Ethernet Communication Design with Service Oriented Architecture (SOA)

A Thesis submitted in partial fulfillment of the requirement for the award of degree of

Master of Engineering
In
Electronics and Communication (ECE)

Submitted By

AASTHA

Roll No. 801761001

Under Supervision of

Dr. HEM DUTT JOSHI

Associate Professor

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT
THAPAR INSTITUTE OF ENGINEERING & TECHNOLOGY
(A DEEMED TO BE UNIVERSITY), PATIALA, PUNJAB
TATA MOTORS LTD. PUNE, MAHARASHTRA

JULY, 2019

DECLARATION

I, **Aastha** hereby declare that the work presented in this thesis entitled **"Ethernet Communication Design with Service Oriented Architecture (SOA)"**, in partial fulfillment of the requirement for the award of degree of **Master of Engineering (ECE)** submitted at **Electronics and Communication Engineering Department**, Thapar Institute of Engineering & Technology (Deemed to be University), Patiala is an authentic record of work carried out under supervision of **Dr. Hem Dutt Joshi** (**Associate Professor, ECED**), **Mr. Ramesh Rathinam (Deputy General Manager, ERC, TML) and Mr. Kevin Rodrigues (Senior Manager, ERC, TML)** from July 2018 to June 2019 in TATA MOTORS Ltd. Pune, Maharashtra. The matter presented in this thesis has not been submitted either in part or full to any other university or institute for the award of any other degree.

The data contained in the thesis is a result of my own evaluation and experiments during the internship period in Tata Motors Ltd.

Date: 04/07/19

Aastha

Aastha

801761001

Date: 04/07/19

Dr. Hem Dutt Joshi

Associate Professor

Department of Electronics and Communication Engineering

Thapar Institute of Engineering & Technology

(A deemed to be University), Patiala, Punjab

ACKNOWLEDGEMENT

This work shall remain unaccomplished unless the painstaking efforts of people, who have guided and supported me in its completion, are acknowledged. By the grace of God, I got this opportunity to thank all those who have helped me in pursuing this task.

I owe a sincere thanks to my guide in Tata Motors, **Mr. Ramesh Rathinam, Deputy General Manager** and **Mr. Kevin Rodrigues, Senior Manager, ERC, Tata Motors Ltd., Pune,** for giving me an opportunity to work on this project and enhance my skills. Their valuable guidance and knowledge lead me through all the challenges that appeared as I progressed during this tenure.

I would like to express my profound gratitude to **Dr. Hem Dutt Joshi, Associate Professor, ECED, Thapar Institute of Engineering and Technology, Patiala,** for his unwavering support and motivation throughout this project.

I would like to extend my sincere gratitude to **Dr. Alpana Agarwal**, **Head of Electronics and Communication Department (ECED), Thapar Institute of Engineering and Technology, Patiala,** for her continual encouragement throughout my work. I am also grateful to our PG coordinator, **Dr. Amit Mishra** for providing me all the prerequisites for getting this project done.

Last but not the least, I am grateful to my parents, friends and family members whose moral support and encouragement made this project a reality.

Aastha
801761001
ME (ECE)

ABSTRACT

The thesis describes how Ethernet Communication design can be created with Service Oriented Architecture using SOME/IP protocol which is defined by AUTOSAR. To define SOA, an ADAS use case has been taken for which various services as per the protocol has been defined, these services are then mapped to ECU's which would be providing and consuming these services. Also the sockets as per the TCP/IP protocol has been defined over which the data contained in the services will be sent on the network. All the above mentioned parameters have been implemented using an E/E architecture tool which was only used to define CAN database, but in this I have proposed how an AUTOSAR compliant architecture on Ethernet can be created. Also in this work, I have shown how diagnostics can be done for different Ethernet using DOIP protocol which will be used in the coming next generation architectures of the vehicles.

An ARXML file which is AUTOSAR 4.3.0 has been exported in the form of system extract, ECU extract, software component extract which can be fed to AUTOSAR compliant toolchains to build the software for the purposed use case.

TABLE OF CONTENTS

List of Abbreviations

ECU – Electronic control unit

OEM – Original Equipment Manufacturer

E/E – Electrical/Electronics

CAN – Controller Area Network

LIN – Local Interconnect Network

LVDS – Low voltage differential signalling

MOST – Media Oriented System Support

ASIC – Application Specific Integrated Circuit

DAS- Driver Assistance System

AE – Automotive Ethernet

SOA – Service Oriented Architecture

MAC – Media Access Control

TCP/IP – Transmission Control Protocol over Internet Protocol

SOME/IP – Scalable Service over Middleware / Internet Protocol

SD – Service Discovery

AUTOSAR – Automotive Open System Architecture

ARXML – AUTOSAR Extensible Markup Language

OABR – Open Alliance BroadR-Reach

BSW – Basic Software

RTE – Run Time Environment

MCAL – Microcontroller Abstraction Layer

I/O – Input/output

OSI – Open Systems Interconnections

API – Application Program Interface

PDU – Protocol Data Unit

ARP – Address Resolution Protocol

NDP – Neighbour Discovery Protocol

DHCP – Dynamic Host Configuration Protocol

VFB – Virtual Function Bus

DOIP – Diagnostics over Internet Protocol

DCM – Diagnostic Communication Manager

SWC – Software Component

LAN – Local Area Network

WLAN – Wireless Network

EMC – Electromagnetic Compatibility

PCS – Physical Coding Sublayer

PMD – Physical Medium Dependent

PAM – Pulse Amplitude Modulation

CSMA/CD – Carrier Sense Multiple Access/Collision Detection

VLAN – Virtual Local Area Network

POC – Proof of Concept

ADAS – Advance Driver Assistance Systems

LIDAR – Light Detection and Ranging

RADAR – Radio Detection and Ranging

UDS – Unified Diagnostic Services

RAM – Random Access Memory

SOAD – Socket Adaptor

DUT – Device under Test

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

In Automotive Industry, the main components that handles the electronics/electrical part of the vehicle are the Electronic Control Units (ECU's). An ECU consists of microcontroller, sensors, actuators, ASIC's etc. to perform various functions in the vehicle. The E/E architecture consists of the placements of these ECU's which communicates on different bus networks and exchange data with each other. The E/E Architecture does not consider implementation of the various E/E components, it just takes care of the fact that a sensor or some other component is required for a particular function in the vehicle. The E/E architecture is designed by an OEM and is continuously evolving as more and more functions are required in the vehicle which leads to more complexity and a large number of ECU's to perform these functions efficiently.

1.1 AUTOMOTIVE E/E ARCHITECTURE

The E/E Architecture consist of different domains such as power train, Chassis, Body Control, Infotainment, Instrument Cluster etc. These domains are connected to each other and communicate via different shared or point to point communication networks such as CAN, LIN, LVDS, MOST and so on depending upon the requirement of the data rate for the particular domain [1]. The present bus topology for the vehicle E/E architecture is shared linear bus topology where all the nodes transmit/broadcast messages on the same bus. These different in vehicle networks have different data rates and different physical mediums such as twisted wire, single wire, optical fibre etc. which are deployed in the vehicle and are chosen based on the functionalities required by the OEM.

1.2 AUTOMOTIVE NETWORKS

The traditional in vehicle network technologies (CAN, LIN, FlexRay, MOST) are all bus systems such that the available bandwidth is shared/utilized among all the nodes connected [2]. The ECU's in present architecture deals with all these technologies in one domain or another to communicate with each other. With Automotive Ethernet the basic concepts of the architecture are completely different as each connection is point to point rather than a shared channel and have higher data rates and payload for next generation architecture consisting of autonomous and connected car features.

1.2.1 CAN (Controller Area Network)

CAN is an automotive-specific bus standard developed by Robert Bosch GmbH released in 1986. The layers 1 and 2 of the OSI network model are defined in the standard. It allows for a maximum bus speed of 1 Mb/s for a maximum length of 40 meters and the payload is of maximum 64 bits in a CAN frame. The CAN frames are sent between two or more ECU's/Nodes and depending upon the arbitration, the messages can be cyclic or event based. To send a CAN frame on the bus, a node should have a CAN

controller which does the work of Data Link Layer and CAN transceiver which is at the physical layer as shown in Figure 1.1

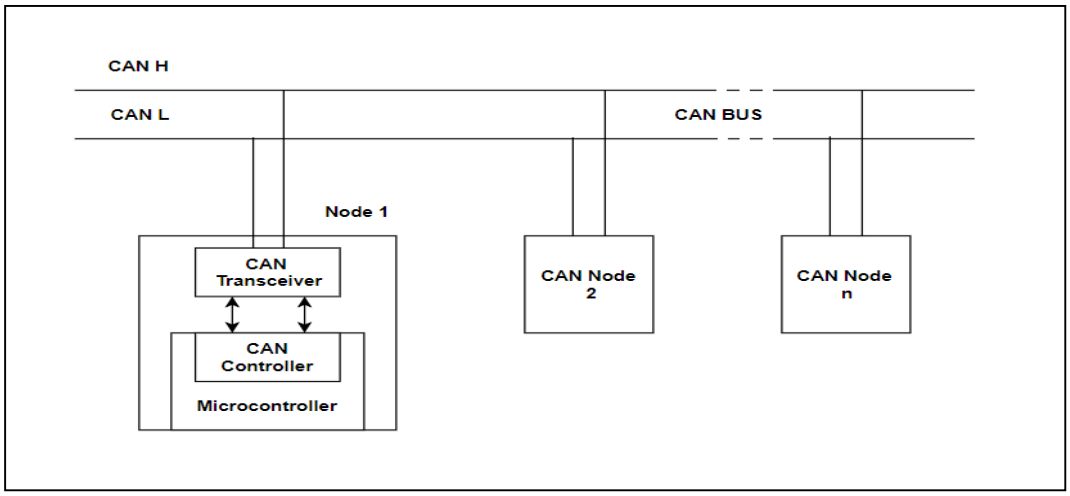

Figure 1 CAN Bus

The CAN frame consists of 11 bit identifier which acts as the logical address in the frame and the different nodes rejects or accept the messages if they are mapped to this ID. The payload of the frame contains the data to be transmitted as shown in Figure 1.2

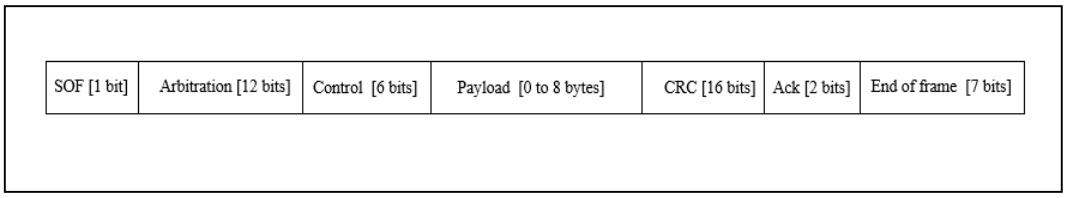

Figure 2 CAN Frame

1.2.2 LIN (Local Interconnect Network)

To meet the demands of automotive industry where low speed, low cost and simple controllers for communication were required, LIN came into use which is also based on bus topology like CAN but transmits messages on a single physical wire. The speed of the bus is up to 20 kb/s and is mostly used in body domain of cars. Also it works in a master slave mode and can have up to 16 nodes [3].

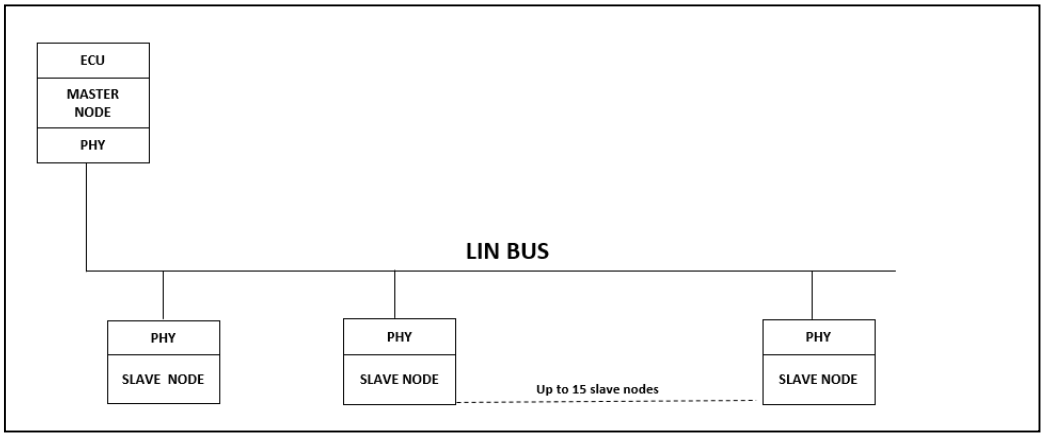

Figure 3 LIN Bus

LIN network has a single master node as compared to CAN which is multi-master bus. The slave nodes can be microcontrollers or ASIC's as shown in Figure 1.3 and 1.4

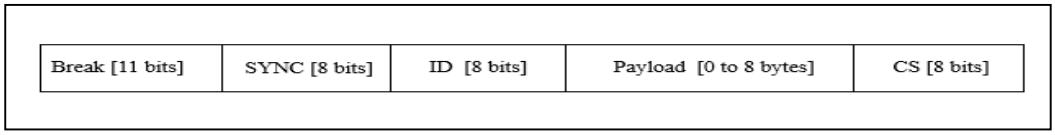

Figure 4 LIN Frame

1.2.3 FlexRay

It was designed for safety-critical applications and supports different topologies such as linear bus, passive star, active star and point to point. The frame ID together with the cycle uniquely identifies the Flex Ray frame similar to arbitration id in CAN. The maximum theoretical bandwidth is 20 Mb/s which makes it unsuitable for infotainment domain. It has larger payload length i.e. up to 256 bytes as shown in Figure 1.5

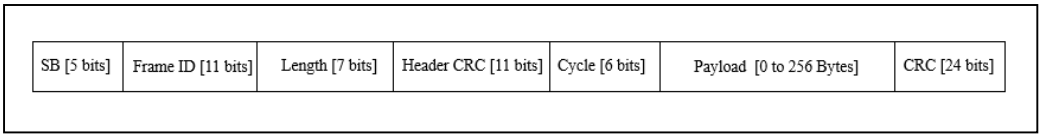

Figure 5 FlexRay Frame

1.2.4 MOST (Media Oriented System Support)

It was designed to address the needs of the Infotainment domain which were not met by the other automotive networks. It has built in channels for streaming audio video data which are required by the audio video entertainment applications such as rear view camera, surround view system etc. [4]. It has bus speed of up to 150 Mb/s which is much greater than other networks. It works in ring topology as compared to star topology used in switched Ethernet network.

1.2.5 Automotive Ethernet

For the next-generation in-vehicle networking infrastructure beyond CAN and FlexRay, the automotive industry has figured out Ethernet as a very promising candidate [5]. Ethernet has the bandwidth capability that is required for new driver assistance and infotainment systems. As existing vehicle control networks like the LIN, CAN and Flex Ray standards are not designed to meet these increasing demands in terms of bandwidth that we expect for various kinds of Driver Assistance Systems (DAS) [6], AE provides scalability and flexibility for next-generation in-vehicle networking Architectures. Over the next decade, CAN and FlexRay will remain for body domain and safety-critical communications.

Automotive Ethernet stems from proven IT technology. Unlike non-automotive Ethernet, the automotive bus uses unshielded, single twisted-pair cabling designed for lower weight and low cost. The automotive standard has its origins in Ethernet, but incorporates significant changes at the physical layer to meet automotive requirements. The Ethernet frame has a payload of up to 1500 bytes and data rate of 100 Mbps which is way greater than what is offered by other networks such as CAN, LIN etc. The frame of automotive Ethernet is similar to standard Ethernet only as shown in Figure 1.6 Automotive Ethernet is just different at the physical layer to meet the automotive requirement and is explained in the coming sections.

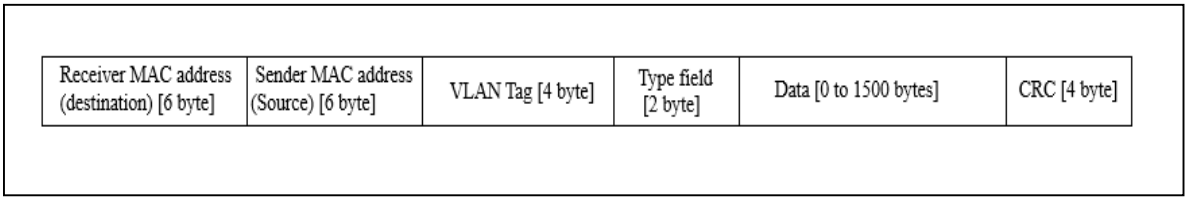

Figure 6 Ethernet Frame

The comparison of different automotive networks discussed above is shown below in terms of bit rate, topology, cabling and other factors.

Table 1.1 Automotive Networks

	BroadR-Reach	CAN	LIN	FLEXRAY	MOST
Maximum Bandwidth (Mb/s)	100	1	0.02	20	150
Messaging	Frames	Cyclic frames	Cyclic frames	Cyclic frames	Cyclic frames/streams
Topologies	Star, Tree	Bus	Bus	Bus, Star, Hybrid	Ring, Star
Media Access Control	Full duplex, Connectionless	Non-destructive arbitration	Time-triggered	Time-triggered	Time-triggered
Cabling	UTP	UTP	UTP	Optical	1-wire
Main Applications	Infotainment, backbone, ADAS	General bus	Switches, doors, seats	Safety critical	infotainment

To support Automotive Ethernet (BroadR-Reach) in automotive industry various industry consortiums and organizations such as One-Pair Ethernet (OPEN) Alliance SIG promotes Broadcom's BroadR-Reach 100 Mb/s Ethernet Physical layer specification as a standard for automotive networking and AUTOSAR ("Automotive Open System ARchitecture") which is the layered software architecture for the development automotive ECU's also supports Automotive Ethernet modules discussed in detail in the next sections has made it easy for the OEM's, its Tier 1 and Tier 2 suppliers to implement AE in vehicles.

1.3 AUTOSAR

AUTOSAR aims to standardize the software architecture of Electronic Control Units (ECUs). In AUTOSAR, the software components are made independent of the hardware and is based on the motto "cooperate on standard, compete on implementation."

1.3.1 AUTOSAR Layered Software Architecture

AUTOSAR is made for the software development of Automotive ECU's. By standardising the software, manufacturers can reuse the same software for different hardware which improve the cost and reduce the time and efforts in developing software for a particular platform.
The layered software architecture consists of:

1. Basic Software (BSW)
2. RTE (Run Time Environment)
3. Application Layer

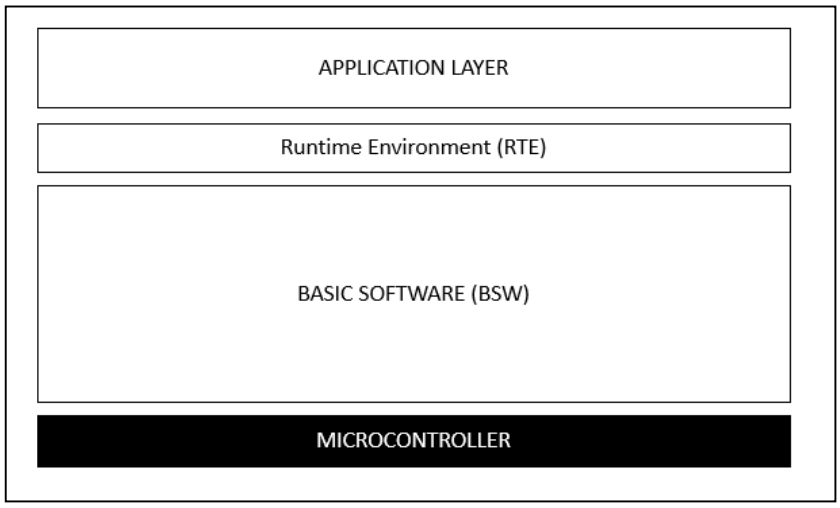

Figure 7 AUTOSAR

In automotive, the ECU consists of microcontroller and its peripherals, the ECU software as per AUTOSAR consists of basic software, RTE and application runs on the microcontroller [7] as shown in Figure 1.7

The application software consist of software components which are piece of software code that perform an algorithm based on some functionality in a vehicle e.g. cruise control, fuel injection etc.

The RTE is run time environment which makes the application software independent of the hardware, hence the code can be built in any programming language and its interaction with the lower layers can be taker care by the RTE.

The basic software consists of Microcontroller Abstraction Layer (MCAL), ECU Abstraction Layer, and Services Layer which have their different functions as shown in Figure 1.8

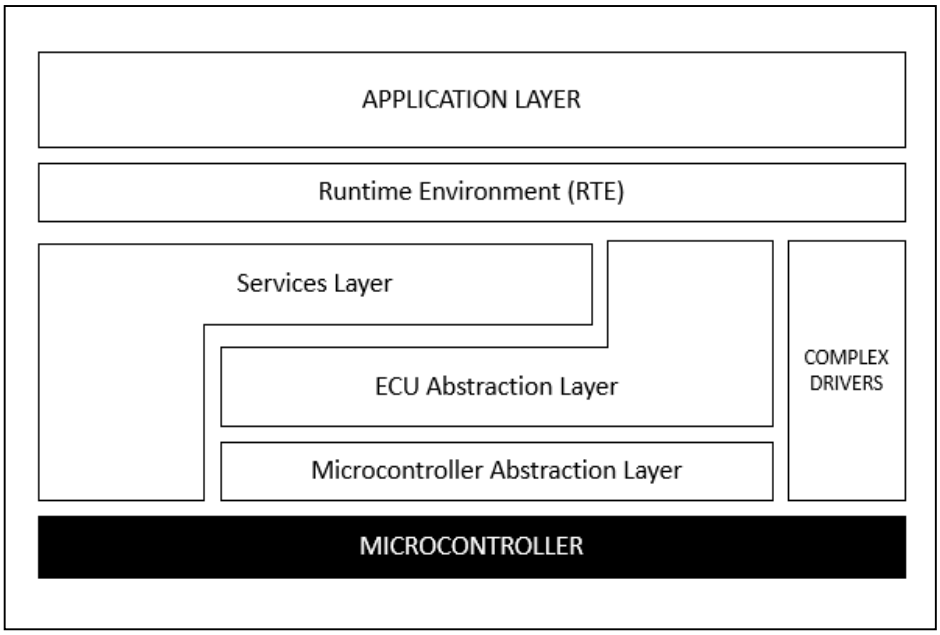

Figure 8 AUTOSAR Layered Architecture

The Microcontroller Abstraction Layer is the lower most layer of AUTOSAR and contains the drivers for internal peripherals of microcontroller which are directly linked to the hardware. It makes the upper layers of the software independent of hardware. The drivers can be for Communication, Memory, I/O etc. as shown in Figure 1.9

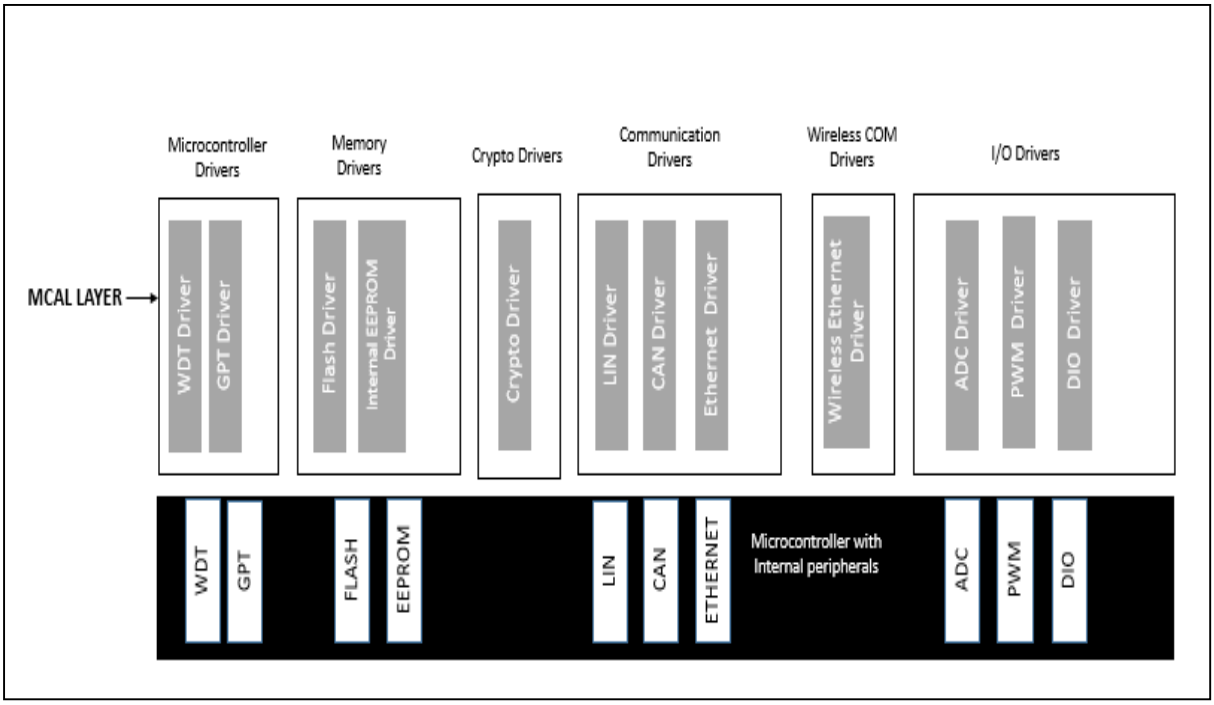

Figure 9 Microcontroller Abstraction Layer

The communication drivers are required for the communication of data on the bus. They reside at the Data Link Layer of the OSI- Model. They take the data from the upper layers and add certain elements of data link layer such as preamble etc. and send it via the controller to the PHY which converts the bits to voltages and sends on the physical media.

The next layer on top of the MCAL is ECU abstraction Layer which contains drivers for external peripherals and provides API's to access the peripherals without actually knowing their location. Like MCAL layer it has also different modules such I/O Hardware Abstraction, Memory Hardware Abstraction, On-board Device Abstraction and COM abstraction which work on top of MCAL Layer as shown in Figure 1.10

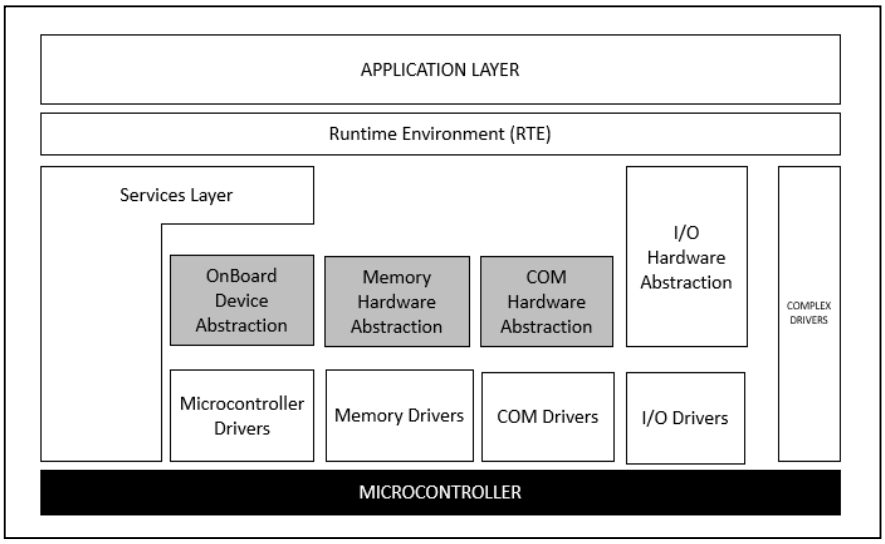

Figure 10 ECU Abstraction Layer

The COM hardware Abstraction module consist of different communication interfaces for CAN, LIN, FlexRay, Ethernet etc. Also it consists of drivers for external communication peripherals. The uppermost layer in the AUTOSAR layered software architecture is the services layer. It offers various functions such as:

- Operating system services
- Communication services
- Diagnostics services
- Network Management etc.

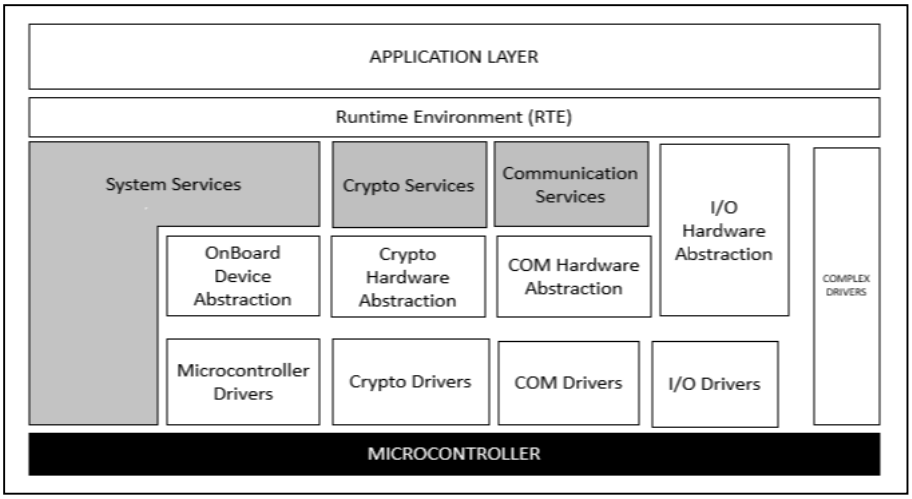

Figure 11 Services Layer

1.3.2 AUTOSAR Ethernet Communication Stack

In this section, the modules necessary for communication over Ethernet are explained as per the AUTOSAR standard. The MCAL layer will have Ethernet driver, the ECU Abstraction layer will have Ethernet Interface which provides access to upper layers and is makes them independent of the Hardware, it will also have drivers for external peripherals such as Ethernet switch and Ethernet transceivers, the services layer will have communication services which consists of different modules necessary for Ethernet communication as shown in Figure 1.12

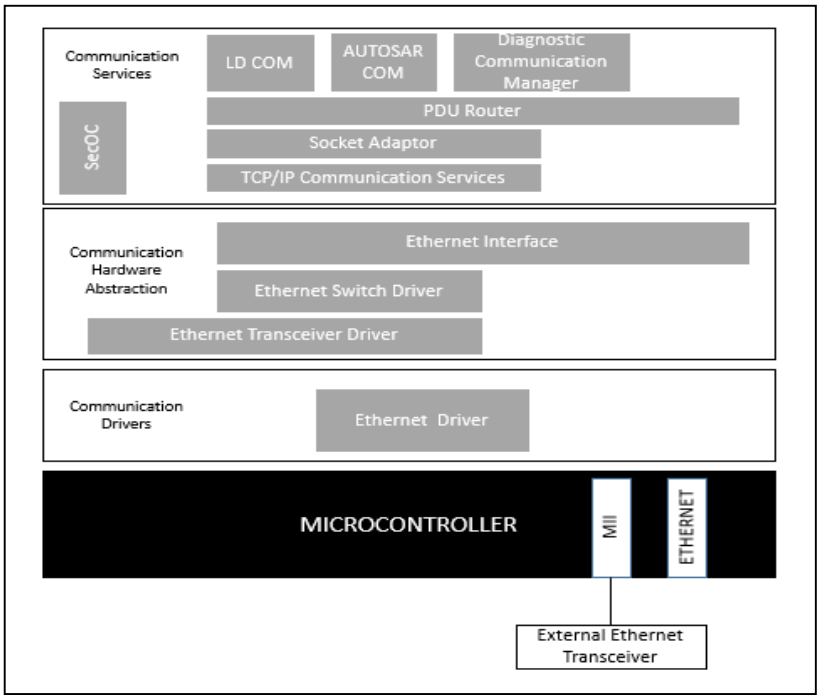

Figure 12 Ethernet Communication Stack

There are different modules in communication services layer consisting of COM module, PDU router, Socket adaptor, TC/IP module etc. which are necessary to define communication infrastructure for the data to be transferred over the Ethernet. Unlike LIN, CAN, FlexRay communication stacks where the data is transmitted on PDU's, Ethernet data in as PDU is mapped to sockets for point to point communication. The brief functioning of various modules in Ethernet Communication Stack is explained below:

a) COM module → The COM module in BSW of an ECU is responsible to convert the signals coming from RTE into protocol data units (PDU's).
b) PDU router →The PDU router passes the PDU to different communication TP module such as for CAN, LIN, Ethernet etc. to which the PDU is mapped in routing table of PDU router [8].
c) Socket Adaptor → Every PDU is mapped to a socket which consist of source port, source IP address, destination port, destination IP address which is necessary for the data to communicate over the Ethernet [9].

d) TCP/IP communication services → It consists of all the modules which supports protocols such as TCP, UDP, IPV4, ARP, NDP, DHCP etc. [10] which are necessary for Ethernet communication as shown in Figure 1.13

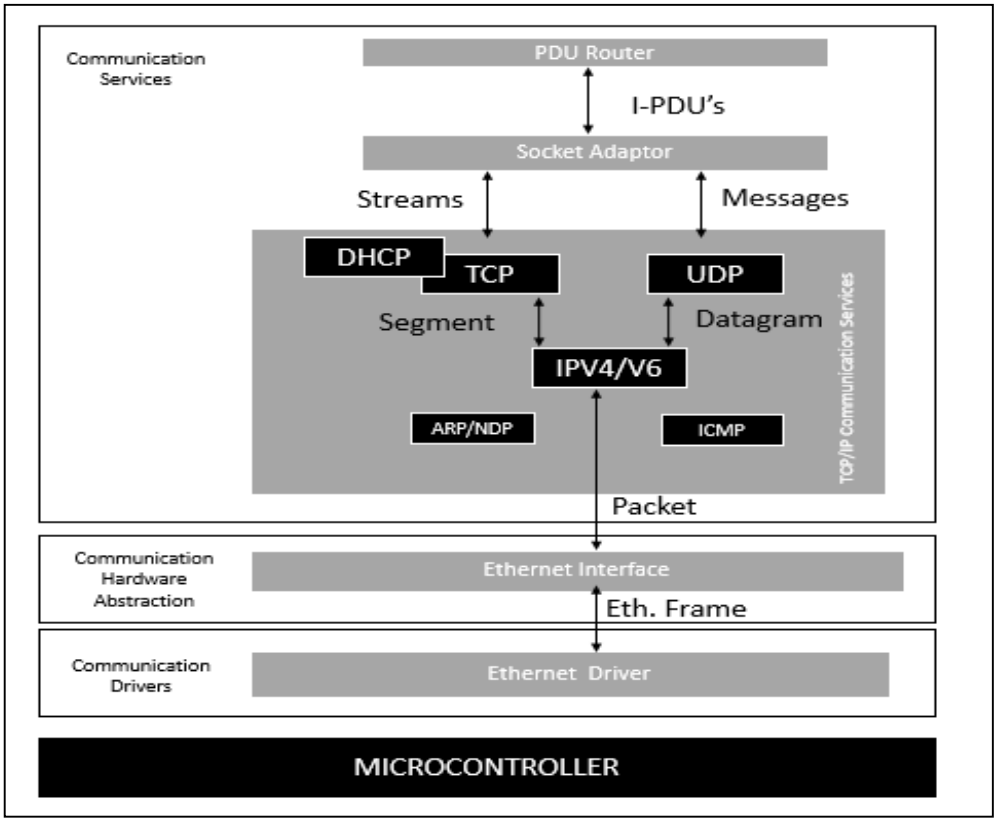

Figure 13 TCP/IP Communication Services

The application software components in the application layer of AUTOSAR communicate over the bus or inside the ECU via well-defined interfaces and ports. They are made independent with the help of VFB (Virtual Function Bus) [11] which is implemented as RTE as shown in Figure 1.14

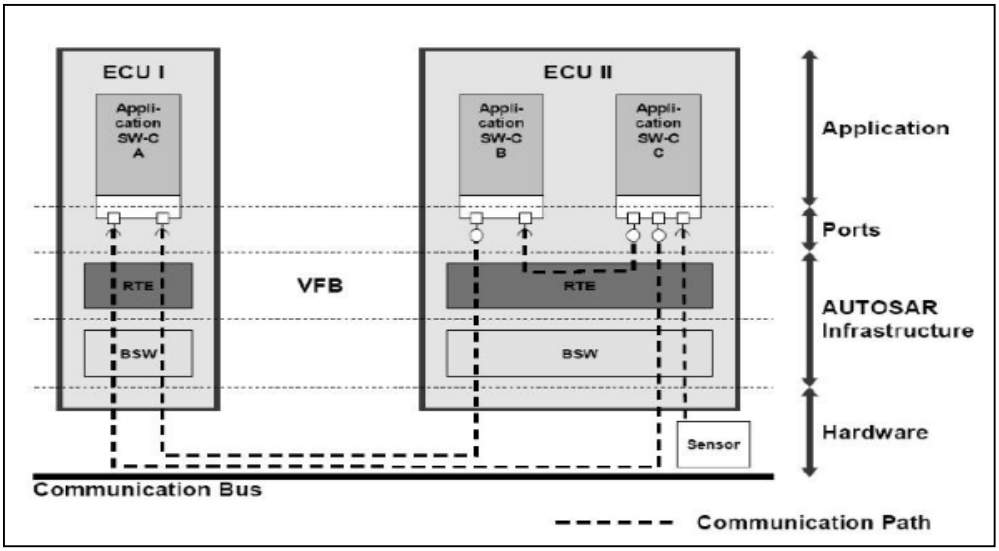

Figure 14 Inter and Intra ECU communication

The AUTOSAR standard supports service oriented communication on Ethernet with its SOME/IP modules namely SOME/IP transformer, SOME/IP- SD, SOME/IP TP and DOIP for diagnostics over IP as shown in Figure 1.15

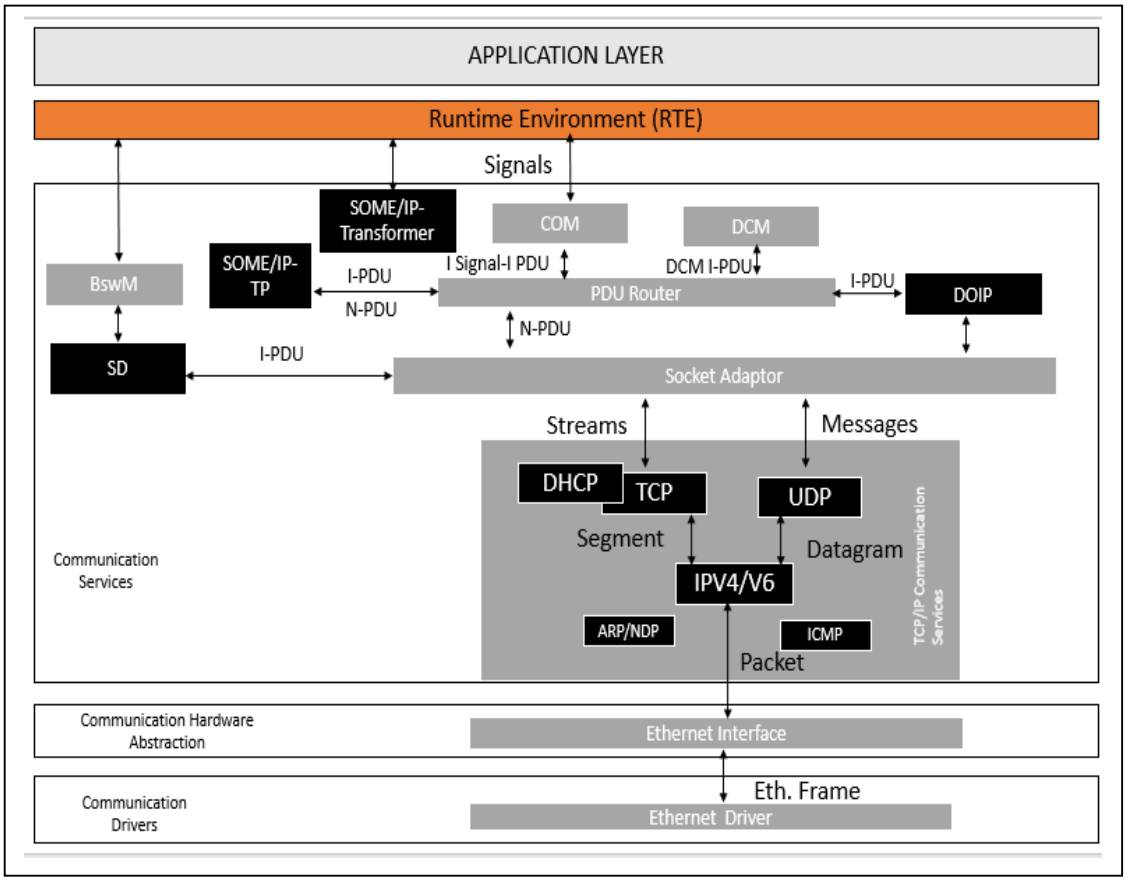

Figure 15 SOA Modules in AUTOSAR

SOME/IP Transformer – This module is responsible for linearizing the data coming from software components via RTE. The complex structured data is converted to linear data with the help of this transforms and generates the client ID which is always unique [12].

SOME/IP TP – This module is responsible for taking care of data which does not fit underlying TCP/UDP packet payload length, hence it creates N-PDU's which will be a part of on unit only but is sent on bus in multiple frames due to payload constraints.

SD – This module is responsible for the service discovery. Though this module a service can be found, offered on the network. Also subscription to various event notifications is sent through this module.

DCM – The DCM module receives the diagnostic messages from PDU router and decodes the UDS messages in the payload [13].

DOIP – The DOIP module receives the diagnostic PDU's from the PDU router to socket adaptor while transmitting or from socket adaptor to PDU router while receiving. It is present in the nodes which supports diagnostics over IP [14].

The AUTOSAR standardises the interfaces through which the different software modules will communicate with each other. These interfaces are:

AUTOSAR Interface: This interface is independent of a single programming language and is used for communication between different software components or from SWC's to I/O hardware abstraction module.

Standardized AUTOSAR Interface: It is a predefined interface and is used by SWC's to communicate with system services offered by modules such as Diagnostic Event Manager (DEM) etc.

Standardized Interface: It is also predefined and the API is written in C language. It is used between different basic software modules and between RTE and/or communication module, operating system as shown in Figure 1.16

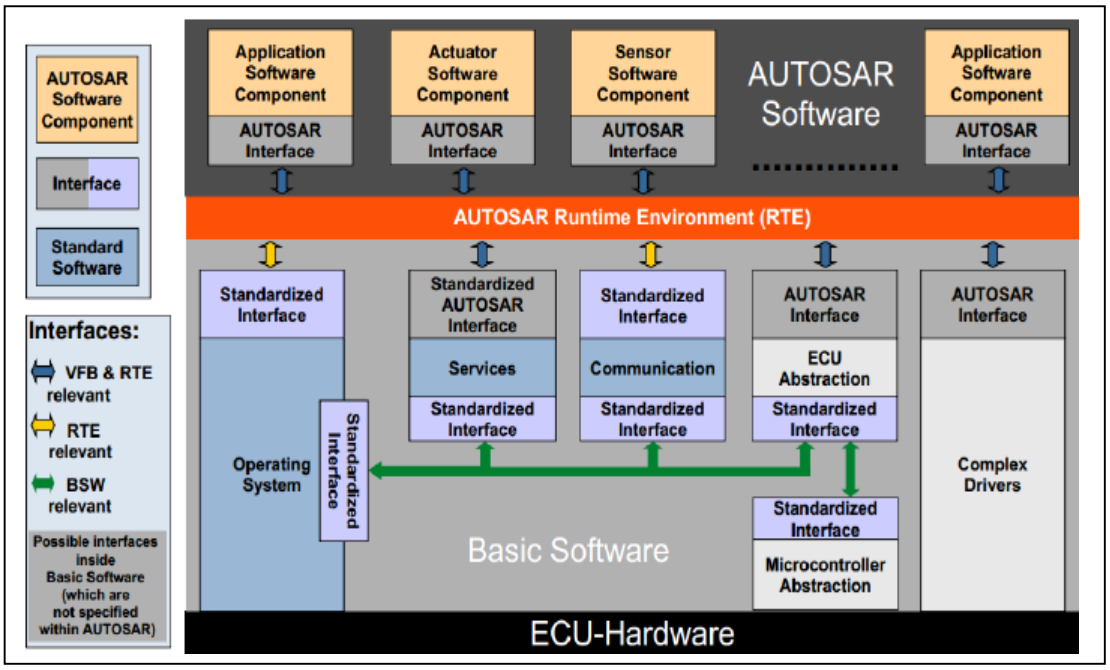

Figure 16 AUTOSAR Interfaces

1.3.2 AUTOSAR Software Development

The OEM's, its Tier 1 suppliers and Tier 2 suppliers play different roles in the development of ECU software. The OEM's design the software architecture and define the signals communicating between different ECU's based on the roles each ECU will play. The data, if design on CAN as network topology is shared in the form of DBC files which is shared with the Tier 1 suppliers who design the software on top of the Hardware purchased from Tier 2 silicon suppliers.

With the AUTOSAR ECU software development, the requirements of an OEM for its network is shared in the form of arxml files which are fed to AUTOSAR compliant tool chains by the Tier 1 Supplier for ECU software development which will generate RTE, BSW, software components for each ECU as shown in Figure 1.17

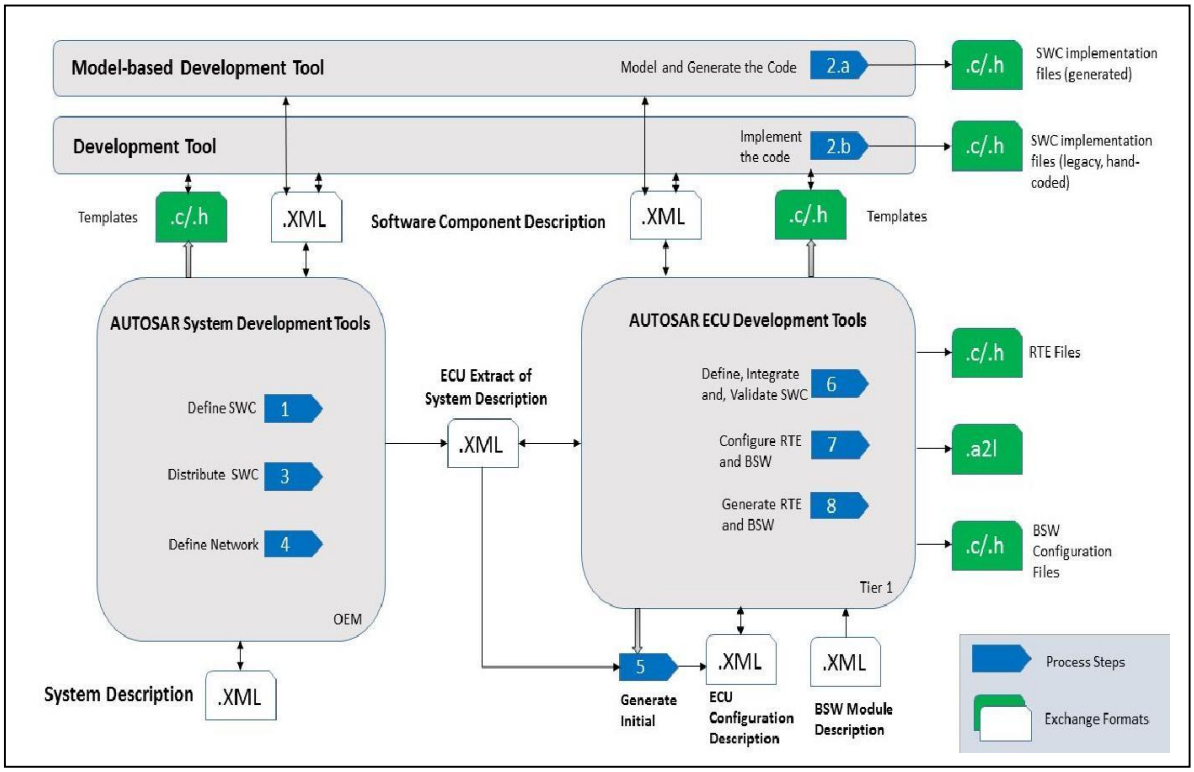

Figure 17 AUTOSAR ECU Software Development

As shown in the above figure, an OEM role is to define software components (SWC's) of applications, to define network i.e. ECU's with their bus topologies and then distributing these SWC's in the network. The defined data is shared in .XML formal called ARXML with the Tier 1 suppliers who will validate the data, configure and generate the RTE and BSW of the ECU. The ARXML file for software components is also fed to model based development tools such as MATLAB which will generate the relevant C code to be integrated with C files of BSW and RTE to build the complete software as per the requirements of the OEM.

1.4 AUTOMOTIVE ETHERNET AND ITS REFERENCE TO OSI MODEL

The OSI (Open Systems Interconnection) model is standardized for communication of data irrespective of the transmission medium. It consists of seven layers, each layer acting as an abstraction layer for the upper layer. Each layer has its own function.

The seven layers of the OSI model are as follows:

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

In automotive industry, the same OSI model has been used with different protocol to meet the automotive requirements for Ethernet communication. The placement of different protocols is shown in Figure 1.18

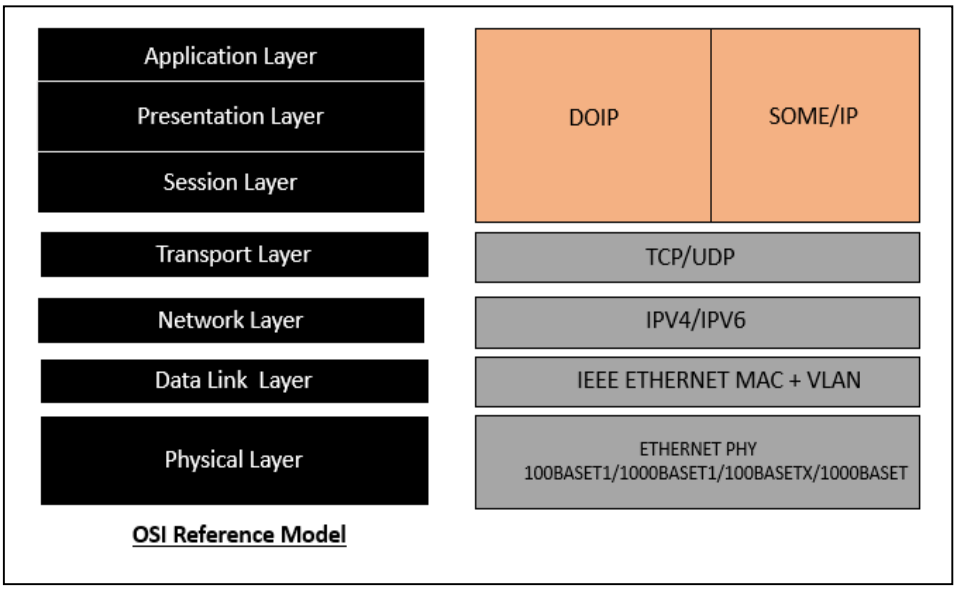

Figure 18 OSI Reference Model

Each protocol required for Ethernet communication in automotive vehicles is briefly discussed in the following section. As it can be seen from the above figure that automotive Ethernet is similar to normal Ethernet used in our day to day life as LAN or WLAN. Just the physical layer and application layer is changed to fit the automotive demands such as EMC etc.

1.4.1 Physical Layer

The first version of the standard is known as BroadR-Reach which was developed by Broadcom and is being supplanted by the IEEE versions known as 100BASE-T1 (P802.3bw) [15] and 1000BASE-T1 (802.3bp). BroadR-Reach interfaces to standard Ethernet MAC layer, so it uses same data link layer logical functions and frame formats as other kinds of Ethernet.

The physical layer of 100BASE-T1 is different from standard Ethernet physical layer or PHY. The encoding and signaling in the PHY is different to achieve the goals of low cost, low weight, compatibility with standard Ethernet etc.

In the PHY/transceiver of the BroadR-Reach, the standard MII passes 4 bits of data from microcontroller to the PHY. The BR-PCS performs 4B3B encoding which converts four bits to 3 bits. Now the block encoding technique used here is 3B/2T in which 3 bits are converted to a pair of ternary symbols. Then line coding done by PMD Layer which is PAM 3 which uses 3 voltage levels (+1, 0,-1) to transmit electrical signals on single pair. The conversion is shown in Figure 1.19 and Figure 1.20

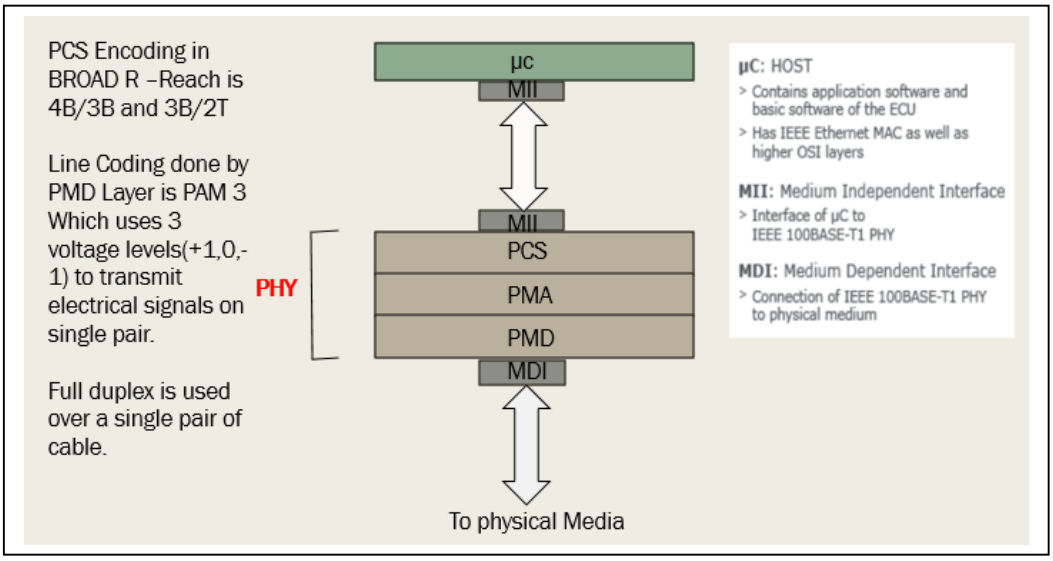

Figure 19 BroadR-Reach PHY

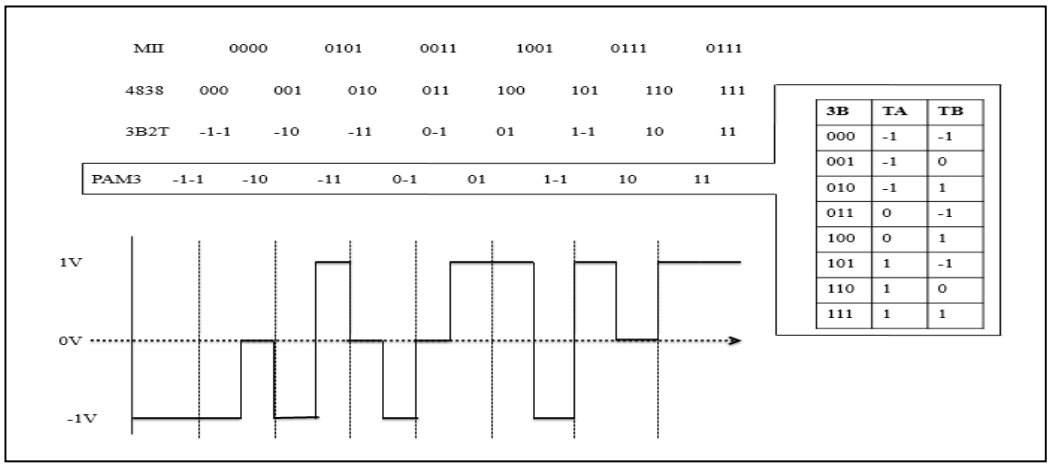

Figure 20 Signalling in 100 BASE T1

Nowadays Ethernet uses point to point links between end devices via a switch. It is different from other technologies which uses CSMA/CD to access the bus which is shared between all the devices. As in case of standard Ethernet 100BASE TX there are two pair of wires, one for reception and for transmission, it is a form of dual simplex transmission whereas BroadR-Reach is full duplex because it transmits messages on a single twisted pair, this has been possible because of the hybrids and echo cancellation techniques used in the PHY shown in Figure 1.21

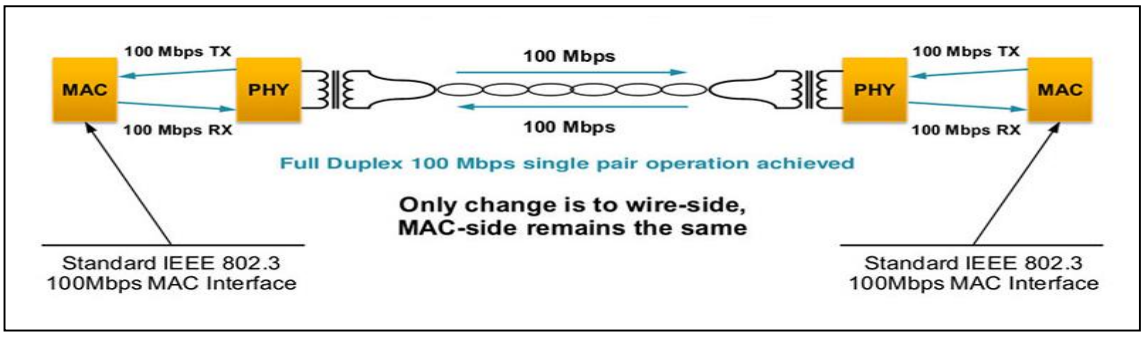

Figure 21 Full Duplex 100 BASE T1

1.4.2 Data Link Layer

The data link layer is responsible for the formation of frames which are transmitted on the physical bus. It consists of two sub layers:

- MAC (Media access control)
- LLC (Logical Link Control)

In the MAC layer, the flow control and collision avoidance mechanisms are taken care off whereas in LLC VLAN tag etc. is implemented. In the MAC sublayer, MAC addresses are used to address different network nodes. In the Ethernet frame, source and destination MAC address are inserted which are 48 bits and are bound to hardware.

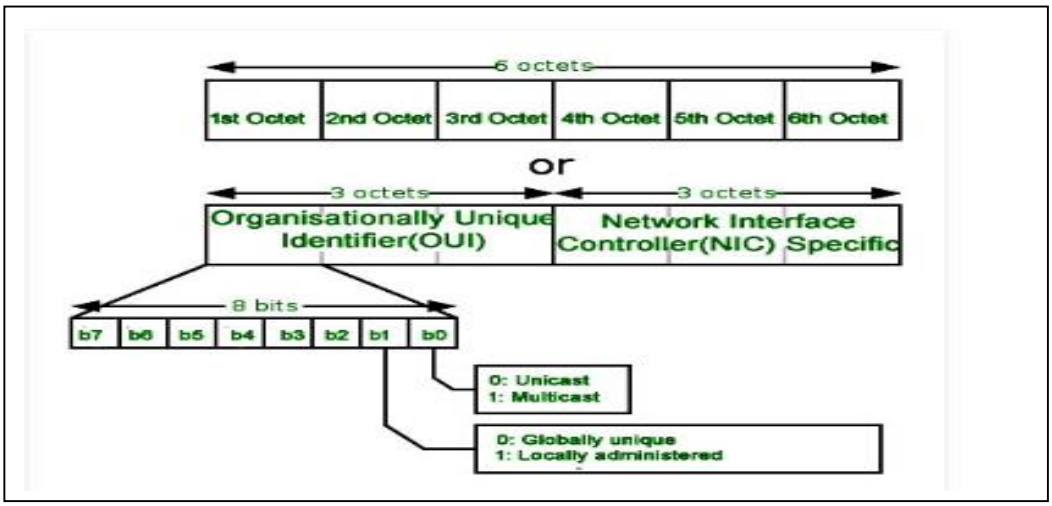

Figure 22 MAC Address

In Automotive Networks, if the nodes are not communicating with the outside work, then the MAC addresses can be locally defined which assigned by network administrator, riding over burned-in address as shown in Figure 1.22

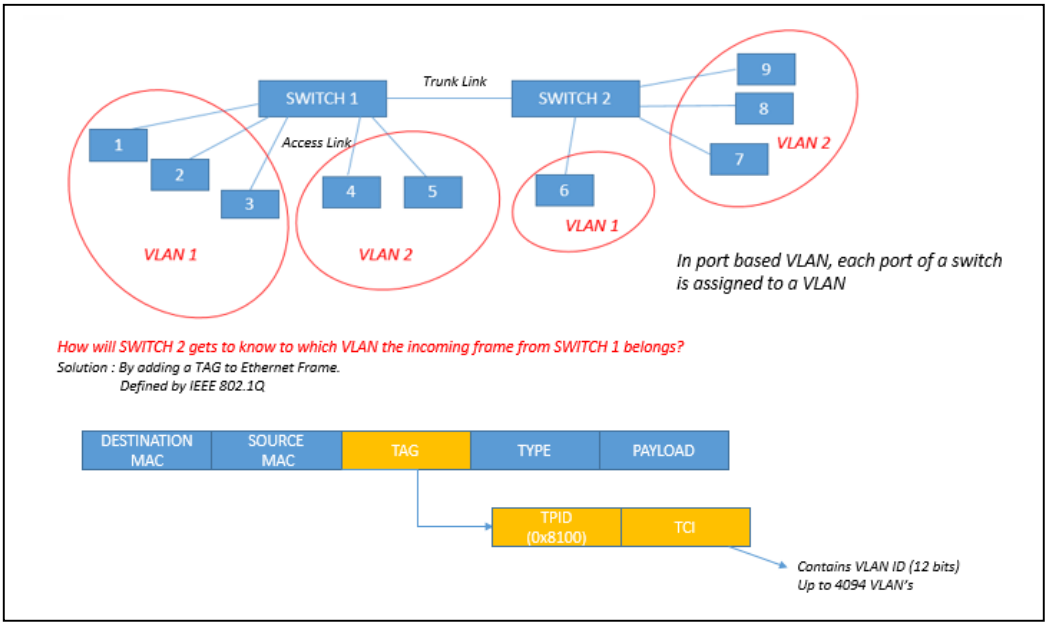

Figure 23 VLAN Tag

The LLC layer is also responsible for VLAN tagging which is necessary to limit the broadcasts from one domains to another. The node in VLAN 1 will transmit data to only other nodes in VLAN 1, but when a switch is part of more than 1 VLAN, then VLAN tagging is necessary to know that to which VLAN the incoming frame is meant for as defined by IEEE 802.1Q shown in Figure 1.23

1.4.3 Network Layer

The IPV4 protocol resides at the network layer and is a 32 bit logical address. For the frame to travel from one device to another, the source and destination IP address needs to known. Also there are two type of IP addresses – private and public within each class A, B, C, D as shown in Figure 1.24

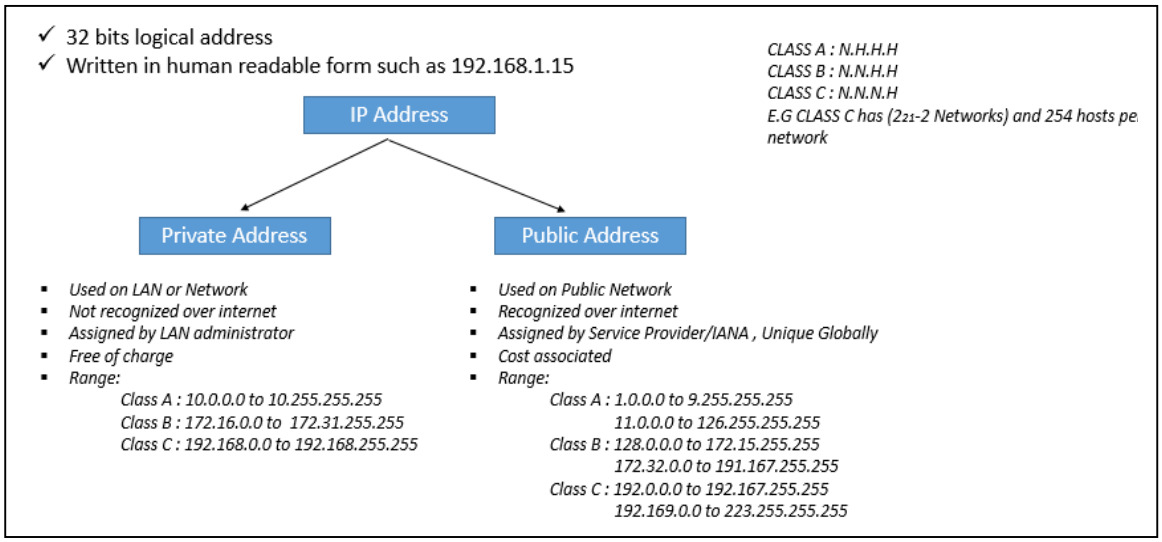

Figure 24 IP Addresses

In Automotive Networks, if the nodes are not communicating with the outside work, then the IP addresses can be private i.e. from the range shown in above figure. The subnet mask and network ID concept is used to minimize the wastage of IP addresses. An example of how they are defined is shown in Figure 1.25

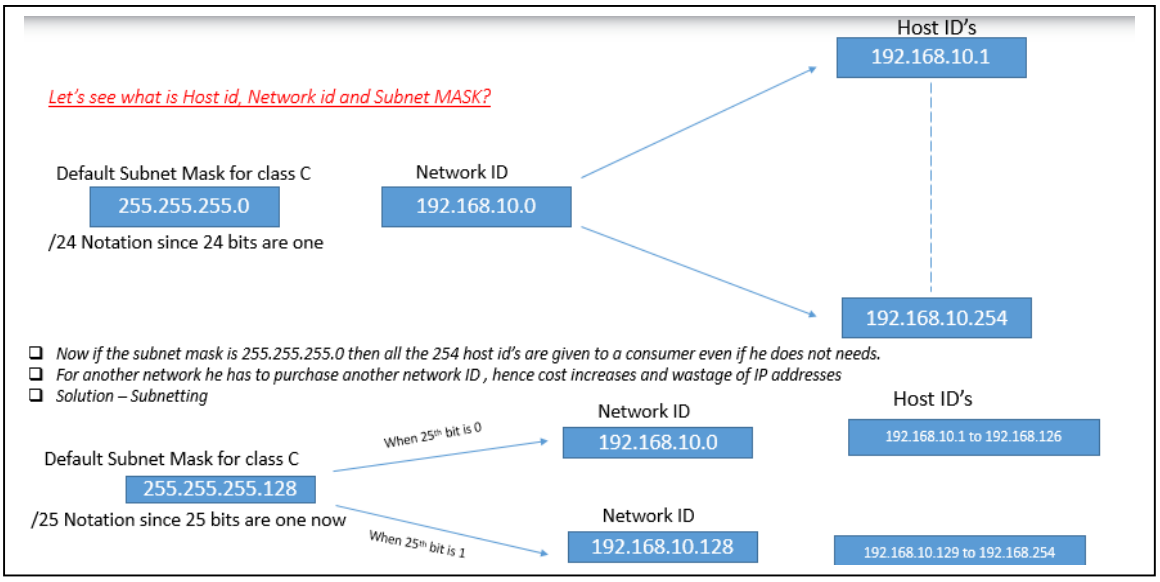

Figure 25 Subnet Mask and Network ID

1.4.4 Transport Layer

The Transport Layer consists of two protocols:

- UDP (User Datagram Protocol)
- TCP (Transmission Control Protocol)

UDP can be used for unicast as well as multicast i.e. when the data is to be sent to n number of nodes and is connection less protocol whereas TCP can be used for only unicast as it is connection oriented protocol and receives acknowledgement for every message sent on the network. Each UDP and TCP needs a port number on which the data from the application is sent to network layer and further on the bus. Hence giving a port number is termed as application end point.

Socket – Since while transmitting Ethernet frames, source port, source IP address, destination port, destination IP address are necessary. All the four combine to form a socket on which the data is sent and received for a particular application.

1.4.5 Application Layer

The SOME/IP and DOIP protocols which are specifically for automotive network works on the application layer. The SOME/IP header format is used to covert the data contained in the payload into a service which is provided by the software component as shown in Figure 1.26

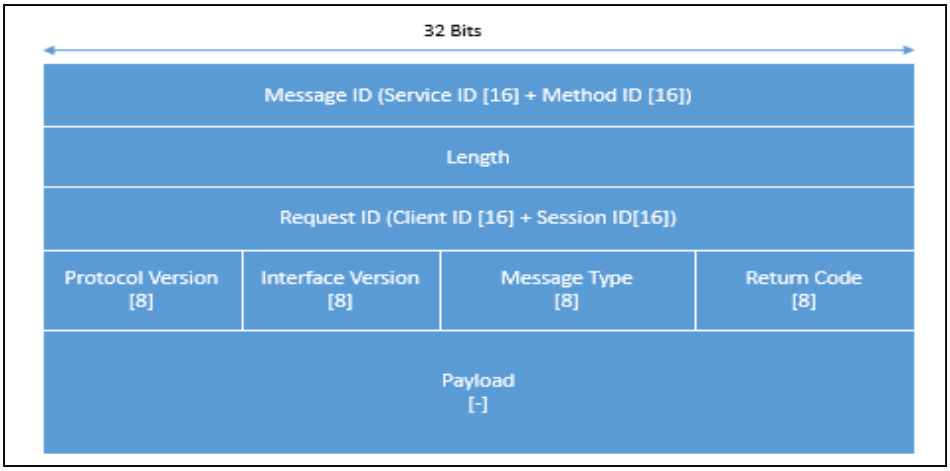

Figure 26 SOME/IP Header

Message ID: Service ID [16 bits] = A service ID is assigned to service which contains methods and events. Method ID [15 bits] = If 17th bit is 0 (Hence for each Service, there are 2^15 Methods. Event ID [15 bits] = If 17th bit is 1 (Hence for each Service, there are 2^15 Events/Notifications)

Request ID: ID of a calling client inside ECU, generated by RTE, has to be unique for single client server combination.

Message Type: 8 bit ID which tells what kind of payload is whether it is a request/response or an event notification as shown in Table 1.2

Table 1.2 SOME/IP Message Type

Number	Value	Description
0x00	Request	A request expecting a Response (even void)
0x01	Request No Return	A fire and forget request
0x02	Notification	A request of notification expecting no response
0x80	Response	The response message
0x81	Error	The response containing an error

The DOIP protocol is specified in ISO 13400 [16]. The Diagnostics data as per the UDS protocol is contained in an Ethernet Frame which increases the payload size. The difference between diagnostics on CAN and IP is shown in Figure 1.27

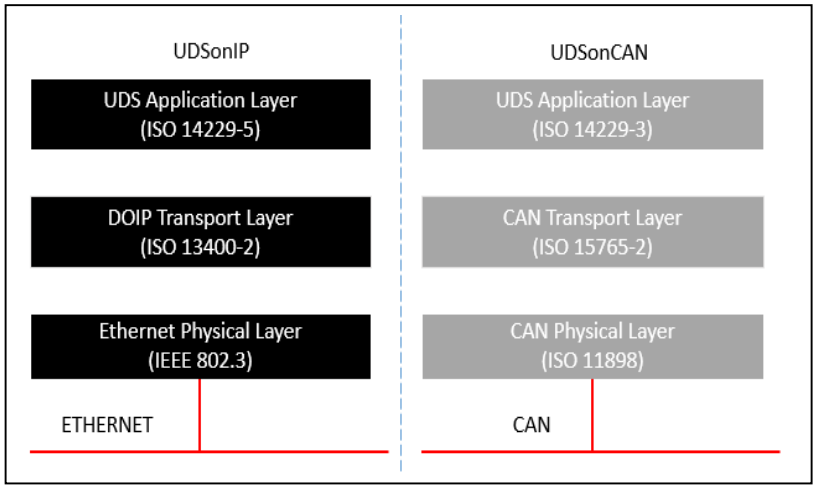

Figure 27 UDS on CAN and IP

The diagnostic data as per the UDS protocol is contained in DOIP payload field of Ethernet Frame as shown in Figure 1.28

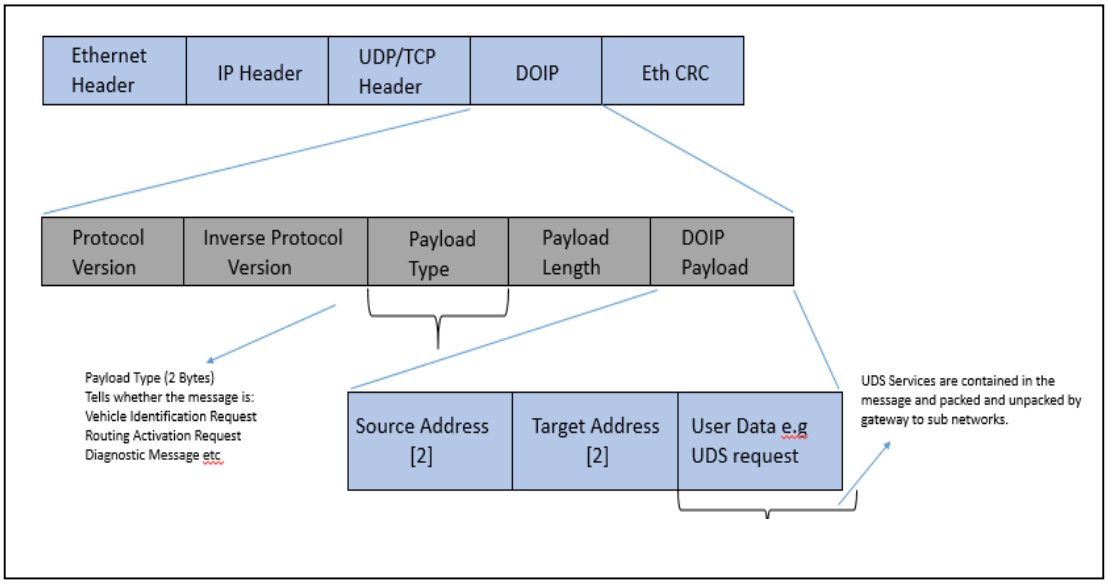

Figure 28 DOIP Header

The payload type field contains what type of message is transmitted in the frame which is shown in Table 1.3

Table 1.3 DOIP Message Type

Payload Type	Description/Value	Transport Protocol
0x0001	Vehicle Identification Request Message	UDP
0x0002	vehicle Identification Request with EID	UDP
0x0003	Vehicle Identification Request with VIN	UDP
0x0004	Vehicle Identification Response Message	UDP
0x0005	Routing Activation Request	TCP
0x0006	Routing Activation Response	TCP
0x8001	Diagnostic Message	TCP

1.5 THESIS OUTLINE

In the next generation E/E Architecture, demands for higher bandwidth and low-latency communication are coming from advance driver assist system and infotainment domains that cannot be achieved by established technologies such as CAN, LIN etc. A new approach towards a flexible highly scalable network is Automotive Ethernet.

With its huge bandwidth, Ethernet provides the potential for the communication of the different automotive domains – e.g. chassis, powertrain or info- and entertainment – on a single shared physical network. Also the established SOA in IT industry [17] [18] can be implemented in vehicle with the introduction of SOME/IP protocol by AUTOAR for automotive Ethernet.

Hence this thesis will be focused on creating a service oriented architecture over Ethernet as its network topology. The use of Ethernet and demand for embedded Ethernet technology solutions in automotive embedded system industry paved the way for use of SOA in automotive embedded system. Different service interfaces which contains the service are defined and typed to the software components of ECU's to communicate with each other. The service interface offers services and find services offered by other nodes in the network. This way the communication between ECU's shifts from signal based to service based communication.

To build a SOA on Ethernet communication, knowledge of automotive Ethernet is required which comprises of all the 7 layers of an OSI model viz. physical layer, MAC layer, network layer, transport layer, session layer, presentation layer and application layer. So a brief study is done on each layer

which is necessary for to define the communication. The application layer protocol called SOME/IP was studied in this chapter which is given by AUTOSAR for service oriented architecture over Ethernet.

Chapter - 2 is based on literature survey which was necessary to understand the trends in the automotive Ethernet technology in vehicles and how the SOA can be used to design Ethernet communication. The survey gave a good understanding to propose this POC.

Chapter - 3 discusses the motivation behind my work, objectives to be achieved and the proposed methodology in which the use case has been proposed for which Ethernet communication design with SOA will be created in the next chapters.

Chapter - 4 consists of the system model in which the services are designed for the use case using SOME/IP protocol, also the Ethernet communication parameters necessary for the implementation are designed.

Chapter - 5, 6, 7 are focused on the implementation of the design in Architecture tool, the results exported after the design to be shared for software development and the future scope respectively.

CHAPTER 2
LITERATURE SURVEY

This chapter includes the work proposed and implemented by different author in the area of service oriented architectures, AUTOSAR, vehicle networking which gave me understanding and support to take forward the thesis work.

1. Jorg Schauffele explains in his paper how two new domains in vehicles – advance driver assist systems and vehicle integration in cloud require high performance communication paradigms such as Ethernet. He also explains how E/E architectures are defined to support a product line of vehicle. He has shown how PREEvision can be used to design and optimise E/E architectures by providing various modelling layers such as software architecture, ECU network topology layer, Communication layer according to AUTOSAR [19].

2. Jelena KociĆ describes the key sensors such as Lidar, Radar & Camera for autonomous vehicles which collects data from the surroundings and after processing assist the driver in driving. The data from these sensors can be analysed and used to assist the vehicle in applying brakes, accelerating or taking some other decisions. It discuss the capabilities and complexities of the highly used camera sensor. Also tells the importance of using radar for features such as Adaptive Cruise Control, warning collision etc. Finally the necessity of sensor fusion is explained since the cameras will not be perfect in domains of lidar or radar and vice versa, hence the sensors are complimentary to each other and combining their data can lead to much accurate results [20].

3. P. Hank [3] sheds light on the automotive Ethernet as a holistic approach in next generation vehicles. It compares the point to point Ethernet switched networks in which the available bandwidth can be used efficiently as compared to CAN based broadcast bus systems. It tells how the Ethernet has evolved in vehicles from being used for diagnostics or ECU flashing to its use in ADAS and infotainment domains and finally towards using Ethernet as network backbone. It shows how automotive Ethernet is different from the normal Ethernet used since it uses only single twisted pair cable to both transmit and receive data. Also shows how the upcoming trends in vehicle architectures demands for high speed which can achieved by using automotive Ethernet [21].

4. Timo Hackel describes how the current ICT architecture does not fulfil demands such as high bandwidth, connectivity to internet, dynamic finding and adding components in the network, hence the SOA architecture over Ethernet is required for new ICT architectures. It explain how SOA can be used to design components of a system into services which can be called from anywhere in the network and are reusable and decoupled components since they are hardware independent. Explains how AUTOSAR has taken the initiative to implement SOA in cars via SOME/IP middleware solution. The paper also shows how the middleware is supported over TCP/IP stack of Ethernet for different OSI layers [22].

5. Wang Dafang explains the importance of VFB in AUTOSAR which is implemented as RTE in an ECU software, the RTE makes the software components independent of a hardware hence relocation of components among different ECU's of the vehicle is possible. It explains how the data is sent via the ports of the components. The different communication patterns such as client/server or sender/receiver are explained and it's mapping to ports of the components. The paper has briefly introduced AUTOSAR and VFB concept which has been used in the thesis [23].

6. Shane Tuohy and others in their review paper has shown how and where different intra vehicle networks such as CAN, LIN, MOST, FlexRay etc. are used which elaborates the need for Ethernet for next generation network having driving assistance features such as lane assist, traffic sign recognition etc. Moreover it talks about the automotive Ethernet which now fulfil the EMC requirements of the vehicle which is cheaper, lightweight and transmit and receive data over single unshielded twisted pair. Also different automotive suppliers and manufacturers are part of one or the other group promoting the use of Ethernet and standardising the protocols. The paper talks about AUTOSAR which is meant for Automotive ECU's software's to be used for high bandwidth domains leading to saving development time and cost. It emphasizes the need to migrate to AUTOSAR development and many OEM's have achieved the same since it will be a gradual process [24].

7. Gopu G.L. in his papers explains how the SOA already there in the IT industry but the protocols for SOA cannot be used in automotive ECU's since the resources are limited. The paper shows home AUTOSAR took up the challenge and created a middleware called SOME/IP for SOA to take place in vehicles. The functionality is converted to a service by attaching SOME/IP header which consists of service ID's and Method ID's or Event ID's for the service [25].

8. Mathias Johanson has proposed in his work how remote diagnostics can happen with Diagnostics over IP (DOIP) standardised by ISO 13400-1, 2, 3 parts. In his paper he has shown a prototype by using the DOIP stack in Linux based telematics unit which is connected to CAN network, the remote tester successfully sends the diagnostic messages to the telematics gateway over WLAN interface which sends the diagnostics request to the CAN network and then takes the response back from network and sends to the external tester [26].

9. Young Seo Lee explains CAN based protocols for diagnostic services or ECU flashing are not suitable for future ADAS and infotainments domain since they provide low data rate and less payload. Whereas Ethernet provides much higher bandwidth to make the diagnostic services faster and less complex. It shows the importance of a Gateway in the vehicle network for diagnostic services to occur. Also for remote diagnostics, he proposed a gateway is necessary which will be connected to internet as well the vehicle's different sub networks. Also a gateway is required since it will only understand the logical addresses which helps the gateway to translate messages and send to the networks which are not on Ethernet. Apart from this, he suggests that the gateway is necessary for security reason to make sure that the data or request are coming from a valid external tester [27].

10. A paper titled “Evolutionary Exploration of E/E-Architectures in Automotive Design” written by Ralph Moritz et.al. discusses about an evolutionary algorithm for electronic and electrical architecture exploration in a car. Techniques for optimizing the communication infrastructure have been shown by defining variation operators in detail. Here, a hierarchical partitioning of ECUs has been presented based on the first level and the second level. The algorithm discussed is suitable for both existing architecture and new designs as well [28].

11. In the year 2009, a paper titled “In-Vehicle Automotive Network Gateway Electronic Control Unit for Low Price Vehicle”, written by T.S. Shamin Dudu et. al. was published in SASTECH. This paper presents an idea to use a gateway ECU to interconnect CAN and LIN field buses for reducing the cost of vehicle. Line transceivers have been used for establishing this interconnection and the specifications have been implemented using a PIC microcontroller for this ECU. This gateway ECU can be later enhanced based on customer needs without any change in its existing cost and hardware [29].

12. In the year 2017, a paper titled “Approaches for In-vehicle Communication – An Analysis and Outlook”, written by Arne Neumann et.al. various available and emerging networks for in-vehicle communications have been analysed. This discussion has been made at the level of communication layers with a special focus on physical layer and medium access protocols. Ethernet has been proposed as a solution to communication related problems and can be implemented in three generations [30].

13. In the year 2013, a paper titled “Automotive Ethernet: In-vehicle Networking and Smart Mobility”, written by Peter Hank et. al. describes the way of bringing Ethernet into automotive applications for next generation in-vehicle networking. Ethernet has been proved as a technique that links vehicle electronics and Internet due to which huge amount of data can be communicated between communication system of the vehicle and cloud based services for optimization of power and cost. A special focus has been laid on the EV communication architecture where Ethernet can accelerate data transmission between vehicles, base stations or charging stations [31].

14. In the year 2013, Article “How to Engineer Tool-Chains for Automotive E/E Architectures?” has been published in a newsletter by Peter Waszeckiet.al. This article provides systematic approaches to prepare tool chain for E/E architecture design process which covers all design and development phases. Different life cycle phases of development have been explained with the V-Model with fine illustration. Usability and functionality aspects of various tools are quantified also taking the findings account, their coverage and compatibility has been checked in terms of forming a tool-chain. Furthermore, it is assumed that some of the studied tools such as PREEvision, ChronSIM/ChronVAL, TargetLink or SIMTOOLS will establish themselves as a standard in the development of E/E architectures [32].

15. The conference paper published in 2014 in IEEE titled “Automotive Ethernet in On-Board Diagnosis (Over IP) & In-Vehicle Networking” by C. Varunet.al. In which a concept has been presented to migrate a CAN based complete Car network to an Ethernet-based approach by just using cyclic send types and uni-cast addressing. OBD system inside the vehicle which has internet connectivity also

provides a user interface to Driver so that he can read any of the sensor values by selecting on the list. Fault diagnosis and on the spot fault information without using OBD scan tool is also described in the system setup. MATLAB software has been used to measure the performance of Ethernet bus. The proposed system has been implemented in hardware using two ARM7 and one ARM 9(server) [33].

CHAPTER 3
PROBLEM STATEMENT

3.1 MOTIVATION

The upcoming age of vehicles will have different self-driving features, for example lane assist, adaptive cruise control, traffic sign recognition and so forth which will assist the vehicles in driving with minimum human efforts. Such autonomous features would require camera, radar, LIDAR, ultrasonic sensors and so forth which will act as senses of the vehicle. These sensors would need to transmit a high amount of information/data to the ECU which would settle on a decision subsequent to analysis of the data received.

Hence Automotive Ethernet was found to be the best choice for physical network to transfer the data from these sensors via a switch to the decision making ECU as it supports large payload (1500 bytes) and high data rate (100 Mbps/1000Mbps). Also there are various standards like AUTOSAR and OPEN Alliance SIG promoting Automotive Ethernet which will make it easy to implement in the vehicle for the various OEM's.

It was found that CAN or FlexRay were based on signal oriented communication in which the signals are statically fit the message layout which considers the scenario of sending empty signals if the signals are not available and broadcast the messages on the respective bus whereas Ethernet supports service oriented communication in which the data is sent to the client when it is asked and has subscribed for the notifications. Also the data elements are serialized when sent on Ethernet by the SOME/IP middleware and is required in ADAS domain because it will save bandwidth and computing resources.

For the nodes communicating through Ethernet network, the diagnostics over IP (DOIP) is required to update the firmware of the ECU's and to detect faults at higher rate as compared to diagnostic on CAN. Every one of these focuses propelled me to plan Ethernet communication design with service oriented architecture (SOA).

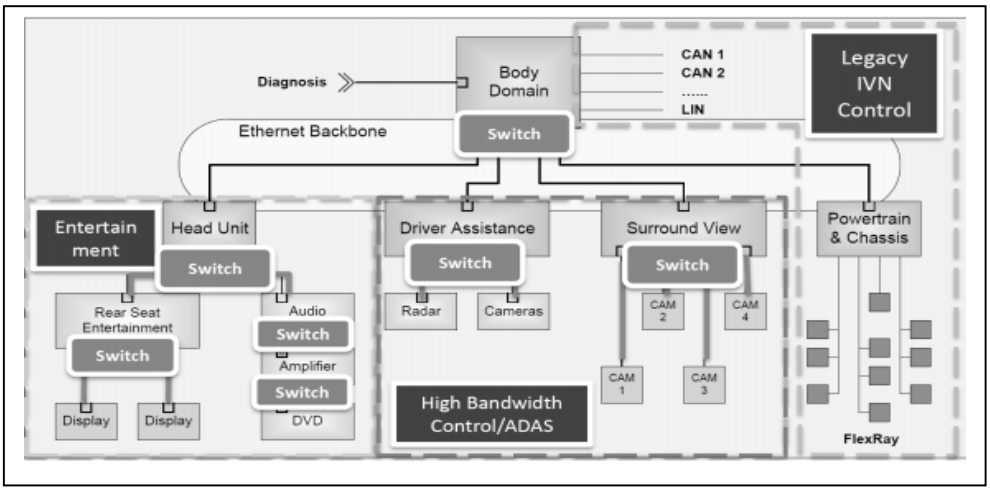

Figure 29 Ethernet in Vehicle Network Topology [34]

3.2 OBJECTIVES

Present E/E Architecture:

- Supports CAN, LIN, LVDS
- Signal Oriented Communication
- Software Development → non AUTOSAR compliant
- Signal's length, position fixed (no serialization)
- Not feasible for upcoming autonomous and connected cars
- Low data rate (0.2 Mbps for CAN)
- Less payload size (8 bytes for CAN)

Proposed E/E Architecture:

- Will support Ethernet Switched Network for ADAS and Diagnostics applications
- Service Oriented Communication
- Software Development → AUTOSAR compliant
- Length and position of data elements variable (need serialization)
- High data rate (100 BASE T1→ 100 Mbps, 1000 Base T1→ 1000 Mbps)
- Higher Payload (1500 bytes)
- Feasible for next gen autonomous and connected cars

At present the ECU software development happens with OEM exchanging DBC files with Tier 1 suppliers in which the CAN messages, signals with their position, length and its attributes are defined. With AUTOSAR compliant ECU software development, the files will be based on XML format to be exchanged between OEM's, TIER 1 and TIER 2 developers which will decrease developed cost and time in future.

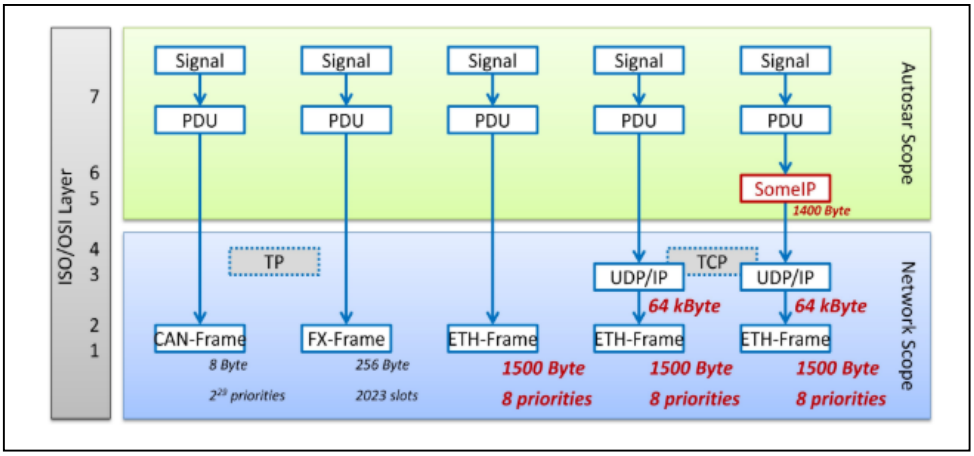

Figure 30 Payload Comparison for different bus systems [35]

Hence the objective of this thesis is to propose how Ethernet communication can be designed on service oriented architecture (SOA) which will be AUTOSAR compliant and to generate XML files for the process of ECU software development.

3.3 PROPOSED METHODOLOGY

Once the objectives were clear, the next step was to propose a use case which would realize Ethernet communication design on Service Oriented Architecture (SOA). Hence for this thesis an ADAS feature named Adaptive Cruise Control (ACC) was chosen to demonstrate SOA.

In present vehicles the Cruise Control feature is based on signal oriented communication. In cruise control, the vehicle runs on the speed set by the user. This feature efficiently works on CAN network present in the vehicle.

Whereas the Adaptive Cruise Control (ACC) is intelligent enough to adjust the speed of the vehicle by sensing the position and speed of the vehicles in the front the car, hence the name Adaptive. The network topology for ACC will consist of ECU's namely Radar, Camera, Central Gateway (CGW), ADAS, Anti-Lock braking system (ABS) and Engine Management System (EMS).

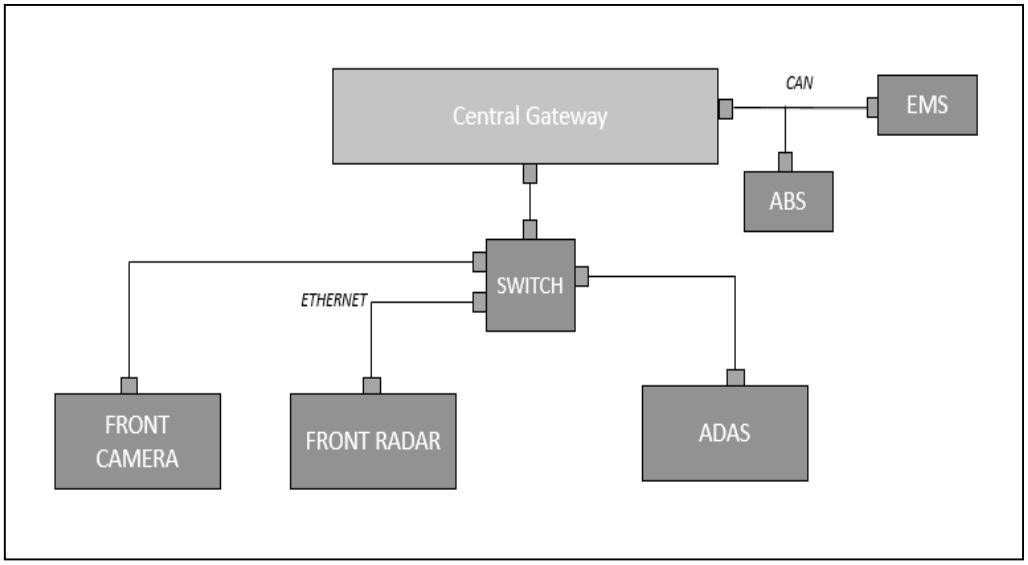

Figure 31 Proposed Network Topology

In the above Figure 3.3, Camera, Radar, ADAS ECU's are connected via switch to the CGW ECU. On an abstract level, the data from camera and radar is sent to ADAS ECU which decides from the data that what should be the adaptive speed of the vehicle. Since the data from camera and radar is large hence they are placed on the Ethernet network.

Subsequent to deciding the adaptive speed of the car, ADAS ECU shares the data with the CGW ECU which acts as gateway between CAN and Ethernet Network. The CGW after receiving the adaptive vehicle speed, determines whether to apply brakes by sending a signal to ABS ECU or to accelerate by sending a signal to EMS ECU.

Since the service oriented communication will happen over Ethernet network consisting of nodes connected to switch, this thesis focuses on defining Ethernet communication design and services design as per the SOME/IP protocol in the following chapter by taking the example of above use case.

CHAPTER 4
SYSTEM MODEL FOR SERVICE ORIENTED ARCHITECTURE (SOA)

4.1 SERVICE DESIGN

In the last chapter, network topology was defined for the Adaptive Cruise Control (ACC) use case which was proposed for Ethernet communication design on service oriented architecture. Before modelling the design in the software tool, services needs to be defined based on SOME/IP protocol for SOA.

The general flow for designing a service is shown in Figure 4.1

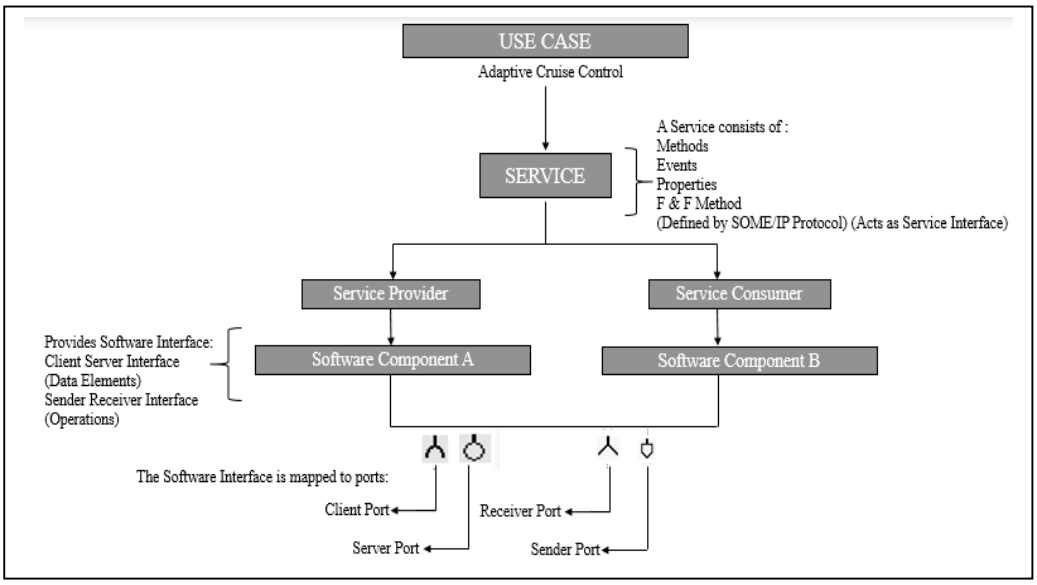

Figure 32 Service Design Flow

Based upon the use case, the services are defined. A service can consist of methods, events etc. A service perform a particular task and is piece of software. A service will consists of:

- Service Interface (describes how the information is transported between ports)
- Service provider (provider of the service)
- Service consumer (consumer of the service)

For each service provider a software component is defined which provides the service to the application component of the consumer of the service. The implementation of the service is different from the service interface, hence the consumer just consumes the service without knowing the actual implementation performed by the service provider component.

This thesis focuses on the definition of services and not the implementation. The service is mapped to the software interfaces which can be:

- Sender Receiver Interface (If the service interface is a method)
- Client Server Interface (If the service interface is event)

The above defined interfaces are mapped to client server or sender receiver port through which the software components communicate the data contained in the functionality of the service to the outside world. A service can contain methods or events as per SOME/IP protocol. Here the first service for the use case proposed is shown in Figure 4.2

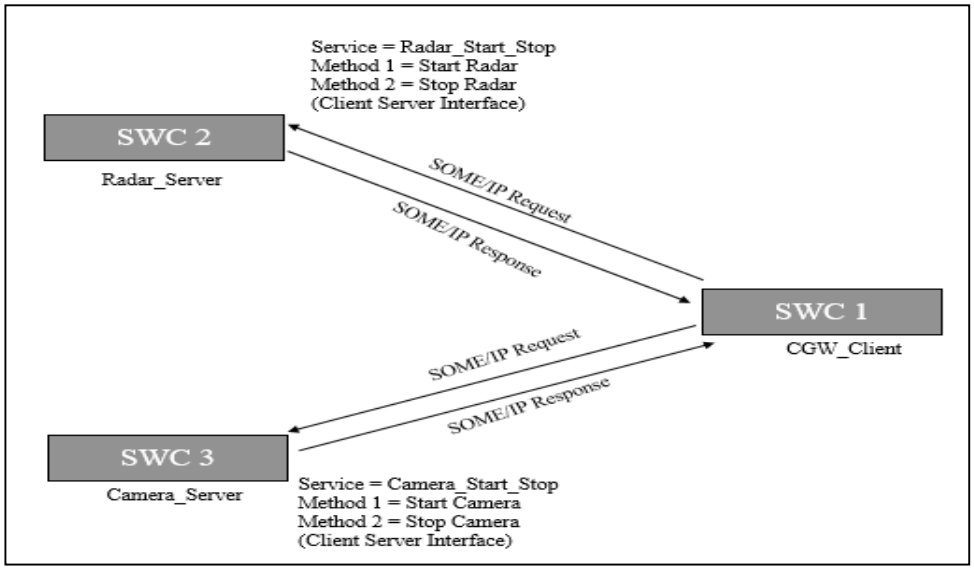

Figure 33 Service (A)

In the first service, the request/response call is generated by the CGW client to start and stop the camera and the Radar_server, Camera_Server components gives back the acknowledgement in response. The SWC1 acts as client which requests the operation on SWC2 and SWC4 server.

The second service is shown in figure 4.3

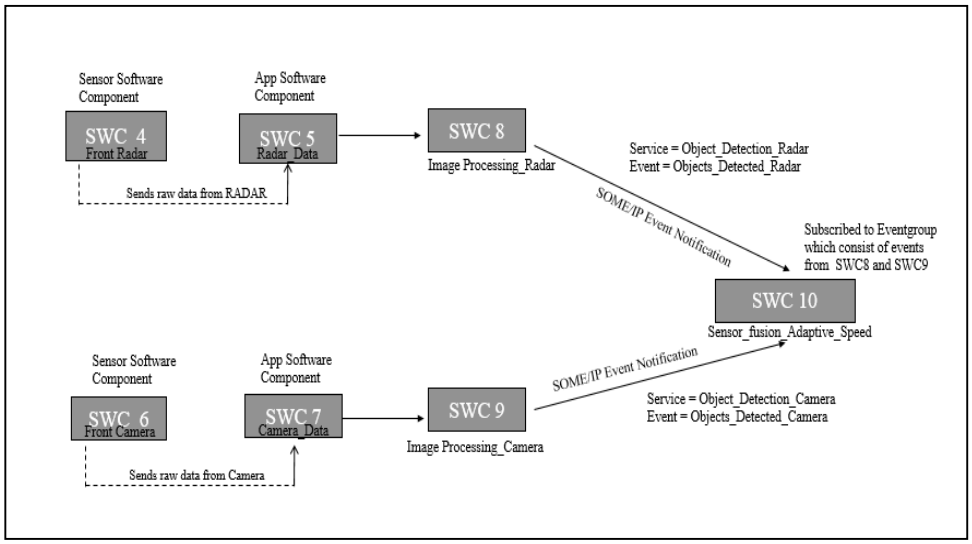

Figure 34 Service (B)

Another service is provided by the camera and radar. After processing the data from camera and radar, the SWC8 and SWC9 will send event notifications to SWC10 whenever the objects are detected by each of them. The notifications are sent when a consumer has subscribed to the service, they are sent on sender receiver interface and are mapped to sender port and receiver port as per AUTOSAR.

The next service defined for the use case is ACC which consists of event and sends notification of the adaptive vehicle speed after sensor fusion of received processed data from radar and camera to the subscribed consumer. It is also shared on sender receiver interface as shown in Figure 4.4

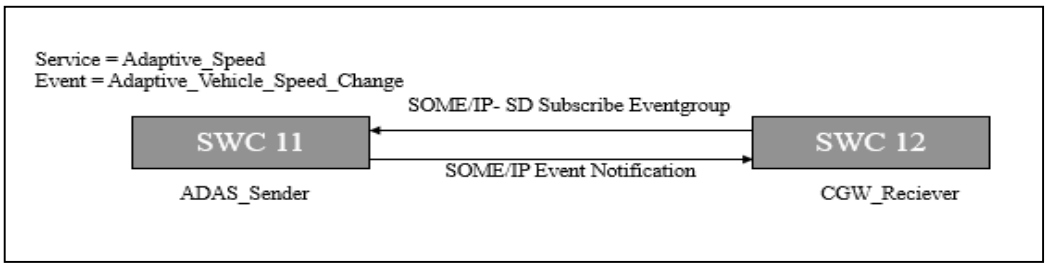

Figure 35 Service (C)

4.2 COMMUNICATION DESIGN

Since the above the defined services will be communicating on the network via Ethernet protocol, the communication endpoints such as MAC address, IP address and TCP/UDP ports have to be defined prior to its implementation in the software.

The below sequence diagram shows the chosen network and application endpoints for the defined services.

The service consisting of method to start radar is shown in below sequence diagram where the MAC address, IP address, UDP port is defined for each ECU. The same sequence goes for the stop radar method of the same service. Both will differ in their method ID's.

The same sequence is valid for the method start and stop camera. The consuming port, MAC address, IP address of the Camera ECU will change.

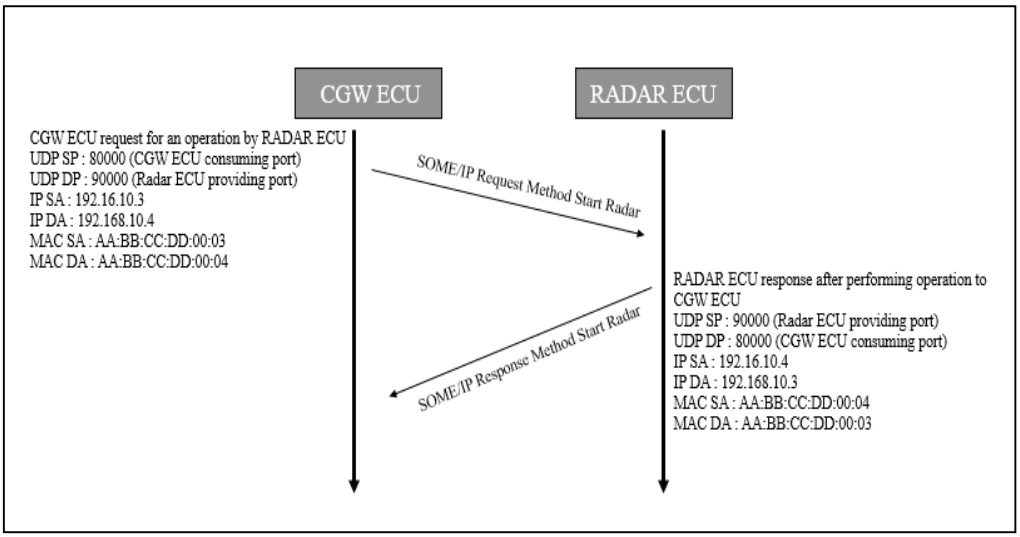

Figure 36 Sequence Diagram (a)

The next sequence diagram shows how the event notification is sent from one ECU to another. It is to be noted that the ECU must have subscribed to the event in order to get the notifications whenever the

event occurs which is shown in the sequence diagram for reference but will not be part of implementation.

The service consists of event notifications from Radar ECU sent to ADAS ECU and provides the data when the objects are detected is shown in below sequence diagram where the MAC address, IP address, UDP port is defined for each ECU. The same sequence goes for the camera event notifications and adaptive vehicle speed event notification but with different service ID's and Event ID's.

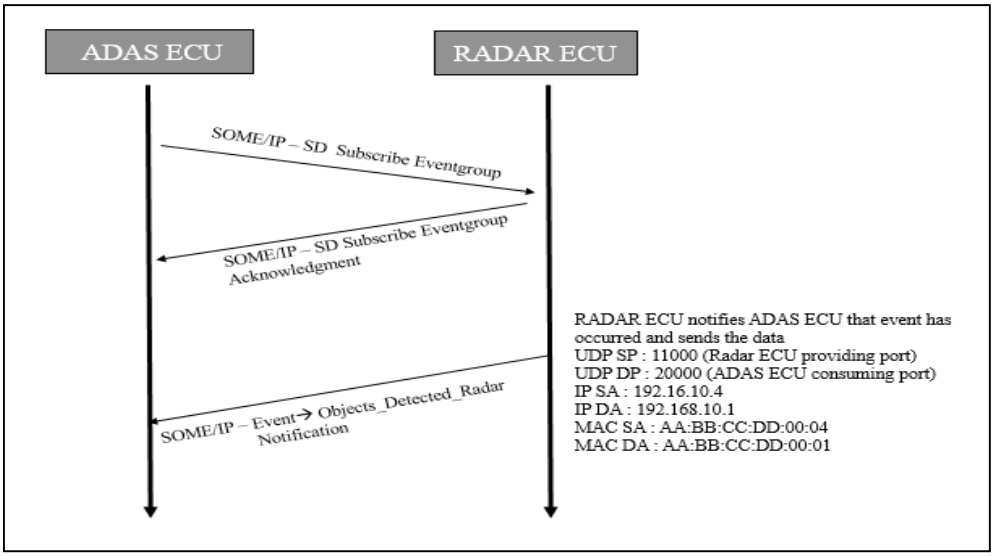

Figure 37 Sequence Diagram (b)

For the diagnostics over the IP (DOIP) for each Ethernet node viz. ADAS, CAMERA, RADAR, CGW the below sequence diagram will be followed. On TCP the actual data and routing activation will happen and on UDP, vehicle discovery will happen as shown in Figure 4.7

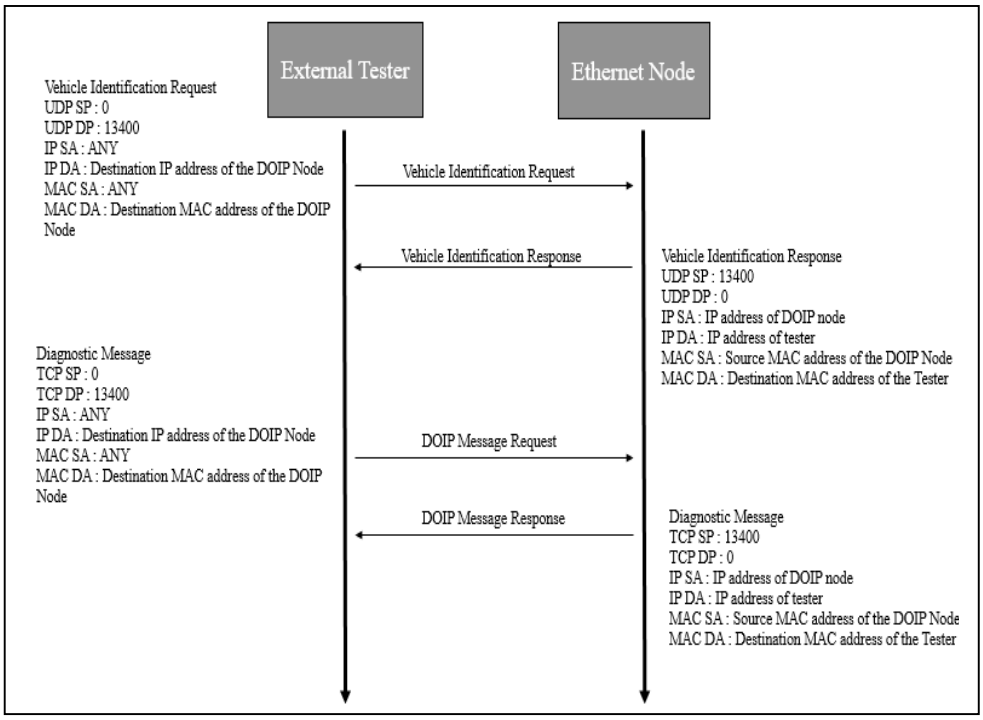

Figure 38 Sequence Diagram (c)

4.3 ECU AND SWITCH DESIGN

After the definition of services and the communication design, the ECU's and switch design needs to be taken care off. In this for a valid network topology, the necessary connections required for communication to take place between ECU's via switch is shown.

The artifacts that have to be modelled, depends on the focus of the design. In this part of the chapter our focus is on the following two:

- ECU in a switched Ethernet design
- ECU with an internal switch

The ECU's which are connected to the switch namely CGW, ADAS, CAMERA, RADAR etc. should have the connections shown in Figure 4.8 below. In order to communicate in the network. All the ECU's are created in a component package. Each ECU has a bus connector which is connected to a bus system. Each ECU connector has a bus interface and a controller configuration which shows its connection to the bus system. The bus system represent the physical channel and in our use case it will of Ethernet type.

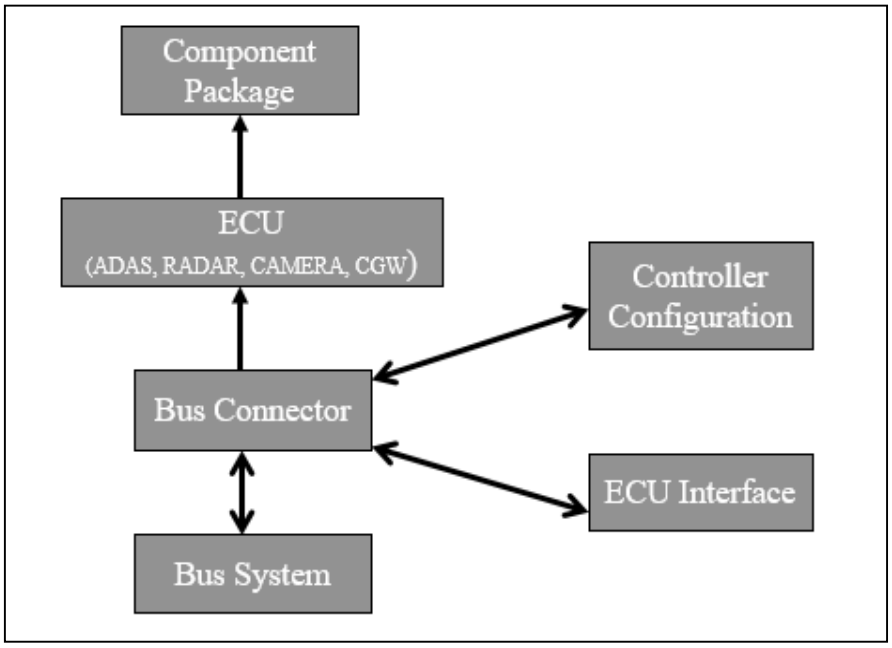

Figure 39 ECU Design

The switch in a switched Ethernet network should have the connections for a valid switch model as shown in Figure 4.9 below. In our model, the switch will be an internal part of CGW ECU. The switch should have bus connectors similar to ECU's and are mapped to a relevant bus system. A switch is a coupling element and it has coupling ports which are needed to connect them to others ECU's or coupling elements.

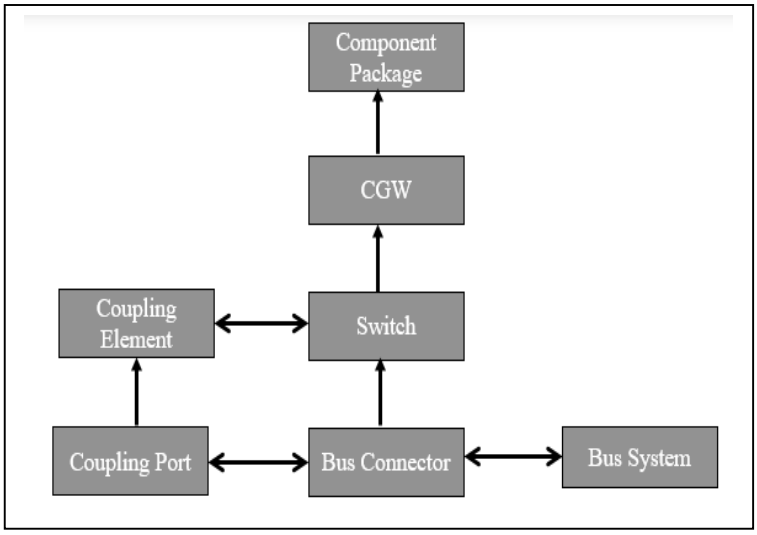

Figure 40 ECU with Internal Switch Design

CHAPTER 5
SOA DESIGN AND IMPLEMENTATION

5.1 SOFTWARE REQUIREMENTS

PREEvision 9.0

- Operating System required → Windows 7, 8, 8.1, 10 (64 bit)
- RAM → 8 GB minimum , 32 GB (recommended)
- Hard Disk → 16 GB minimum should be available on Machine

5.1.1 PREEvision 9.0

The tool PREEvision supports various features to design Service Oriented Communication in Automotive Domain such as:

- AUTOSAR Classic 4.3.0 service oriented design
- Automotive Ethernet switched networks
- SOME/IP Protocol
- ARXML import/export

The above listed features were required for the implementation of SOA for the proposed Automotive Adaptive Cruise Control use case.

Also for diagnostic communication over IP the features provided in PREEvision are:

- Defining Tester's and DUT's
- Assigning gateway functionality
- Support of DOIP on UDS (ISO 13400)

5.2 SOA WORKFLOW

This section demonstrates the steps followed to model SOA for ACC in PREEvision. Each step creates the communication parameters which are AUTOSAR compliant [36].

Layers in PREEvision for E/E architecture:

1. Product Goals
2. Logical Function Architecture
3. Wiring Harness
4. System Software/Service Architecture compliant with AUTOSAR
5. Hardware Architecture
6. Communication

The defined SOA for ACC use case is based on Software Architecture, Hardware Architecture and communication layers.

The service oriented architecture (SOA) workflow is shown in Figure 5.1 The design starts with defining the services with their service interfaces based on the defined use case and are deployed to SOME/IP as the transport protocol for Ethernet. The software components are created for each service (provider and consumer) and the service interface is mapped to software interface which can be a client server or a sender receiver interface. The interface is mapped to port of the SWC's, the ports can be required or provided ports which actually sends the data contained in functionality of the SWC. In parallel the hardware architecture is defined which comprises of ECU's, switches, physical Bus systems etc.

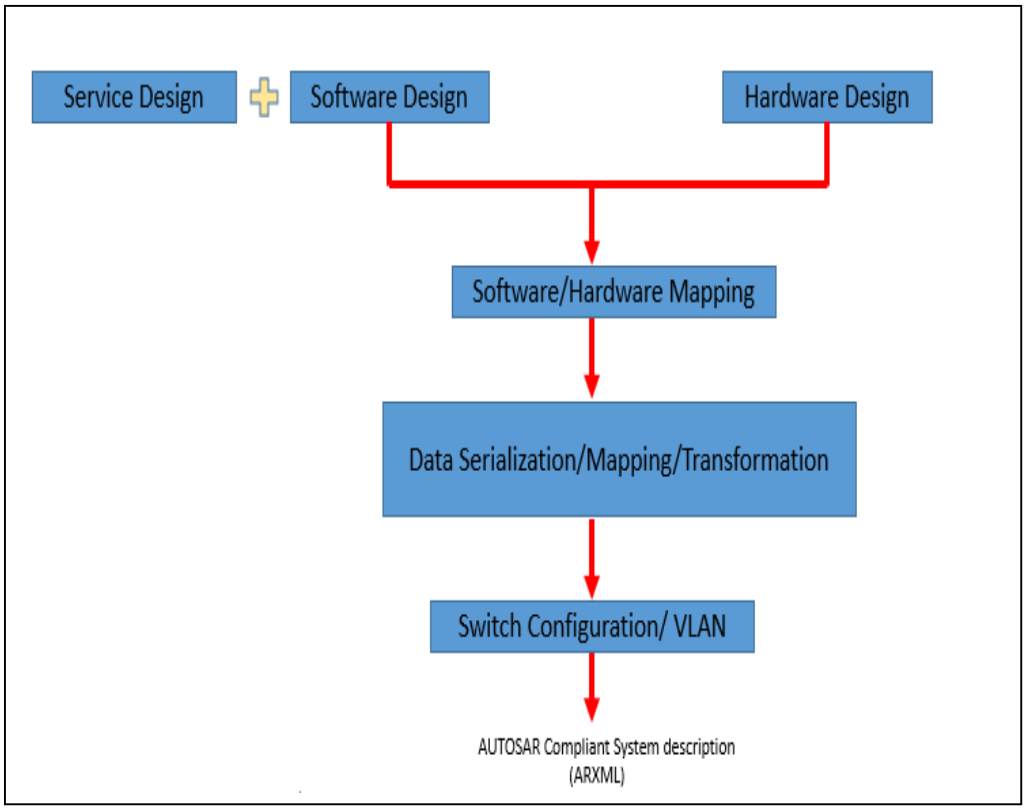

Figure 41 SOA Workflow

Once the above steps are implemented, the software hardware mapping is done. The sockets (IP address + UDP/TCP port) are defined for ECU's and the service producers and consumers are mapped to them. The data serialization is a step to serialise the data coming from the application layer which is done by the SOME/IP transformer which is required to convert the data into a bit stream for Ethernet Communication. Also the switch configuration is done since the Ethernet nodes are point to point connections connected via a switch. VLAN's can be configured in the switch for secure communication and to keep data segregated within its own network. The signal router then finds paths through the network and synthesize Ethernet relevant communication artifacts such as signals, PDU's etc. After the design and implementation is completed, an ARXML file is exported.

5.3 SOA IMPLEMENTATION

PREEvision supports SOA and Ethernet explorer to implement the use case defined in Chapter 4. The design and implementation starts with creating a product line which would later contain the Software Architecture, Hardware Architecture and communication packages for the proposed Architecture.

5.3.1 Service Design & Implementation

The software architecture starts with defining services with service ID which is of 16 bits as per SOME/IP protocol header as shown in Figure 5.2

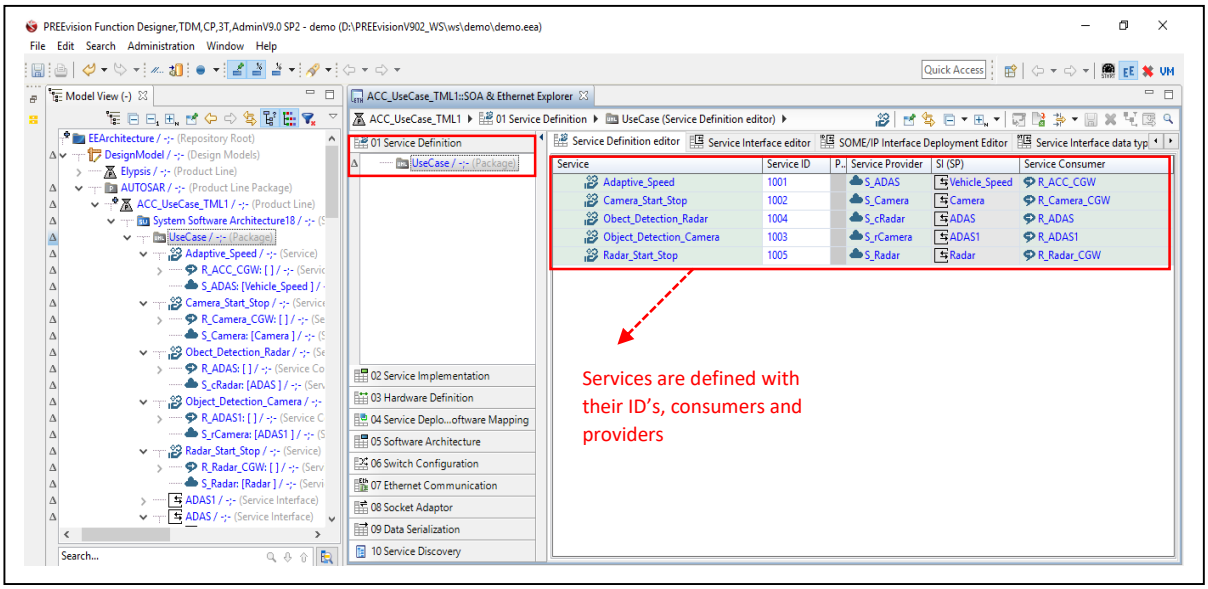

Figure 42 Service Design

The next step is to define the service interface for each service which consists of methods and events e.g. Object_Detection_Camera is an event as shown in Figure 5.3

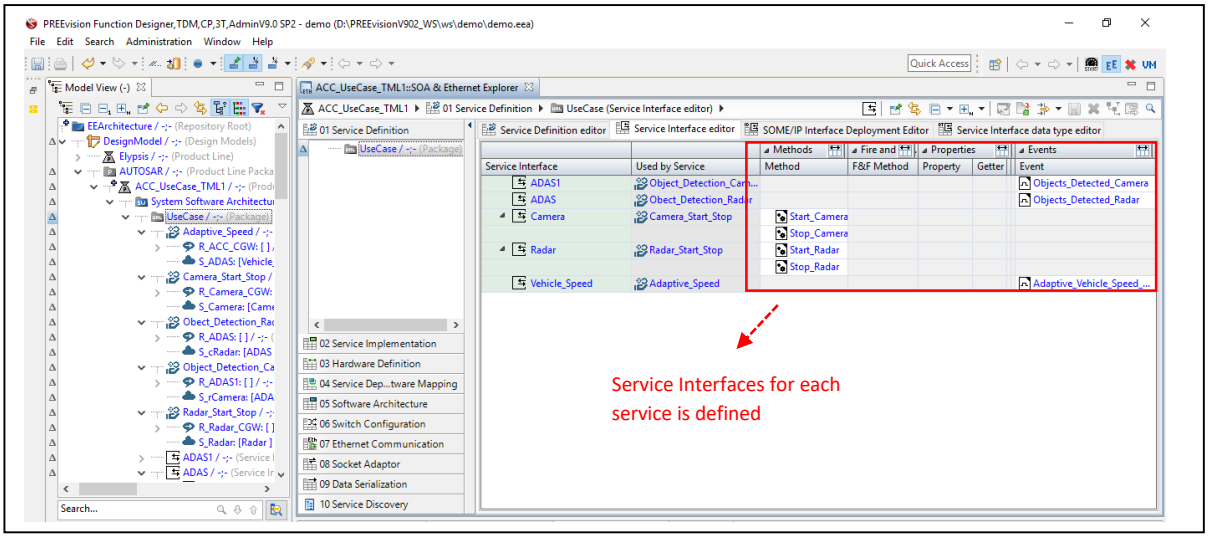

Figure 43 Service Interface

The above defined services are deployed on SOME/IP as the protocol for Ethernet Communication which is synthesized by the tool itself and is created under the software deployment package as shown in Figure 5.4

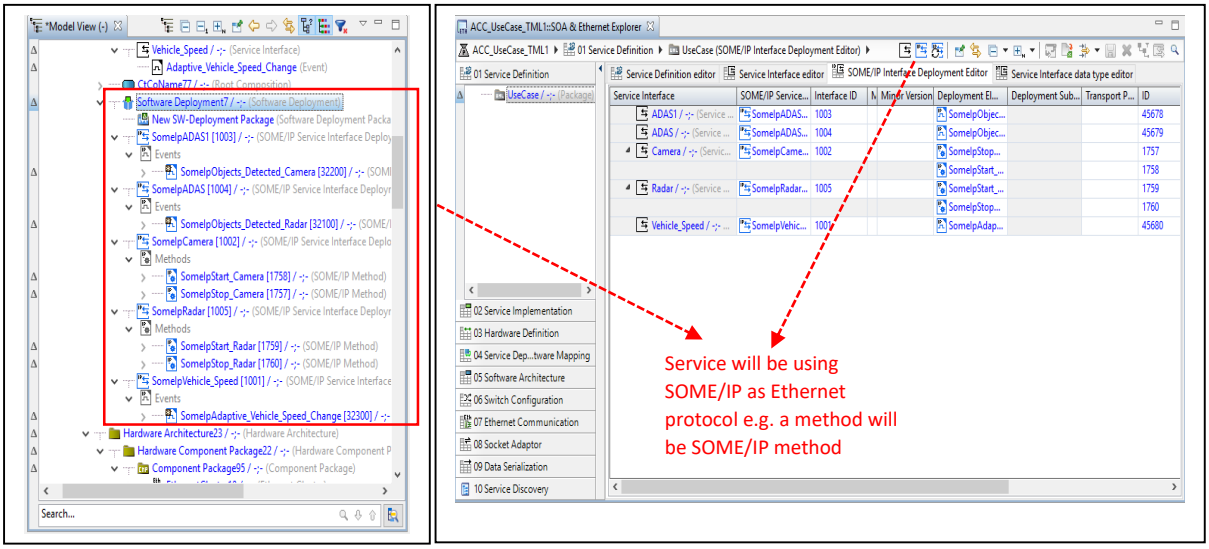

Figure 44 SOME/IP Deployment

Once the services with their interface were defined, the tool has generated software components (SWC's) for each service provider and consumer of the service. The service interfaces are mapped to software interfaces as per the AUTOSAR classic technology shown in Figure 5.5 Application software component type with their prototypes are created having required ports and provided ports on which sender receiver and client server interfaces are mapped. The software component types and its prototypes are created in the library of the product line.

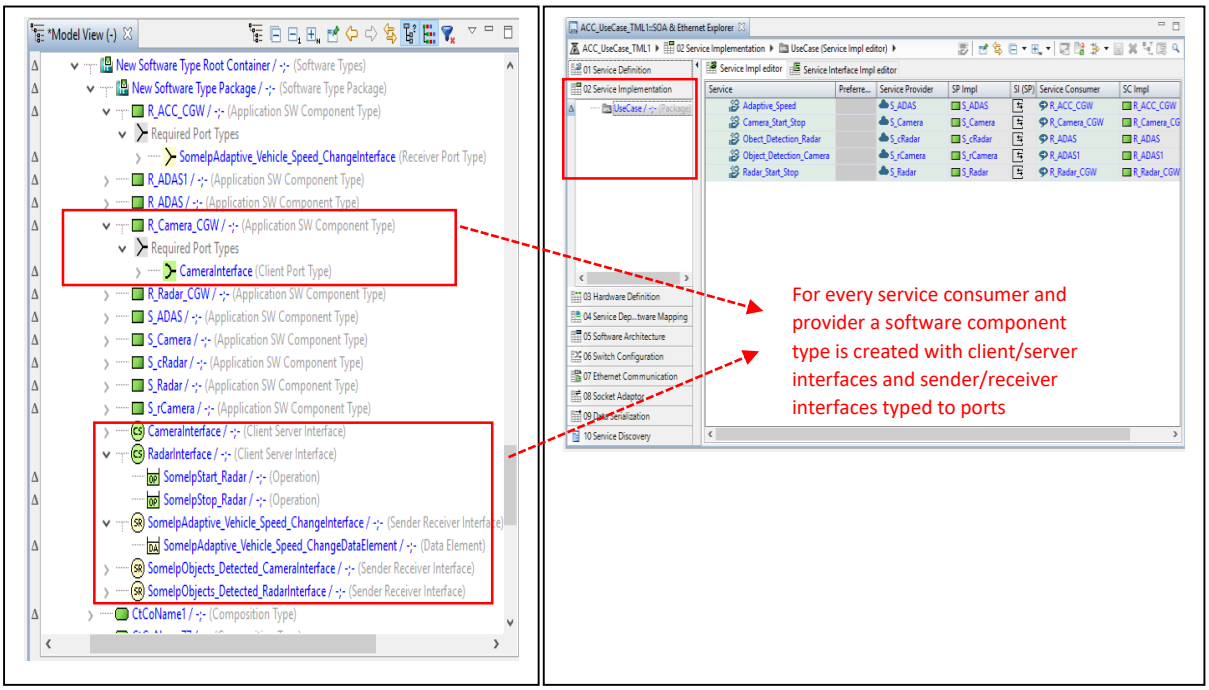

Figure 45 Service Implementation

5.3.2 Hardware Definition

In the hardware definition, the ECU's with their processing units are created. A switch is also created which facilitates the communication between different Ethernet nodes/ ECU's. They are created in the hardware architecture package of the product line and can be seen in the hardware definition tab of the SOA and Ethernet explorer shown in Figure 5.6

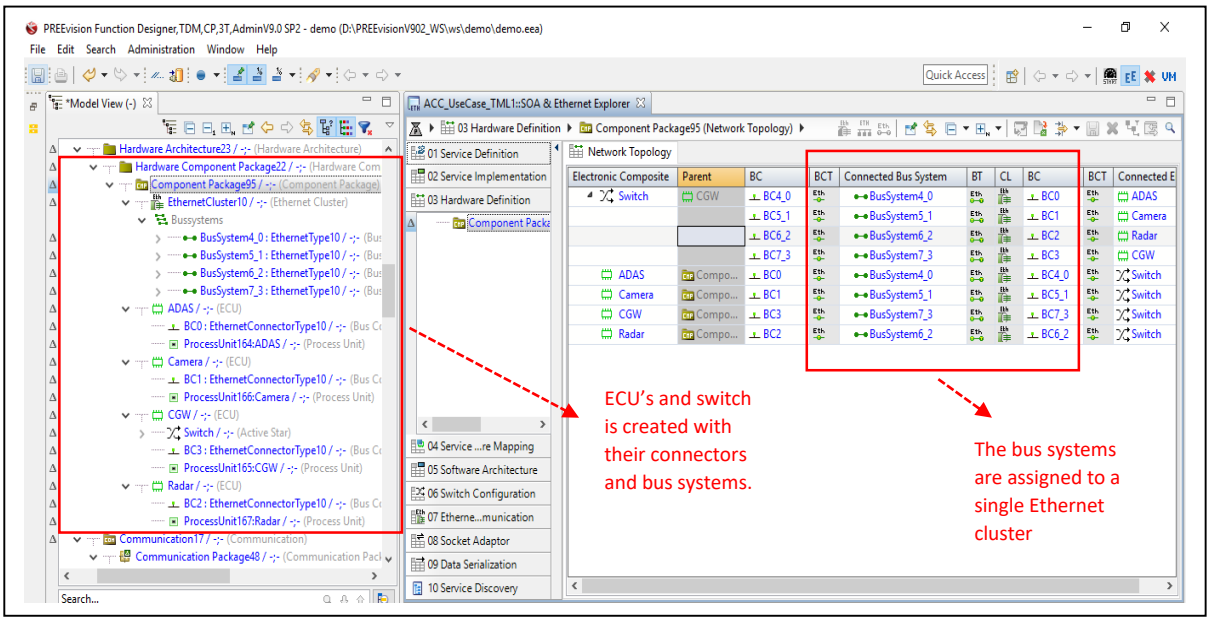

Figure 46 Hardware Definition

Each ECU is assigned a bus connector and connected to one of the connector of switch via a bus system. A SOA and Ethernet explorer helps to assign the bus systems under one Ethernet cluster. A network diagram is created which shows how the ECU's are connected via a switch as shown in Figure 5.7

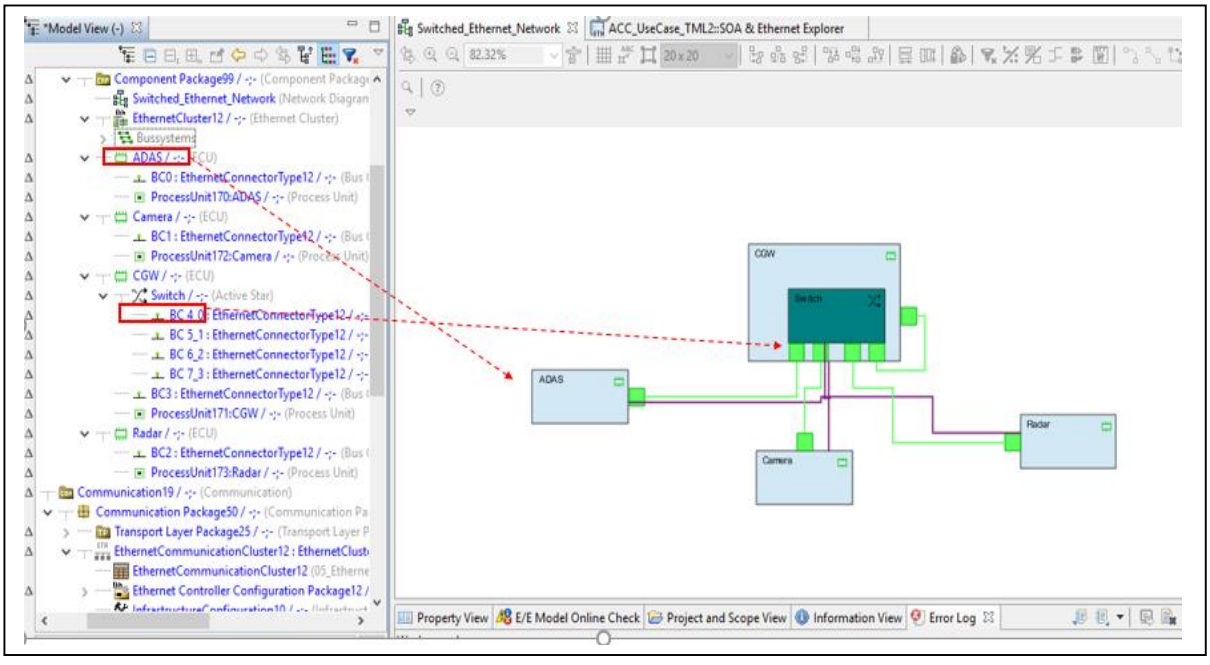

Figure 47 Network Diagram

5.3.3 Software/Hardware Mapping

Once the software architecture and hardware architecture is defined in the product line. The service providers and consumers of the services are mapped to the ECU's where they will actually perform the functionality as shown in Figure 5.8

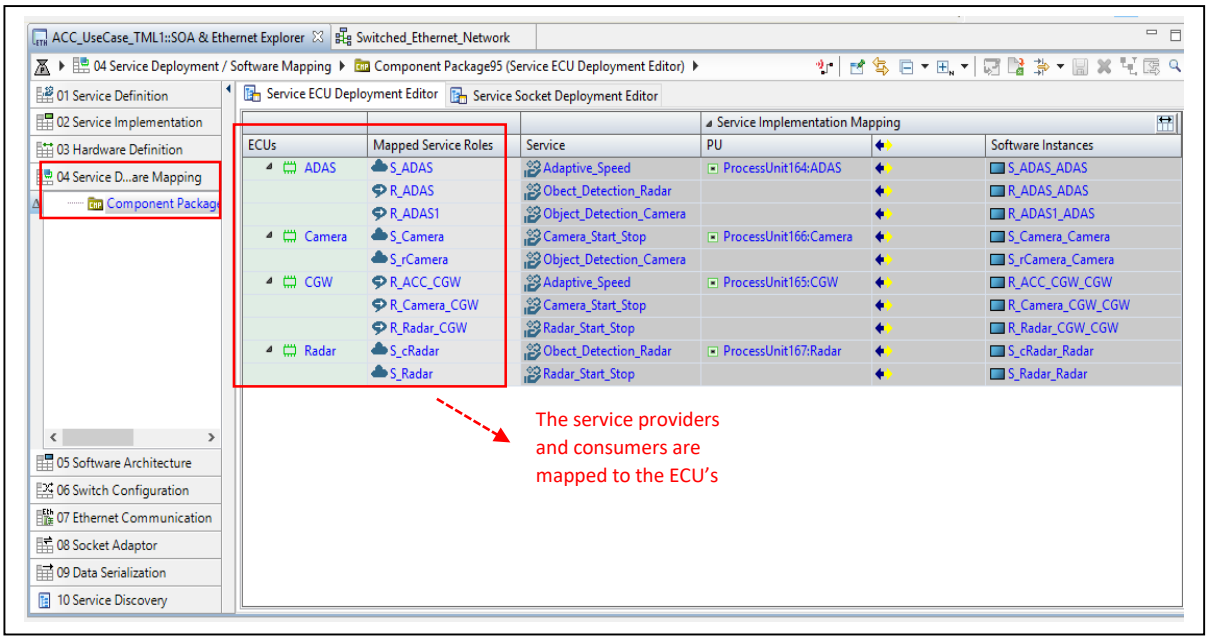

Figure 48 Software/Hardware Mapping

After the mapping, the provided (Server/Sender) and required (Client/Receiver) ports of the software components are connected for each service. A provided or required port which is mapped to a client/server or a sender/receiver interface are connected once the software/hardware mapping is done. The SOA and Ethernet explorer provides a software architecture tab to make the required connections as shown in Figure 5.9

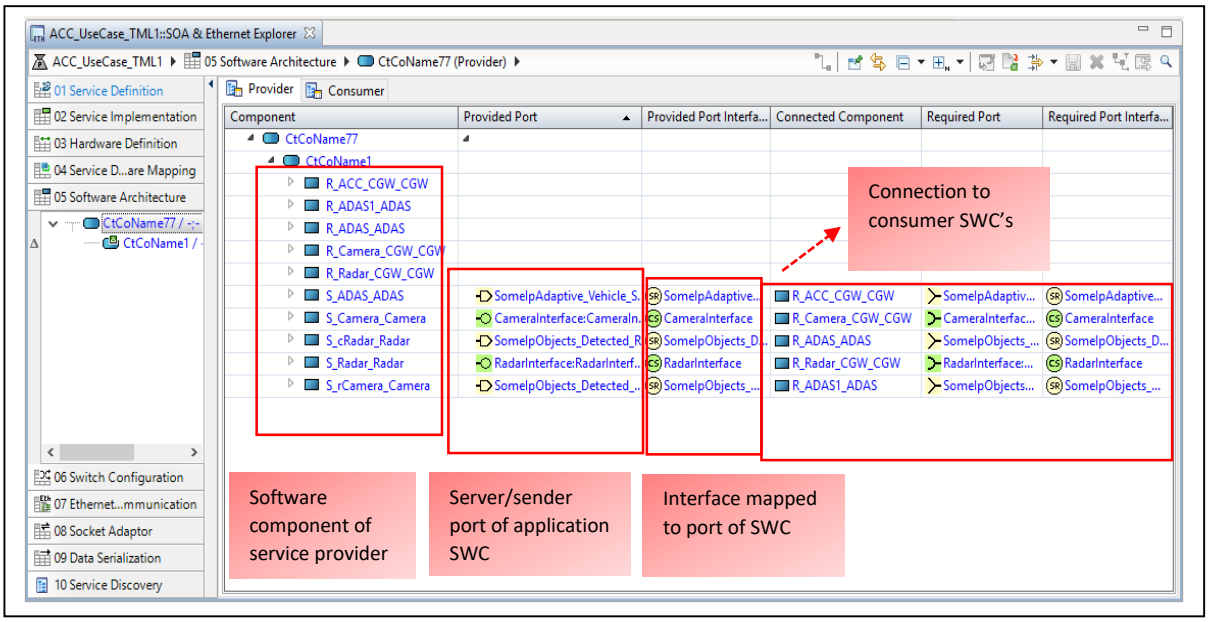

Figure 49 Software Architecture

5.3.4 Switch Configuration and Ethernet Communication

The virtual Lan (VLAN) is configured in the switch. The VLAN's are assigned on port by port basis. Since the proposed use case does not require different VLAN's we have set the default VLAN at each port of the switch for both ingress and egress traffic. For ingress traffic default VLAN is created at each port i.e. if the data is untagged it belongs to default VLAN. Also for egress traffic VLAN membership is created for default VLAN and the option send-untagged data is set as shown in Figure 5.10

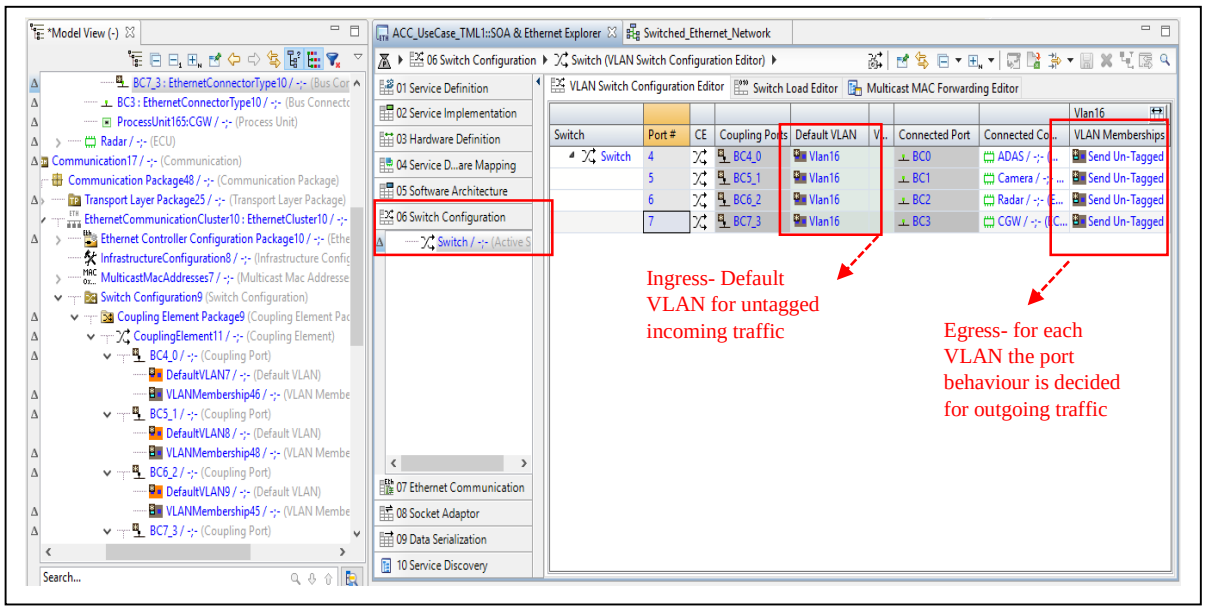

Figure 50 Switch Configuration

In the next text layer the artifacts related to OSI Layer 1 (Physical Layer) and OSI layer 2 (Data Link Layer) are created. The data is transmitted on 100 BASE T1 physical layer and the MAC addresses for each controller is defined for Ethernet communication as shown in Figure 5.11

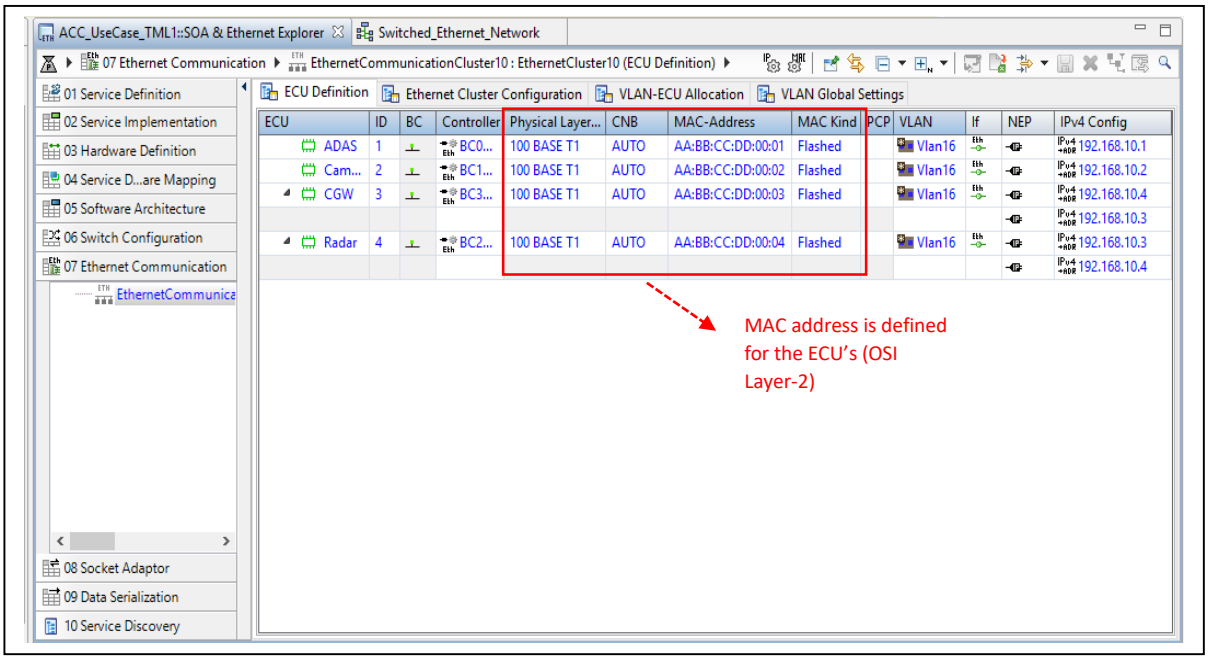

Figure 51 Ethernet Communication

Similarly artifacts for OSI Layer 3 (Network Layer) and OSI Layer 4 (Transport Layer) are defined. The private IPV4 addresses are defined for each ECU interface which makes the network endpoint. Also the UDP ports are defined which makes the application endpoint. Together the IP address and UDP port constitutes a socket address on which the data is transmitted. This is shown in Figure 5.12

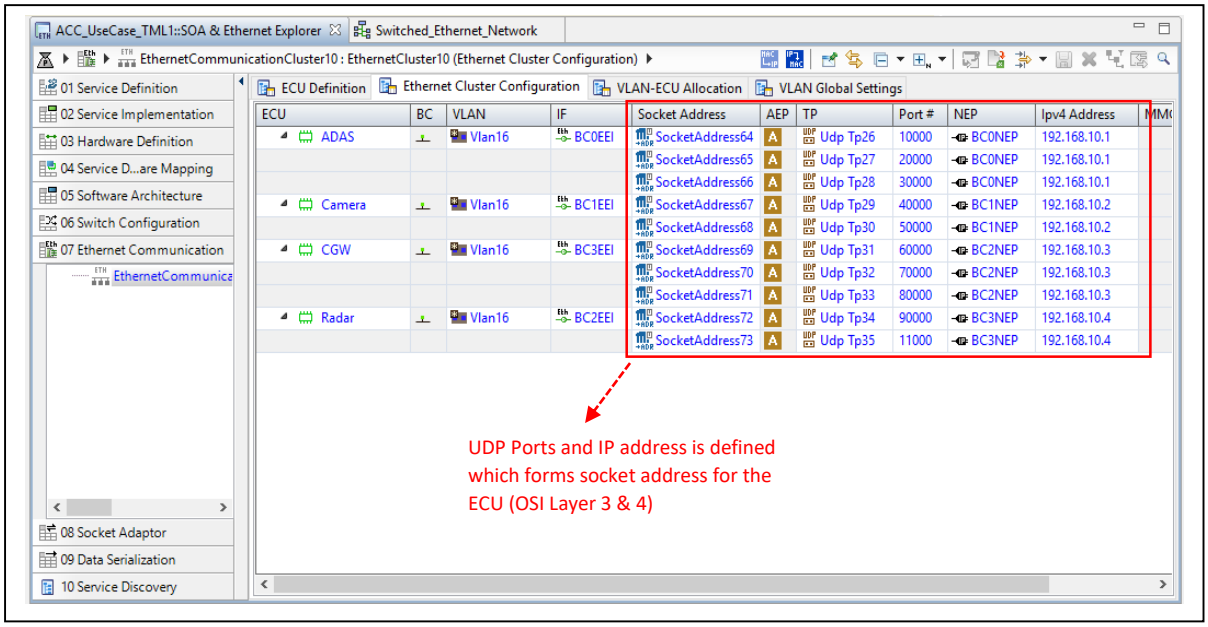

Figure 52 AEP creation

In the next step the VLAN ID's is created which was earlier used in switch configuration step. Here the VLAN's created are assigned VID which is of 12 bits. Each VLAN is assigned to different ECU's. Also the IP address allocation is done to be static and not by DHCP server. The IP subnet ID and network mask are also defined for private IP addresses used as shown in Figure 5.13

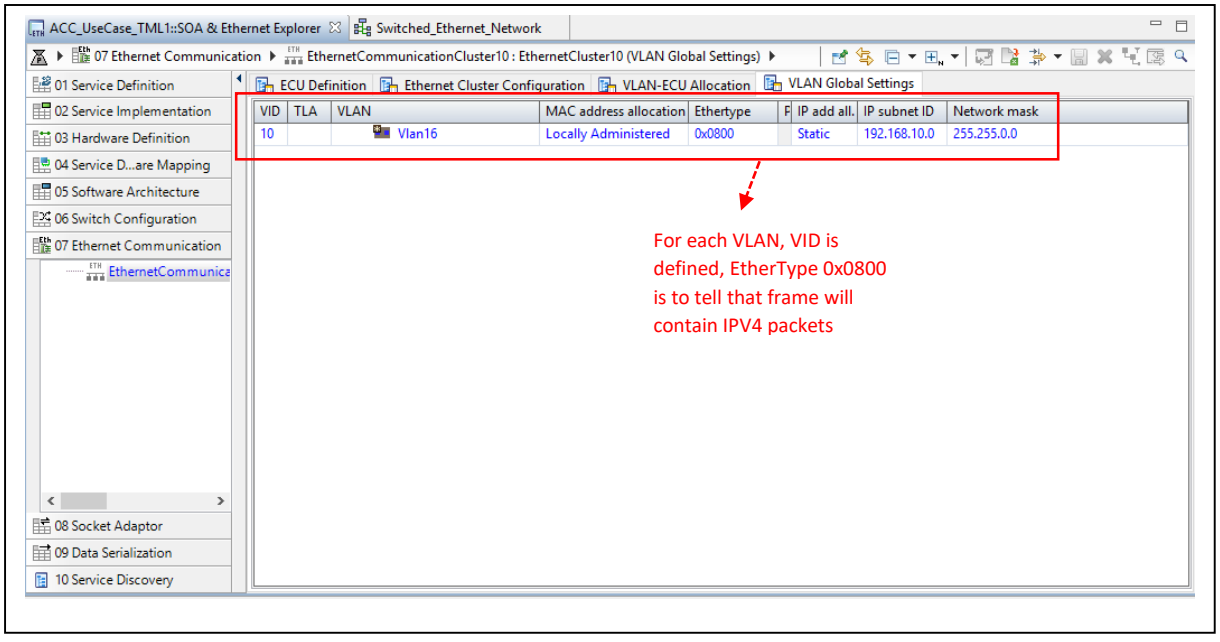

Figure 53 VLAN Definition

Once the artifacts for Ethernet communication such as VLAN, MAC, AEP's, NEP's are created. They are mapped to service providers and consumers in the service socket deployment editor. Since the services are mapped to ECU's in the earlier steps but they must be assigned socket addresses since Ethernet communication takes place through sockets. This mapping is shown in Figure 5.14

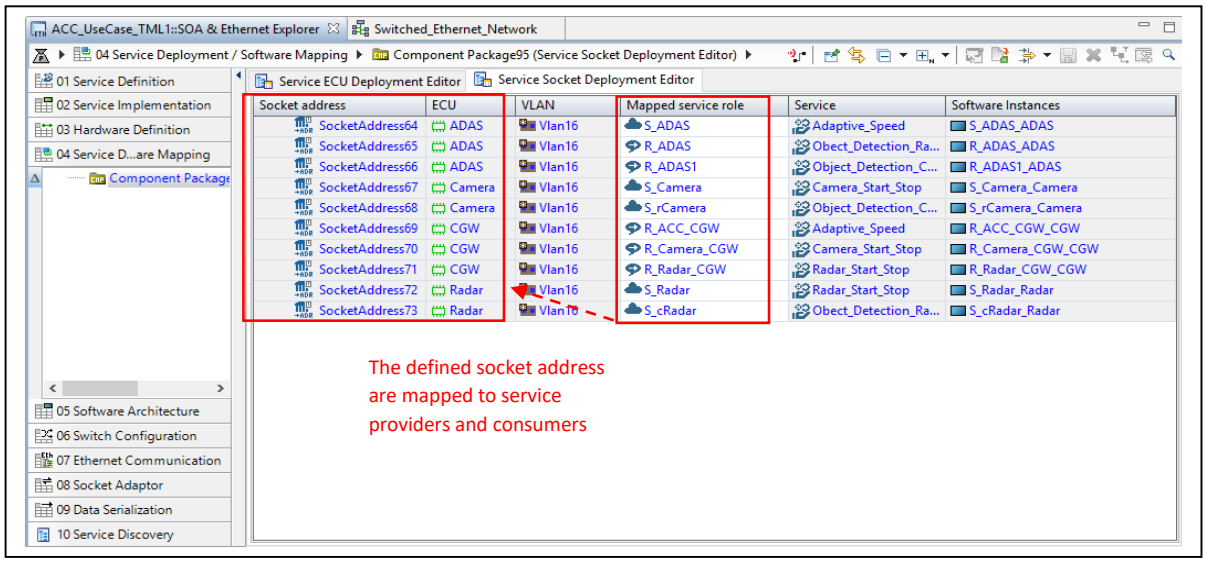

Figure 54 Service to Socket Mapping

5.3.5 Socket Adaptor and Data Serialization

The signal router is a mechanism which will find the best possible route/path through the given network. It will create required system signal mapping i.e. linking the data elements/operations to signals and their instantiation on the bus as signal transmission. Prerequisites for signal routing – system software architecture and hardware architecture should be defined. The process applies on the service providers which is shown in Figure 5.15

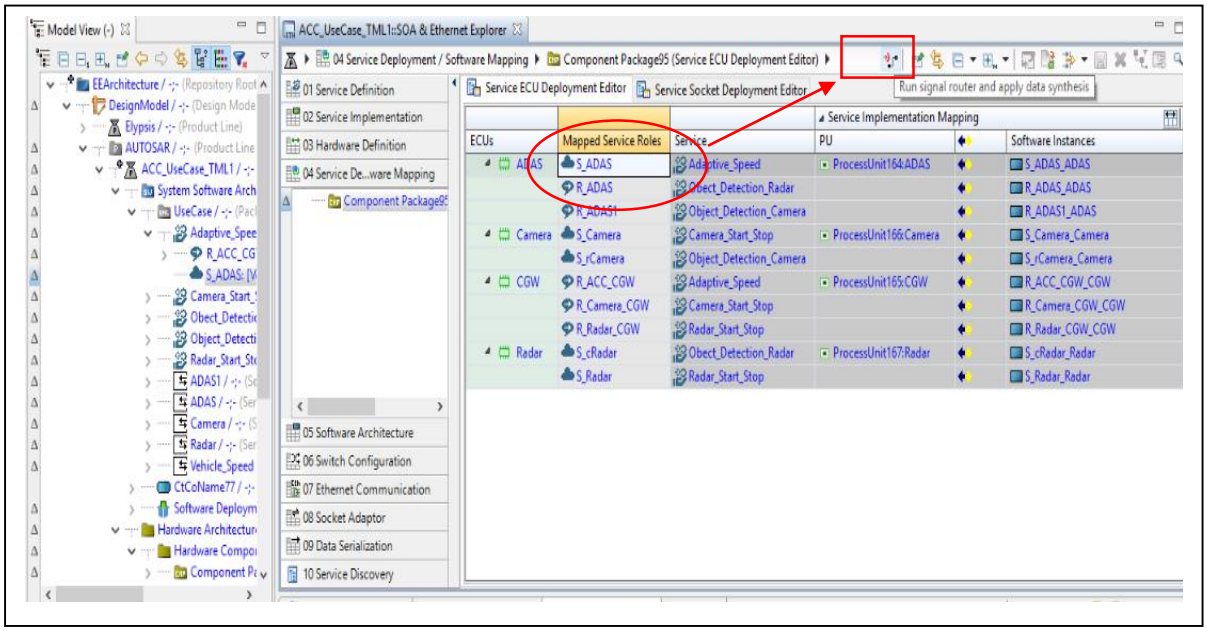

Figure 55 Signal Router

After the signal routing on each service provider the PDU's are generated which are mapped to respective sockets on the ECU's on which the PDU's will be transmitted and received. As per the TCP/IP stack of AUTOSAR, the mapping takes place in SOAD (socket adaptor) module. This mapping is shown in Figure 5.16

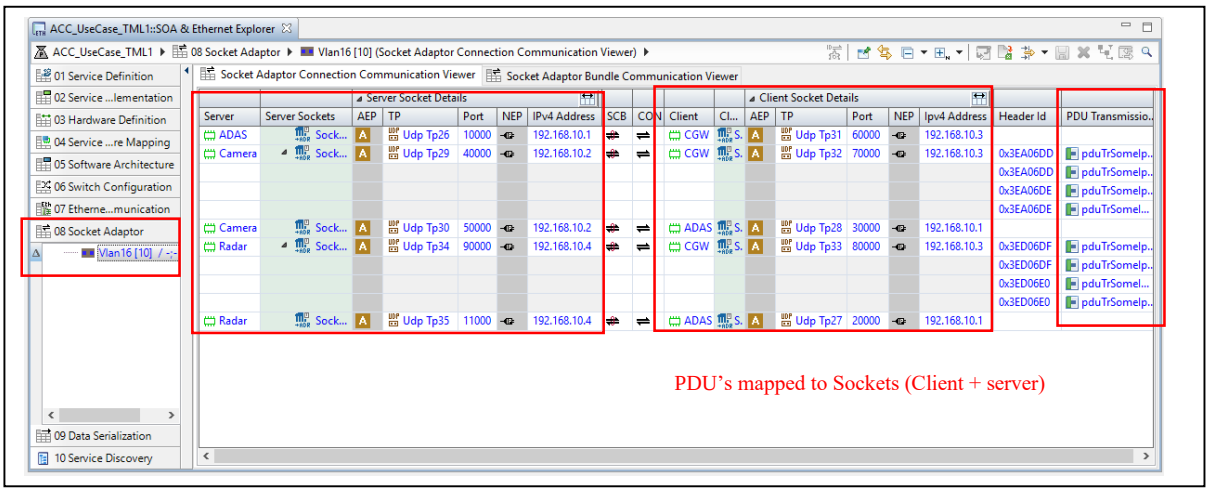

Figure 56 Socket Adaptor

The SOME/IP transformer is responsible for the serialization of data, it sends the serialized data to the RTE which sends the signals to COM module as per AUTOSAR specification. The SOME/IP transformer generates the SOME/IP header except the message ID and length such as message type for each service which consist of method request/response and event notification as shown in Figure 5.17

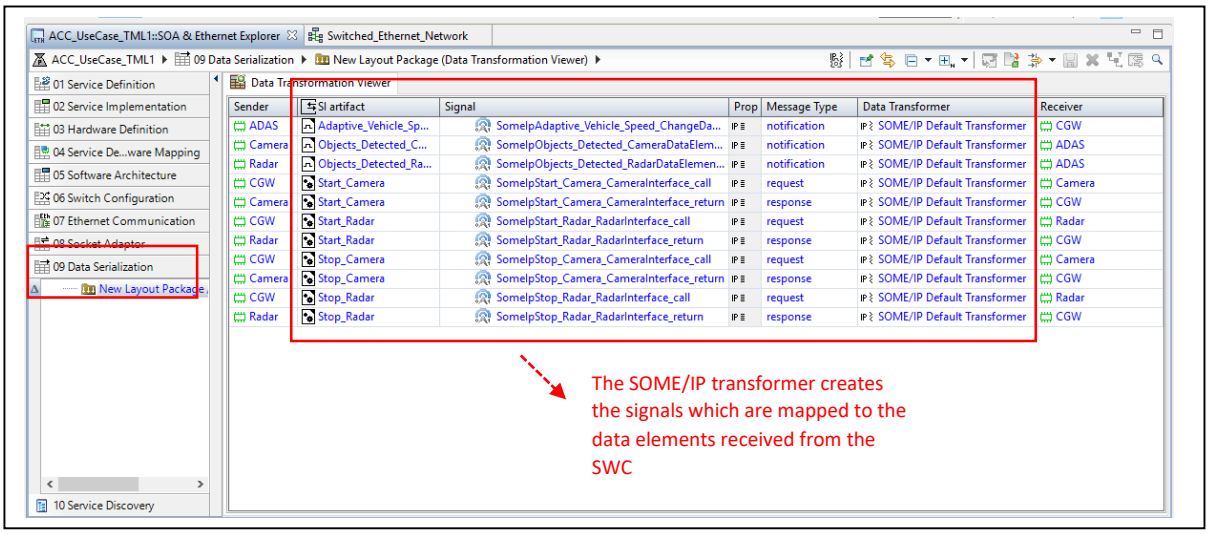

Figure 57 Data Serialization

After all these steps, the structure has been created where the data element from a sender port of the SWC is mapped to signal IPDU which is mapped to the signal (I-signal) and the system signal. When a consumer asks for a service, then the SWC containing the service provider will transmit the data element from the service and software interface of SWC to the RTE and RTE will create signals and transmit to the PDU router which will route the IPDU to socket adapter module so that the data is sent

through Ethernet channel. The software has checked for errors in the SOA design created as shown in Figure 5.18 and Figure 5.19 which were corrected at the end for the final export of system descriptions.

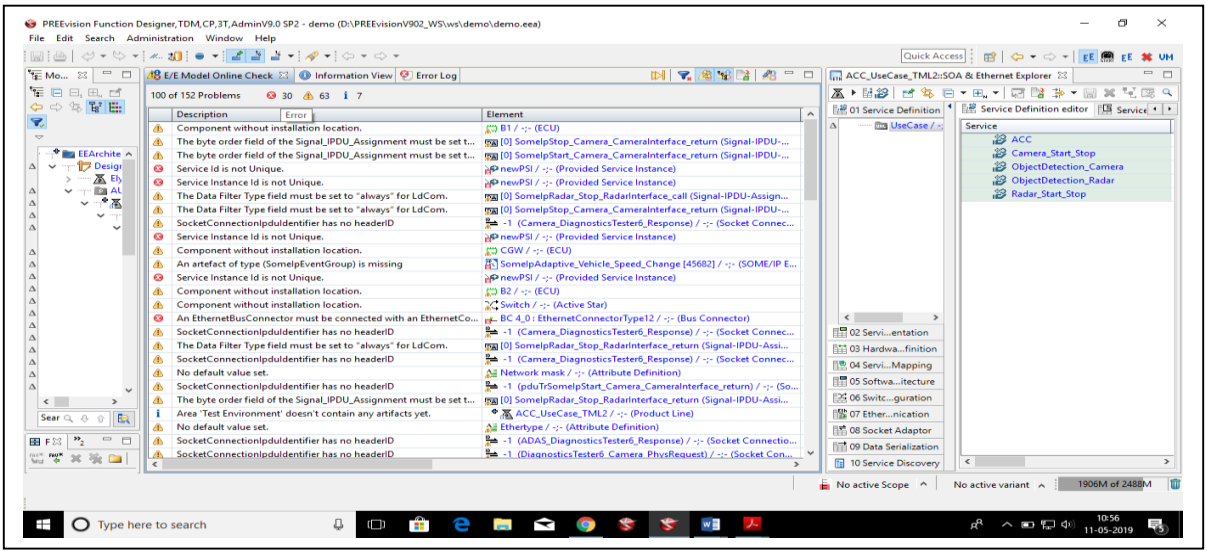

Figure 58 E/E Model check

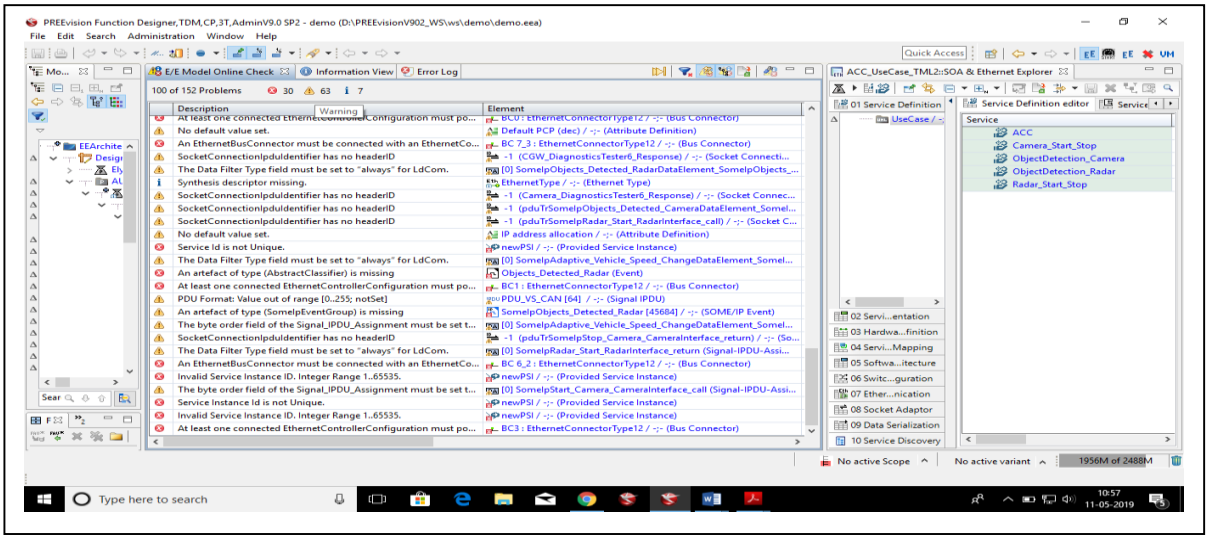

Figure 59 E/E Model check

5.4 DOIP WORKFLOW

For the above defined network topology in which the ECU's are connected to each other via Automotive Ethernet, the complex parameters such as sockets, socket connections and PDU's needed for DOIP messages can be defined in PREEvision [37].

The workflow for the DOIP is shown in Figure 5.20 This workflow starts with defining the network topology for E/E architecture. The network topology was defined in SOA implementation of ACC use case in the previous steps. This is necessary to implement DOIP in the software. The next step is to create an external tester and assign DUT's to it. Then communication path through the network defined

are found in software on which DOIP tester will communicate with DUT's. After selecting the desired path, the infrastructure for diagnostic communication to take place will be synthesized.

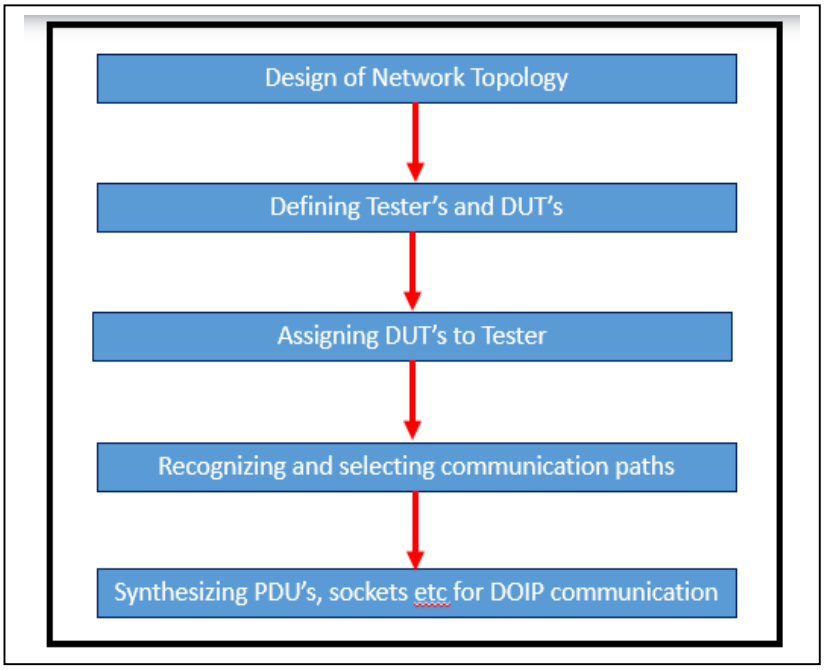

Figure 60 DOIP Workflow

5.5 DOIP IMPLEMENTATION

The above illustrated workflow is supported in PREEvision through Diagnostic Communication Explorer. The product line under which the use case is defined is opened with explorer where the workflow will be used for DOIP.

5.5.1 Network Topology with External Tester

The network topology defined for ACC which connected Ethernet nodes via switch will include a port connected to the external tester as shown in Figure 5.21

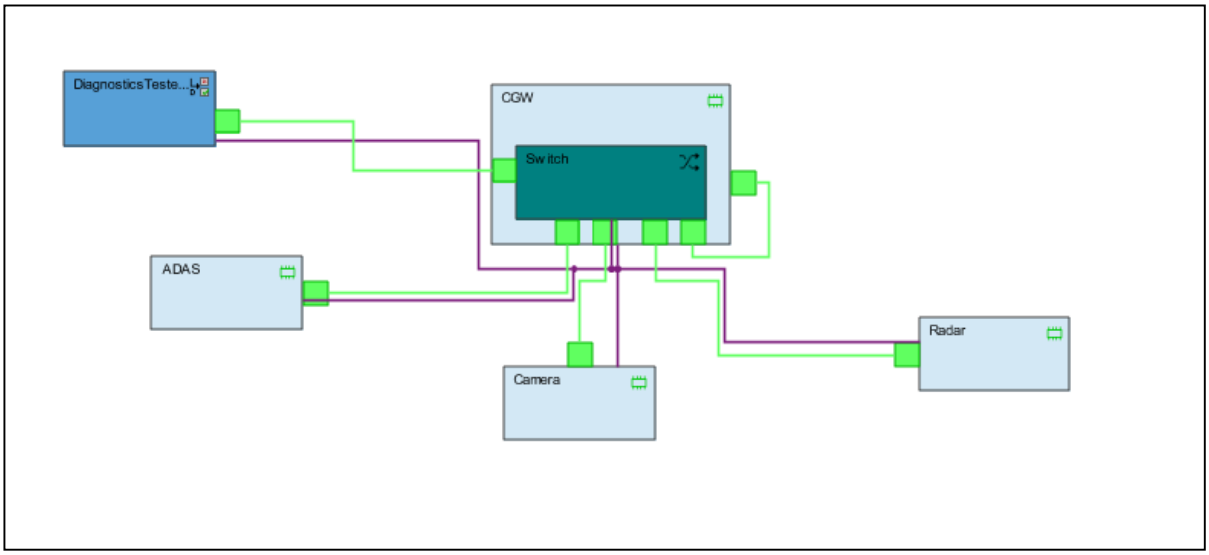

Figure 61 Network Topology

5.5.2 Assigning DUT's to External Diagnostic Tester

The components created in the hardware architecture can be assigned as external tester/internal tester and DUT's (Device under test). In our case, the diagnostic tester connected via a switch is assigned as an external tester and the CGW, Radar, Camera and ADAS ECU's are assigned as DUT's as shown in Figure 5.22

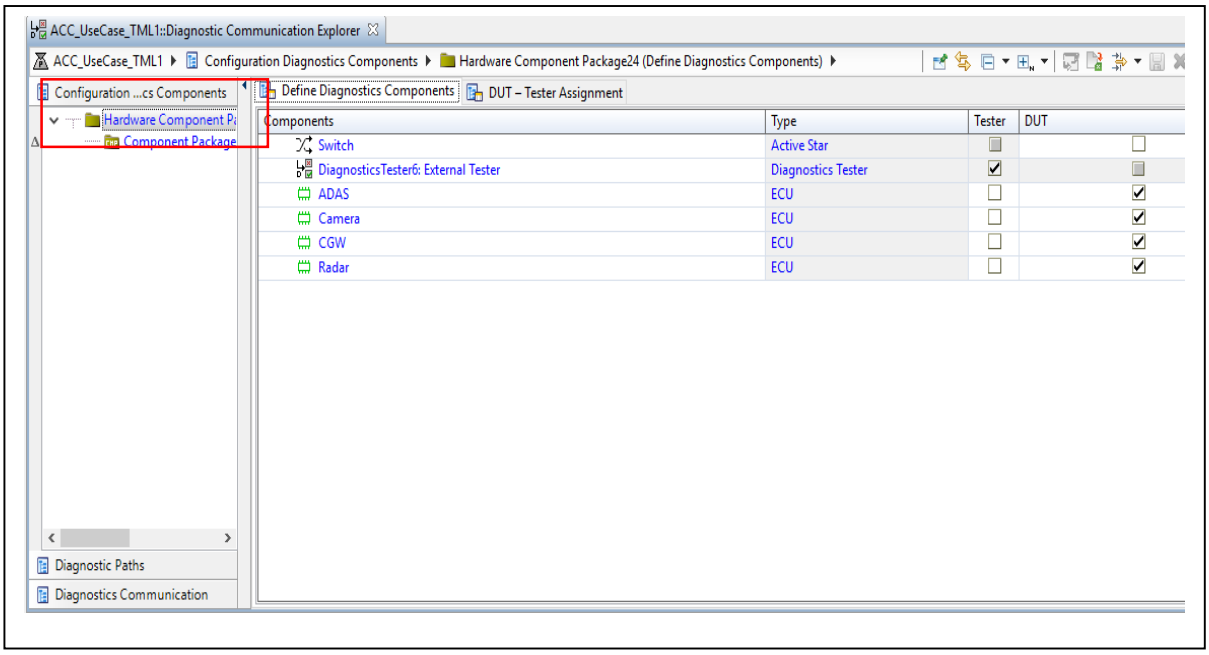

Figure 62 Diagnostic Components Definition

The next step is to assign whether a gateway is required to route the diagnostic messages to DUT's via a switch or the tester directly communicates with the DUT's. Here we have not assigned any gateway since that will provide a delay by routing the messages first to the gateway (CGW) and to the respective Ethernet nodes as shown in Figure 5.23 But when the DUT's are say on CAN network, the diagnostic information needs to pass through the gateway which has CAN interface.

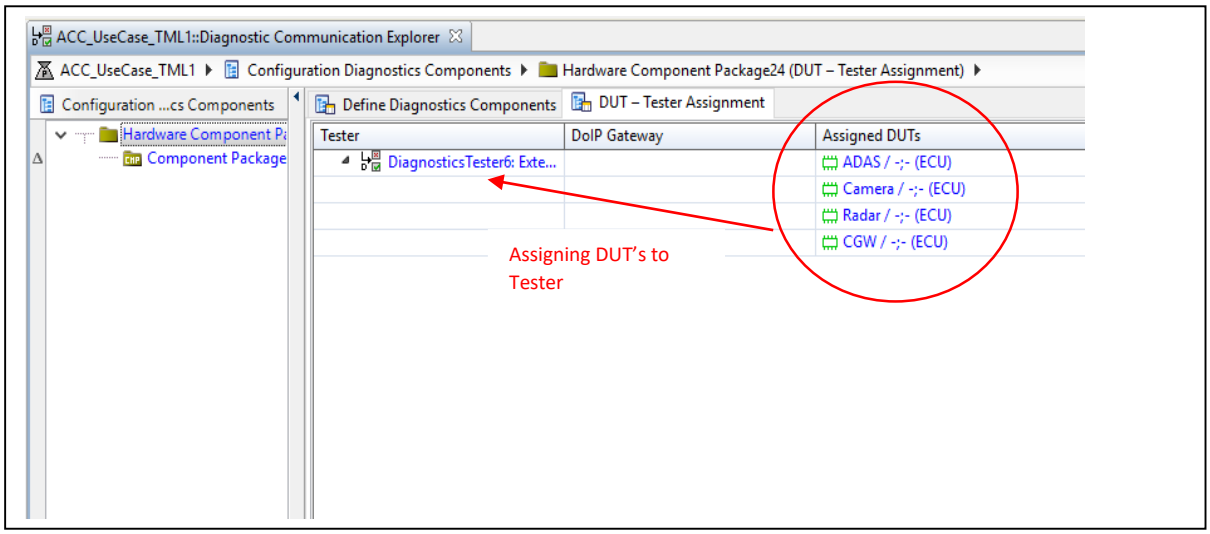

Figure 63 DUT-Tester Assignment

5.5.3 Recognizing Diagnostic Path

After the above steps, the software tool generates the possible communication path's through the network to send the diagnostic messages. If more than one path exist to diagnose the DUT in the network, we can select the one required. The communication path selected for the diagnostics of the DUT's of our use case is shown in Figure 5.24

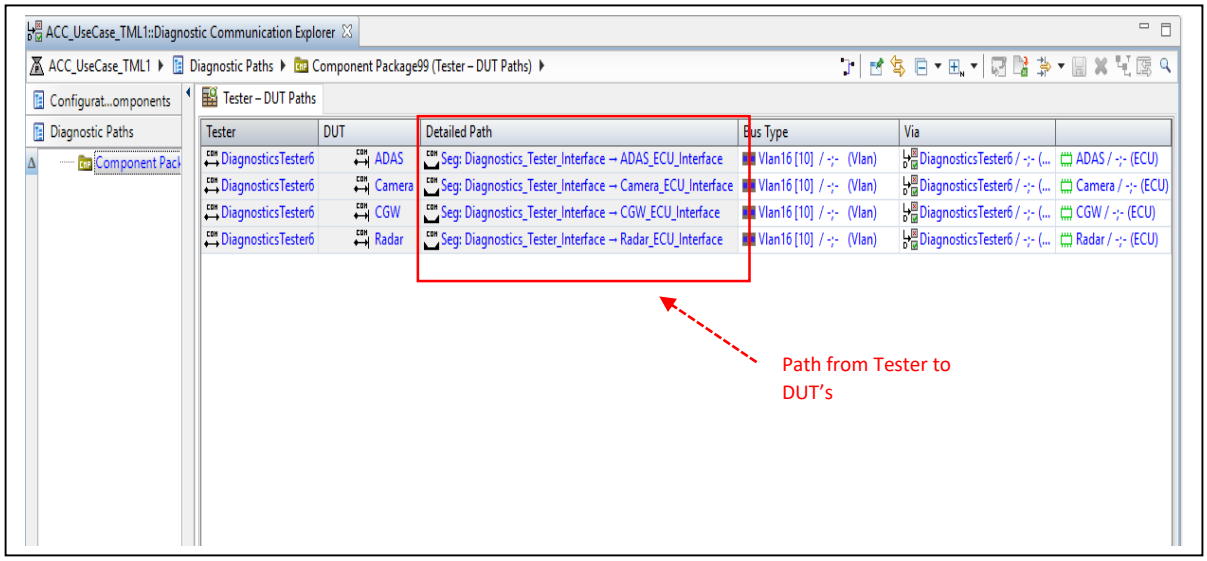

Figure 64 Tester-DUT Paths

After the path setup for the DOIP nodes (i.e. DUT's on Ethernet), the communication infrastructure required for the diagnostic communication are created such as sockets, DCM PDU's etc. As per the AUTOSAR communication stack the DCM IPDU's from DCM module are mapped to sockets for Ethernet Communication. The sockets can be UDP/TCP as shown in Figure 5.25

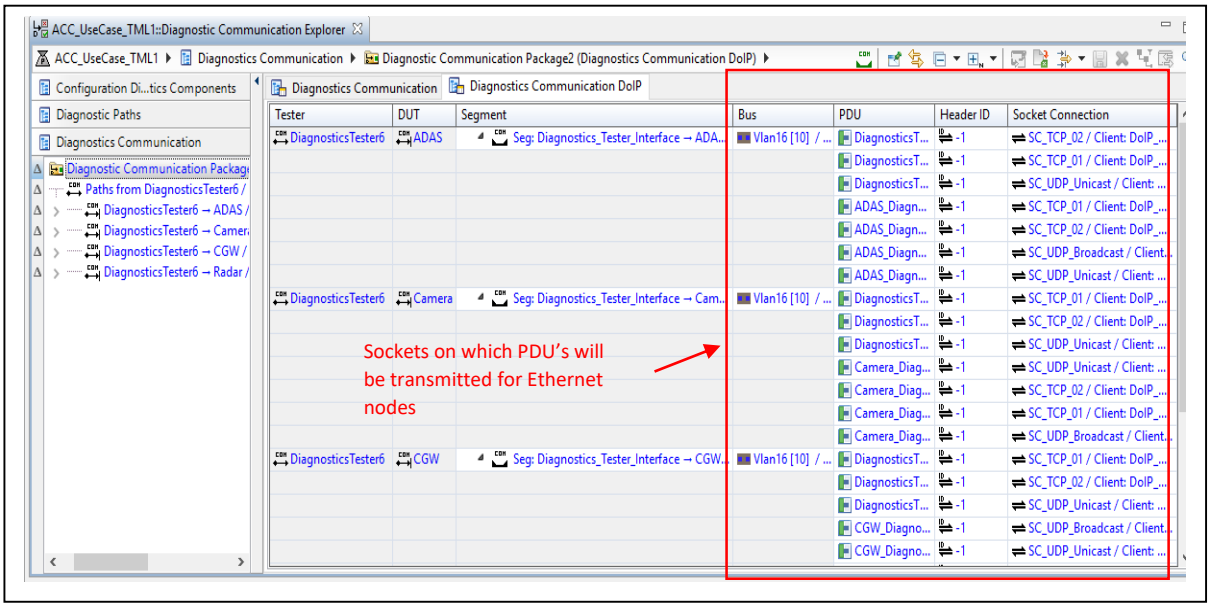

Figure 65 DOIP sockets

The vehicle identification messages are sent on UDP sockets whereas the routing activation request & response as well as diagnostic messages are sent on TCP sockets. TCP port 13400 is used for diagnostic data and UDP port 13400 is used for diagnostic discovery given by IANA as shown in Figure 5.26

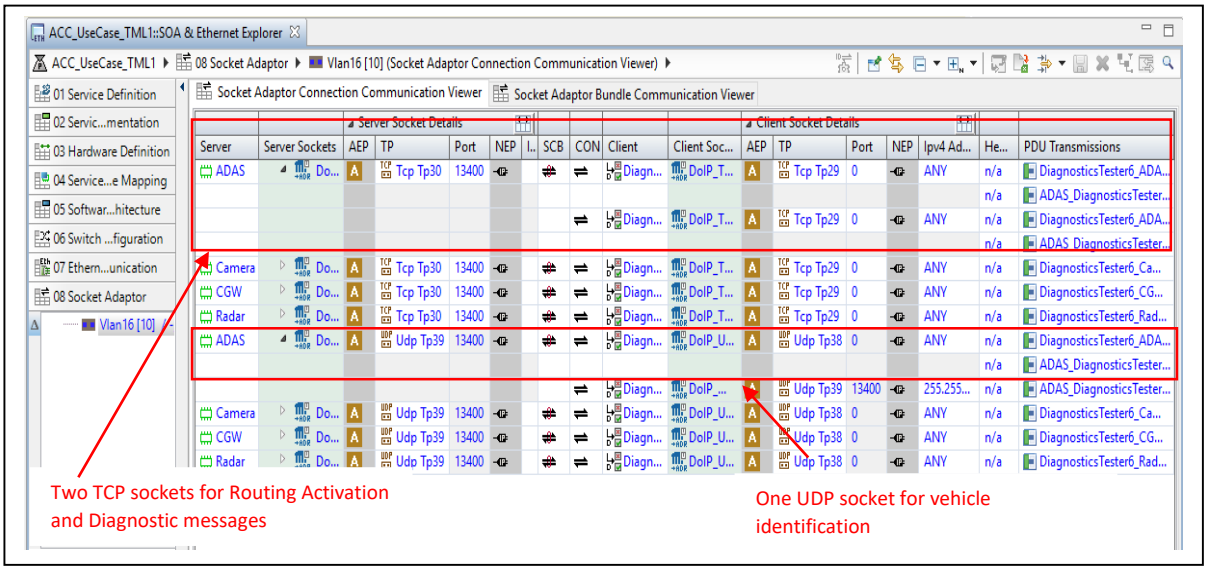

Figure 66 DOIP Communication

CHAPTER 6
RESULTS AND DISCUSSION

6.1 RESULTS & OUTPUTS

After the design, we were able to generate arxml codes for the following:

1. AUTOSAR System Description
2. ECU Description
3. SW Component Description

The AUTOSAR system description consists of the all the system information and contains the data that will communicate between different ECU's.

The ECU description contains the extract of single ECU and is a sub part of system description. It will consist of vehicle bus related information such as IP addresses, TCP/UDP ports etc. This information is necessary for the code generation of that particular ECU.

The SW component description includes the interfaces (Sender/Receiver or Client/Server), the relevant mapped ports, data elements/operations, data types that were defined in the implementation and necessary for software components development in ECU.

The below screenshots are generated arxml code for CGW ECU:

```
<SHORT-NAME>SoftwareTypes</SHORT-NAME>
<AR-PACKAGES>
  <AR-PACKAGE UUID="5a099a20b9943503bdbfb2d67e1094a4">
    <SHORT-NAME>ComponentTypes</SHORT-NAME>
    <ELEMENTS>
      <APPLICATION-SW-COMPONENT-TYPE UUID="0ac15fe8a16b03194bf844bfaX0ac15fe8a16b03194bf844bf900">
        <SHORT-NAME>R_ACC_CGW</SHORT-NAME>
        <PORTS>
          <R-PORT-PROTOTYPE UUID="0ac15fe8a16b03194bf844bfax0ac15fe8a16b03194bf844c0900">
            <SHORT-NAME>SomeIpAdaptive_Vehicle_Speed_ChangeInterface</SHORT-NAME>
            <REQUIRED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">/SoftwareTypes/Interfaces/SomeIpAdaptive_Vehicle_Speed_ChangeInterface</REQUIRED-INTERFACE
          </R-PORT-PROTOTYPE>
        </PORTS>
      </APPLICATION-SW-COMPONENT-TYPE>
    </ELEMENTS>
  </AR-PACKAGE>
  <AR-PACKAGE UUID="343af3c2383e360e8acf537589d92d54">
    <SHORT-NAME>Interfaces</SHORT-NAME>
    <ELEMENTS>
      <SENDER-RECEIVER-INTERFACE UUID="0ac15fe8a16b03194bf844c01X0ac15fe8a16b03194bf844c0000">
        <SHORT-NAME>SomeIpAdaptive_Vehicle_Speed_ChangeInterface</SHORT-NAME>
        <IS-SERVICE>false</IS-SERVICE>
        <DATA-ELEMENTS>
          <VARIABLE-DATA-PROTOTYPE UUID="0ac15fe8a16b03194bf844c04X0ac15fe8a16b03194bf844c0300">
            <SHORT-NAME>SomeIpAdaptive_Vehicle_Speed_ChangeDataElement</SHORT-NAME>
          </VARIABLE-DATA-PROTOTYPE>
        </DATA-ELEMENTS>
      </SENDER-RECEIVER-INTERFACE>
    </ELEMENTS>
  </AR-PACKAGE>
</AR-PACKAGES>
```

Figure 67 ARXML Code – Sender/Receiver Interface Definition

Figure 68 ARXML Code – Client/Server Interface Definition

The above figures shows arxml code for software components which consist of data elements and operations via sender/receiver and client/server interfaces typed on ports.

Figure 69 ARXML Code – System Signal Definition

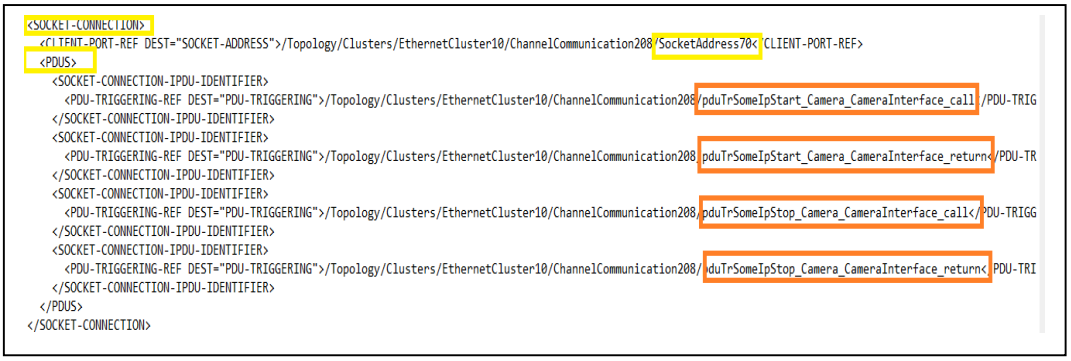

Figure 70 ARXML Code – Socket Connections/PDU to socket mapping

Figure 71 ARXML Code – SOME/IP Transformation

Figure 72 ARXML Code – Application Endpoints Definition

The above figures shows the application endpoints i.e. TCP/UDP ports defined through which the data is provided or consumed in the ECU.

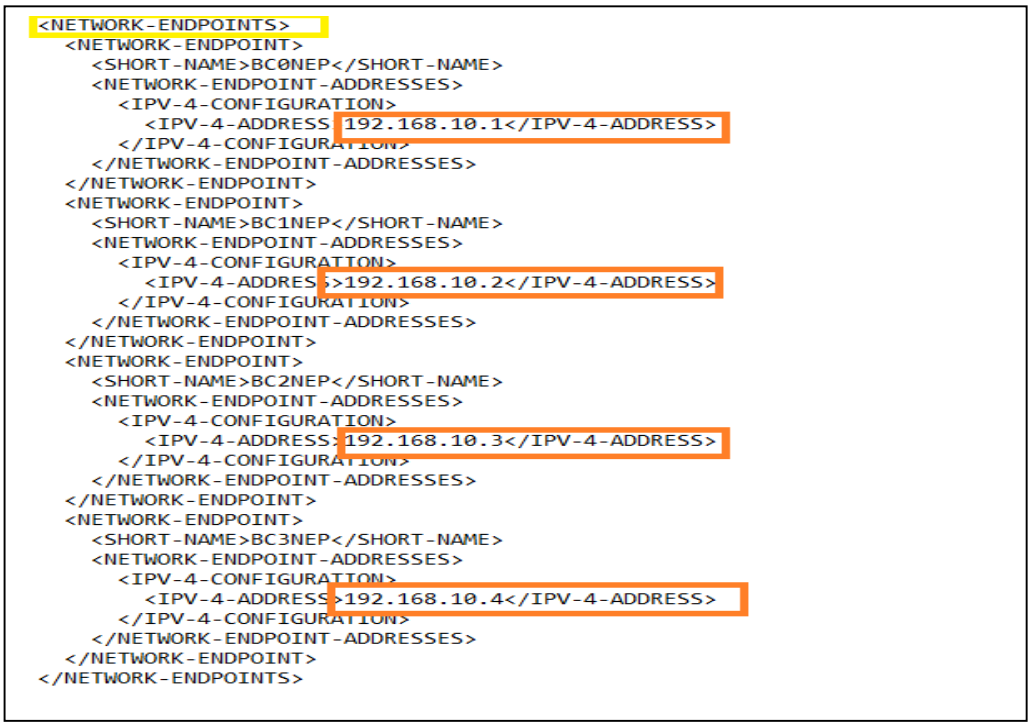

Figure 73 ARXML Code – Network Endpoints definition

Figure 74 ARXML Code – Physical Layer MAC definition

6.2 DISCUSSION

The above results shows the arxml code generated after designing the SOA, these arxml files which are AUTOSAR 4.3.0 compliant contains communications parameters required for software components to communicate over Ethernet. The arxml files can be fed into various tool chains such as DaVinci configurator Pro by vector, EB tresos studio by Elektrobit, MATLAB etc. for the AUTOSAR software development comprising of base software, RTE and application software components which is the end outcome of defining a use case over Service Oriented Architecture (SOA).

CHAPTER 7
CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

This thesis focused on Ethernet Communication Design with Service Oriented Architecture (SOA) using the SOME/IP protocol supported by Automotive Ethernet and AUTOSAR. PREEvision, a model based development tool has been chosen to create software components (SWC's) which implements the services, distribute the SWC's in the network, and the network itself.

To propose the SOA architecture, Adaptive Cruise Control use case has been taken for which the software architecture, hardware architecture has been defined in the model. After the modelling, the arxml files have been generated for the system description, ECU description and software component description which is the work area of an OEM. The system descriptions are AUTOSAR 4.3.0 compliant. Hence a prototype SOA has been defined in this thesis in order to move from signal to service oriented communication and from CAN, LIN, Flex Ray towards Automotive Ethernet.

7.2 FUTURE SCOPE

This thesis can be seen as a start point for the OEM's to define their next generation architecture which would have Ethernet as their physical channel to communicate the data over the network. Moreover the data elements and operations of the software components, the network management communication relevant parameters can be defined further. Also how Ethernet can be used as a backbone in the architecture for SOA can be explored.

Going ahead with the proposed architecture, the arxml files generated can be shared with TIER1 suppliers of the OEM's to generate the base software and RTE of different ECU's involved to implement the use case which will be AUTOSAR compliant. Similarly the software component descriptions can be imported to Model Based tools to generate the code for the application, hence building the complete system which would realize the function in a vehicle for the OEM.

REFERENCES

[1] F. Fikke, "Electric/Electronic-Architectures: Automating and optimizing communication matrices," Telecommunications Sensing, Delft University of Technology, 2016.

[2] Matheus K., & Thomas K. (2017). *Automotive Ethernet* (2nd ed.)

[3] J. Huang, M. Zhao, Y. Zhou and C.-C. X. Xing , "In-Vehicle Networking: Protocols, Challenges, and Solutions," IEEE Network, 2018.

[4] C. Kozierok et al., *Automotive Ethernet — The Definitive Guide*, Interpid Control Systems, 2014.

[5] A. Kern, "Ethernet and IP for Automotive E/E-Architectures," 2012.

[6] P. Hank, T. Suermann, and S. Müller, "Automotive Ethernet, a holistic approach for a next generation in-vehicle networking standard," in *Advanced Microsystems for Automotive Applications*. Berlin, Germany: Springer-Verlag, 2012, pp. 79–89.

[7] AUTOSAR, https://www.autosar.org/ *Layered Software Architecture*, 4.3.1 edition, 2017.

[8] AUTOSAR, https://www.autosar.org/ *Specification of PDU router*, 4.3.1 edition, 2017.

[9] AUTOSAR, https://www.autosar.org/ *Specification of Socket Adaptor*, 4.3.1 edition, 2017.

[10] AUTOSAR, https://www.autosar.org/ *Specification of TCP/IP Stack*, 4.3.1 edition, 2017.

[11] W. Dafang, L. Shiqiang, H. Bo, Z. Guifan, and Z. Jiuyang, "Communication mechanisms on the virtual functional bus of AUTOSAR," in *Proc. Int. Conf. Intell. Comput. Technol. Autom.*, May 2010, pp. 982–985.

[12] AUTOSAR, https://www.autosar.org/ *Specification of SOME/IP Transformer*, 4.3.1 edition, 2017.

[13] AUTOSAR, https://www.autosar.org/ *Specification of Diagnostic Communication Manager*, 4.3.1 edition, 2017.

[14] AUTOSAR, https://www.autosar.org/*Specification of Diagnostic over IP*, 4.3.1 edition, 2017.

[15] IEEE p802.3bw-2015-IEEE Standard for Ethernet Amendment 1: Physical Layer Specifications and Management Parameters for 100 Mb/s Operation over a Single Balanced Twisted Pair Cable (100BASE-T1). [Online]. Available: https://standards.ieee.org/standard/802_3bw-2015.html

[16] ISO 13400-2, "Road vehicles — Diagnostic communication over Internet Protocol (DoIP) Part 2: Transport protocol and network layer services," 2012

[17] Conner Peter, Robinson Stewart, “Service oriented architecture” ,Google Patents, 2007

[18] Paul Patrick, Ashok Aletty, Jayram kasi, Chet Kapoor, Tolga Urhan, Matthew Mihic, “Service oriented architecture”, Google Patents, 2006.

[19] J. Schäuffele, "E/E Architectural Design and Optimization using PREEvision," *SAE International*, 2016.

[20] J. Kocic, N. Jovičić and . V. Drndarević, "Sensors and Sensor Fusion in Autonomous Vehicles," in *TELFOR*, 2018.

[21] P. Hank, S. Müller and T. Suermann, "Automotive Ethernet, a Holistic Approach for a Next Generation In-Vehicle Networking Standard," *Electrified Vehicles*, pp. 79-89.

[22] T. Hackel , "A Middleware Solution for Open and Dynamic ICT Architectures in Future Cars," Hamburg, Germany.

[23] W. Dafang, L. Shiqiang, B. Huang, Z. Guifan and Z. Jiuyang, "Communication Mechanisms on the Virtual Functional Bus of AUTOSAR," in *2010 International Conference on Intelligent Computation Technology and Automation*, 2010.

[24] S. Tuohy, C. Hughes, E. Jones, M. Trivedi and L. Kilmartin, "Intra-Vehicle Networks: A Review," *in IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS*, 2014.

[25] G. Gopu, K. Kavitha and J. James, "Service Oriented Architecture based connectivity of Automotive ECUs," *in 2016 International Conference on Circuit, Power and Computing Technologies [ICCPCT*], 2016.

[26] M. Johanson, P. Dahle and A. Söderberg, "Remote Vehicle Diagnostics over the Internet using the DoIP Protocol," in ICSNC 2011 : The Sixth International Conference on Systems and Networks Communications, 2011.

[27] Y. S. Lee, J. H. Kim and J. W. Jeon, "Automotive Diagnostic Gateway using Diagnostic over Internet Protocol*," IEIE Transactions on Smart Processing and Computing*, vol. 3, no. 5, pp. 313-318, 2014.

[28] Ralph Moritz, Tamara Ulrich, and Lothar Thiele "Evolutionary Exploration of E/E-Architectures," Germany.

[29] T. S. Shamin Dudu, S. G. Shivaprasad Yadav, M. Arun Kumar, Nisha C. Rani, "In-Vehicle Automotive Network Gateway Electronic Control Unit for Low Price Vehicle," *SASTECH*, vol. 8, no. 2, pp. 79-86, 2009.

[30] Arne Neumann, Martin Mytych, Derk Wesemann, Lukasz Wisniewsk, "Approaches for In-vehicle Communication – An Analysis and Outlook," in *Communications in Computer and Information Science*, 2017.

[31] Peter Hank, Ovidiu Vermesan, Steffen Müller, Jeroen Van Den Keybus, "Automotive Ethernet: In-vehicle Networking and Smart Mobility," in *EDAA*, 2013.

[32] Waszecki, Peter, Martin Lukasiewycz, Alejandro Masrur, and Samarjit Chakraborty. "How to Engineer Tool-Chains for Automotive E/E Architectures?" *Special Issue on the 5th Workshop on Adaptive and Reconfigurable Embedded Systems*,new york: news letter, december 2013.

[33] C., Varun, and Kathiresh M. "Automotive Ethernet in On-Board Diagnosis (Over IP) & In-Vehicle Networking." *International Conference on Embedded Systems*. IEEE, 2014. 255-260

[34] Hinrichsen, "The road to autonomous driving," in *Deterministic Ethernet Forum*, Vienna, April 2015.

[35] https://automotive.electronicspecifier.com/air-conditioning/real-time-automotive-ethernet

[36] M. Helmling, "Service-oriented Architectures and Ethernet in Vehicles: Towards Data Centers on Wheels with Model-based Methods,"Automotive Ethernet, March 2017.

[37] D. D. Gebauer, "AUTOSAR-Conformant Vehicle Diagnostics over DoIP:Developing Diagnostic Communications for E/E Systems," *Automobil Elektronik*, December 2018.