control system...............................................25

Figure 5: MATLAB implementation of transmission actuator control system.........26

Figure 6: CP1 clutch actuator............................................................................27

Figure 7: Position output of CP1 actuator step test at 23.58kg load......................30

Figure 8: System identification of CP1 actuator................................................31

Figure 9: CAN simulator structure with three nodes........................................42

Figure 10: CAN node simulator.......................................................................44

Figure 11: CAN node mask...............................................................................45

Figure 12: CAN bus simulator..........................................................................49
Figure 13: CAN bus simulator mask .................................................................49

Figure 14: Distribution of simulated intergeneration and inter-arrival times ..........57

Figure 15: Distribution of inter-arrival time on CAN network ...................................58

Figure 16: Arrival rate of CAN messages .............................................................59

Figure 17: Quantization effects on CAN based signals ..............................................61

Figure 18: CAN bus signal – bit transmission time .................................................62

Figure 19: CAN bus activity compared to CAN simulator .......................................63

Figure 20: CAN bus activity of CP1 compared to CAN simulator .............................64

Figure 21: Inter-arrival time and packet loss detection ............................................66

Figure 22: Packet loss fraction ..............................................................................66

Figure 23: Structure of the CAN simulator based control system ............................70

Figure 24: CAN controls simulator process flow diagram .......................................71

Figure 25: CAN network for closed loop control .....................................................72

Figure 26: CAN network for supervisory control ....................................................74

Figure 27: Reference input and ideal output performance for transmission actuator plant .........................................................................................................................77
Figure 28: Reference input and ideal output performance for clutch actuator plant........77

Figure 29: RMS error guide ........................................................................................................79

Figure 30: Quantization effects on control performance of transmission actuator (A).....82

Figure 31: Quantization effects on control performance of transmission actuator (B).....84

Figure 32: Quantization effects on control performance of clutch actuator .....................85

Figure 33: Packet loss effects on control performance of transmission actuator.............86

Figure 34: Packet loss effect on control performance of clutch actuator .......................87

Figure 35: Cumulative effect of packet loss and quantization on control performance....88

Figure 36: Baud rate effect on control performance of transmission actuator..............89

Figure 37: Baud rate effect on control performance of clutch actuator .........................90

Figure 38: Effect of the priority of control signal on control performance .....................91

Figure 39: Factors affecting bus utilization on CAN network........................................93

Figure 40: Control system with fractional delay .................................................................95

Figure 41: Effect of fractional delay approximation on control system .......................96

Figure 42: Step response of the TransAct-Loop without the plant delay .......................98

Figure 43: CAN based control system’s loop delays .......................................................98
Figure 44: Fractional delay approximation model for TransAct-Loop .......................100

Figure 45: Relationship diagram – network parameters vs. control performance ..........101

Figure 46: Effect of quantization on 1st order plants of different time constants ..........104

Figure 47: Effect of delays and loop gains on pole location of TransAct-Loop ............108

Figure 48: Gain scheduler middleware implementation on SimEvents ......................109

Figure 49: Gain scheduler middleware performance on TransAct-Loop ...................110

Figure 50: Delay estimation and scheduled gains of GSM .......................................110
CHAPTER 1: INTRODUCTION

The transportation requirements of today’s world are satiated by non-renewable fossil fuels. It is no secret that the fossil fuels are a finite resource that takes millions of years to replenish. It also has led to global warming and degraded air quality over the years. Hence, automotive industry has been seeking alternatives to the fossil fuels to power the next generation of vehicles. To stimulate the innovation and educate the younger engineers, Department of Energy and General Motors organized the EcoCAR 2 competition. It is a three year student engineering competition which requires the students to design and build an energy efficient and low emission vehicle from a stock 2013 Chevrolet Malibu. The competition tests the vehicle for its performance, efficiency and consumer acceptability.

The Ohio State University’s EcoCAR 2 team designed a parallel-series plug in hybrid electric vehicle. The vehicle is powered by two electric motors and an inline 4-cylinder Honda engine. The drivetrain also includes a six-speed automated manual transmission coupled to the front electric motor via a belt drive and the engine and, a single speed gearbox coupled with the rear electric motor.
The design process included extensive model based vehicle simulations and model based controls development for the vehicle. The supervisory controls implementation is done on a dSPACE MicroAutoBox (MABx) which communicates with various devices like the engine controller, inverters for the motor, etc. through controller area networks (CAN). The engine controller also houses the transmission control logic. The transmission controller controls linear actuators to perform gear shifting and the MABx is responsible for controlling the clutch using a smart electro-hydraulic clutch actuator on the CAN network. The vehicle has multiple CAN networks running at 500kbits/s and 1Mbits/s for communicating with all the devices on the network.

1.1 Motivation

Networked control systems are characterized by their use of a communication network in the control loop. A networked control system may use communication network to send the control signal from the controller to the plant or to transmit a feedback signal from a sensor. The effects of communication network on the control performance can be significant in normal operating conditions due to quantization, delays and packet losses.

The advantages of networked control systems are immense. It allows for fewer wiring between components, making the entire system less cumbersome and more reliable. It also introduces the capability of relaying more information than what is generally possible over an analog channel. Hence, allowing more complicated control algorithms to be adopted in the vehicles.
The OSU EcoCAR2 vehicle uses networked control systems extensively over a CAN network. The control system design and simulation are performed on a simulation platform like Simulink with continuous time system models and discrete time controllers. This simulation framework can simulate certain effects of a networked control system but it does not encompass the effects of the communication networks in the control loop in its entirety. Therefore a CAN network model, tailor made for the purpose of simulating control systems is required. This is the primary motivation for developing a CAN network simulator on Simulink platform.

A secondary motivation for the project is the availability of smart actuators in the market. The electro-hydraulic clutch actuator manufactured by FTE is used in the OSU EcoCAR2 vehicle. It can be controlled by sending control signals over a CAN network. The feedback from the smart actuator is also sent over the CAN network. Integrating such actuators into a simulation environment inevitably requires a CAN network simulator which acts as an interface between the actuator model and rest of the components.

The work presented in this thesis presents a CAN network simulator based on Simulink platform for control systems simulations. It also discusses analysis techniques, design guidelines and some control solutions for countering the effects of networked control performance.
CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

Modeling and simulation of CAN behavior has been recorded extensively in the literatures. This section provides an overview about the existing methods and implementations of CAN simulators. The section also presents the existing work in networked control system modeling, delay approximations and CAN based control systems.

2.2 Networked Control System

Networked control systems are characterized by the use of a communication network in their control loop. Control signals and feedback signals are exchanged over a communication network. In the case of CAN, the data exchange happens using data packets called ‘messages’. Several different approaches for modeling the networked control system are available on the literature. Generally, each approach concentrates on capturing a particular trait of the NCS.

The NCS is modeled as a coupled dynamic system of three systems in [1]. The controller, plant and the network are modeled as separate dynamic systems interacting with one
another. The model also incorporates a Try-Once-Discard (TOD) protocol, it tries to send a packet on the network and if the bus is busy at that time the packet is discarded. The arbitration for transmitting is implemented by another protocol called Maximum Error First (MEF). The MEF sends the message which has the highest error metric assigned to it. The error metric is calculated as the difference between the reference signal and the last plant feedback. The backbone of the simulator is a dynamic scheduling methodology to change the prioritization of messages on the fly. [2] models the network of the NCS as a switching and hold device which updates at regular interval. The control signal is considered to reach the plant directly and not through the network. The paper also presents necessary and sufficient conditions for the stability of the system.

In another approach to NCS modeling which is closer to CAN, multiple actuators and sensors which have limited number of communication pathways or channels is considered [3]. Each node is made to wait until the bus is free. It also shows a simplified model of the NCS by ignoring sensor and actuator data which are non-responsive. An extended model which explores the different communication sequences that preserve reachability and observability of the plant is also presented. The specific communication sequences which can make the NCS reachable and observable can be obtained by the algorithm developed in the paper. In [4] a very detailed study on stability of the NCS with quantization and packet loss is presented. The paper details the process for developing a sufficient condition for stability of the system with respect to quantization levels and maximum number of continuous packet loss. [5] presents the NCS model with network induced delays. The delays are model in two different modes, one less than the
sampling interval and another more than the sampling interval. The paper also presents the stability regions for various cases network delays and also explores stability of the NCS with packet loss. The paper also provides delay bounds for stability of the NCS. Another paper [6] proposes a logic based control and stability analysis for a class of linear NCS. The system is modeled with time invariant parameters with time varying perturbations. The paper proposes an NCS modeling problem can be reduced to finding an asymptotically convergent state observer. [7] proposes several control strategies for the NCS. The paper explores different delay characteristics and effects of the delay on the control loop, then it presents different control design methodologies for the NCS. This work can be used for determining a suitable control design method.

Methods for stabilization of NCS with random delays are explored in [8]. The closed loop systems resulting from random delays are treated as jump linear systems and analyzed. Another very important work relevant to NCS research is [9]. It presents the current trends in the NCS and summarizes the ongoing research in the field. It also describes all the fundamental components of an NCS which is very helpful in understanding NCS.

2.3 Controller Area Network Modeling

The CAN has been modeled and simulated by people from diverse fields of study such as computer science, communication engineering, automotive engineering, etc. The fidelities of the simulation differs based on what the designer is looking for. [10] presents a comparative study of effects of modeling a CAN network at different fidelities.
Transaction Level Model, Arbitrated Transaction Level Model and Bus Functional model are the three modeling schemes used for comparison. Further, different analysis parameters are compared in the paper. The paper concludes that as the fidelity increases the simulations time and accuracy of simulations also increases.

A CAN bus behavioral model is presented in [11], it uses VHDL for implementing the behavior of the CAN transceivers. The performance of the simulator is analyzed for three different configurations of the transceiver model architectures. Another simulation method using CANoe software is presented in [12]. The CAN bus is simulated on the CANoe software platform with virtual buses and virtual nodes. The CANoe software is configured to show certain graphic user interfaces for representing dials and gauges. The Communication Access Programming Language is used to automate the virtual nodes. CAPL is a C-like language integrated into the CANoe software platform. The simulations results are analyzed by using certain performance metrics like peak bus load, tracking the data frame for accurate transmissions, etc. The work presented in [13] is very similar to the modeling method used in this thesis. It uses MATLAB/Simulink platform to develop a CAN simulator. SimEvents platform is used to model certain parameters of the simulator. However, the implementation of arbitration and replication is implemented by constructing specific gated algorithms. It also implements the retransmission feature of the CAN network. Certain results for the response time or worst case queuing delay are presented and compared with simulation results from CANoe.
The CAN based control systems are subjected to delays as a consequence of various transmission and queuing delays. Further, digital communications introduce quantization into the signals carried by a CAN message. [14] provides a methodology for determining the pole-zero displacement of systems with coefficient quantization. Although the coefficient quantization effects are not directly relevant to NCS, the methodology is a good way to think about formulating effects of delays. In NCS the input and output signals are quantized. Another paper, [15] presents various techniques to represent a delay in a NCS. FIR and IIR filters are explored and their accuracy in representing a pure delay is examined. The paper also talks about representing fractional delays which are encountered in a NCS framework. The approximations of the delays with such filters are not accurate representation of the delays, but they provide a very valuable tool in analyzing the effects of delays on stability and performance.

A gain scheduled compensation for the effects of time delay on NCS is presented in [16]. The paper explores robust stability of the GSM in the face of varying network induced delays. The results from robust stability analysis is then used for determining a suitable gain scheduled compensator, finally, a gains scheduled $H_\infty$ control is developed. Another such method presented in [17] uses an online delay estimation scheme to determine gains for the gain scheduler design. The paper applies the control techniques to an NCS based on TCP-IP protocol. The concept of Gain Scheduler Middleware is introduced in this paper. The GSM enables the controller node to modify the control input based on network traffic conditions.
CHAPTER 3: CAN NETWORK MODELING TOOLS AND METHODS

CAN network modeling involves concepts and tools from various fields like computer networks, digital communication, discrete event systems, sampled data systems, etc. Some of the methods used in the formulation of this thesis and some basic concepts are explained in this chapter

3.1 Computer Networks

A computer network is defined by interconnection of multiple computers through electronic circuits, communication cables or wireless transmissions for the purpose of exchanging data with one another. A computer network is therefore an entity which enables communication among different computers.

3.1.1 Open System Interconnection Model

Open system interconnection model is a conceptual model for computer networks divided into seven layers assigned with specific functions in a networks context. The model was developed by International Organization for Standardization (ISO) and it is considered as the primary architectural model for computer networks.
The OSI model’s layers have specific task groups assigned to them and are self-contained in their functions. This enables independent implementation of each layer and thereby ensuring no adverse effects on other layers when one layer is modified.

The different layers of the OSI model are,

1. Layer 7: Application layer
2. Layer 6: Presentation layer
3. Layer 5: Session layer
4. Layer 4: Transport layer
5. Layer 3: Network layer
6. Layer 2: Data link layer
7. Layer 1: Physical layer

Generally each layer can interact with at most three OSI layers; two adjacent layers and a peer layer in another computer. The following section provides very brief explanation for each of the layers. Note that this is not an exhaustive list of functions and complexities of the different layers.

Physical layer handles the transmission and reception of bit level data on a physical bus. The bus can be an electrical circuit with cables or a wireless communication link.

Data link layer is responsible for managing error-free communication of data from one node to another. It houses the logic which triggers the physical layer functions. Its other
functions include assembling the data in appropriate frames, appending identifiers, start and end sequences, etc.

Network layer works at a network topology level, its main purpose is to determine the path in a network that a data packet should take based on network conditions. It also handles priority of service, translation of bit level data to physical signals and vice versa.

Transport layer is responsible for management of the data for transmitting on the network. It splits and tailors the larger data fragments into appropriate size for transmission on the network.

Session layer establishes a communication session between two nodes. Its functions include keeping track of session establishment and termination, security and logging of data.

Presentation layer translates the signals received from the network into a user accessible format. For example scaling a signal from 0-1024 values to appropriate pressure values in kPa by scaling and shifting.

Application layer is the user interface which interacts with the user by displaying the signals and accepting commands.

3.1.2 Classification

The computer networks can classified by scale and network topology. This section provides an introduction to network classification.
A node is a device or computer that connects to other devices through a network. The communication medium, usually in the form of a wire, optical cables or wireless link is called a bus.

### Table 1: Network classification by scale

<table>
<thead>
<tr>
<th>Network Type</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personal Area Network</td>
<td>Characterized by proximity to the user and network reach is about 20-30 feet.</td>
<td>Bluetooth, ZigBee, Infrared communication</td>
</tr>
<tr>
<td>Local Area Network</td>
<td>Covers a small physical area like a home, office, school, airport, <em>etc.</em> Lower data transfer rates</td>
<td>Wi-Fi</td>
</tr>
<tr>
<td>Campus Area Network</td>
<td>Connects local area networks in a limited geographical area</td>
<td>University Wi-Fi services</td>
</tr>
<tr>
<td>Wide Area Network</td>
<td>Covers a broader area, usually on a national or international level. It connects local area networks in its region.</td>
<td>Internet</td>
</tr>
</tbody>
</table>
Network topology is the configuration of various elements of a computer network. The topology can be defined physically, based on the cabling, or logically, by the flow of data through the network.

Table 2: Network classification by topology

<table>
<thead>
<tr>
<th>Network Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus - Linear</td>
<td>All the nodes are connected to the same bus but the bus can have only two end points.</td>
</tr>
<tr>
<td>Bus Distributed</td>
<td>All the nodes are connected to the same bus and it can have more than two end points</td>
</tr>
<tr>
<td>Star</td>
<td>All the nodes are connected to a central hub and all the data passes through the hub.</td>
</tr>
<tr>
<td>Ring</td>
<td>Every node is connected to two other nodes. The end points are connected to one another.</td>
</tr>
<tr>
<td>Mesh</td>
<td>Every node is connected to every other node. The number of connections increase exponentially with the addition of each node.</td>
</tr>
<tr>
<td>Tree</td>
<td>Hierarchical system where a ‘central’ node is connected to lower nodes, which are connected to another set of lower nodes and so on.</td>
</tr>
<tr>
<td>Hybrid</td>
<td>A combination of two different network topologies.</td>
</tr>
</tbody>
</table>
3.2 Controller Area Networks

CAN is a serial, two-wire, half-duplex, high-speed communication network. The transmission happens by transmitting the data bits serially on a bus at bit rates of up to 1000 kbits/s. The network is called half-duplex because the nodes communicate with one another but only one node gets to communicate on the bus at a time.

CAN network was developed by Robert Bosch GmbH for the automotive industry in 1986. The network was designed to fulfil the communication requirements between multiple ECUs in a vehicle. Since point-to-point communication was restrictive, a multi-master communication system was developed. Intel corporation fabricated the first CAN chip in 1987. The information furnished in this section is an excerpt of CAN specifications document [18].

The CAN bus can have the speeds of 10kbits/s to 1Mbits/s. However, the choice of bit rate usually depends on the physical length of the CAN network. For example, a CAN network running at 1 Mbits/s is recommended not to have more than 40m of bus length.

3.2.1 CAN Basics

CAN network uses CSMA/CD+AMP (Carrier Sense Multiple Access / Collision Detection with Arbitration on Message Priority). That is, a CAN node should check if the bus is busy before it tries to access the bus. This ensures that the nodes do not try to access the bus while other node is transmitting a message, risking the corruption of both
the messages. The event in which two nodes access the bus simultaneously is called a collision.

CAN identifier: In the CAN network the communication is not primarily based on the device addresses. Each CAN message carries a message identifier called CAN identifier (CAN ID). The CAN messages are transmitted on the network, each node on the network receives the message. Then based on the identifier on the CAN message the node may decide to keep the message or discard it.

Arbitration: When a collision occurs on a CAN network the arbitration mechanism decides which node gets to transmit first. A node which is trying to send a message with the lower CAN identifier gets the preference in transmission. On a bit level, two nodes wait until the bus is idle and then start transmitting their CAN ID bits on the network. Each node also keeps listening to the bus activity, hence each node knows when a collision is happening but they continue transmitting the CAN ID bits as long as both are transmitting the same kind of bits. Since all CAN IDs need to be unique, a dissimilar bit transmission happens at some point, the node which transmits a dominant bit or ‘0’ wins the arbitration and gets to transmit the rest of the message while the other node has to stop transmitting and wait for bus idle. This scheme is called CD+AMP (collision detection and arbitration on message priority).

Frame Types: CAN message transfer is done based on 4 different frames.

1. Data frame carries the data from transmitter to receiver
2. Remote frame is used by a remote node to request another node to send a particular message.

3. Error frame is used for notifying all the nodes that an error has occurred in transmission.

4. Overload frame is used to provide delay between successive Data or Remote frames.

The CAN network is governed by the ISO standards ISO-11898 (high-speed up to 1Mbps) and ISO-11519 (low-speed up to 125kbps). Further the CAN network communication is classified based on the number of identifiers a CAN message uses.

1. Standard CAN (V 2.0A) uses 11 bit identifiers.

2. Extended CAN (V 2.0B) uses 29 bit identifiers.

The two CAN networks also differ in their structure of message frames and identifier positions in a CAN message. Theoretically the CAN V2.0A can have 2032 unique IDs.

The structure of the CAN message’s data frame is shown in Figure 1. The bit length of each section varies with the CAN standard used.
Figure 1: CAN message data frame

The following section explains each section of the CAN message for CAN 2.0A protocol.

Start of frame is the beginning of a data frame and it is 1 bit long and is always a dominant bit or ‘0’ digital level.

Arbitration field consists of the CAN ID and the Remote Transmission Request bit (RTR). The identifier field is 11 bits long and the RTR bit is one bit long and in a data frame it has to be dominant or ‘0’.

Control field has six bits and it carries Data Length Code (DLC – 4bits) and two bits reserved for future expansion. The DLC is used to specify the number of bits in the data field.
Data field is generally 1 to 8 bytes long or 8 to 64 bits long. This field carries the information from one node to another. The information in this field is also called as payload.

CRC stands for Cyclic Redundancy Check. It is an error detection mechanism and it is 16 bits long including the delimiter bit.

The acknowledgement field consists of two bits. The first bit is transmitted by the transmitter as a recessive bit but other nodes overwrite it with dominant bits signifying successful reception of message. This is another form of error detection and reporting scheme. If at least one node detects an error and conveys it in the acknowledgement field, all the nodes discard the message and the transmitter tries to retransmit the message. The second bit is a delimiter bit.

The acknowledgement field is followed by an End of Frame field which is seven bits long. After the end of frame field there is a 3 bit interframe space after which the bus is considered to be idle.

Further, the number of bits present on the CAN bus during the period of message transmission can be more than the sum of all the fields listed here because of bit stuffing. Bit stuffing is a process of inserting a high bit after five consecutive low bits to let the receiving nodes know that the transmission is still on. The receiving nodes destuff the message before using it. However, if the receiver nodes detect more than five consecutive low bits they issue an error.
3.3 Delays Approximations in Control Systems

Time delays manifest in control systems as an effect of the physical system being modeled and the control system itself. Some physical systems like chemical reactions and interconnected structures have an inherent delay in responding. In the case of control systems the delay in processing speed and signal transmission speed from sensors or to actuators cause the delay. Approximating delays numerically can help in analyzing the control system’s performance analytically [19]. There are several ways to approximate delays, two of them have been described in this section.

3.3.1 Padé Approximant

In continuous time systems a time delay of \( t_d \) seconds is expressed as \( e^{-t_d s} \) in laplace domain and in discrete time systems it is expressed as \( z^{-k} \) where \( k \) is an integer value representing the number of time steps of delay time. Padé approximation of the time delay in continuous time systems is a method of matching infinite series expansion of the \( e^{-t_d s} \) with a rational function of ‘s’. The method truncates the series shown in (1) and (2) and picks the coefficients \( a_i \) and \( b_i \) such that (1)-(2) is minimized. The Padé approximant is never accurate unless the number of coefficients extend to infinity. However, the accuracy can be improved sufficiently for controls purposes by choosing the order to be greater than 10. Such approximations help greatly in visualizing the effect of variable delays on the position of poles. In MATLAB Padé approximant is can be calculated by using the function ‘pade()’ [20].
\[ e^{-tds} = 1 - s + \frac{s^2}{2} + \frac{s^3}{3!} + \frac{s^4}{4!} + \ldots \]  \hspace{1cm} (1)

\[ td(s) = \frac{a_N s^N + a_{N-1} s^{N-1} + \ldots + a_0}{b_M s^M + b_{M-1} s^{M-1} + \ldots + b_0}; M \geq N \]  \hspace{1cm} (2)

### 3.3.2 Thiran Filter

In discrete domain the delays are represented as an integer multiple of the sampling interval. But in many cases the time delay is not an integer multiple, such delays are classified as fractional delays. Often, fractional delays are ignored by rounding it off to the closest integer value or to the closest higher integer value.

The fractional delays can be modeled more accurately by several methods described in [21] and [15], using FIR and IIR filters. One such method proposed by J.P. Thiran uses an IIR filter to model fractional delays. This method is especially attractive to control system analysis because the filter coefficients are given by a closed form equation (3). In MATLAB the Thiran filter can be obtained by the command ‘thiran()’ [20].

\[ a_k = (-1)^k \binom{N}{k} \prod_{n=0}^{N} \frac{D - N + n}{D - N + k + n}; k = 0,1,\ldots,N \]  \hspace{1cm} (3)

Where, D=delay/sampling interval and \( \binom{N}{k} = \frac{N!}{k!(N-k)!} \) is a binomial coefficient.

\[ T(z) = \frac{(a_N z^N + a_{N-1} z^{N-1} + \ldots + a_1)}{(a_0 z^N + a_1 z^{N-1} + \ldots + a_N)} \]  \hspace{1cm} (4)
Where, $N = \lceil D \rceil$ is the order of the filter and $k$ is the index.

### 3.4 System Identification

System identification is the process of determining a mathematical model of a physical system using the response of the system to various inputs. The system identification is classified into three main categories based on the amount of information known a priori about the system.

1. White box models: The behavior of the system and its parameters are known.
2. Gray box models: The behavior of the system is known but its parameters are not.
3. Black box model: Neither the behavior nor the parameters are known.

The first step in a system identification process involves experimentation with the physical system to record its outputs to various controlled stimuli. After enough data about the output of the system is available, an iterative process of choosing a model structure and parameters to match the outputs of the physical system with corresponding inputs. Model identification of a dynamic process is rather difficult and a large number of iterations are required to get an accurate model. Hence the process is generally done with the help of a software like MATLAB. The choice of the model structure is made based on the general idea about the system.
3.4.1 MATLAB – System Identification Toolbox

The MATLAB system identification toolbox can be called by the command ‘ident’. The GUI tool lets the user to load input and output traces of the physical system from multiple experiments. Once all the data is loaded, several experiments can be merged into one. At this point the user is prompted to choose a model structure in the form of a transfer function, correlational models, state space model, polynomial models, non-linear models, etc. The system identification toolbox then attempts to fit the parameters of the model to recreate the outputs of the system with corresponding inputs. Multiple models structures can be used at the same time and compared with one another based on their error metric. The system identification toolbox also allows the user to use one of the experimental dataset for a validation test. Figure 2 shows the interface for the system identification toolbox.
The parameters of the final fit model can be extracted by clicking on the model.

3.5 Model Description

The CAN network simulator uses two real world plant and controller models for studying the effect of CAN network on the control performance. The transmission actuator model was developed by Teng Ma as a part of his master’s thesis and the clutch actuator (CP1) is developed using system identification methods.

3.5.1 Transmission Actuator

The transmission actuator is a combined system of a SKF CAHB-10 linear actuator powered by PMDC motor, levers and cables. The actuator was modeled in great detail by
Teng Ma in his master’s thesis [22]. The same model and controller is used in simulations on CAN network. The following section briefs the model parameters and structure.

The linear actuation is made possible by a lead screw mechanism housed inside the SKF CAHB-10. Figure 3 shows the schematics of the actuator along with the linkages [22].

![Figure 3: Transmission actuator schematics](image)

The plant model of the transmission actuator involves a mathematical model of the PMDC motor, kinematic linkages (levers and gears), friction and lumped equivalent inertias reflected on the motor shaft. The model also incorporates a lumped transport and sensor delay into it.
The equations (5) and (6) shows the transfer functions corresponding to the transmission actuator.

\[
\frac{X(s)}{V(s)} = \frac{1}{9.28800518 \times (L_a s + R_a)(J_m s + B_m) s + K_t K_b s} e^{-0.024 s} K_t
\]

\[
\frac{V(z)}{E(z)} = \frac{33.68 z^3 - 64.62 z^2 + 30.98 z - 0.03474}{z^3 - 0.9047 z^2 - 0.09316 z - 0.002102}
\]

The controller \(C(z)\) is designed using discrete time controller design methods, it is a combination of a discrete time PI controller and a compensator to address the output delay of the system.

The described system is implemented on MATLAB/Simulink platform as shown.
in Figure 5. The system is split into controller and the plant model for implementation on the CAN network at the dashed red line.

Table 3: Transmission actuator parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electric motor inductance ($L_a$)</td>
<td>514.4e-06 H</td>
</tr>
<tr>
<td>Electric motor resistance ($R_a$)</td>
<td>0.88397 ohm</td>
</tr>
<tr>
<td>Back-emf constant ($K_b$)</td>
<td>0.0190</td>
</tr>
<tr>
<td>Torque constant ($K_t$)</td>
<td>0.0190</td>
</tr>
<tr>
<td>System equivalent inertia ($J_m$)</td>
<td>1.111e-5 kg. m²</td>
</tr>
<tr>
<td>System equivalent viscous coefficient ($B_m$)</td>
<td>1.2867e-5 N.m/rad</td>
</tr>
</tbody>
</table>

Figure 5: MATLAB implementation of transmission actuator control system
3.5.2 CP1 Clutch Actuator

Clutch actuator is a collection of a smart electrohydraulic actuator and throw-out bearing piston. The smart electrohydraulic actuator is comprised of an electric motor with linear actuation mechanisms which actuates a master piston. The piston moves a column of hydraulic fluid, transferring the force to the throw-out bearing piston which is then used to actuate the clutch. The electrohydraulic actuator is ‘smart’ because it has on board controller which helps in accurate positioning of the master piston. The position of the master piston can be requested by an external device by sending appropriate CAN messages to the electrohydraulic actuator. The electrohydraulic actuator is manufactured by FTE Automotive GmbH and it is named CP1 (Controlled Piston). However, in this thesis the combination of the CP1 and the throw-out bearing is termed as CP1.

![Figure 6: CP1 clutch actuator](image)
For the modeling purposes all the components were lumped together and treated as a single system. The Figure 6 shows the CP1, the actuator can be controlled only through CAN messages. The on board CAN transceiver broadcasts the current position of the master cylinder and line pressure of the hydraulic fluid at regular intervals.

Table 4: CP1 communication CAN messages

<table>
<thead>
<tr>
<th>CAN Message</th>
<th>CAN identifier</th>
<th>Purpose</th>
<th>Interval [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCU_HYB_01</td>
<td>0x7F</td>
<td>Position and Pressure feedback</td>
<td>5 (cyclic)</td>
</tr>
<tr>
<td>MSG_HYB_01</td>
<td>0x3A6</td>
<td>Sleep Acknowledgment</td>
<td>100 (cyclic)</td>
</tr>
<tr>
<td>MSG_HYB_15</td>
<td>0x96</td>
<td>Position Request and Mode selection</td>
<td>10 (cyclic)</td>
</tr>
</tbody>
</table>
Table 4 describes the CAN messages used for communicating with the CP1. The main control message 0x96 has a 10ms cycle time which can be considered as the sampling time for the system.

A MicroAutoBox (MABx) was coded to communicate with the CP1 through CAN messages. The dSPACE ControlDesk application was used to address the operation of the MABx in real time. This setup was used for conducting experiments required for the system identification. An oscilloscope was used to record the bus level electrical activity on the CAN bus. A CAN message logger was used as a data acquisition device.

The CP1 was subjected to several step request tests and step loading tests. The position request and load were considered as two inputs to the system. Response of the CP1 to these inputs were logged through CAN messages.

It is important to note that the CP1 is an actuator with on board control system to maintain its state at a desired value. Hence, any identification test we perform will be of the composite plant-controller. The CP1 is designed to accept position command in terms of the spindle position of the motor in degrees. The range that was used in the experiments were from 0 to 3500 degrees.

The step tests were conducted by giving position requests to the CP1 and loading the throw-out bearing with standard weights. The position requests were from [0-1500] and [0-3000] at loads of [17.46, 23.58, 29.25, 34.92] kg.
Figure 7: Position output of CP1 actuator step test at 23.58kg load

Table 5: Load test details on CP1

<table>
<thead>
<tr>
<th>Test number</th>
<th>Initial Load (kg)</th>
<th>Step increase (kg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17.46</td>
<td>23.58</td>
</tr>
<tr>
<td>2</td>
<td>17.46</td>
<td>29.25</td>
</tr>
<tr>
<td>3</td>
<td>17.46</td>
<td>34.92</td>
</tr>
</tbody>
</table>
The loading test was performed by providing a constant position request to CP1 and increasing the load on the throw-out bearing as a step. The load tests were conducted with two position requests at 1500 and 3000 with the loading pattern shown in Table 5.

The logged CAN data was resampled at 1 kHz to use as raw data for the system identification toolbox. Since the data collected was for the controller-plant composite system, a position request to position out model was chosen. The model structure was 2\text{nd} order continuous time transfer function. The system identification results and GUI are shown in Figure 8. Equations (7) and (8) are the transfer functions of the position output with respect to position request and load.

\[
\frac{Pos_1(s)}{PosReq(s)} = \frac{[-0.8142s + 896.2]}{[s^2 + 47.95s + 896.5]} \tag{7}
\]
\[
\frac{Pos_2(s)}{Load(s)} = \frac{[-0.3369s + 0.02634]}{[s^2 + 0.02957s + 5.914]}
\] (8)

The transfer functions are implemented on MATLAB/Simulink platform and are used in CAN network simulations as an example of supervisory control over CAN network. Smart actuators are very useful in control applications as it reduces the load on the control software; on the downside it gives less control on the intricacies of the plant dynamics.
CHAPTER 4: CAN NETWORK SIMULATOR

4.1 Introduction

CAN communication is a carrier-sense, multiple access protocol (CSMA) that has priority arbitration and collision avoidance (CD-AMP) built into it. The CAN communication happens by transmitting and receiving packets of data called “messages” on a shared bus. Further a CAN network can be explained in terms of the OSI model. The physical layer corresponds to the physical electrical bus and the digital communication happening through two voltage levels for representing a digital 0 and 1. The data link layer corresponds to the logic that governs the arbitration, bus access and assembling the parts of the message according to the protocol being used. On a physical level, a dedicated microcontroller is generally used to execute the CAN communication; these devices are also known as CAN transceivers.

The CAN network can be modeled at several different levels and with varying fidelities depending on the purpose of the simulation. A comprehensive and detailed simulator will have to implement bit level communication along with the higher level logic on the same platform. Such a simulator would be very helpful in determining the bit level behavior of the CAN networks; although also be computationally laborious. The focus on building
the described simulator is on control systems simulation over CAN networks and bit level communication implementation is omitted while preserving the behavior of the CAN network.

Several different simulation platforms can be used to establish the behavior of CAN network; but two primary approaches are time-based and event-based simulation. In time-based simulations, the behavior of the network activity is lumped into network delays and logic for implementing the CSMA/CA. In event-based simulation the behavior of the network is modeled as discrete events with network behavior represented by movement of the discrete entity through the network. It is simulated closer to the actual CAN network and network delays manifest as a result of the behavior. In this simulator event-based simulation is used.

The packet data or messages can be modeled as a discrete entity with information. The transmission and reception of such messages are discrete events. A detailed modeling approach is presented in this chapter.

4.2 SimEvents Platform

The discrete event simulation can be performed on several commercially available platforms like MATLAB®/Simulink, Flexsim, Lanner, NetSim, Arena, etc. and open source platforms like adevs, Fascimile, MASON, etc. The current simulator is implemented on MATLAB®/Simulink platform along with the discrete events toolbox SimEvents®.
SimEvents provides simulation tools which comprise of entities, queues, routing methods, generators, servers and signal management tools. The different components of the SimEvents toolbox and their relevance to the CAN network simulation are presented in this section.

4.2.1 Entities

Entities, as defined in the SimEvents framework is analogous to a physical object which is defined by its properties or the set of information it carries. The entity may be a block of aluminum to be machined in a CNC, a tray of food at a restaurant or a packet of data in a communication network. An entity can be transported inside the simulation system through queues, servers, gateways, switches, etc. An entity is associated with certain attributes and the attributes are preserved as long as the entity is preserved. An entity is generated by an entity generator block; entities can be generated based on time or based on events by using their respective blocks. Every entity, like a physical object has to be accounted for. Once an entity is generated it has to go to an entity sink after being propagated through other entity handling blocks.

4.2.2 Attributes

Attributes are properties or data associated with the entities; they are the information contained in an entity. An example of an attribute is the data on a CAN message, while a CAN message is an entity. More than one attribute can be associated with an entity by the set attribute block. An attribute can be associated with an entity from an external source
or defined internally. A typical external attribute would be the data on a CAN message and an internal attribute would be a constant CAN message identifier or CAN ID. Similarly, an attribute can be extracted from an entity using the get-attribute block.

4.2.3 Gateways

Gateways are the blocks which convert a time-based signal to an event-based signal and vice versa. This is a requirement of the simulation platform and does not have significance for the network simulator. Anytime a signal in time from Simulink platform has to be added to the SimEvents model or when an event based signal has to be sent to the Simulink platform, the signal has to go through a gateway block.

4.2.4 Queues

Queues are the blocks that work as a storage unit for entities. An entity is stored in a queue while it is waiting to pass through the consecutive block either because it is occupied by another entity or because a switch leading to next block is blocking the path.

Three types of queues are available on the SimEvents; FIFO, LIFO and priority queue.

First-In First-Out (FIFO) queue can store a predefined maximum number of entities and they are released in the order they entered the queue. Last-In First-Out queues behaves similar to the FIFO queues but it releases the latest entity to enter it, first.

Priority queues are qualitatively very different from the LIFO and FIFO queues; the order in which the entities are released are determined by an attribute or an external signal and
do not depend on the order of the entities entering it. The entity release sequence can be controlled by assigning a priority attribute to each entity and configuring the priority queue to release them in the ascending or descending order of the priority attribute’s value. Queuing is especially helpful to simulate systems in which entities share a common resource; for example if the production of aluminum blocks to be machined by a CNC is faster than the machining time, then the aluminum block will have to wait in a queue until the CNC machine is freed up. In the context of a CAN network, the queues hold the CAN messages before they are transmitted on the network bus.

4.2.5 Routing

The routing blocks help direct the entities along different paths in the discrete event simulator. There are several routing methods in the SimEvents, each having a significantly different role than the others.

Input Switch: Multiple paths enter the input switch block and only one path leads out. The input path that should connect to the output path is specified in the block and it is treated as a direct connection. All other paths leading to the input block are blocked and no entities may pass through.

Output Switch: The output switch is the counterpart of the input switch, but it has certain special features. A single path enters the block and the entity can be routed to any one of the output paths. The choice of the output path a certain entity should take can be specified in the block or from an external signal or an attribute. The routing mechanism
which can be triggered by an attribute, is extremely useful while implementing sorting of entities. In the case of CAN networks, every CAN node receives all the signals but only certain signals with a specific CAN ID are useful to the node. Using the output switch the messages can be filtered based on the CAN ID.

Path Combiner: Entities can enter the block through multiple paths but they are all routed to one output path. The path combiner is very similar to the input port but it does not selectively block any of the input paths. However, it does block all the input paths if the output path is blocked.

4.2.6 Servers

A server is used to represent a process which takes a finite time to complete. The entities enter the block and after a finite time they are released through the output port. A server can be used to simulate transport delays, process time and number of independent handlers of the entity coming in. For example, the time delay as a consequence of transporting an aluminum block on the conveyor through a distance can be simulated; the time spent in a CNC machine for machining the aluminum block also can be simulated. Further in the context of CAN networks, the transmission delays can be simulated using servers. There are three types of server blocks on SimEvents. The infinite server block can accept infinite number of entities from the input port and release them to the output port after a defined time period. This represents having an infinite number of pathways for the entity to travel from one end to the other. It is analogous to having an infinite number of CNC machines to process the aluminum blocks. The N-Server block has N
pathways and a single server has one pathway. The single server will accept only one entity, store it for a time period and release it through the output path. During this period of time the input port will be blocked. If, at the end of the specified time interval the output path is blocked, then the entity will stay inside the server until the output port becomes free.

The single server block is useful to simulate the CAN network bus, since the bus can be occupied by only one CAN message at a time. It also simulates the transmission delay over the network.

4.2.7 Generators

The generators are a class of blocks in the SimEvents which generate the entities or discrete event based samples. Entity generators, as the name suggests, generates entities which are then used to carry attributes across the system. Entity generators are of two types; time based and event based. Time based entity generator produce an entity at periodic time intervals. The time interval can be specified internally in the block or governed by an external signal. The event based entity generator is governed by an external signal. The block can be configured to produce an entity when the external signal meets a certain criterion.

Function call generators produce a trigger signal to execute a function when a certain event happens. The event that drives the function call generator can be a periodic time
interval or an external event. The execution of the function takes precedence over the system, much like calling an external function while executing a larger program.

Signal generators contain an event based random signal generator. The happening of the event is back propagated to block, which prompts it to generate a random number. The random number generator can be configured to have a specific distribution for the random numbers it produces.

4.3 CAN Network Modeling

The CAN network model for control systems simulation relaxes certain requirements and adds certain extra constraints as compared to an ideal simulator. The SimEvents platform explained in the last section is used to simulate the parameters and features of the CAN network. The CAN network protocol V2.0A (ISO11898) described in section 3.2 is implemented on the simulator. The following sections explain the specifications and features of the CAN simulator.

4.3.1 Fidelities and Assumptions

The CAN simulator is designed to accurately represent the following features of a CAN network,

1. Collision avoidance protocol.
2. Arbitration based on message priority defined by CAN identifier.
3. Carrier sense multiple access protocol for sharing the common bus.
4. Multi-master architecture with address independent nodes which can perform without the knowledge about other nodes.

5. Message filtering based on CAN identifier.

6. Modularity of transceiver nodes to enable flexibility in modifying the number of nodes on the network.

7. Faster simulation to accomplish integration onto other simulators.

The simulator does not simulate bit level behavior of the network. The consequences of such a simulation on individual parameters of the network is explained in the following section. A major difference in the simulator and the real CAN network is the ability to perform error handling and retransmissions. The transmission of data frames is implemented, while error frames and remote frames are not.

**Structure of the simulator:** The simulator is built in the architecture of an automotive CAN network as shown in Figure 9. The CAN bus is built independent of the nodes, so that the number nodes in the network can be modified easily. The CAN bus block handles the transmission, routing, packet loss, replication and arbitration of the messages.

The nodes are designed for modularity; nodes can be added to the simulator by copying an existing node and changing its identifiers. Nodes also handle the filtering of relevant messages.

This structure is very similar to having multiple controllers on a single CAN network. The addition or removal of a certain controller does not affect the bus structure.
4.3.2 CAN Simulation Parameters

4.3.2.1 Quantization

Quantization of the signals transmitted on the CAN network is an effect of discrete time sampling and finite bit length of the CAN messages.

Discrete time sampling uses analog to digital converter (ADC) along with a sample and hold device to read the analog signals. The digital equivalent of the analog signal is stored in a buffer, which is then accessed by the controller to do the processing. Any ADC is characterized by the resolution to which it can convert the analog signal, most
commonly used ADCs have 10, 12, 14 or 16 bit resolution. The cost of an ADC increases with higher resolution and speed of conversion.

The resolution of the ADC determines the smallest detectable change in the signal being sampled. Analogous to the sampling frequency’s relationship to the maximum reproducible frequency content of the signal being sampled, the resolution determines the minimum change in signal that can be reproduced. The following section illustrates the effect of resolution.

The minimum change detectable from an analog signal is given by,

\[ \Delta S = \frac{\max(S(t)) - \min(S(t))}{2^{\text{resolution}}} \]  

Where, \( S(t) \) is the analog signal being digitized, \text{resolution} is the number of bit resolution the ADC delivers. Note that the equation applies for an ADC which can handle the signal levels of \( S(t) \).

Once digitized, the reproduction of the analog signal depends on the digital to analog converter (DAC) resolution. It is generally cheaper to have higher resolution DACs than higher resolution ADCs, hence the limiting factor for resolution is the ADC.

Message length and number of bits used for representing a signal in a CAN message also account for quantization. A CAN message can carry 64 bits of data but often the 64 bits are used for sending multiple signals of shorter bit lengths. The quantization effect due to the number of bits used for representing the signal in a message is given by,
\[ \Delta S = \frac{\max(S(t)) - \min(S(t))}{2^{\text{bit length}}} \]  \hspace{1cm} (10)

Where, \textit{bit length} is the number of bits representing the message. Note that if a signal is sampled with \(N\) bit resolution, having a higher number of bits to represent it in a message does not add any resolution to the signal. Similarly the DAC also acts as a limiting factor for resolution while reproducing the signal.

Figure 10 shows the implementation of the CAN node simulator, the quantization block quantizes the signal based on the bit length of the message segment used to represent the signal and the extreme values that the signal can take. Figure 11 shows the mask for the node block which translates the higher level information like bit length chosen to represent the signal, into the quantization interval.

![Diagram of CAN node simulator](image)

Figure 10: CAN node simulator
4.3.2.2 CAN Message Identifier

Every CAN message sent on a network must have a CAN identifier which is a unique number assigned to each message. The CAN ID is used as a key to decode the content of the data a message carries.

The CAN ID intended for a particular message is associated with an entity as an attribute. Figure 11, the CAN node mask enables the higher level function of assigning a CAN ID to the outgoing message. The outgoing message will always carry the attribute, a constant CAN ID and preserve the attribute throughout the communication pathway.
4.3.2.3 Payload

The data carried by the CAN message is called the payload. Generally the data tends to be a sampled signal or a status of a process, in either case the data changes over time based on events or time. The data is associated with the CAN message as an attribute called value and it is sent out. Note that this data is quantized because of the sampling and choice of a finite bit length to represent it.

4.3.2.4 Message Generation Scheme

A CAN message is generated in two ways

1. Based on an event: The event can be a change in the value of an incoming signal or an internal event in the node like an error detection.
2. Based on time: A message is generated every $T_{int}$ seconds called intergeneration time.

The CAN network specification does not restrict when a message has to be generated and transmitted on the network. However, there is a physical limit to the periodicity of messages. Once a message is generated, it takes a certain time to transmit it on the network, so it is not feasible to have $T_{int}$ smaller than the transmission time.

The time-based message generation is implemented on the simulator using the time-based entity generator block. The entity generator block is given the $T_{int}$ as an input and the
block generates an entity every $T_{\text{int}}$ second interval. The intergeneration time is specified in the CAN node simulator mask.

4.3.2.5 Message Length

A CAN message is a composition of several segments of information which is illustrated in the following table. [18]

Table 6: CAN message composition

<table>
<thead>
<tr>
<th>Segment Description</th>
<th>Number of bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Start of frame (SOF field)</td>
<td>1</td>
</tr>
<tr>
<td>2 Arbitration field (CAN ID &amp; RTR)</td>
<td>12</td>
</tr>
<tr>
<td>3 Control field (Reserved and DLC)</td>
<td>6</td>
</tr>
<tr>
<td>4 Data field</td>
<td>64</td>
</tr>
<tr>
<td>5 CRC field</td>
<td>16</td>
</tr>
<tr>
<td>6 Acknowledgment field</td>
<td>2</td>
</tr>
<tr>
<td>7 End of frame</td>
<td>7</td>
</tr>
<tr>
<td>8 Intermission field</td>
<td>3</td>
</tr>
<tr>
<td>9 Bit stuffing – worst case scenario</td>
<td>22</td>
</tr>
</tbody>
</table>
From the Table 6, there are 111 bits in a CAN message. However, the length of the CAN message can vary depending on the number of data bits it is carrying and bit stuffing. It is assumed that the data field will always have 64 bits. Bit stuffing is a technique used to detect errors in communication and it works by inserting a bit of opposite polarity when five consecutive bits are of the same polarity. The receiving node removes the stuffed bits to retrieve the original message. The worst case scenario for bit stuffing is 22 bits; for the current simulator it is assumed that all messages will have 9 stuffed bits.

The message length may change if a different version of CAN is used (CAN V2.0B). As mentioned earlier, the simulator does not implement bit level behavior and hence all the bits of the CAN message are lumped into one data packet or entity in the SimEvents platform. The sending and receipt behavior of the CAN messages is preserved, however, changes in message length due to stuffed bits are not represented well.
Figure 12: CAN bus simulator

Figure 13: CAN bus simulator mask

Figure 13 shows the CAN bus simulator mask; the message size parameter along with the bit rate determines the time taken to transmit a message.
4.3.2.6 Bit Rate

Bit rate or baud rate is the physical transmission speed of the electrical bus, it is measured in terms of bits transmitted per second. CAN communication supports 33 kbits/s to 1Mbit/s. The CAN protocol requires all the nodes on the network to be at the same baud rate. The baud rate along with the message length determines the time taken for transmission of one message.

\[ T_{\text{trans}} = \frac{\text{Message Length}}{\text{Baud rate} \times 1000} \text{ [s]} \]  

(11)

Where, \textit{Message Length} is expressed in number of bits, spanning SOF field to intermission field, and \textit{baud rate} is expressed in kBits/s.

Consider a 500 kbits/s baud rate network, the transceiver and the bus is capable of switching and transmitting 500,000 digital voltage levels per second; i.e. one bit can be transmitted in 2µs. Therefore, a CAN message of length 120 bits can be transmitted in 240µs. Common baud rates are 125 kbits/s, 250 kbits/s and 500 kbits/s.

The simulator uses the server block to represent the time taken to transmit a packet of data. The data packet enters the server block and leaves after \( T_{\text{trans}} \) seconds. In the bus simulator mask shown in Figure 13, the bit rate or baud rate is specified. Bit rate along with the message length parameter determines \( T_{\text{trans}} \).
4.3.2.7 Arbitration based on Message Priority (AMP) and Collision Avoidance

The CAN network relies on the collision detection and avoidance scheme for robustness in message transfer, and relies on priority arbitration to achieve faster communication under high bus load conditions.

The CAN network is built to deal with simultaneous attempts to access the bus and transmit a message by the use of priority arbitration. When two nodes try to access the bus at the same time, the nodes start transmitting their CAN IDs, this is called collision. On a bit level, the node which transmits a lower arbitration bit wins the arbitration and gets to complete the transmission while the other node withdraws transmission and waits. This helps to resolve collisions. Compared to the Ethernet protocol, in which all the nodes that try to transmit shut down when they detect a collision, they wait for a random time span and then try to transmit again. This is less efficient and can lead to high level of uncertainty in transmission at high bus loads.

The simulator employs the server block to implement the collision avoidance and the behavior of waiting of a lower priority message. The server block as explained in 4.2.4 holds a packet of data or message and releases it after a specified length of time. During this time interval no new message is allowed to access the bus and they are made to wait in their respective queues.

The priority arbitration is implemented using priority queues as explained in 4.2.4. The priority queues act as a pathway from the nodes to the bus in normal conditions when
messages try to access the bus sequentially without overlap. However, when a message is being transmitted through the bus (when the server is holding an entity and hasn’t released it) the priority queue acts as a queuing device. Further, if two messages are in the queue waiting to be transmitted, the priority queue releases them in the order of their CAN IDs, in ascending order with lower CAN ID value getting higher priority. The blocks are connected as shown in Figure 12.

The bus model also implements the CSMA feature by providing the nodes access to every message transmitted on the bus. All the entities on the network are replicated and each node is sent a copy of the entity. The nodes then have to filter and retain the messages they are interested in processing further. Figure 10 shows the mechanism by which a node filters the incoming messages and passes on the relevant ones. The node mask shown in Figure 11 specifies the message that a node is interested in using.

### 4.3.2.8 Bus Utilization

Bus utilization is a metric used to express the traffic density on the CAN bus. It can be defined as,

\[
U_{\text{bus}}(t) = \frac{1}{t} \int Bus \text{ BusyFlag}(t) \, dt
\]  

(12)

Where, \(U_{\text{bus}}(t)\) is the utilization of the bus at time \(t\), \(Bus \text{ Busy Flag}\) is an indicator of the bus being used for transmission of a message. It takes the value 1 when the bus is busy and 0 otherwise.
The bus utilization heavily depends on the periodicity of messages defined by $T_{\text{int}}$ and the transmission time defined by $T_{\text{trans}}$. If all the $N$ messages on the network have the same periodicity of $T_{\text{int}}$ then in a time interval spanning 0 to $T_{\text{int}}$,

$$U_{\text{bus}}(T_{\text{int}}) = \frac{T_{\text{trans}}N}{T_{\text{int}}}$$  \hspace{1cm} (13)

Note that if $N*T_{\text{trans}}$ exceeds $T_{\text{int}}$ the bus load becomes 1 and the messages in the queue will be timed out and lost. Hence the $N*T_{\text{trans}} < T_{\text{int}}$ is a condition for safe operation of the network. The bus utilization can be observed in real time on the simulator through the statistics of the server block.

### 4.3.2.9 Delays

In the networks context, a delay can be generally defined as the time elapsed between the generation of a data packet and receipt of the data packet by another node. Note that the time taken by the CAN transceivers to assemble and process the messages are not included in the delay definition. This is because the transceivers generally work above 26 MHz and time to handle a CAN message would be in the order of couple of micro seconds.

Due to the requirement of the CSMA/CA-AMP implementation the transmission of messages can be carried out only in a certain sequence, first the transceiver has to wait for the bus to become free or idle, it has to then wait for its turn to transmit as explained in 4.3.2.7 and finally when it transmits it takes a finite time to finish transmission.
As defined earlier, $T_{\text{trans}}$ is the time taken to transmit all the bits of a CAN message on a network with a specified baud rate. Then the total time delay can be expressed as,

$$T_{\text{CAN\_delay}} = p \cdot T_{\text{trans}} + N \cdot T_{\text{trans}} + T_{\text{trans}}$$

(14)

Where, $p \cdot T_{\text{trans}}$ is the wait time for the bus to become idle while another node is transmitting a message on the bus and $p$ is a factor in the interval $[0,1]$. Note that the value of $p$ cannot be determined analytically. $N$ is the number of messages with higher priority in the queue. $N \cdot T_{\text{trans}}$ is the wait time for letting all the messages of higher priority to be transmitted on the network. Since $p$ and $N$ are probabilistic in nature, $T_{\text{CAN\_delay}}$ is expected to have a distribution. The $T_{\text{CAN\_delay}}$ is also bounded by $[T_{\text{trans}}, (2+N_{\max}) \cdot T_{\text{trans}}]$.

4.3.2.10 Packet Loss

Data packets or messages are lost in a CAN network through various mechanisms, one of the main reasons for packet loss is bit transmission error. When the other nodes detect a faulty transmission, the data packet is discarded by all the nodes and the sender is asked to retransmit the message. Another reason for packet loss includes time-outs while waiting in a queue. Since packet loss rate cannot be determined analytically, a probability distribution for packet loss is implemented in the simulator.
4.3.2.11 Cyclic Redundancy Check (CRC)

CRC is one of the error detection mechanism in communication networks. In the context of CAN communication, CRC is a 15 bit code appended after the data field of the CAN message. It is used for determining whether the transmission has happened correctly at the receiving node.

4.4 Model Calibration and Validation

This section focuses on the methodology used to determine the nominal values for the parameters of the simulator and validation of the same by comparison with experimental data is explained in this section. Nominal parameters for the system includes some design parameters such as baud rate, which we have complete control over. Some dependent parameters, like transmission time ($T_{trans}$), manifest as an effect of the design parameter values. Certain parameters, like loss rate, are completely dependent on the behavior of the CAN bus.

The experimental data is a collection of CAN activity logs of the OSU EcoCAR2 tests. The data contains hundreds of CAN messages collected over different tests. The data contained in the CAN logs are not used, but the message generation and arrival times are studied from it. To augment the experimental data, bus level electrical activity recorded from an oscilloscope is used.

The CAN activity log is processed in MATLAB and the inter-arrival times of CAN messages are extracted. Only the CAN message sequences, which have more than 1000
sample messages are used. The process went through 104 different CAN message sequences and about 2.5 million samples of messages. This ensures that there is sufficient data to draw conclusions about distribution of inter-arrival times.

4.4.1 Intergeneration Time - Mean and Distribution

The intergeneration time $T_{\text{int}}$ is the time between creations of two messages of the same CAN ID. A CAN node or a controller is generally set to a constant intergeneration time. Some of the commonly used intergeneration times are 1ms, 2ms, 5ms, 10ms, 50ms and 100ms. A nominal value of 4ms intergeneration time is used for simulations; the intergeneration time is set to have a uniform distribution spanning the interval [4ms, 4.1ms]. Figure 14 shows the simulation results of the 4ms intergeneration time for a CAN message under different bus loading conditions. It was observed that while the intergeneration time remains the same, inter-arrival times at the receiving node changes with bus load. This behavior is an effect of the constant transmission delay and variable queue waiting delay. The inter-arrival time was calculated by determining time interval between two consecutive message arrivals. Hence, the distribution also shows inter-arrival times lesser than the $T_{\text{int}}$ also. As the bus loading increases and the number of messages on the bus increases, the inter-arrival distribution tends to become normal.
Figure 14: Distribution of simulated intergeneration and inter-arrival times

The three cases had a mean of 4ms and standard deviation of 1.02e-4, 1.35e-4 and 3.83e-4 respectively.

The trend seen in Figure 14 indicates that although the distribution is becoming wider, the data packets are coming only between discrete intervals. This is expected because the message packets can be delayed by a multiple of the transmission time and a fraction of transmission time. Similar trends can be seen in Figure 15 also. However, as the simulation time is increased and the number of messages on the network is increased the inter arrival time distribution will become more continuous.

The experimental data corroborates the simulation behavior by showing similar trends in the inter-arrival time distribution.
Figure 15: Distribution of inter-arrival time on CAN network.

Figure 15 shows the distribution of many different CAN message sequences, each sub-figure shows the distribution around a different mean value. It is important to note that the intergeneration time was set to the mean value of the observed distributions. The top right corner of the sub-subfigures indicates the number of distributions plotted in it.
It is observed that the distribution around the mean is not the same and varies according to the bus traffic as shown in the simulations. Figure 16 shows the map of CAN message inter-arrival time pattern. Each marker on the graph corresponds to a CAN message sequence’s inter-arrival time distribution. It is observed that the mean value of the distributions are very close to their nominal values. However, the standard deviations of the distributions varies significantly. In a CAN network with a lot of messages and high bus load, higher priority messages tend to have lower standard deviations and vice versa.

![Figure 16: Arrival rate of CAN messages](image)

Figure 16: Arrival rate of CAN messages
4.4.2 CAN ID and Priority

CAN ID can be arbitrarily chosen for a message as long as care is taken to ensure the CAN IDs are in the same order as the priority of the messages. A CAN ID with smaller magnitude has higher priority during arbitration. For the simulation, CAN IDs 1 and 2 are used.

4.4.3 Bit Length

The effect of bit length on quantization is explained in section 4.3.2.1. For the simulation, bit length of 8 is used for representing a message which varies between 0 and 25. From equation (10) the quantization interval is calculated to be 0.0977. Figure 17 shows the signals transmitted on the CAN network. The left sub-figure is the amplitude of the simulated signal plotted at time=0. On closer observation it is seen that the smallest difference between two data points is not less than the quantization interval.
The right sub-figure is taken from a CAN log, it represents a pressure sensor output which varies between 0 and 51.2 bar. The bit length assigned to the signal is 8, hence the calculated quantization interval is 0.2. In the figure it can be seen that the signal varies but it doesn’t take a value between the quantization intervals.

### 4.4.4 Baud Rate

The baud rate for the simulation is set to 500 kbits/s because it is the standard baud rate used by many applications and on EcoCAR2 vehicle. Baud rate determines the transmission time of each bit on a network. At the baud rate 500 kbits/s, it is calculated that the time taken to transmit one bit is 2µs. Figure 18 shows the bus electrical activity over time. The two data tips are at the rising edge (t = 0.002273s) and the falling edge (t = 0.002275s) of a data signal. It can be seen that a bit transmission is completed in 2 µs.
Figure 18: CAN bus signal – bit transmission time

The shown signal is known to be a single bit because it is the acknowledgement bit. Only the acknowledgement bit is followed by so many low bits.

4.4.5 Length of Message and Transmission Time

The length of the message in bits is determined mostly by the protocol being used. The simulator assumes that the CAN message is made of 120 bits, a more detailed reasoning for this choice is explained in section 4.3.2.5. The choice of number of bits in a message also determines the transmission time ($T_{int}$) required to send a message on a network
working at a specific baud rate. Since we have chosen 500 kbits/s baud rate, the time taken to transmit a single message is calculated to be 240 μs.

As explained in section 4.3.2.6 the server block is used to simulate the transmission time $T_{int}$. Figure 19 shows the CAN simulator’s bus busy flag compared to the CAN bus activity recorded from an oscilloscope. The bus busy flag goes high when the server is sending out a message and the bus is occupied. The CAN bus activity shows two consecutive messages being transmitted. It is important to note that the CAN bus activity ends before the CAN simulator’s bus busy flag goes down. This is because the CAN

Figure 19 : CAN bus activity compared to CAN simulator
simulator keeps the bus busy till the end of interframe field. After the interframe field a new message can be transmitted on the bus.

Figure 20 shows a similar graph to Figure 19, the figure shows a relatively longer period of simulation. It is observed that over the period of time there was very little drift. Hence, both at bit level and message interval level behavior is accurately represented by the CAN simulator.
4.4.6 Packet Loss Probability

Packet loss is implemented as a probabilistic event in the CAN simulator. A random event generator produces a ‘loss signal’ with Bernoulli distribution; when the loss signal takes the value of 1 the data packet at that instance is discarded by the simulator. The nominal value for the probability of the Bernoulli distribution is determined by experimental data.

Each CAN message sequence is processed and the inter-arrival time is extracted by determining successive difference in the time instances of message arrival. The inter-arrival time is converted to a zero mean signal by subtracting its mean from it. A packet loss occurs if the zero mean inter-arrival time sequence has deviations greater than 70% of the mean of the inter-arrival time. Figure 21 shows a CAN message’s inter-arrival time; Note that the signal has zero mean and the red line indicates the mean value of the inter arrival time. Any inter arrival time higher than the 70% of the mean value of inter arrival time means that the message arrived at a node too late for the current update cycle. Hence the packet is lost or irrelevant. This method is a conservative way of calculating packet loss because in this method a period of inactivity on the CAN bus is considered to be just one packet loss.
Figure 21: Inter-arrival time and packet loss detection

Figure 22: Packet loss fraction
Figure 22 illustrates the packet loss fraction for each one of the CAN message sequences processed. The packet loss fraction is calculated by,

\[
Packet\ Loss\ Fraction = \frac{N_{\text{loss}}}{N_{\text{total}}}
\]  

(15)

Where, \( N \) is the number of data points in the message sequence. \( N_{\text{loss}} \) is the number of data points of the inter-arrival time that deviates more than 70% of the mean inter-arrival time.

Table 7: Packet loss simulation

<table>
<thead>
<tr>
<th>Simulation Time [s]</th>
<th>Messages Generated</th>
<th>Messages Lost</th>
<th>Message Loss Fraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>4985</td>
<td>6</td>
<td>0.0012</td>
</tr>
<tr>
<td>40</td>
<td>9968</td>
<td>12</td>
<td>0.0012</td>
</tr>
<tr>
<td>60</td>
<td>14954</td>
<td>17</td>
<td>0.0011</td>
</tr>
</tbody>
</table>
The cumulative packet loss fraction over all the message sequences that was processed is 0.0019. Hence, in the simulation a nominal loss probability of 0.0015 is used to account for the variation in packet loss fraction of individual sequences.

The CAN simulator was run with this nominal loss probability and the results are tabulated in Table 7. The results indicate that the packet loss fraction is close to what is expected on a real CAN bus.
CHAPTER 5: EFFECTS OF CAN ON CONTROL PERFORMANCE

In an automotive CAN network the CAN bus is shared by multiple devices which share data with one another through the network. The data exchange utility is used to send control signals and receive feedback by spatially separated controller and plant. The CAN simulator described in chapter 4 is specifically constructed to simulate the effects of using the CAN network as a channel for exchanging control-feedback signals. The effects of CAN network parameters on control performance is presented in the following sections.

5.1 Structure of the Simulator

The CAN network simulator is structured in the same way as automotive CAN networks. Separate devices host control logic, actuator controls and sensor managements. Each controller is then connected to one another using the CAN network. Earlier systems housed all the functions of a digital controller in a single device and depended on extensive cabling to reach the different physical parts of the control loop. Figure 23 shows the structure of the simulator with controller interacting with one node and the plant interacting with another. The controller block houses a discrete controller to
regulate the performance of the plant. The plant block simulates the model of a physical system along with actuators and sensors.

![Figure 23: Structure of the CAN simulator based control system](image)

Each node is assigned its messaging parameters based on the signals it is required to transmit. For example a control signal may vary from -10 to 10 and the corresponding plant feedback may vary from 0 to 50. Suppose the designer, for such a system wishes to keep the quantization interval similar then the bit length used to represent the control signal will be less than that of the plant signal. The intergeneration time for transmitting messages is set to the sampling interval of the discrete controller. The intergeneration time smaller than the sampling interval can be used at the cost of increasing bus load.

More controllers and plants can be inserted into the same system with minor changes to the bus block. This feature allows for faster customization of the simulation platform.
The background network traffic plays an important role in the performance of the CAN network. It determines the bus availability and queue delays. The background traffic is implemented by introducing a node that sends messages on the network at a controlled rate. The network traffic is controlled by assigning the intergeneration time for the background traffic node by the number of messages the node generates.

![Figure 24: CAN controls simulator process flow diagram](image)

Figure 24 shows the different processes and the flow of information through the CAN network based control system. Green indicates the controller, red indicates the plant and blue indicates the CAN network processes. Note that the controller and plant nodes have to use the same pathway to transmit their respective messages.
5.2 Architectures

The CAN network can be used in a control system with many architectures based on what part of the control system needs to transfer data on a communication network. In principle any pathway between two components can be replaced with a CAN network with two nodes on either end. In the following section two different architectures are considered for evaluating effects of CAN network in a control system.

5.2.1 CAN for Closing the Control Loop

The architecture explained in this section uses the CAN network as a part of the closed loop of the control system; Figure 25 illustrates the architecture. It is assumed that the controller is trying to drive the plant output in the trajectory provided by the reference block.

Figure 25: CAN network for closed loop control.
The controller is a digital transfer function implemented numerically on a processor based device like MicroAutoBox®. The controller communicates internally with a CAN transceiver node; the transceiver is responsible for gathering the control signal from the controller and transmitting it on the network. The reference for the controller could be an internal reference like a constant value or a pre-programmed trajectory; an external reference like an analog sensor input, or a set point request by another controller. The control signal is transmitted as the payload on a CAN message and the message is received by all the nodes on the network. The node on the plant side is generally a part of another microprocessor based device, which also houses a DAC and ADC. The node filters the messages and relevant messages are let through. The message with the control signal is translated into an appropriate control signal by the node and sent to the actuator. Although the Figure 25 shows only a DAC sending signals to an actuator; in reality there are several different ways to do the same job. Some actuators can be controlled by digital signal levels using Pulse Width Modulation (PWM) or by just activating a gate using a digital signal level which then controls the actuator.

The plant reacts to the control signal and the change in output is recorded by the sensor. The sensor signal is then converted to digital format by the ADC and it is sent to the node for transmission. The node transmits the message on the bus and the controller side node uses the message to determine the feedback signal going to the controller.
In the simulations the transmission actuator model described in 3.5 is used in this architecture.

5.2.2 CAN for Supervisory Control

CAN network can be used as part of the supervisory control in which the control loop is closed locally on the node housing the ADC and DAC. Figure 26 illustrates the supervisory control architecture where the set point or reference trajectory is sent to the controller through the CAN network.

![Figure 26: CAN network for supervisory control](image)

Although this architecture seems trivial, the control performance is significantly affected by the CAN parameters. The transmission actuator and the clutch actuator model
described in 3.5 are used in this architecture for the simulations. The advantage of this architecture is that the delays in control system are reduced and higher sampling frequencies can be used; however, on the down side it requires the node to have higher computational power to run a numerically implemented controller.

5.3 Effects of CAN network

The use of CAN network as a part of control systems is on option which has its own advantages and disadvantages; therefore, it is important to understand the effects of CAN network on control performance. Simulation experiments are conducted on the CAN simulator to bring out the trends in performance of the control system as a function of the CAN network parameters.

The simulator was initialized with nominal values for the network parameters. The experiments varied a single network parameter while holding all other parameters at nominal value and simulating the CAN simulator with control systems.
Table 8: Nominal parameters of the CAN simulator

<table>
<thead>
<tr>
<th>Simulator Parameter</th>
<th>Nominal Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Baud Rate</td>
<td>500 kbits/s</td>
</tr>
<tr>
<td>2 Message size</td>
<td>120 bits</td>
</tr>
<tr>
<td>3 Packet Loss Probability</td>
<td>0.0015</td>
</tr>
<tr>
<td>4 Intergeneration time for controller and plant node</td>
<td>4ms (+0.1ms)</td>
</tr>
<tr>
<td>5 CAN ID (Controller message, Plant message)</td>
<td>(1, 2)</td>
</tr>
<tr>
<td>6 Bit length</td>
<td>16</td>
</tr>
<tr>
<td>7 Background traffic</td>
<td>1 packet every 0.1s</td>
</tr>
</tbody>
</table>

The effects of the variation in different parameters are presented in the next section.

5.3.1 Performance Evaluation Metrics

The choice of an input trajectory and a performance measure is required to quantify and compare the performance of the control system. The input trajectory or the reference signal was constructed with a step signal and a ramp signal, further, the plant was
subjected to an external load. The simulation duration was 40 s and the ODE45 solver on Simulink was used with a variable-step solver with relative tolerance set to 1e-9.

Figure 27: Reference input and ideal output performance for transmission actuator plant

Figure 28: Reference input and ideal output performance for clutch actuator plant
The performance measure for the simulations is chosen to be the root mean squared error of the output of the plant compared to two references. The first reference is the reference signal or input trajectory and the second reference is the ideal performance of the controller without the CAN network in the control loop.

\[
\text{Performance Metric Ref} = \sqrt{(x_p(t) - r(t))^T(x_p(t) - r(t))} \tag{16}
\]

\[
\text{Performance Metric Ideal} = \sqrt{(x_p(t) - \tilde{x}_p(t))^T(x_p(t) - \tilde{x}_p(t))} \tag{17}
\]

Where, \(x_p(t)\) is the output of the plant, \(r(t)\) is the input reference signal and \(\tilde{x}_p\) is the ideal plant output. \(\tilde{x}_p\) is obtained by running an ideal plant and controller without the CAN network’s involvement.

Figure 27 and Figure 28 show the input signal, load signal and ideal performance of the controller for the transmission actuator and clutch actuator plants.

For the sake of simplicity the following terminology is used to describe different simulation experiment configurations.

1. TransAct-Loop: Experiments on transmission actuator using the CAN network to close the control loop. (Graph legend – CAN loop closing)
2. TransAct-SV: Experiments on transmission actuator using CAN network for sending only the reference signal. In other words using the CAN network for supervisory control. (Graph legend – CAN for SV)

3. CP1-SV: Experiments on clutch actuator using the CAN network for supervisory control. (Graph legend – CAN for SV)

Figure 29: RMS error guide

Figure 29 shows the response of the two plants and their respective RMS errors, it gives a good idea what an RMS error magnitude means in terms of performance.

5.3.2 Sampling Interval

In a CAN network framework, the intergeneration time limits the sampling interval. For control design the plant model is required to be discretized using a zero order hold
method. The dependence of the discretized plant dynamics on the sampling interval is very well studied [23]. In the context of the simulation experiments, the sampling interval is relevant only for the transmission actuator plant. The continuous time transfer function for the plant including an output delay is given in (18).

\[
\frac{X(s)}{V(s)} = \frac{e^{-0.0024s} \cdot 0.002046}{5.624 \cdot 10^{-9} s^3 + 9.676 \cdot 10^{-6} s^2 + 378.5 \cdot 10^{-6} s} \tag{18}
\]

The plant is discretized with zero order hold and several different sampling intervals (Ts) and the discrete time z-domain transfer functions and their respective pole locations are shown below,

TS = 0.001s, poles = [1.0000, 0.9607, 0.1863]

\[
\frac{X(z)}{V(z)} = z^{-24} \frac{4.134e - 05 z^2 + 0.0001136 z + 1.766 \cdot 10^{-5}}{z^3 - 2.147 z^2 + 1.326 z - 0.179} \tag{19}
\]

TS = 0.004s, poles = [1.0000, 0.8520, 0.0012]

\[
\frac{X(z)}{V(z)} = z^{-6} \frac{0.001233 z^2 + 0.001898 z + 6.437 \cdot 10^{-5}}{z^3 - 1.853 z^2 + 0.8542 z - 0.001027} \tag{20}
\]

TS = 0.008s, poles = [1.0000, 0.7258, 0.000001453280981]

\[
\frac{X(z)}{V(z)} = z^{-3} \frac{0.005417 z^2 + 0.006378 z + 5.7 \cdot 10^{-5}}{z^3 - 1.726 z^2 + 0.7258 z - 1.055 \cdot 10^{-6}} \tag{21}
\]
From (19), (20) and (21) it is obvious that the poles move with change in discretization interval, hence changing the dynamics of the system. It is implied that to match a particular performance, different controllers have to be designed for each discretized model of the same plant. It is important to note that the CAN network supports intergeneration time $\geq 1\text{ms}$ for keeping the bus load low. Very slow sampling times can be used at the cost of response time and settling time. The choice of sampling interval is not straightforward but certain guidelines are available to decide on an acceptable sampling interval. It is also often limited by hardware limitations such as clock frequency of the microprocessor used for sampling or the CAN network in this particular case. For the current simulations the sampling time of $T_s=0.004\text{s}$ is used.

### 5.3.3 Quantization

Quantization is determined by the choice of bit length used to represent a signal in a CAN message. In the simulation experiment the number of bits used to represent the signals is varied from 3 bits to 16 bit.

The transmission actuator model is simulated in both the architectures described in section 5.2. The figures presented in this section shows the RMS error between the plant output and the two different references explained in the 5.3.1. The blue lines are the control performance with reference to the reference signal and the blue lines are the control performance with respect to the nominal or ideal performance of the controller without the CAN network.
Figure 30 shows the simulation results for TransAct-Loop and TransAct-SV with the same message length for both control signals and feedback signals.

In the TransAct-Loop simulations, it is observed that the control performance was unchanged from about 10 bits to 16 bits of message length. Indicating that arbitrarily increasing the number of bits to represent a signal has diminishing returns in terms of control performance. On the other hand, reducing the signal bit length below 10 bits shows increasing degradation of control performance.

Figure 30: Quantization effects on control performance of transmission actuator (A)
In the TransAct-SV simulations, the control performance did not get affected significantly by the bit length variation. However, it is important to note that the performance degradation compared to the reference signal is lower than performance degradation compared to the nominal performance. The underlying reason for such behavior is that the tracking performance is very less affected by quantization of the reference signal. However, as quantization interval increases the ability of the input system to provide accurate reference signal degrades.

Figure 31 shows the simulation results for TransAct-Loop with independent variation in bit lengths of control signal and feedback signal. Note that the figure shows performance index with respect to nominal performance. The effect of controller signal quantization is more significant on the performance degradation.
The same experiment was not conducted for TransAct-SV case because there is no feedback signal being transmitted on the network.

Figure 32 shows the effect of quantization on the control performance of the clutch actuator CP1 working in the supervisory control architecture (CP1-SV). The observed trends are similar to the ones seen in the case of transmission actuator. The RMS error magnitude is higher because of the difference in scales of the output of the two systems as shown in Figure 27 and Figure 28.
Figure 32: Quantization effects on control performance of clutch actuator

5.3.4 Packet Loss Probability

Packet loss is a phenomenon in which the CAN message transmitted on the network does not reach the target node successfully. The effect of packet loss on control performance is very significant. For example, if the controller does not get an updated feedback signal it assumes that the output of the plant has not changed and issues a stronger control signal; which can cause performance degradation and eventually lead to instability.

The simulation experiment varies the packet loss probability from 0.0015 to 1. Figure 33 shows the simulation results for TransAct-Loop and TransAct-SV as a function of packet loss probability. The packet loss probability in normal operating conditions is less than
0.002, however the packet loss fraction increases due to other factors like bus load. Hence, this set of simulations consider the extreme cases for packet loss probability.

It is observed that the control performance deteriorates very rapidly beyond the packet loss fraction of 0.7; the plant output is unstable above 0.8. It is important to note the performance is relatively unchanged until the probability of 0.1 which is in the region of normal operation. TransAct-Loop is more vulnerable to the packet loss effects than the TransAct-SV.

Figure 33: Packet loss effects on control performance of transmission actuator.
Figure 34 shows the simulation results for CP1-SV. The trends are very similar to TransAct experiments.

![Effects of Packet Loss on control performance - clutch actuator](image)

Figure 34: Packet loss effect on control performance of clutch actuator

Figure 35 shows the simulation results for TransAct-Loop and TransAct-SV when both quantization interval and packet loss probability are varied. The packet loss has more effect on the performance than the quantization interval.
5.3.5 Bus Speed

The bus speed is defined by the number of bits that can be transmitted on a network in a second, usually specified in kbits/s in CAN networks. The bus speed determines the time delay between two consecutive transmissions and hence becomes a limiting factor for the number of messages that can be sent in a period of time. As a consequence it limits the maximum number of messages that can be on the network, without degrading the network performance to $\min(\frac{T_{\text{int}}}{T_{\text{trans}}})$. For example at $\min(T_{\text{int}}) = 0.004s$ and baud rate of 500 kbits/s, only 16 messages can be sent on the network without causing more than 0.004s delay.

Figure 36 shows the simulation results for TransAct-Loop and TransAct-SV as a function of bus speed, which was varied between 50kbits/s to 1000kbit/s. As the bus speed decreases the control performance also degrades. The magnitude of degradation is very
small because the effect of lowered bus speed is nullified by the lack of high background traffic. Figure 37 shows the simulation results for CP1-SV and the trend looks similar to the TransAct simulations. Note that in these simulations the bus utilization level was not more than 0.7

Figure 36: Baud rate effect on control performance of transmission actuator
5.3.6 Priority of Messages

The CAN ID defines the priority of a message in a CAN network. When multiple messages are waiting in a queue to be transmitted, the AMP ensures that the messages are transmitted on the network in the order of their CAN IDs. A message having a lower CAN ID gets the higher priority when transmitting on a network. The experiment conducted to simulate the effect of priority of the control signal on control performance is conducted on TransAct-Loop and it requires modified CAN parameters. The change from nominal parameters are listed below.

1. Background traffic is increased to 3 messages every 1ms.
2. For low control priority: Background CAN ID – 1; Control CAN ID – 3
3. For high control priority: Background CAN ID – 3; Control CAN ID – 1

Figure 38: Effect of the priority of control signal on control performance

Figure 38 shows the difference in performance with the same network traffic conditions but different control message priority. In both cases the bus utilization was at 0.99 and it should be noted that high bus utilization levels is primary reason for control performance degradation. The results indicate that assigning higher priority to a control signal could improve the performance significantly.
5.3.7 Network Utilization

The effects of sampling period or the intergeneration time on discretized plant models was discussed in section 5.3.2. Further, in each of the previous discussions on effects of network parameters on control performance the bus utilization level was mentioned when applicable. Section 5.3.6 not only shows the effect of priority of signals but also the effects of high bus utilization on control performance. The current discussion illustrates the effect of parameters like intergeneration time and number of messages on the network on the bus utilization.

Figure 39 shows curves of bus utilization as a function of the intergeneration time. Each curve is associated with a fixed number of messages on the CAN network. For example the black curve represents a total of six periodic messages on the bus with an intergeneration time specified by the x-axis and bus utilization level on the y-axis.

It is observed that as the intergeneration time decreases the bus utilization also increase with same number of messages on the network. Further, as the number of messages on the network increases the bus utilization increases at all intergeneration times. Hence it is concluded that number of messages on the network and intergeneration time can increase bus utilization, thereby increasing the chances of degradation in control performance.
Figure 39: Factors affecting bus utilization on CAN network
6.1 Analysis techniques

The delays caused in the CAN network are combinations of the intergeneration time or the sampling interval and fractional transport delay. The delays which are integer multiples of the sampling intervals can be represented in as a time step delay in discrete systems. However, for fractional delays other methods are required. Several different methods are presented in [21] for designing digital filter which can be used to represent the fractional delay effects. An IIR all-pass filter was suggested by Thiran in 1971 for the fractional delay representation. While other methods require iterative design process to calculate the filter coefficients, Thiran’s filter coefficients can be calculated by the closed form equation (3) and (4). The MATLAB platform has implemented the Thiran filter as a function and can be used to approximate the response of a fractional delay.

The availability of such a function allows for approximating the effects of delays on the location of poles and zeroes. It is important to keep in mind that such methods for representing delays are accurate only to a certain degree and it is important to verify the simplified approximate models with the simulations.
Consider the following second order system with a delay: \( P(s) = \frac{2000 e^{-0.01s}}{(s+40)(s+50)}. \)

Discretizing the system with zero order hold and at sampling interval of 0.01s we get

\[ P(z) = z^{-1} \frac{0.074z + 0.0552}{(z^2 - 1.277z + 0.4066)}. \]

An arbitrary controller with a fractional input delay of 0.015s is chosen to be, \( C(z) = \frac{1}{z+0.5}. \) Figure 40 shows the described system.

![Figure 40: Control system with fractional delay](image)

Now, the fractional delay can be expressed as a 1.5 time steps delay. However, in the standard method of represented digital controllers there is no scope for the extra 0.5 time steps. Therefore, in most case it is convenient to round the delay to the previous or next integer value. Figure 41 shows the effect of such rounding or flooring of fractional delays to an integer value.

To account for the effects of fractional delay the controller can be appended with a Thiran filter obtained by the MATLAB command ‘thiran(d,Ts)’, where d is the fractional delay and Ts is the sampling interval for the system. From Figure 41 it is clear that the
performance with the Thiran filter matches the continuous time system more closely than other approximations.

In order to apply the techniques presented in the previous section for the CAN network delays it is important to understand the mechanism by which delays are generated.

Figure 42 shows the control and feedback signals along with the CAN signals which represent them. A step input is applied as the reference signal to the controller at T=0.005s. Note that the circular markers are the intergeneration interval for the CAN messages (0.004s). Since the step input is given in the middle of two sampling intervals

Figure 41: Effect of fractional delay approximation on control system
the input is not be detected until the next time step. At $T=0.008$, the controller picks up the increased reference signal and computes an appropriate control signal that gets sent to the CAN node for transmission, this signal is represented in blue in Figure 41. The control signal is transmitted on the CAN network and after 480 µs the control signal is provided to the plant; this is represented in green in Figure 41. The plant (red) immediately starts reacting to the control signal and increases in amplitude. Only at the next time step are the sensors read and transmitted back to the controller after the transmission delays; the signal is represented in cyan. It is also important to note that the delay between the receiving node getting the control signal and producing a physical signal has not been considered. It is generally very small compared to the transmission delays. As a consequence of this assumption the node on the plant side is assumed to run at a faster clock frequency than the intergeneration time. Hence decreasing the delay due to the zero order hold from 0.004 to an insignificantly small value. From these observations a model for CAN based control loop delays is developed and is shown in Figure 43.
Figure 42: Step response of the TransAct-Loop without the plant delay

Figure 43: CAN based control system’s loop delays
In Figure 43, a and b are integers which represent the uncertainty in the delay due to bus load conditions. Note that the $T_{\text{trans}}$ is a fractional delay compared to $T_{\text{int}}$. The following discussion is based on the assumption that the $(a T_{\text{trans}})<T_{\text{int}}$.

An important observation is that the location of fractional delays in a control loop plays an important role in determining the accuracy of the delay models. Consider the fractional delay $(a T_{\text{trans}})$ between the controller and the plant; this delay manifests directly as the delay in output of the plant and it is very important to include it in the model. The other fractional delay $(b T_{\text{trans}})$ is sandwiched between two other delays. In an ideal system, all these delays would get added into one lumped output delay of the plant. However, in the context of CAN network this fractional delay is insignificant because regardless of the value of ‘b’, as long as the $(b T_{\text{trans}})<T_{\text{int}}$, the intergeneration time happens only at a set time. In other words the entire system’s feedback delay is governed by the intergeneration time and the control input delay is governed by the transmission time. Hence, it is advised that while dealing with fractional delays, analyzing how it manifests into the system should be done carefully.
The delay model obtained by implementing the model described in Figure 43 by using thiran function is shown below. The numerator and denominator coefficient for the discrete time (z-domain) closed loop transfer function with $T_s=0.004$ is as follows,

Numerator = [0.03264 0.02915 -0.08043 -0.03939 0.05612 0.001926 -2.236e-06 0]

Denominator = [1 -1.972 0.2707 1.312 -0.5488 -0.06043 -0.001333 1.696e-6 0.03264 0.02915 -0.08043 -0.03939 0.05612 0.001926 -2.236e-06]

The step response of this system along with the step response of the control system on the CAN network is presented in Figure 44. It is observed that the responses match closely.
6.2 Design Guidelines

The CAN network simulator is used to test and understand the effects of network parameters on the control performance. A consolidated overview of all the results and their use in the system design are presented in this section.

Figure 45: Relationship diagram – network parameters vs. control performance

Figure 45 shows the relationships of various CAN network parameters with the control performance. The direction of the arrow indicates the cause and effect relationship; the
color of the arrows indicate whether the two parameters are directly related or inversely related. Further, double sided arrows indicate a tradeoff or interdependent parameters.

Some of the parameters have complex relationships with the control performance which can be realized by tracing the arrows carefully. Most of the parameters affect the control performance in more than one way, in some cases this leads to a tradeoff and choosing an optimal value for the parameter becomes difficult. The following section discusses some of the design guidelines that can be derived from the Figure 45.

Bit length of messages determines the quantization interval of the signals. Lower bit length results in larger quantization interval. Quantization effects on control performance is discussed in section 5.3.3. As quantization interval increases the control performance gets worse. Bit length of the messages also result in increase in the number of messages that are needed to transmit all the data on a CAN network. Assume that there are 10 signals that needs to be transmitted by a node and the designer chooses to use a uniform bit length of 8 bits/signal, then there will be 2 messages on the network to transmit all the 12 signals. On the contrary, if 6 signals use 8bits/signal and 4 signals use 4bits/signal, then only one message is required, which reduces the bus utilization by half.

The number of messages on the network increases the bus load as discussed in 5.3.7. Increased bus load causes increased packet loss as discussed in 5.3.6. Increased packet loss degrades performance. Thus, bit length used for a signal can better and degrade control performance.
Similarly higher sampling rate used for control signals can improve the control performance but increased sampling rate results in shorter intergeneration time and hence higher bus loads which degrades controls performance. Note that arbitrarily increasing the sampling interval for a discrete time system can make it unstable.

Priority of signals play an important role in the performance of the control system under high bus load conditions. It inversely effects transmission delay of the high priority message, and thereby reducing the chance of packet loss. Decrease in transmission delay and packet loss, preserve the control performance. Therefore, providing the control signals higher priority is a safer bet. As discussed in 6.1, the fractional delay location in a control loop plays an important role in determining the entire loop delay. From the results of the discussion it is concluded that on a CAN network based control system the control signal should be assigned higher priority than the feedback signal. The feedback signal and control signals should always have higher priority than all the other background signals when possible. Further, a message with very small intergeneration time may be assigned lower priority if it is acceptable. When such messages have higher priority they tend to hog the bus and essentially block the slower signals with lower priority.

On the same note, it is also important to remember that the sampling rate or intergeneration rate is limited by the hardware limitations of the CAN bus. CAN bus operating at different baud rates and depending on the length of the network will have different limits on smallest intergeneration time for safe operation.
Baud rate inversely affects the transmission delay and packet size affects the transmission delay directly. The baud rate or bit rate along with packet size determine the maximum number of messages that can be on the network.

Figure 46: Effect of quantization on 1st order plants of different time constants

Figure 46 shows the effect of quantization on various 1st order systems. In order to be able to compare the performances 5 first order plants with different time constants were taken. The 5 corresponding controllers were designed to have the same closed loop performance. Then the effects of quantization were introduced. It can be clearly seen that as time goes on, the plants with higher time constants do worse. But as the quantization interval is decreased the performance tends to become the same across all the plants.
The effects of CAN networks on supervisory control is also discussed in 5.3. It should be noted that choice of an intergeneration time for sending the reference signal to the controller should be made in the same way a sampling frequency is chosen to reproduce a signal. The intergeneration time should be such that it captures the highest relevant frequency content of the reference signal. If the intergeneration time for a supervisory signal is chosen poorly, then having a very good controller becomes useless.

6.3 Control Solutions

A CAN network based control system’s performance can be optimized by using the design metrics discussed in section 6.2. However, some of the factors that affect the CAN network are dynamic. They appear with certain network conditions like high bus load or high packet loss rate. A great deal of research has been done into developing control strategies for networked control systems. One such method is gain scheduler middleware (GSM) reported in [17]. The GSM uses the external gain scheduler scheme to modulate the amplitude of the control signal. The gains are set based on the real time network traffic situations. The advantage and disadvantage of the GSM is its reliance on the network delay parameter. While it makes the control less conservative by providing larger design space, it requires knowledge of network delays in real time. The method described in [17] is implemented on a TCP-IP network and the network delay is approximated by round trip delay of a data packet. It is important to note that the GSM is not a part of the traditional controller but it is implemented at the node level.
\[ u^* = f(n \, T_{\text{trans}}) \, u \] (22)

Where, \( u \) is the control signal generated by the controller and \( u^* \) is the new control signal scaled by the gain scheduler. The gain is a function of the total network delay.

The TCP-IP network is qualitatively very different from the CAN network in priority arbitration, collision management and flow of data packets. The most important difference in the two is that in automotive CAN networks the messages are sent periodically which makes the roundtrip delay less useful for effectively representing the network traffic. Further, since the nodes of the CAN network are not synchronized to a global time it becomes harder to evaluate the network delay based on time stamps on the CAN messages. Hence a practical method to determine the network is developed for CAN networks.

The network delay approximation is based on the knowledge about the nominal intergeneration time of a CAN message. A particular CAN message is generated every \( T_{\text{int}} \) seconds, therefore its arrival at the controller node should be spaced \( T_{\text{int}} \) seconds in ideal conditions. If there is network delay the inter-arrival time deviates from \( T_{\text{int}} \).

Let the inter-arrival time, \( T_{\text{Arr}} \) be equal to \( T_{\text{int}} \) at time some time instance \( t_k \). In ideal conditions the next CAN message with the same identifier will be expected to arrive at \( t_k + T_{\text{int}} \). Also the \( T_{\text{Arr}} = t_{k+1} - t_k \). When \( T_{\text{Arr}} > T_{\text{int}} \) the network delay is above the nominal level. Further, for the next time instance, \( t_{k+2} \) if the network traffic comes back to normal then, \( T_{\text{Arr}} < T_{\text{int}} \). This does not, however, mean that the network traffic has reduced below
the nominal values because the $T_{\text{int}}$ is a constant quantity. This information is very helpful as it gives a measure for severity of the network load.

The determination of $T_{\text{Arr}}$ is done by generating an interrupt every time a message with a certain identifier is received. The interrupt prompts the GSM to record the current time $t_k$, the next instance when the interrupt is generated the GSM records the time $t_{k+1}$ and so on. The $T_{\text{Arr}}$ is calculated by determining the difference between the two time recordings. A network delay parameter can be formulated by using $D = T_{\text{Arr}} - T_{\text{int}}$. Now the GSM gain can be made a function of $D$. The next task is to find such a function and test its performance.

To gain an understanding about the nominal system’s behavior on changes to the loop delay due to the network traffic, a pole-zero map is constructed. Figure 47 shows the pole-zero map of the TransAct-Loop with different loop delays. The poles of the system move significantly in the range of delays that is used for the simulation. Since there are 14 poles it is hard to describe their motion in simple terms. A similar simulation experiment is conducted to see the effect of loop gains on the poles of the nominal system. The objective of the GSM is to keep the poles at its original location to restrict change in performance. It is observed that as the loop delay increases the loop gain should decrease in order to keep the poles at its original location.
Figure 47: Effect of delays and loop gains on pole location of TransAct-Loop
Hence the GSM is required to modulate the gain inversely with the network delay. A prototype GSM is implemented on the CAN network based control system TransAct-Loop as shown in Figure 48.

![Figure 48: Gain scheduler middleware implementation on SimEvents](image)

The GSM was tuned by trial and error in this simulation. A more formal tuning by iterative optimization can help improve the performance of the GSM. The delay estimation can be made better by using watchdog timers or reset counters while implementing on a physical controller. The results of the simulation are presented in Figure 49 and Figure 50.
Figure 49: Gain scheduler middleware performance on TransAct-Loop

Figure 50: Delay estimation and scheduled gains of GSM

The GSM improved performance significantly during the simulation tests with similar high bus load conditions mentioned in 5.3.6. The delay estimation performance matches
the expected trends. The gains of the GSM varies between 0-1 based on the state of network delay. The reduction in the loop gain results in degraded performance in terms of rise time and settling time, although it certainly helps the system from going into violent oscillations. A well-tuned GSM can improve the performance in a much more balanced way.
7.1 Conclusions

The CAN network simulator on SimEvents platform was formulated based on the known behavior of the CAN network. The network parameters like bus speed, quantization, packet loss, intergeneration time, and network induced delays and transmission delays were represented satisfactorily in the simulator. Nominal values for the simulator parameters were determined based on experimental data. The behavior and the choice of simulator parameters were validated against experimental data from the CAN network logs and oscilloscope logs.

The simulator was then used to demonstrate the effects of CAN network on control systems. Effects of quantization, packet loss, bus speed, etc. were studied with models of two systems. The effects were studied in two architectures; CAN for supervisory control and CAN for control loop. The results were presented and some design guidelines were developed based on the results.

The delays in the CAN network were discussed and a way to represent the effect of such delays in terms of pole-zero map was also discussed. A control technique, Gain Scheduler Middleware (GSM), to correct the effects of delays in the control loop was...
discussed and its effectiveness demonstrated. The work presented in this thesis developed an effective behavioral simulator of the CAN network for simulating networked control systems.

### 7.2 Future Work

The CAN simulator developed in this work is for the specific purpose of simulating control systems on a CAN network with high simulation speeds. As a consequence, the fidelity of the simulator is limited. Only the data frames are used in the current simulator, in the future other frames like error frames, overload frames and remote frames can be included in the simulations. The inclusion of these frames can model the CAN network even more accurately but it might increase the simulation time.

The current simulator does not allow retransmissions on the network when an error occurs. The retransmission behavior can be implemented by using the current error detection mechanisms or by developing a dedicated error detection and handling mechanism.

The simulator was based on the MATLAB 2014a platform which does not allow accelerated simulations for SimEvents based models. In the future, if the MATLAB develops ways to accelerate the simulations then the model’s accuracy can be improved without compromising on the simulation speed.
The gain scheduler middleware concept can be exploited further to the control’s advantage by optimizing the gain scheduling function. MATLAB functions like fmincon or fminsearch can be used to do the same. Alternatively, the gain schedule map can be developed for different loop delay conditions.

The online network delay estimation can be improved by utilizing the built in functions like watchdog timers on an ECU. Such an estimator would estimate the current inter arrival time as compared to the previous inter arrival time implemented in this simulator. It should be kept in mind that such a delay estimator may need a saturation limit when it affects the gain scheduling. Without a saturation, the increasing delay may eventually drive the gains to negative values resulting in positive feedback to the system. Further a PI control can be considered with the network delay as the error parameter.
BIBLIOGRAPHY


APPENDIX: LIST OF SYMBOLS AND ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVTC</td>
<td>Advanced Vehicle Technology Competition</td>
</tr>
<tr>
<td>ADC</td>
<td>Analog to Digital Converter</td>
</tr>
<tr>
<td>CAN</td>
<td>Controller Area Network</td>
</tr>
<tr>
<td>CAPL</td>
<td>Communication Access Programming Language</td>
</tr>
<tr>
<td>D</td>
<td>Total Delay as a multiple of sampling interval</td>
</tr>
<tr>
<td>DAC</td>
<td>Digital to Analog Converter</td>
</tr>
<tr>
<td>DOE</td>
<td>Department of Energy</td>
</tr>
<tr>
<td>ECU</td>
<td>Electronic Control Unit</td>
</tr>
<tr>
<td>EOF</td>
<td>End Of Frame</td>
</tr>
<tr>
<td>EPA</td>
<td>Environmental Protection Agency</td>
</tr>
<tr>
<td>EV</td>
<td>Electric Vehicle</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>EVT</td>
<td>Electronically Variable Transmission</td>
</tr>
<tr>
<td>FEM</td>
<td>Front Electric Machine</td>
</tr>
<tr>
<td>GM</td>
<td>General Motors</td>
</tr>
<tr>
<td>GSM</td>
<td>Gain Scheduler Middleware</td>
</tr>
<tr>
<td>HEV</td>
<td>Hybrid Electric Vehicle</td>
</tr>
<tr>
<td>HIL</td>
<td>Hardware-in-the-Loop</td>
</tr>
<tr>
<td>MEF</td>
<td>Maximum-Error-First</td>
</tr>
<tr>
<td>NCS</td>
<td>Networked Control System</td>
</tr>
<tr>
<td>SOF</td>
<td>Start Of Frame</td>
</tr>
<tr>
<td>T_{Arr}</td>
<td>Inter-arrival Time</td>
</tr>
<tr>
<td>T_{int}</td>
<td>Intergeneration Time</td>
</tr>
<tr>
<td>T_{trans}</td>
<td>Transmission Time</td>
</tr>
<tr>
<td>TOD</td>
<td>Try-Once-Discard</td>
</tr>
<tr>
<td>PHEV</td>
<td>Plug-in Hybrid Electric Vehicle</td>
</tr>
<tr>
<td>u</td>
<td>Control Signal</td>
</tr>
</tbody>
</table>
$U_{bus}$  Bus Utilization

VHDL  Verilog Hardware Description Language