Universität Stuttgart
Institut für Fahrzeugtechnik Stuttgart

Ermittlung Verbindungsauslastungen in Automotive Ethernet-Netzwerken

Estimation of Link Loads in Automotive Ethernet Networks

von

Yanzhou Chen

Matrikelnummer. 3676237

Betreuer: Master.-Ing. Maier Jonas, Master.-Ing. Mohammad Zaheri,
Dr.-Ing. Andreas Vetter
Prüfer: Prof. Dr.-Ing. Hans-Christian Reuss

Vorgelegt an der Universität Stuttgart

Institut für Fahrzeugtechnik Stuttgart
Lehrstuhl für Kraftfahrzeugmechatronik

2024

Contents

Abbreviations V

Symbols VII

List of Figures IX

List of Tables XIII

Kurzfassung XIV

Abstract XV

1 Introduction 1

1.1 Motivation 1

1.2 Structure of this Work 3

2 State of the Art 5

2.1 Basic Concept....5

2.1.1 AUTOSAR and ARXML....5

2.1.2 ARXML Visualizer 5

2.1.3 Signal....6

2.1.4 Signal Groups....7

2.1.5 Frame....7

2.1.6 Signal Transmit Types....7

2.1.7 Types of Bus Systems in vehicle....8

2.1.8 Frame Transmission Trigger 9

2.1.9 NCD....10

2.1.10 PDU Multiplexing....10

2.2 Ethernet Throughput....10

2.3 Automotive Ethernet Throughput 11

2.4 Existent Link Load Estimation Approachs 13
2.4.1 PREEvision 13
2.4.2 Existent Tables for Load Calculation 14

3 Static Link Load in Ethernet 17
3.1 Static Bus Load Analysis in CAN-Busses 17
3.2 Methodology for Link Load Calculation in Ethernet 18
3.3 Ethernet Communication Model 19
3.4 Estimation of Frame Size 20
3.4.1 Frame Packet Structure 20
3.4.2 PDU Packet Size 21
3.4.3 Other Protocol Format 24
3.4.4 Total Frame Size 27
3.5 Further Conditions 27
3.5.1 Sending Frequency 27
3.5.2 Baud Rate 28
3.5.3 Redundant Frames 28
3.6 Calculation of Static Link Load and Table Establishment 28
3.6.1 Calculation 28
3.6.2 Virtual Networks 29
3.6.3 Table Establishment 30

4 Dynamic Link Load Estimation in Ethernet 35
4.1 Dynamic Bus Load Analysis in CAN-Busses 35
4.2 Data Collection 36
4.3 Probability Distribution Analysis 40
4.4 Segmentation and Classification 45
4.4.1 Previous Works 45
4.4.2 Clustering Algorithm 47

4.5 Representation Learning 50
4.5.1 Principle Component Analysis 50
4.5.2 Causal Convolution Network 53
4.5.3 Decomposition Deep Encoder Model 54
4.5.4 Unet Diffusion AutoEncoder 55
4.5.5 Loss Function 58
4.5.6 General Procedure 60
4.6 Experiments 61
4.6.1 Causal Convolution Network 61
4.6.2 Decomposed Convolution Network 66
4.6.3 Unet Diffusion AutoEncoder 68
4.6.4 Voting and Merging Process 70
4.7 Classification Results and Inspections 72

5 Results and Discussions 75
5.1 Results 75
5.2 Discussions 81
5.3 Future Works 82

6 Conclusions 83

7 Literature 85

Appendix 89
A.1 Source Code for Static Link Load Calculation 89
A.2 Source Code for Table Establishment 94
A.3 Part of the Code from the UDAE 99
A.4 Code for Voting and Merging Process 105
A.5 Link Load Tables from Classes 6 to 10 107
A.6 SQLite Data Processing code 116

A.7 Part of the Code for Causal CNN 118
A.8 Unfinished Code for Negative Binomial Mixture 125

Abbreviations

ARXML	AUTOSAR XML
API	Application Interface
CAN	Controller Area Network
CNN	Convolutional Neural Network
CRC	Cyclic Redundancy Check
DAE	Diffusion Autoencoder
DPGMM	Dirichlet Process Gaussian Mixture Model
ECU	Electrical Control Unit
FLOSS	Fast Low-cost Online Semantic Segmentation
FTP	File Transfer Protocol
HDP-GP	Hierarchical Dirichlet Process - Gaussian Process
NCD	Network Communication Description
NHPP	Non-Homogeneous Poisson Process
OSI	Open Systems Interconnection
PCA	Principal Component Analysis
PDU	Protocol Data Unit
PduR	Protocol Data Unit Router
SMS	Short Message Service
SMSC	Short Message Service Center
SoAd	Socket Adaptor
SWC	Software component
TELNET	TELecommunication NETwork
VAE	Variational Autoencoder
VLAN	Virtual Local Area Network
UDAE	Unet Diffusion Autoencoder

Symbols

N_{Frame}	Byte	Number of Bytes of a frame
N_{PDULen}	Byte	Number of Bytes of a PDU payload
N_{Trans}	Byte	Number of Bytes added in the transportation layer
$N_{FrameWithGapPerBit}$	bit	Number of bits in one frame data contained in the frame transmission, considering frame gaps
$Baudrate$	Bit/s	Transmission speed of the link
$Load_{FrameTrans}$:	bit	Load generated by a Frame transmission
T_{cycle}	s	Cycle time of the signal transmission
sFE	-	Statical frame error rate
λ	s^{-1}	average rate of occurrence
$K_{\zeta}(T', T)$	-	kernel density estimation
C_i	-	is the i th cluster
r_{ik}	-	posterior probabilities for each data point i belonging to each component k with form:
π_k	-	weight of component k
α	-	Dirichlet prior concentration parameter
x_i	-	data points
μ_0	-	prior mean
O	-	output
W	-	weight
b	-	bias
z_{lat}	-	the latent subcode
o_{ki}	-	denotes the i-th window in the k-th group.
c_k	-	the embedding center for the k-th group
$\{x_{t_k:t_k+w}\}_{k=1}^{M}$	-	inter-state samples

$\{x_{t+i:t+w+i}\}_{i=1}^{N}$.	-	intra-state samples
$S(t_{i-1}, t_i)$	-	Time segment between t_{i-1}, t_i
S_1'	-	New time segment on the left
S_2'	-	New time segment on the right
$N_{PDUsSent}$	-	Number of PDUs sent in a second

List of Figures

Figure 2.1 Welcome page of the ARXML Visualizer 6
Figure 2.2 Throughput under different packet sizes 11
Figure 2.3 Average of Link Load based on the measured bandwidth 12
Figure 2.4 TCP throughput with and without SACK and timestamp options. 12
Figure 2.5 UDP can significantly exceed TCP in throughput 13
Figure 2.6 Screenshot of the Communication Structures shown in PREEvision 14
Figure 2.7 Screenshot of calculated loads 14
Figure 3.1 The communication between ECUs and switch from Mercedes internal website 19
Figure 3.2 Protocol overview for automotive Ethernet 21
Figure 3.3 Header format of NmPdu 22
Figure 3.4 Header format of SOMEIP-SEGMENTED-IPDU 23
Figure 3.5 SOME/IP Padding Example 23
Figure 3.6 IPv4 format 25
Figure 3.7 Ethernet packet format 25
Figure 3.8 AUTOSAR Layered Software Architecture for Diagnostic PDUs 26
Figure 3.9 Static link load table in ARXML Visualizer 33
Figure 4.1 Database result from the first .mfx file of trace 1 36
Figure 4.2 Database result from the first .mfx file of trace 2 37
Figure 4.3 Bug Hunter Interface 38
Figure 4.4 Poisson distribution of common λ values 41
Figure 4.5 PDU data segment on 2024.05.28 44
Figure 4.6 PDU data segment on 2024.06.07 44
Figure 4.7 Illustration of a gaussian mixture 48
Figure 4.8 Most significant features of trace 52

Figure 4.9 Scree Plot of PCA....53
Figure 4.10 Latent representations....54
Figure 4.11 Structure of proposed model: UDAE with DPGMM....57
Figure 4.12 Layered structure of the model....58
Figure 4.13 General model structure....61
Figure 4.14 Total segmentation with window=50, step=5....62
Figure 4.15 Total segmentation with window=100, step=10....62
Figure 4.16 Total segmentation with window=200, step=20....63
Figure 4.17 Total segmentation with window=300, step=30....63
Figure 4.18 Total segmentation with window=500, step=50....63
Figure 4.19 Segmentation 1 with window=50, step=5....64
Figure 4.20 Segmentation 1 with window=100, step=10....64
Figure 4.21 Segmentation 1 with window=200, step=20....64
Figure 4.22 Segmentation 1 with window=300, step=30....64
Figure 4.23 Segmentation 1 with window=500, step=50....65
Figure 4.24 Segmentation 10 with window=50, step=5....65
Figure 4.25 Segmentation 10 with window=100, step=10....65
Figure 4.26 Segmentation 10 with window=200, step=20....66
Figure 4.27 Segmentation 10 with window=300, step=30....66
Figure 4.28 Segmentation 10 with window=500, step=50....66
Figure 4.29 Total segmentation with window=50, step=5....67
Figure 4.30 Total segmentation with window=200, step=20....67
Figure 4.31 Segmentation 1 with window=50, step=5....67
Figure 4.32 Segmentation 1 with window=200, step=20....68
Figure 4.33 Segmentation 10 with window=100, step=10....68
Figure 4.34 Segmentation 10 with window=200, step=20....68
Figure 4.35 Total segmentation with step=5....69
Figure 4.36 Total segmentation with step=10....69

Figure 4.37 Segmentation 1 with step=5 69
Figure 4.38 Segmentation 1 with step=10 69
Figure 4.39 Segmentation 10 with step=5 70
Figure 4.40 Segmentation 10 with step=5 70
Figure 4.41 Voted and merged result of segment 1 71
Figure 4.42 Voted and merged result of segment 10 71
Figure 4.43 Overall classification result 72
Figure 5.1 Selection of scenarios 75
Figure 5.2 Link load average over the entire trace in ARXML Visualizer 77

List of Tables

Table 2.1: Frame triggers and their descriptions [13] 9
Table 3.1: Part of the static link load table 30
Table 4.1: Part of the converted data 38
Table 4.2: Part of the table with PDU names, time sent and PDU numbers 39
Table 4.3: Mean value of PDU numbers sent per Class per Second 72
Table 4.4: Threshold value of PDU numbers sent per Class per Second 73
Table 5.1: Link load average over the entire trace 75
Table 5.2: Link load under scenario 1 77
Table 5.3: Link load under scenario 2 78
Table 5.4: Link load under scenario 3 80

Kurzfassung

Diese Arbeit konzentriert sich auf die genaue Schätzung der Linkauslastung innerhalb von Automotive-Ethernet-Netzwerken, eine kritische Aufgabe aufgrund der begrenzten Bandbreite der Fahrzeugkabel. Frühere Forschungen in diesem Bereich stützten sich auf pseudo-automotive Netzwerke, wie z.B. Beispielkommunikations-Setups mit Raspberry Pi. Im Gegensatz dazu liegt der Fokus dieser Forschung auf der Schätzung dynamischer Linkauslastungen durch die Analyse von Trace-Dateien, die aus tatsächlichen Fahrzeugen gesammelt wurden. Darüber hinaus wurde ein Modell entwickelt, um statische Linkauslastungen durch die Analyse von Daten innerhalb von ARXML-Dateien und Protokollen zu berechnen. Verschiedene Methoden wurden eingesetzt, um die dynamische Linkauslastung zu schätzen, wobei ein kritischer Bestandteil die Schätzung dynamischer Signalübertragungsraten war, die aus den Trace-Dateien abgeleitet wurden. Dieser Prozess erforderte die Segmentierung und Klassifizierung der Traces in unterschiedliche Szenarien, wofür maschinelle Lerntechniken wie DGPMM-Fitting, Kausale Konvolutionale Netzwerke und U-Net Diffusion Autoencoder angewendet wurden. Die Experimente führten zur Erstellung einer umfassenden Linkauslastungstabelle. Diese Studie schließt eine bedeutende Lücke in der genauen Schätzung der Linkauslastungen in Automotive-Ethernet-Netzwerken und bietet Ingenieuren wertvolle Einblicke in das Signalverhalten und die Kabelauswahl für Ethernet-Verbindungen. Für zukünftige Forschungen wird empfohlen, unterschiedliche PDU-Übertragungstypen einzeln zu untersuchen, alternative maschinelle Lernansätze zur Verbesserung der Segmentierung zu erkunden und verschiedene Fahrzeugkommunikations-architecturen zu untersuchen.

Abstract

This thesis focuses on accurate estimation of link load within automotive Ethernet networks, a critical task due to the limited bandwidth of in-vehicle cables. Previous research in this area relied on pseudo-automotive networks, such as sample communication setups with Raspberry Pi. In contrast, this research focuses on estimating dynamic link loads by analyzing trace files collected from actual vehicles. Additionally, a model was developed to calculate static link loads by analyzing data within ARXML files and protocols. Various methods were employed to estimate dynamic link load, with a critical component being the estimation of dynamic signal sending rates derived from trace file data. This process necessitated the segmentation and classification of traces into distinct scenarios, for which machine learning techniques—such as DGPMM fitting, Causal Convolutional Networks, and U-Net Diffusion Autoencoders—were applied. The experiments culminated in the creation of a comprehensive link load table. This study addresses a significant gap in the accurate estimation of link loads in automotive Ethernet networks, offering engineers valuable insights into signal behavior and cable selection for Ethernet links. For future research, it is suggested to examine different PDU sending types individually, explore alternative machine learning approaches for enhanced segmentation, and investigate various vehicle communication architectures.

1 Introduction

Motivation

In recent years, the automotive industry has experienced a significant shift towards the integration of increasingly advanced electronic systems within vehicular architectures, together with introduction of new concepts such as the "software-defined vehicle" [1]. This transformation is primarily driven by the substantial demand for connected and autonomous driving technologies, alongside the increasing consumer expectations for sophisticated in-vehicle entertainment systems. Consequently, vehicular communication networks have evolved to become highly complex, requiring enhanced bandwidth and stringent time-sensitive networking capabilities [2]. Within this context, automotive Ethernet has emerged as a critical component of in-vehicle networks, with the potential to serve as the primary communication medium in modern vehicles [3].

However, the increasing complexity of these networks has led to significant cost of automotive harnesses. Current estimations suggest that communication harnesses constitute more than 5% of the total vehicle cost [4], a figure that is anticipated to rise in tandem with the increasing complexity of vehicular communication systems. Additionally, the cost disparity between Ethernet cables of varying speeds is considerable, with high-speed cables potentially costing up to ten times more than their lower-speed counterparts [5]. As a result, a deployment of high-speed cables throughout a vehicle could make the overall costs unaffordable.

Considering these challenges, it becomes imperative to accurately estimate link loads within automotive Ethernet networks to ensure both the reliability and cost-efficiency of vehicular communication systems. This paper seeks to address this problem through a computational and empirical analysis of signal loads, augmented by investigations based on ARXML files and real-world vehicle data. By quantifying link loads and identifying potential bottlenecks, the study offers valuable insights that can guide the selection of cables and the design and optimization of communication protocols as well as overall architecture of vehicle communication systems. At the same time, these research findings can also provide automotive engineers with a deeper understanding of network behaviors and insights into another aspect of in-vehicle network performance, namely the required computational capacity, thereby establishing a robust foundation for future advancements in automotive engineering, particularly in domains such as autonomous driving. The subsequent sections of this paper elaborate on the methodology employed to analyze link loads, present the

experimental results, and explore directions for future research, thus contributing to the ongoing evolution of vehicular communication systems.

.

Structure of this Work

In chapter one, the motivation behind this work is introduced, with a detailed rationale provided for the significance of estimating Ethernet link load in vehicular networks.
Chapter two presents the foundational knowledge relevant to this study and offers a comprehensive review of prior research in the field of Ethernet link load estimation.
In chapter three, the methodology for calculating static link load is discussed. This involves utilizing information from ARXML files and analyzing protocol specifications to estimate packet header lengths, ultimately leading to the development of a table that presents the static link loads.
Chapter four addresses the estimation of dynamic link loads, which requires empirical studies. This chapter begins by outlining the data collection and conversion processes, followed by an investigation into the application of statistical methods to model the collected data using specific distributions. Various machine learning techniques are then explored to classify different scenarios from real-life trace data, enabling the separate estimation of dynamic link loads based on the classified results.
Chapter Five presents the results of the study, accompanied by a discussion of the findings and reflections on potential future research directions.

State of the Art

Basic Concept

AUTOSAR and ARXML

AUTOSAR, which stands for Automotive Open System Architecture, is a global development partnership aimed at creating an open and standardized software architecture for the automotive industry [6]. By defining standardized interfaces, protocols, and methodologies, AUTOSAR enables the development of modular, flexible, and efficient automotive software that can be seamlessly integrated into vehicles from various manufacturers [7]. AUTOSAR's architecture is structured into several layers, including the application layer, runtime environment, basic software layer, and hardware abstraction layer, each serving specific functions and providing standardized interfaces for communication and interaction between software components [8]. Furthermore, AUTOSAR supports both traditional automotive ECUs (Electronic Control Units) and emerging technologies such as connected cars, autonomous vehicles, and electric vehicles, ensuring adaptability to evolving industry trends and requirements [9].

ARXML (AUTOSAR XML) is a standardized format used within the AUTOSAR (AUTomotive Open System ARchitecture) framework to describe and configure automotive electronic systems. It helps with the exchange of information between various tools used in the development of automotive software [6]. ARXML files encapsulate detailed descriptions of software components, including data types, interfaces, and configurations of the automotive systems, enabling interoperability and seamless integration across different vendors and development stages. By adhering to the AUTOSAR standard, ARXML ensures consistency and reduces the complexity of software development in the automotive industry [10].

1.1.1 ARXML Visualizer

The ARXML Visualizer is a powerful tool designed for working with AUTOSAR XML files inside Mercedes, offering a range of features to view and manipulate ARXML files. It provides an intuitive interface that simplifies the navigation and visualization of AUTOSAR files in a clear, tree-like structure. Other than that ARXML Visualizer can also show the SWC topology, Hardware topology as well as listing the Major AUTOSAR objects separately.

The Visualizer can also perform fundamental editing tasks such as renaming objects, changing references, editing attribute values and add and delete nodes. During these operations, the tool can analyze the entire files, ensuring all related references and paths are adjusted according to ARXML schema.

Other than that Visualizer is equipped to compare the differences between two ARXML files and also merge two ARXML files flawlessly.

Figure Error! No text of specified style in document.**.1** Welcome page of the ARXML Visualizer

The aforementioned functions facilitate a more intuitive and efficient examination of automotive electronic systems by engineers. Additionally, the Visualizer extends its capabilities to include the analysis of data contained within ARXML files, offering various tabular views that present unique perspectives on the underlying ARXML data. This specific functionality is the focus of development in this thesis..

1.1.2 Signal

A signal is the smallest entities for the exchange of information. Its classification branches into two quintessential categories: status signals and physical signals [11].

Status signals serve as emissaries of state, ranging from elementary, requiring merely one bit (e.g., 1 denoting sensor activation, 0 indicating sensor cessation), to multifaceted states necessitating multiple bits for representation. In the realm of simple states, the binary duo of 0 and 1 delineates inactivity and activity, respectively.

Conversely, complex states entail an assignment table for exhaustive state enumeration.

Physical signals, in contrast, epitomize the tangible, embodying sensor actual values, actuator set values, or computed results. For instance, a physical signal might encapsulate the current distance from colligation, spanning a potential range of up to 10 m to a crucial distance of 10 cm. The establishment of conversion factors becomes imperative, mandating meticulous consideration prior to signal utilization [12].

A signal according to Mercedes standard is generally consists of [13]:

Signal name (derived from the English description)

Signal description

Signal length

Signal init value

Used data type

1.1.3 Signal Groups

A signal group combines signals to ensure data consistency. Signal groups, the amalgamation of signals necessitating synchronized transmission, transcend the dichotomy of physical or status delineation. These cohesive units facilitate the simultaneous dissemination of disparate signal types, fostering streamlined data transmission. According to Mercedes files when two or more signals have to be transmitted with time coherent values these signals shall be defined in one signal group [14].

1.1.4 Frame

A frame is like the container for sending data across a network. It holds all the bits and bytes being sent, along with some extra information needed for handling the data correctly. This extra information is called protocol overhead and is placed at the beginning or end of the frame. While it's crucial for the frame to be received properly, it doesn't include the actual user data, which is called the payload. So, whenever data is sent using a Protocol Data Unit (PDU), it's always wrapped up in a frame. In short, a Frame represents a general design object that is used to describe the layout of the included PDUs as a reusable asset [11].

1.1.5 Signal Transmit Types

The transmit type describes when, or under which conditions a signal shall be:

triggered for transmission. The transmit type is signal-specific and aggregated to the

PDU transmit type. Signal groups are treated like a single signal, which means all signals of a signal group have the same transmit type. The following transmit types are defined according to Mercedes standards:

"Cyclic": A "Cyclic" signal shall be transmitted on a regular periodic schedule, independent of a change of the signal,

"Cyclic On Change": A "Cyclic On Change" signal shall be transmitted on a periodic frequency schedule. This signal additionally triggers transmission once whenever the signal changes,

"Cyclic If Active": A “Cyclic If Active" signal shall be only transmitted at a specific periodic rate t_C when the signal does not equal the default value. The transmission shall be discontinued after transmitting the default value for $n + 1$ times,

"Dual Cycle": A "Dual Cycle" signal shall be transmitted with a fixed cycle time t_C as long as the signal equals the default value, and with a different fixed cycle time as long as the signal is not equal to the default value,

"On Change": An "On Change" signal shall be transmitted $m + 1$ times if the signal value changes,

"Spontaneous": A "Spontaneous" signal shall be only transmitted once if the application triggers transmission.

The transmit types are split into static and dynamic cases. The static transmit types sent signals almost deterministically, while dynamic transmit types sent signals undeterministically. Cyclic transmit type is usually defined as static, while the others to be dynamic. When there is a Container-frame, the frame transmission trigger also plays a role, in such case, static transmission cases may also count as dynamic.

Types of Bus Systems in vehicle

Controller Area Network (CAN):

Developed by Bosch in the 1980s, CAN is one of the most widely used vehicular networks. It provides robust, deterministic communication between electronic control units (ECUs) in real-time. CAN supports multiple topologies and data rates, making it suitable for various automotive applications, including engine control, transmission control, and chassis control [12].

Ethernet:

Ethernet is increasingly being adopted in automotive networks due to its high bandwidth capabilities, which are essential for handling the growing volume of data generated by advanced driver assistance systems (ADAS) and autonomous driving features. Automotive Ethernet offers faster communication speeds, enabling the integration of high-resolution cameras, LiDAR, and other sensors [12].

FlexRay:

Initially developed to meet the demands of safety-critical applications in automotive systems, FlexRay provides deterministic, fault-tolerant communication suitable for applications requiring high reliability and real-time performance. However, its adoption has been limited compared to CAN and Ethernet [12].

Local Interconnect Network (LIN):

LIN is a low-cost, low-speed network primarily used for non-critical applications such as interior lighting, seat control, and infotainment systems. It is often used in conjunction with CAN to reduce the overall cost of the vehicle's network architecture [12].

Media Oriented Systems Transport (MOST):

MOST is a high-speed multimedia network primarily used for infotainment and multimedia applications in vehicles. It supports the transmission of audio, video, and control data over a single network, providing a reliable and efficient solution for in-car entertainment systems [12].

1.1.6 Frame Transmission Trigger

A frame transmission trigger refers to the mechanism used to trigger the transmission of data frames within the communication stack. This mechanism defines the conditions under which a data frame should be transmitted over the network. Sometimes multiple PDUs are aggregated into container frames. Instead of individually sending each PDU in a separate frame, PDUs, along with their PDU headers, are grouped together in a Container-frame and transmitted as a single entity. To form container frames, PDUs must be temporarily stored and arranged based on their arrival order, which can occur within a control unit during transmission or a gateway during forwarding. The triggering of Container-frames depends not on the transmission types of the PDUs, but on the trigger specified in table 2.1.

Table Error! No text of specified style in document.**.1:** Frame triggers and their descriptions [13]

Trigger	Description
MaxTxDelayFrame	Defines the time after which the frame is sent. The time is started when the first PDU is put into the frame
PDU	A PDU can be configured to trigger the transmission of the frame immediately. This equates to MaxTxDelayPDU = 0
Buffer	The frame is sent if the frame is full or a configured buffer threshold is reached
MaxTxDelayPDU	A PDU can have a distinct MaxTxDelayPDU which can reduce MaxTxDelayFrame to MaxTxDelayPDU at the time the PDU is put into the frame

The EthernetCommunicationController has a property named maximumTransmitBuffer-Length which specifies the buffer threshold [13]. For simplicity, the PDUs are considered to be sent individually to get the maximum possible load.

1.1.7 NCD

NCD stands for Network Communication Description and is synonymous with the communication matrix. It defines the complete data configuration of communication and networks [14].

1.1.8 PDU Multiplexing

Multiplexing is the process of combining multiple signals or data streams into one. In PDU multiplexing, this involves combining several PDUs into a single frame or message, allowing multiple pieces of data to be sent together over a shared communication channel. In this research, PDU multiplexing was not considered for simplicity [13].

Ethernet Throughput

Throughput is defined as the rate of successful message delivery over a communication channel and is considered one of the most vital Quality of Service (QoS) parameters for networks.

Nazrul Islam et al [15] analyzed the Quality of Service of Ethernet network based on package size. They used the Riverbed network simulator to observe the behavior of an Ethernet network under different packet sizes. The experimental setup includes 16

workstations, two hubs, and a switch, all connected via 10 Mbps links. The experiment considers packet sizes of 128, 256, 512, 1024, and 2048 bytes, with segmentation applied to packets larger than the Maximum Transmission Unit (MTU) of 1518 bytes.

The simulation results show that the throughput threshold increases with larger packet sizes. In the Ethernet network used by Mercedes, the package size is generally set to be 1500, but some packages may be even over 4000. That means the throughput may be very high.

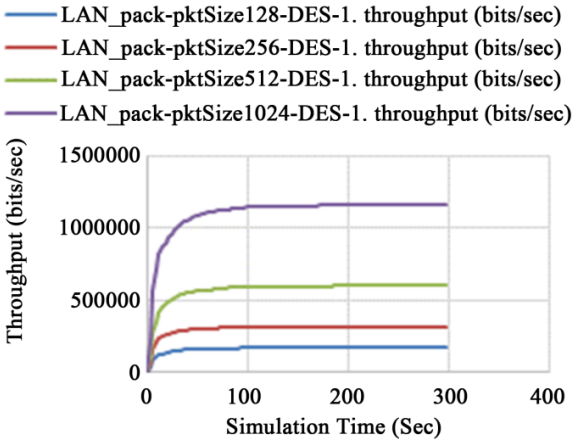


Figure Error! No text of specified style in document..**2** Throughput under different packet sizes [1]

Automotive Ethernet Throughput

There are already some research conducted over Automotive Ethernet throughput.

Hyung-Taek Lim et al [16] analyzed different network topologies (star-based, daisy-chain, and tree) and traffic types, such as control data, driver assistance camera data, navigation data, and multimedia data and used OMNeT++ and the INET-framework to simulate and measure end-to-end delays and network load under various conditions.

The results suggest that the prioritization mechanisms have no affect on the link load, while network topologies have only minor affect on link loads.

Measured Link	Topo1		Topo2		Topo3	
	without Prio [Mbit/s]	with Prio [Mbit/s]	without Prio [Mbit/s]	withPrio [Mbit/s]	without Prio [Mbit/s]	with Prio [Mbit/s]
Switch1 → HeadUnit	28.7754	28.7733	29.1889	29.1866	29.1889	29.1866
Switch2 → RSE	55.3968	55.3968	55.4090	55.4090	55.3984	55.3984
RSE → Switch2	x	x	1.6959	1.6959	1.6960	1.6960
Switch2 → RSE	55.3968	55.3968	55.4090	55.4090	55.3984	55.3984

Figure Error! No text of specified style in document.**.3** Average of Link Load based on the measured bandwidth [16]

Sangrok Han and Hyogon Kim [17] conducted experimental tests on AUTOSAR TCP performance, the researchers used Raspberry Pi to simulate ECUs, enabling them to manipulate clock speed and TCP features. They found that firstly, in the in-vehicle network, the speed and workload of ECUs dominate TCP throughput. Secondly, Window Scale option can improve TCP throughput in some cases, especially for slower ECUs. Finally, UDP can achieve higher throughput than TCP in some cases, but there are also some potential issues such as reduced UDP throughput due to insufficient receiver processing capacity. The window size equals to the socket buffer and the author are using only 3 simulated ECUs. The Figures shows the throughput result of TCP and UDP protocol respectively. In the STAR-35 and STAR-3 communication structure of Mercedes Cars, there are TCP and UDP packages sent through the cable. The TCP packages consist of timestamp and a link can have more then 3 ECUs and the clock rate might be over 2 GB so the throughput of the Ethernet might be exceeding the baud rate of the cable even if the cable is able to support 1Gbit/s baud rate.

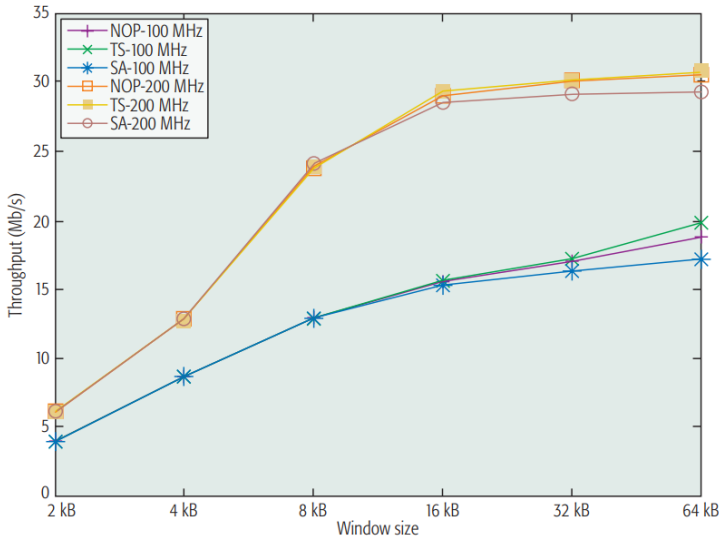


Figure Error! No text of specified style in document.**.4** TCP throughput with and without SACK and timestamp options. [17]

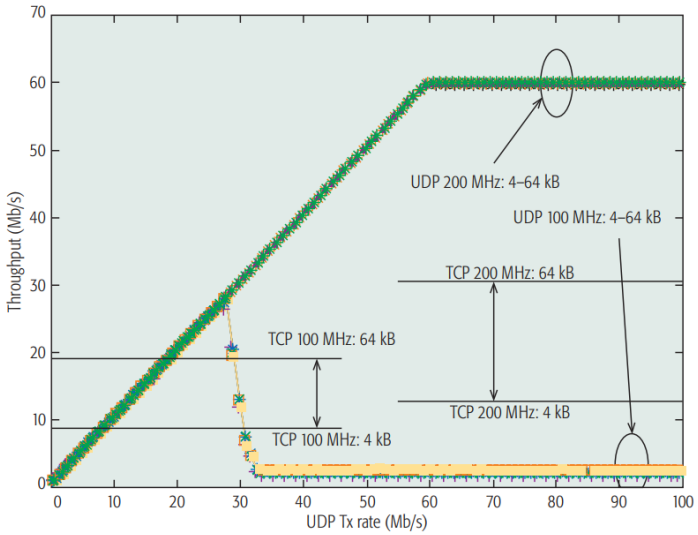


Figure Error! No text of specified style in document.**.5** UDP can significantly exceed TCP in throughput [17]

Existent Link Load Estimation Approachs

1.1.9 PREEvision

One method for estimating link load involves using Vector PREEvision, a specialized tool designed for analyzing E/E properties, including bus load calculations that can integrate data across various PREEvision layers through the use of NCDs. While PREEvision is beneficial for certain analyses, it proves inadequate for estimating link load due to its requirement for manual input of traffic amount for each link. This manual process is impractical, especially given the large number of links and the complexity of empirically estimating traffic amounts. Moreover, PREEvision lacks the capability to estimate the load on different VLANs within a single link. In contrast, the approach proposed in this paper simplifies the process. Once the model is constructed using

empirical data, it can automatically estimate link load by extracting relevant information directly from the NCD/ARXML files.

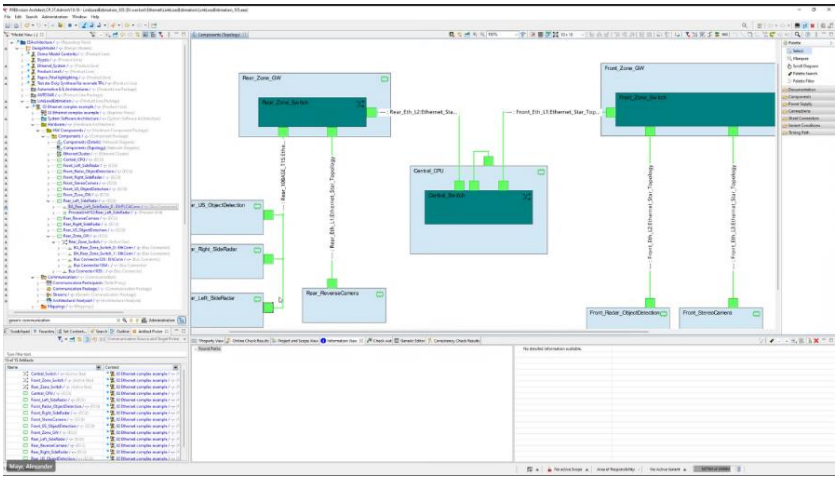


Figure Error! No text of specified style in document.**.6** Screenshot of the Communication Structures shown in PREEvision

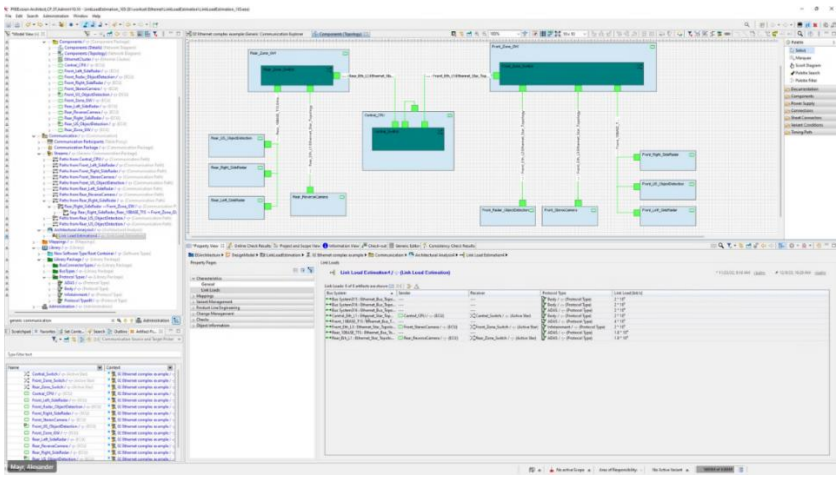


Figure Error! No text of specified style in document.**.7** Screenshot of calculated loads

1.1.10 Existent Tables for Load Calculation

CANFD Busload Overview: Contains information on frame instances, PDU instances, predicted send times, and durations, applicable when there is a CAN-FD cluster.

Ethernet Timing Overview: Formerly "EthernetBusloadOverview.java", this includes predicted timings and durations for Ethernet packets, used when there is an Ethernet cluster.

Routing Load Overview: Shows the load across network channels, including incoming and outgoing PDUs per second.

Network Load Overviews: Separate tables for general networking load and specific to Ethernet VLANs, showing best-case frame rates and data transfer rates.

Static Link Load in Ethernet

Static Bus Load Analysis in CAN-Busses

Although there is limited research on estimating automotive Ethernet link load, significant approaches exist for estimating link load in Controller Area Network (CAN) systems.

Thomas Nemenz [18] proposed three methods for calculating bus load performance in CAN busses in his paper, which are static Excel-based calculation method, the Generic-Busload Metrics within PREEvision software, and an 2/3 method.

Excel-Based Static Bus Load Calculation Method:

These method includes three steps.

a) Data Preparation: Relevant communication data is extracted from the vehicle network, including each frame's name, payload size, and transmission cycle. This data is typically stored in a NCD Excel spreadsheet.

b) Calculate Individual Frame Load: Using the payload size and transmission cycle for each frame, the method calculates the percentage of bus bandwidth occupied by each frame over a one-second interval. The frame size calculation includes both the message bits and any protocol overhead bits..

c) Summarize Total Load: The bus loads of all frames are aggregated to determine the total network load percentage.

This method is simple and intuitive, but it cannot handle the complexity of dynamic communication

Generic-Busload Metrics in PREEvision:

a) Modelling: An E/E architecture model is created in PREEvision, including all relevant ECUs, bus connections, and communication signals.

b) Configuration and Execution: The model's attributes, such as signal sizes, transmission cycles, and modes, are configured. The Generic-Busload Metrics tool is then run to calculate the bus loads.

c) Result Analysis: The tool generates detailed data on the load for each signal, PDU, and bus. These results enable users to analyze network performance and identify potential bottlenecks or overload situations.

This method is capable of handling complex E/E architectures and providing detailed load analysis. However, it requires extensive preconfiguration, which is actually impractical in real-world scenarios.

In general, the methods used for static bus load calculation in CAN networks can be adapted to the calculation of static link load in Ethernet networks. However, there are certain details that require careful consideration.

Methodology for Link Load Calculation in Ethernet

As discussed in previous sections, signals are transmitted in discrete units across a network via physical cables. The data is encapsulated in fundamental data containers known as frames. It is important to note that frames of the same type generally maintain a consistent size, except for those containing signals with dynamic arrays. When data is added to frames of consistent size, the unused fields are typically padded with a specific pattern to ensure data integrity.

Under these circumstances, the calculation of link loads can be systematically achieved in three steps by adapting methods from CAN network calculations.

First, the size of each frame type must be determined. This information can be obtained from ARXML files, which are meticulously designed by engineers to describe signal transmission. These files provide detailed specifications for each frame type, allowing for the derivation of frame lengths.

Second, the transmission frequency of each frame type must be obtained. For static frames, this information is readily available in the ARXML files. However, for dynamic frames, the transmission frequency must be estimated empirically, as it can vary depending on operational conditions.

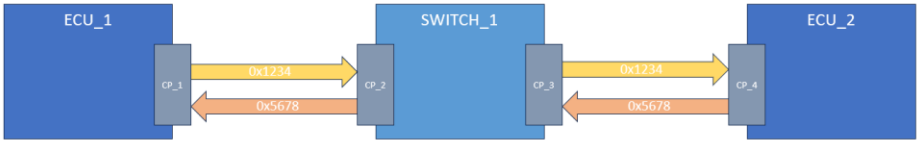
Finally, the total throughput is calculated using the frame sizes and transmission frequencies obtained in the first two steps.

Previous discussions have highlighted the different types of PDU transmissions, which can be categorized into static and dynamic. The initial focus is on static types, as they represent a more straightforward case for analysis. Since frame sizes are specified in the documentation, the process of determining frame size is equally applicable to dynamic link load estimation. The primary difference between the analysis of static and dynamic cases lies in the method used to obtain the transmission frequency, which, for dynamic cases, requires empirical estimation.

Ethernet Communication Model

Before estimating frame size, it is essential to model the communication process within Ethernet links.

When two ECUs are connected through a switch and a message is sent from ECU_1 to ECU_2 (e.g., a PDU with id 0x1234), that PDU will pass through four coupling ports. Figure 3.1 shows this procedure and the information regarding the coupling ports: To which ECU/Switch do they belong, which direction the information travels through that specific port, etc.



ECU/Switch	CouplingPort	CP-Direction	PDU-Header	Source	Target
ECU_1	CP_1	OUT	0x1234	ECU_1	ECU_2
SWITCH_1	CP_2	IN	0x1234	ECU_1	ECU_2
SWITCH_1	CP_3	OUT	0x1234	ECU_1	ECU_2
ECU_2	CP_4	IN	0x1234	ECU_1	ECU_2
ECU_1	CP_1	IN	0x5678	ECU_2	ECU_1
SWITCH_1	CP_2	OUT	0x5678	ECU_2	ECU_1
SWITCH_1	CP_3	IN	0x5678	ECU_2	ECU_1
ECU_2	CP_4	OUT	0x5678	ECU_2	ECU_1
Local perspective on transmission. The direction concerns the CouplingPort!			End-to-End perspective on transmission. Source has OUT direction; Target has IN Direction.		

Figure Error! No text of specified style in document.**.8** The communication between ECUs and switch from Mercedes internal website

A previous model was developed to extract and represent Ethernet communication information from an ARXML file. However, this model did not include the modeling of network links. The initial model creates data structures to store communication endpoints, endpoint connections, routes, and other pertinent information. It identifies communication endpoints for socket connections and SOME/IP services, categorizes them based on directionality, and establishes connections between sender and receiver endpoints.

To facilitate the estimation of link loads, an additional link model was integrated into this communication model. This link model represents the connection between two network ports, encapsulating details such as the physical characteristics (e.g., baud rate), associated VLAN memberships.

Moreover, transmission properties are added to each connection between sender and receiver endpoints, specifying the frame size and PDU transmission frequency.

Furthermore, a HashMap is employed to track the distribution of load across different VLANs as well as the total load associated with the link. With this enhanced model, AUTOSAR objects regarding Ethernet link load calculation can be efficiently referenced and manipulated, enabling comprehensive analysis of both static and dynamic link loads within the Ethernet communication network.

Estimation of Frame Size

In ARXML file, there is no explicit information regarding the size of frames transmitted across cables. To address this, it is essential to identify the outermost data packages specified within the ARXML files, thereby reducing the need for detailed protocol examination and complex calculations. Within these ARXML documents, the outermost data package is represented by PDU. Consequently, it is unnecessary to account for the individual sizes of signals. However, before progressing further, it is imperative to discuss the packet structure of the frames.

Frame Packet Structure

For a standard Ethernet packet, it is essential to account for all redundant bits introduced by the protocol formats during the packet encapsulation process. The widely recognized five-layer OSI model, as specified in ISO/IEC 7498-1, includes the following layers: Physical Layer, Data Link Layer, Network Layer, Transport Layer, and Session Layer.

In automotive Ethernet, the structure generally mirrors this model. As illustrated in Figure 3.2, different PDU types correspond to different session formats. These include the MAC protocol with VLAN, the IP protocol, and various transport protocols..

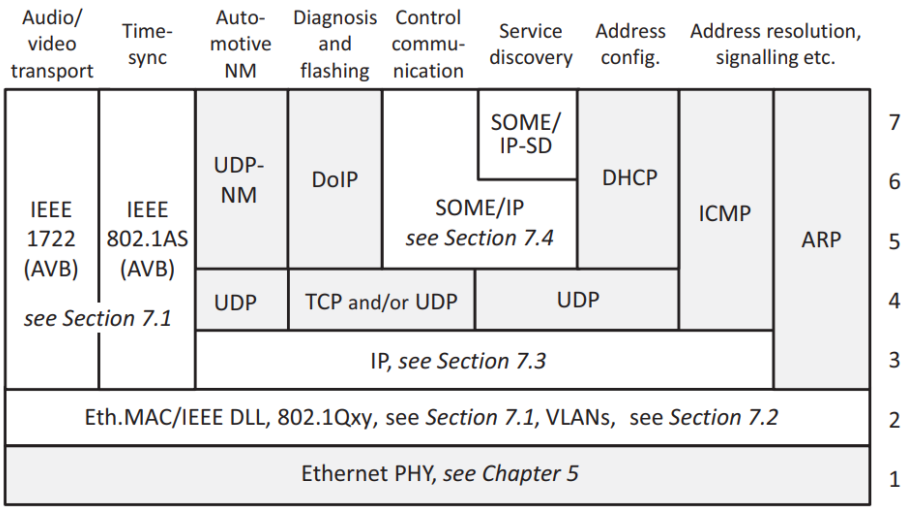


Figure Error! No text of specified style in document.**.9** Protocol overview for Automotive Ethernet [11]

Therefore, it is necessary to consider the packet formats specified at each layer of

the OSI model. Since the outermost encapsulation of signals within the ARXML model is typically represented by the PDU, PDU packet size is first calculated and all subsequent layers and their respective formats will be accounted for afterwards.

PDU Packet Size

The PDU packet formats are first considered at the session layer. The possible PDU types are defined according to the AUTOSAR System Template [13]:

ISignalIPdu:

Represents the IPdus handled by Com. The ISignalIPdu assembled and disassembled in AUTOSAR COM consists of one or more signals. This is the basic type od PDU.

NmPdu:

Network Management PDU for managing network-related communication and activities within a vehicle's communication network.

GeneralPurposeIPdu:

This element is used for AUTOSAR PDUs without attributes that are routed by the PduR. That means, GeneralPurposeIPdu is utilized for a variety of purposes within the automotive communication framework, particularly those that do not fall under more specific categories like ISignalIPdu or NmPdu. For GeneralPurposeIPdu, the categories are standardized in the AUTOSAR System Template [13]:

SOMEIP_SEGMENTED_IPDU: For UDP, The SOME/IP message too immense to be transported directly with UDP. Therefore, the PDUs are segmented into pieces and transported.

DoIP: Diagostic information

SD: Service Discovery, which communicates the availability of functional entities as well as controlling the send behavior of event messages

XCP: PDUs of parameters and measured variables read from the memory of an ECU.

SecuredIPdu:

A Secured I-PDU is an AUTOSAR I-PDU that contains Payload of an Authentic I-PDU supplemented by additional Authentication Information. The separate

Authentic I-PDU is described by the PDU that is referenced with the payload reference from this SecuredIPdu.

Different types of PDUs have distinct headers, resulting in varying data lengths, even if the payloads are identical. The payload lengths of the PDUs can be determined from the <LENGTH> attribute associated with each PDU.

IsignalIPdu:

The header length is 8 bytes, as specified from the Mercedes document.

NmPdu:

The header length is 8 bytes, derived from Figure 3.3 from AUTOSAR_SWS_UDP NetworkManagement [19]

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CBV	NID	User Data		PNC bit vector			
0x40	0x00	0xFF	0xFF	0x12	0x8E	0x80	0x01

Figure Error! No text of specified style in document.**.10** Header format of NmPdu [19]

GeneralPurposePdu:

There are various categories of PDUs under this type. By checking the <CATEGORY> attribute, the category of a certain PDU can be determined.

For different categories:

a) SOMEIP-SEGMENTED-IPDU:

The header size equals to 20 bytes according to the Figure 4.19 in SOME/IP Protocol Specification: [20]

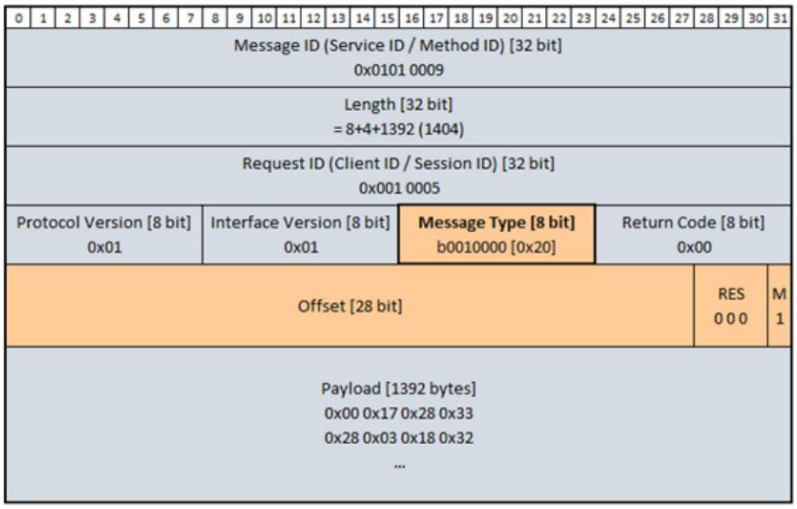


Figure Error! No text of specified style in document.**.11** Header format of SOMEIP-SEGMENTED-IPDU [20]

DoIP-PDU, SD-PDU and XCP-PDU:

The network layer package size equals to the maximum transmission unit size (EthernetCommunicationConnector.maximumTransmissionUnit) minus the sum of IP header, UDP header as well as Socket Adaptor PDU Header according to TPS_SYST_02117 due to paddings from the AUTOSAR_TPS_SystemTemplate [13]

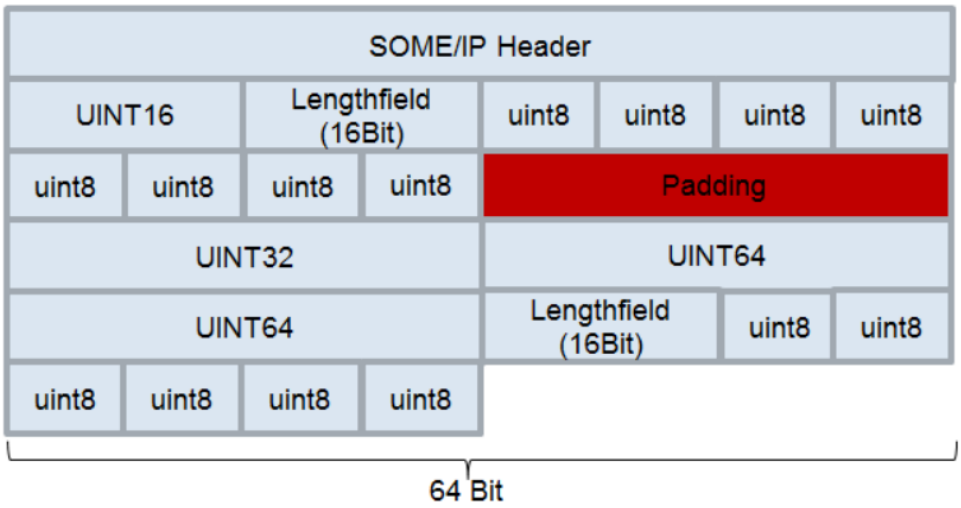


Figure Error! No text of specified style in document.**.12** SOME/IP Padding Example [21]

Notice that the SOME/IP header is already included in the header of category SOMEIP-SEGMENTED-IPDU, and is neglected due to paddings for DoIP-PDU, SD-PDU and XCP-PDU.

SecureIPdu:

The header length is 16 bytes, which is given by Mercedes documents.

a) SecOC [13]:

Aside from the fixed PDU header, there is also SecOC standard protocol which appends a security header to each PDU, which includes authentication and freshness information to verify that the message has not been tampered with and is from a legitimate source and prevents replay attacks by incorporating counters or timestamps. The length of these additional information is indicated in the <SECURE-COMMUNICATION-AUTHENTICATION-PROPS> <AUTH-INFO-TX-LENGTH> and <SECURE-COMMUNICATION-FRESHNESS-PROPS> <FRESHNESS-VALUE-TX-LENGTH> leaves which are referenced by each PDU instances.

b) E2E [13]:

Another protocol is the AUTOSAR End-to-End (E2E) protocol, which is a standardized method to ensure the integrity and authenticity of communication between ECUs. It uses Cyclic Redundancy Check (CRC), counters, and data identifiers to detect errors, prevent data tampering, and protect against replay attacks. For certain PDUs, E2E header will be added to messages if the PDU reference of <I-SIGNAL> is referencing a certain <TRANSFORMATION-TECHNOLOGY> with <PROTOCOL> value of E2E.

The length of the E2E header is specified in the <BUFFER-PROPERTIES>

<HEADER-LENGTH> leaf. The values are in bits.

Other Protocol Format

Transport layer headers:

UDP consumes a length of 8 bytes and TCP of 30 bytes (containing time stamp). Whether a PDU should be using UDP or TCP is specified in the <SOCKET-ADDRESS> <APPLICATION-ENDPOINT> <TP-CONFIGURATION> leaf [13].

Network layer headers:

There are IPv4 and IPv6 formats.

The version of IP protocols used can be derived from whether from the <NETWORK-MASK> leaf or the simply the network addresses under <NETWORK-ENDPOINT-ADDRESSES> [13].

For automotive Ethernet, only IPv4 with no additional options is used, therefore the header size is 20 bytes [22].

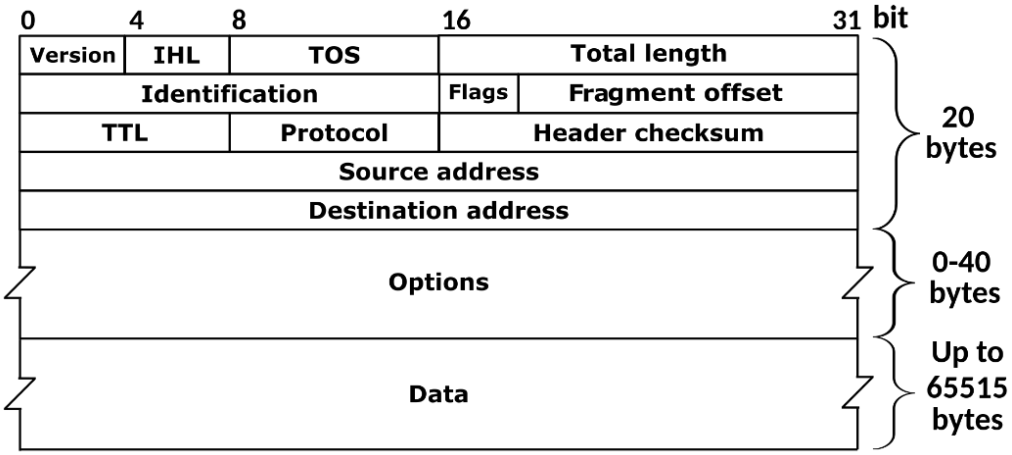


Figure Error! No text of specified style in document.**.13** IPv4 format [22]

Ethernet layer header:

Figure 3.7 shows the Ethernet layer format with VLAN.

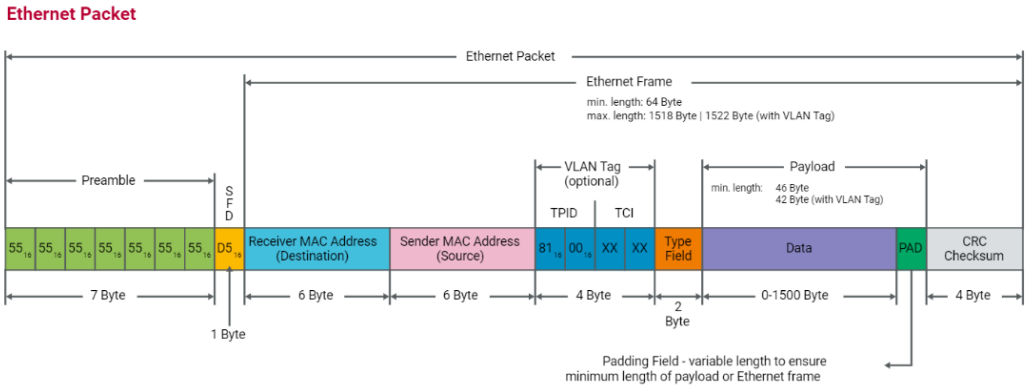


Figure Error! No text of specified style in document.**.14** Ethernet packet format [11]

Since the smallest Payload data = 20 + 8 + 8 + 7 = 43 Bytes > 42 Bytes, the padding can be neglected.

Therefore, the Ethernet layer header size is 30 bytes.

Socket Adpator header:

As shown in Figure 3.8, for certain VLANs, there are socket adaptors over the TCP protocol.

The Socket Adaptor (SoAd) module in AUTOSAR provides a standardized Socket API that enables upper-layer applications and services to communicate over networks without dealing with the complexities of the underlying TCP/IP stack. This can be used anywhere inside automotive Ethernet.

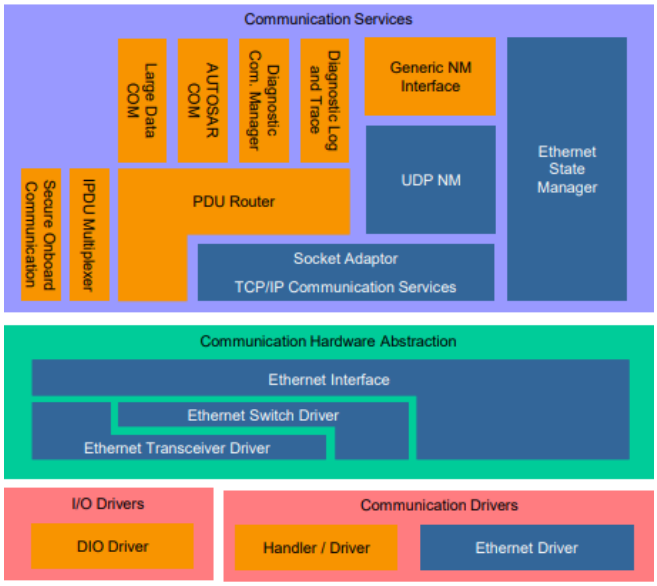


Figure Error! No text of specified style in document.**.15** AUTOSAR Layered Software Architecture for Diagnostic PDUs [8]

The SoAd PDU header shall consist of a 4 byte ID field for unique identification of the PDU at the receiver and a 4 byte length field specifying the data length of the PDU, according to [SWS_SoAd_00198] in Specification of Socket Adaptor. [23]

Whether a VLAN is using SoAd can be determined by checking if there is <SO-AD-CONFIG>leaf under the VLAN.

SOME-IP related:

a) Dynamic Array [24]:

A dynamic array can adjust its size during runtime, making it ideal for storing signals with significant fluctuations in size over time, such as those used for transporting logs. It is important to note that the length header is added before the signals themselves, rather than before the PDUs. The maximum length of all signals, including the length field, is already accounted for within the PDU's maximum length. Therefore, for simplicity, the individual length headers will be disregarded, and the maximum length specified for each PDU will be used.

b) Alignment [13]:

Some signals need to be aligned in the memory to improve the efficiency of memory access and processing by the CPU. The memory address multiple could be 4, 8 or other numbers. However, since all paddings are considered in PDU sizes, this part can also be neglected.

Total Frame Size

After obtaining all the necessary values, the total size of the network packet can be determined as follows:

$$N_{Network} = \begin{cases} N_{MTU} & if\ PDU\ \in DoIP, SD, XCP \\ N_{PDULen} + N_{PDUHeader} + N_{Trans} + N_{IP} & else \end{cases} \quad (3.1)$$

N_{Frame}: Number of Bytes of a frame [Byte]

N_{PDULen}: Number of Bytes of a PDU payload [Byte]

$N_{PDUHeader}$: Number of Bytes of a PDU header [Byte]

N_{Trans} : Number of Bytes added in the transportation layer [Byte]

N_{IP}: Number of Bytes added in the network layer [Byte]

And the frame size is:

$$N_{Frame} = \begin{cases} N_{network} + N_{Eth} + N_{SoAd} & if\ SoAd\ is\ used \\ N_{network} + N_{Eth} & if\ SoAd\ is\ not\ used \end{cases} \quad (3.2)$$

N_{Eth}: Number of Bytes added in the Ethernet layer [Byte]

N_{SoAd} : Number of Bytes of a socket adaptor [Byte]

The inter frame gap of 12 bytes should also be added. Thus, the total frame data in bit is:

$$N_{FrameWithGapPerBit} = 8 \cdot (N_{Frame} + 12) \quad (3.3)$$

Further Conditions

Sending Frequency

The cyclic timing period can be obtained from the <TIME-PERIOD><CYCLIC-TIMING> in the <TRASMISSION-MODE-TRUE-TIMING><TRASMISSION-MODE-DECLARATION><PDU-TIMING> leaf of the each PDU [13].

The frame sending frequency can be considered as the number of PDUs sent per second by taking the reciprocal of the cyclic timing period. Only the PDUs of transmission type “Cyclic” or “Dual Cyclic” are considered. The transmission types of the PDUs are implied in the ARXML files by the combination of values in the leaves

<EVENT-TIMING> leaf, <NUMBER-OF-REPETITIONS>, <TRASMISSION-MODE-TRUE-TIMING>, <TRASMISSION-MODE-FALSE-TIMING> [13].

Baud Rate

The transmission baud rate is derived from the value in the <PHYSICAL-LAYER-TYPE> and <MAC-LAYER-TYPE> [13]. In this leaves different Ethernet standard of the ports are implied, for example 1000BASE-T, 100BASE-TX, etc.

There are 8 cases of Baud rate [13]:

1000BASE-T: Ethernet Standard (IEEE 802.3ab) to support 1 Gbit/s over 4 twisted pairs.

1000BASE-T1: Ethernet Standard (IEEE 802 3bp) to support 1 Gbit/s over a single twisted pair cable.

100BASE-T1: Ethernet Standard (IEEE 802.3bw) to support 100 Mbit/s over a single twisted pair cable. 100BASE-T1 is the IEEE Standardized version of BroadRReach.

100BASE-TX: Ethernet Standard (IEEE 802.3u) to support 100Mbit/s over two twisted pairs.

10BASE-T1S: Physical layer interface (10 Mbit/s, 2 pairs). Used for automotive.

X-MII: Mac layer interface (data) bandwidth class 10-100 Mbit/s (e.g., RMII; RvMII, SMII, RvMID

XG-MII: Mac layer interface (data) bandwidth class 1 Gbit/s (e.g., GMII, RGMII, SGMII, RGMII, USGMII)

XXG-MII: Mac layer interface (data) bandwidth class 10 Gbit/s

Redundant Frames

Redundant frames are transmitted because of error occurrence during transmission. Therefore, error rate should also be considered. According to the BroadReach standard, in the physical layer, the possibility of error occurrence is $10^4 \sim 10^6$ [24]. Therefore, the static frame error variable is chosen to be 0.0001.

Calculation of Static Link Load and Table Establishment

Calculation

The Frame Data Load per Link is:

$$Load_{FrameTrans} = \frac{N_{FrameWithGapPerBit} \cdot (1 + sFE)}{Baudrate} \cdot \frac{1}{T_{cycle}} \quad (3.4)$$

$Load_{FrameTrans}$: Load generated by a Frame transmission. Specified as a ´BigDecimal´ numbers (arbitrary-precision decimal)

$N_{FrameWithGapPerBit}$: Number of bits in one frame data contained in the frame transmission, considering frame gaps[bit]

$Baudrate$: Transmission speed of the link [Bit/s].

sFE: Statical frame error rate

T_{cycle}: Cycle time of the signal transmission [s].

The first fraction calculates the ratio of occupied to free bus time, while the second fraction determines the frequency of frame transmissions per second.

By summing the overall PDU data loads across a specific link, the total load on that link is calculated as a percentage:

$$Load_{total} = \sum Load_{FrameTrans} \cdot 100 \quad (3.5)$$

Virtual Networks

A VLAN (Virtual Local Area Network) is a networking technology that allows for the logical segmentation of a single physical network into multiple virtual networks. Each VLAN functions as an independent network, even though it shares the same physical infrastructure with other VLANs.

Single Tagged VLAN: The Single Tagged VLAN, as per the IEEE 802.1Q standard, supports the tagging of each network packet with a VLAN identifier (VLAN ID). This tagging allows for the segmentation of network traffic into distinct logical networks by adding VLAN tags to Ethernet frames, enhancing network security and management.

VLAN ID Assignment: According to Mercedes standards, all vehicle communication is to be assigned to specific VLAN-IDs. This ensures that different types of vehicle communication are logically separated within the network infrastructure, aiding in traffic prioritization and network optimization.

No Retagging of VLAN IDs: It is specified in Mercedes standards that VLAN IDs should not be retagged, except for mirroring purposes. This means that once a VLAN ID is assigned to a packet, it should remain unchanged throughout its journey within the network, unless specifically required for monitoring or analysis purposes.

No Untagging of VLAN IDs: Mercedes standards mandates that VLAN IDs should not be untagged, except for egress traffic destined for external vehicle access. This ensures that VLAN-tagged packets maintain their integrity and remain associated with their respective VLANs until they reach their intended destination.

Since all frames are mandated to have a VLAN tag, it is necessary to calculate the link load according to each VLAN tag.

Since all frames are required to have a VLAN tag, it is necessary to calculate the link load according to each VLAN tag. By reading the values in the elements <COMMUNICATION-CONNECTOR-REF>, <COMMUNICATION-CONNECTOR-REF-CONDITIONAL>, <COMM-CONNECTORS>, and <PHYSICAL-CHANNEL>, the corresponding VLANs of the PDU can be identified [13]. This allows for the calculation of link load across each individual VLAN.

$$Load_{totalPerVLAN} = \sum Load_{FrameTransPerVLAN} \cdot 100 \quad (3.6)$$

Table Establishment

By combining the results from the previous analyses, a comprehensive table can be established that displays the static link loads both overall and for each VLAN. The calculations and modeling code used to derive these results are included in the appendix for reference.

It is important to note that in some links, the calculated values may be zero. This is because the dynamic link load has not been considered in these instances.

Table 3.1 presents the static link loads for different links with both the overall load and the distribution of load across different VLANs, while Figure 3.9 illustrates the graphical representation of this static link load table shown in the ARXML Visualizer.

Table Error! No text of specified style in document.**.2**: Part of the static link load table

Start of link	End of Link	Baudrate	Load in Percent (%)	MAIN_1 in Percent (%)	MAIN_10 in Percent (%)	MAIN_100 in Percent (%)	MAIN_101 in Percent (%)	MAIN_110 in Percent (%)	MAIN_111 in Percent (%)	MAIN_120 in Percent (%)	MAIN_127 in Percent (%)
A	B	100000000	0.36472	0.00000	0.10038	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C	D	100000000	2.05773	0.00000	2.05701	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
E	B	100000000	0.67771	0.00000	0.02517	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
F	B	100000000	1.71610	0.00000	0.10536	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
G	B	100000000	2.19817	0.00000	0.01757	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
H	B	1000000000	0.36300	0.00000	0.05146	0.00000	0.00000	0.00000	0.00000	0.00000	0.00496
B	I	100000000	3.10464	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

B	G	100000000	1.28382	0.00000	0.00288	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
B	A	100000000	0.54301	0.00000	0.22669	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
B	D	100000000	11.18057	0.00000	5.56849	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	J	100000000	4.40601	0.00000	4.40457	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	F	100000000	4.92813	0.00000	1.83339	0.00000	0.00000	0.00000	0.00000	0.00000	0.02557
B	K	100000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	L	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	M	100000000	0.53672	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	N	100000000	7.24524	0.00000	0.24300	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	O	100000000	0.02824	0.00000	0.00627	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
B	E	100000000	0.01128	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
B	H	1000000000	2.53233	0.23093	0.71769	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
L	B	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
P	M	1000000000	0.05498	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
M	P	1000000000	0.01745	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
M	Q	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
M	B	100000000	0.11538	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
M	R	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
S	D	1000000000	0.62023	0.00000	0.05917	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
T	D	10000000000	0.02935	0.00000	0.02934	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	C	100000000	0.01643	0.00000	0.01568	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
D	B	100000000	2.09682	0.00000	1.17605	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
D	T	10000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	S	1000000000	0.20945	0.00000	0.11745	0.00000	0.00000	0.00000	0.00000	0.00000	0.00015
U	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
V	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
W	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
X	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Y	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Z	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AB	AC	100000000	2.62772	0.00000	2.62700	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AC	J	100000000	0.13170	0.00000	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AC	AB	100000000	0.13170	0.00000	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AD	J	100000000	1.78149	0.00000	1.78077	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	AC	100000000	2.62772	0.00000	2.62700	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	B	100000000	0.13603	0.00000	0.13453	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
J	AD	100000000	0.00754	0.00000	0.00678	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AE	1000000000	0.29762	0.00000	0.00326	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
N	X	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	W	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AP	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AF	1000000000	0.39852	0.00000	0.03942	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	AG	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	B	100000000	4.09728	0.00000	0.42685	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
N	V	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AH	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	U	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AE	N	1000000000	0.20081	0.00000	0.00129	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AH	N	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AF	N	1000000000	0.53804	0.00000	0.02301	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	B	100000000	0.02115	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00451
I	AI	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	Y	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AJ	1000000000	0.19584	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
I	AK	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AL	1000000000	0.19762	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	Z	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AM	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AJ	I	1000000000	0.13031	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AI	I	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AN	AL	1000000000	0.18318	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AL	AO	1000000000	0.00177	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AL	I	1000000000	0.00513	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AL	AN	1000000000	0.00336	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AO	AL	1000000000	0.01444	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AP	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
O	B	100000000	1.21525	0.00000	0.00442	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AG	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Q	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
R	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AM	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

AK	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
K	B	100000000	2.30926	2.30926	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Start of link	End of Link	Baudrate	Load in Percent (%)	MAIN_1 in Percent (%)	MAIN_10 in Percent (%)	MAIN_100 in Percent (%)	MAIN_101 in Percent (%)	MAIN_110 in Percent (%)	MAIN_111 in Percent (%)	MAIN_120 in Percent (%)	MAIN_12
[illegible]	[illegible]	100000000	0.34683	0.00000	0.08890	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	2.05773	0.00000	2.05701	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	0.63901	0.00000	0.02447	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	1.59839	0.00000	0.10210	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	1.78553	0.00000	0.01757	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	1000000000	0.35974	0.00000	0.05007	0.00000	0.00000	0.00000	0.00000	0.00000	0.00489
[illegible]	[illegible]	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	0.01128	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
[illegible]	[illegible]	100000000	4.37654	0.00000	4.37510	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	4.73939	0.00000	1.82862	0.00000	0.00000	0.00000	0.00000	0.00000	0.02557
[illegible]	[illegible]	100000000	7.20201	0.00000	0.24150	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	0.49512	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	0.54112	0.00000	0.22669	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
[illegible]	[illegible]	100000000	3.15555	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	11.13808	0.00000	5.54100	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	1000000000	2.39702	0.23093	0.71588	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	1.28382	0.00000	0.00288	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
[illegible]	[illegible]	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	1000000000	0.05082	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
[illegible]	[illegible]	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
[illegible]	[illegible]	100000000	0.11538	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
[illegible]	[illegible]	1000000000	0.01745	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
[illegible]	[illegible]	1000000000	0.61781	0.00000	0.05825	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	10000000000	0.02916	0.00000	0.02916	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	0.01643	0.00000	0.01568	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
[illegible]	[illegible]	1000000000	0.19446	0.00000	0.11130	0.00000	0.00000	0.00000	0.00000	0.00000	0.00015
[illegible]	[illegible]	100000000	1.94692	0.00000	1.11456	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
[illegible]	[illegible]	10000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	2.61217	0.00000	2.61145	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	0.12856	0.00000	0.12781	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
[illegible]	[illegible]	100000000	0.12856	0.00000	0.12781	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
[illegible]	[illegible]	100000000	1.76606	0.00000	1.76534	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	0.13440	0.00000	0.13290	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
[illegible]	[illegible]	100000000	2.61217	0.00000	2.61145	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	0.00754	0.00000	0.00678	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
[illegible]	[illegible]	1000000000	0.37328	0.00000	0.03942	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
[illegible]	[illegible]	100000000	0.01771	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
[illegible]	[illegible]	100000000	0.01771	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
[illegible]	[illegible]	100000000	3.88008	0.00000	0.42685	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
[illegible]	[illegible]	100000000	0.01771	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075

Figure Error! No text of specified style in document..**16** Static link load table in ARXML Visualizer

Dynamic Link Load Estimation in Ethernet

Dynamic Bus Load Analysis in CAN-Busses

In the study by Thomas Nemenz and Oliver Lein [18], a method for calculating dynamic bus load is also proposed, known as the $2/3$ method.

The fundamental concept of this method is that, of this approach is based on the observation that, in dynamic communication environments, approximately $2/3$ of potential dynamic messages are likely to be transmitted, while the remaining $1/3$ may not be sent due to unmet conditions or other factors. The method involves the following steps:

Message Classification: The communication messages within the network are categorized into static and dynamic groups. Static messages, which are transmitted at fixed intervals, are calculated using traditional methods. In contrast, dynamic messages, whose transmission is less predictable, are estimated using the $2/3$ method.

Dynamic Load Calculation: It is assumed that $2/3$ of the dynamic messages will be transmitted within a given time cycle. The maximum possible dynamic load is first determined by calculating the theoretical load if all dynamic messages were to be transmitted. The estimated bus load is then derived as two-thirds of this maximum possible dynamic load.

Total Load Summation: The total network load is obtained by summing the static load and the dynamically estimated load calculated using the $2/3$ method.

This method simplifies the complexities associated with dynamic communication, maintains a balance between accuracy and computational efficiency. However, it is still having a lot of limitations. The $2/3$ method relies on assumptions derived from empirical observations, which may not be true across different scenarios and bus types. Its accuracy is also compromised by the assumption that the dynamic link load consistently consumes two-thirds of the maximum potential load. Moreover, the method does not account for the possibility that the actual load could exceed the estimated value provided by this approach. Consequently, while the $2/3$ method serves as a useful approximation, there is a compelling need for more precise methods to accurately calculate Ethernet link load, especially to guide engineers in the design and optimization of automotive network systems.

Data Collection

To more accurately analyze the dynamic behavior of PDUs, it is essential to collect real-world data from vehicles. At Mercedes, extensive databases are available that contain recorded bus communications from various test drives. The data utilized in this analysis was collected from a single vehicle.

The available dataset comprises over 10,000 traces from different vehicles, vehicle types, and topologies. For this study, two specific traces were selected from the same vehicle but under significantly different driving conditions. The following figures illustrate the vehicle status for the two selected traces used in the analysis.

Figure 4.2 displays the signal activity over time corresponding to Trace 1, illustrating the vehicle's status during testing from April 3, 2024, to April 9, 2024. Throughout this period, the vehicle remained stationary. In contrast, Figure 4.3 presents the signal activity for Trace 2, showing a different vehicle status during testing from May 28, 2024, to June 21, 2024. During this time, the vehicle was driven for approximately one and a half hours.

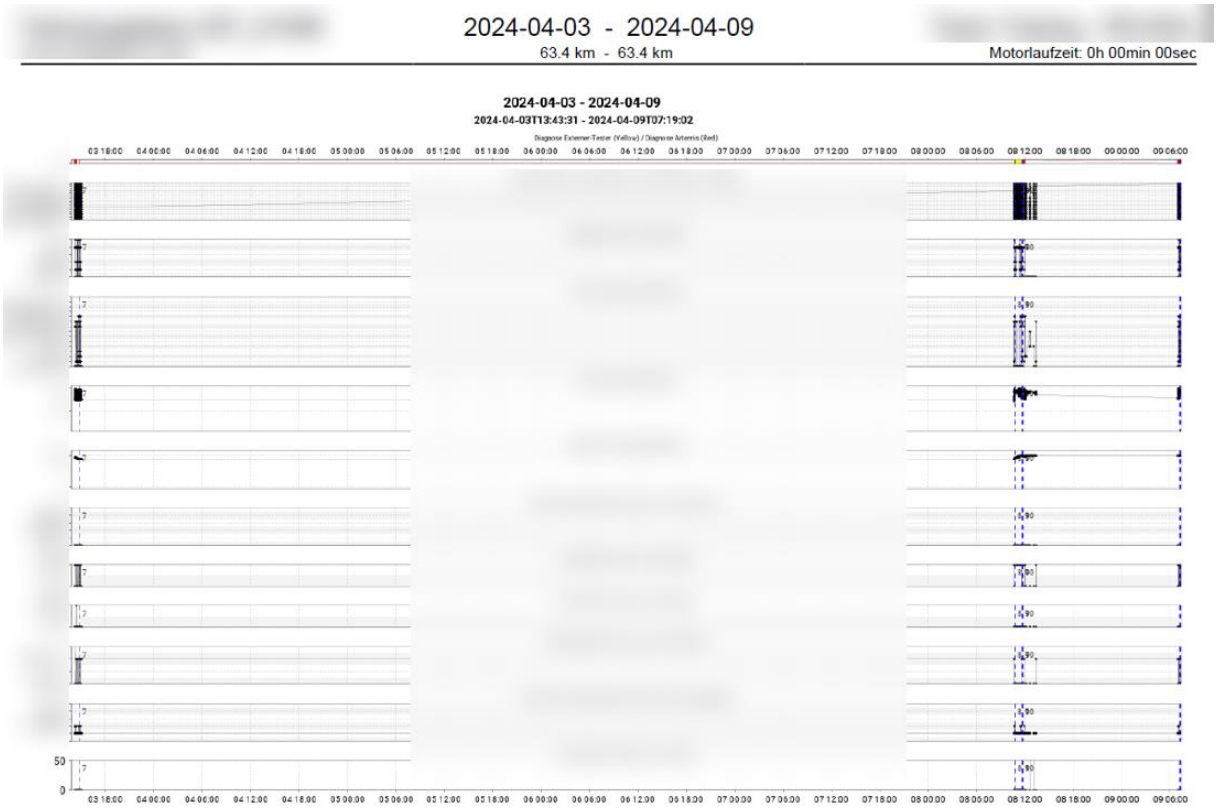


Figure Error! No text of specified style in document.**.17** Database result from the first .mfx file of trace 1

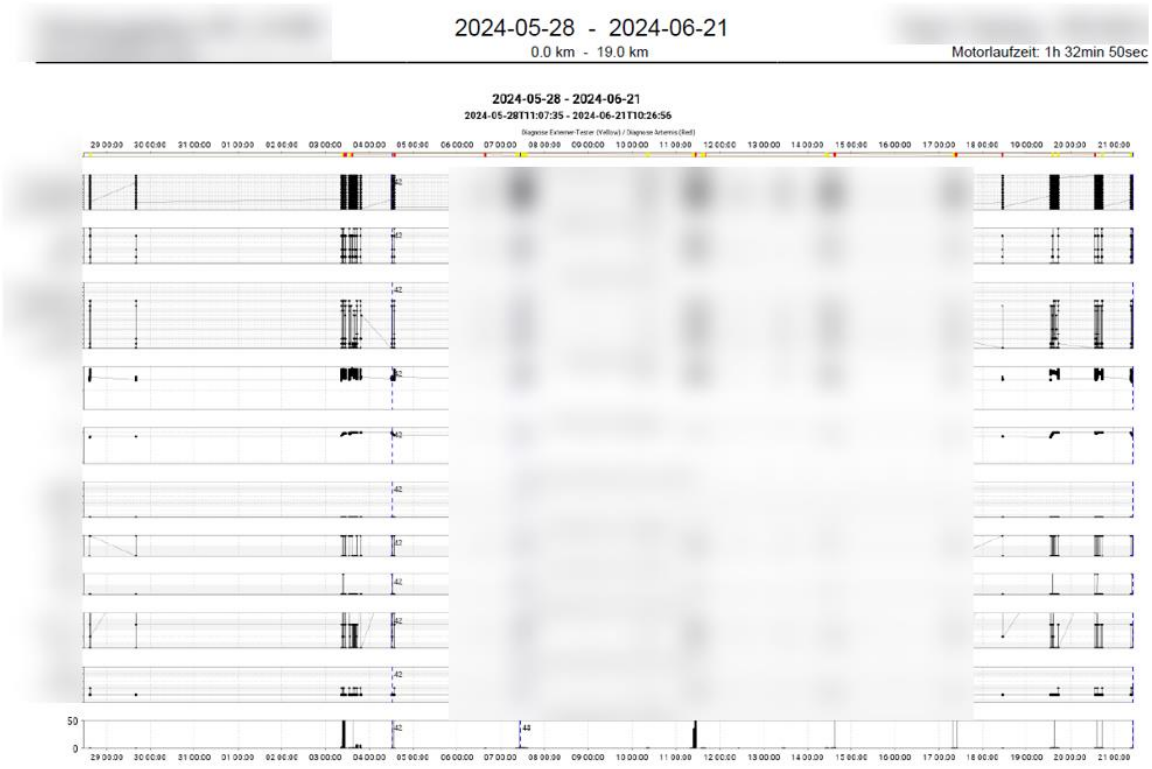


Figure Error! No text of specified style in document.**.18** Database result from the first .mfx file of trace 2

The original files are in the .mfx format, which is not human-readable. While they can be converted using the Xoraya tool, the process is both cumbersome and highly time-consuming, requiring over eight hours to process a single trace file. Consequently, a significant challenge lies in converting these files into a human-readable format. To address this, the internal tool "Bug Hunter" was employed. This tool is capable of interpreting PDU names, VLAN tags, timestamps, and other relevant information from the .mfx files according to the corresponding ARXML file. By developing a Python script to interpret the trace files in batches and organize the data into an SQLite database, the conversion process becomes significantly more efficient and structured. Additionally, the script collects the number of PDUs transmitted within 10-second intervals, facilitating a more straightforward analysis of PDU frequency.

Figure 4.4 illustrates the interface in "Bug Hunter" where the .mfx files are converted. Python scripts are developed within the “Bug Hunter Vault” and executed concurrently in the terminal below as the trace files are being interpreted. These two processes are carried out simultaneously due to the limited RAM size.

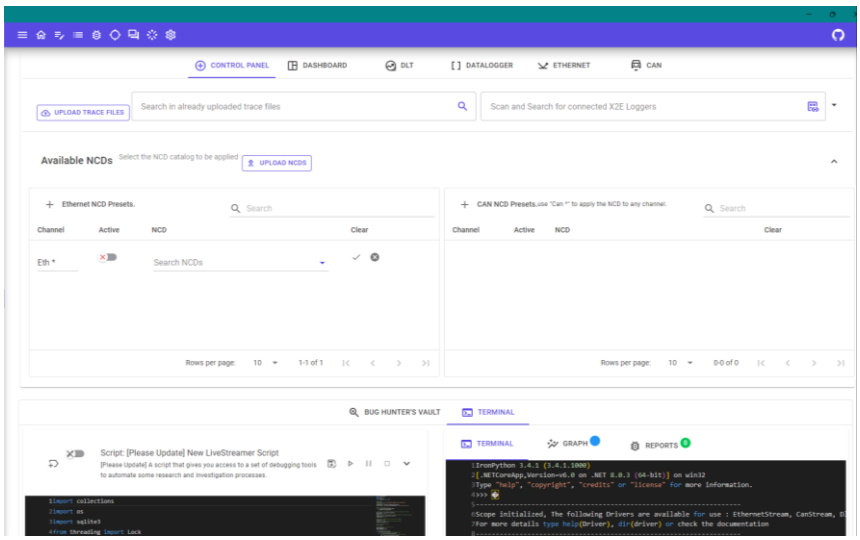


Figure Error! No text of specified style in document.**.19** Bug Hunter Interface

Table 4.1 displays the converted data of trace files. The table includes timestamp interval which is the time range during which the data was captured, PDU Name from the corresponding ARXML file, VLAN tag associated with each PDU, frequency of PDU transmissions within 10 seconds, datetime interval and PDU ID.

Table Error! No text of specified style in document.**.3:** Part of the converted data

timestamp_interval	ipdu_name	vlan	frequency	datetime_interval
1903.4847992-1913.5186963000001	A	10	216	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	B	20	2287	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	C	10	162	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	D	20	1479	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	E	20	4852	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	F	10	1954	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	G	10	196	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	H	10	978	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	I	20	1461	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	J	10	63	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	K	10	374	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	L	20	297	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	M	20	2985	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	N	10	1976	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	O	10	978	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	P	20	306	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	Q	10	288	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	R	20	437	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	S	20	212	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	T	10	485	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	U	20	2985	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	V	20	19546	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	W	10	2943	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	X	10	3723	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	Y	10	2008	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM

1903.4847992-1913.5186963000001	Z	20	2925	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	A	20	469	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	B	10	792	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	C	10	1424	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	D	20	18003	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	E	20	396	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	F	10	126	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	G	20	198	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	H	10	4880	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	I	20	1952	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	J	20	849	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	K	20	1179	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	L	10	674	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM
1903.4847992-1913.5186963000001	M	20	4875	4/3/2024 12:15:15 PM - 4/3/2024 12:15:25 PM

To ensure a more accurate analysis, the converted data was initially segmented into distinct time periods. The first level of segmentation was based on the date each PDU (Protocol Data Unit) was transmitted, allowing for daily data to be treated as separate entities. Subsequently, a finer segmentation was performed by examining the time intervals between consecutive PDU series. Specifically, if the time gap between two adjacent PDU series exceeded 120 seconds, this was interpreted as an indication of vehicle idleness or a possible interruption in data collection. Such a substantial time gap suggests a potential shift in context or scenario, thereby necessitating that the data preceding and following the gap be treated as distinct, non-continuous sequences.

Following segmentation, the data within each identified interval was meticulously compiled. PDUs from different virtual networks were treated as separate entities, each tagged with a corresponding VLAN identifier. To facilitate subsequent analyses, the final cleaned dataset was converted to a .npy file format, which enhances processing speed in downstream tasks. Given the vast volume of data, SQLite was employed for data management. The structured data, in word format, is presented follows:

Table Error! No text of specified style in document.**.4:** Part of the table with PDU names, time sent and PDU numbers

Timestamp/Signal Names	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
2024-04-03 12:15:59	806	85	90	782	173	180	876	450	0	1239	6264	2682	6264	0	2222	477	18	2656	0	0
2024-04-03 12:16:02	669	86	89	772	159	180	878	450	0	1225	6253	2679	6253	0	2250	442	18	2657	0	0
2024-04-03 12:16:03	564	77	80	699	125	162	788	405	0	1099	5614	2406	5614	0	1985	378	16	2360	0	0
2024-04-03 12:16:04	566	84	87	763	123	190	874	475	0	1204	6113	2619	6113	0	2170	422	18	2597	0	0
2024-04-03 12:16:05	476	83	86	745	96	170	864	425	0	1190	6027	2583	6027	0	2175	387	18	2594	0	0
2024-04-03 12:16:06	469	84	95	755	97	170	852	425	0	1197	6022	2580	6022	0	2170	360	9	2576	0	0
2024-04-03 12:16:09	277	94	87	753	107	170	872	425	0	1197	6041	2589	6041	0	2140	285	0	2595	0	0
2024-04-03 12:16:12	414	85	87	760	116	188	880	470	0	1211	6111	2619	6111	0	2165	323	0	2619	0	0

2024-04-03 12:16:13	415	78	80	700	120	174	810	435	0	1113	5644	2418	5644	0	2030	312	0	2442	0	0
2024-04-03 12:16:14	510	87	89	786	161	178	884	445	0	1232	6251	2679	6251	0	2245	340	0	2685	0	0
2024-04-03 12:16:15	601	88	90	792	180	180	894	450	0	1253	6344	2718	6344	0	2240	378	0	2691	0	0
2024-04-03 12:16:16	609	87	90	783	179	180	908	450	0	1309	6344	2718	6344	0	2245	378	0	2712	0	0
2024-04-03 12:16:20	790	88	98	785	180	180	896	450	0	1309	6393	2739	6393	0	2275	351	0	2718	0	0
2024-04-03 12:16:22	785	87	90	808	154	180	886	450	0	1234	6304	2700	6304	0	2275	345	0	2706	0	0
2024-04-03 12:16:23	712	77	80	724	150	160	784	400	8	1101	5601	2399	5601	8	1990	282	0	2376	0	0
2024-04-03 12:16:24	827	86	90	812	161	180	898	450	9	1241	6300	2698	6300	9	2240	318	0	2694	0	0
2024-04-03 12:16:25	836	95	90	840	169	180	900	450	9	1234	6223	2665	6223	9	2240	291	0	2697	0	0
2024-04-03 12:16:26	829	87	90	842	161	180	884	450	9	1234	6270	2686	6270	9	2270	291	0	2676	0	0
2024-04-03 12:16:30	883	85	90	890	158	180	888	450	9	1229	6223	2665	6223	9	2245	318	0	2676	0	0
2024-04-03 12:16:32	882	85	90	887	191	180	872	450	9	1239	6240	2676	6240	9	2225	321	0	2640	0	0
2024-04-03 12:16:33	784	75	80	781	154	160	778	400	0	1092	5505	2359	5505	0	1970	285	0	2349	0	0
2024-04-03 12:16:34	883	85	90	891	174	180	876	450	0	1232	6198	2656	6198	0	2220	348	0	2643	0	0
2024-04-03 12:16:35	883	85	90	893	182	180	876	450	0	1237	6266	2686	6266	0	2220	375	0	2643	0	0
2024-04-03 12:16:36	882	84	91	884	173	180	870	450	0	1237	6280	2692	6280	0	2260	375	0	2640	0	0
2024-04-03 12:16:40	886	84	89	879	173	180	870	450	0	1249	6285	2695	6285	0	2260	426	0	2652	0	0
2024-04-03 12:16:42	890	85	89	881	173	180	888	450	0	1244	6287	2693	6287	0	2235	456	0	2664	0	0
2024-04-03 12:16:43	792	76	79	790	154	160	788	400	0	1111	5577	2391	5577	0	1990	429	0	2373	0	0
2024-04-03 12:16:44	880	86	88	879	174	180	868	450	0	1307	6333	2715	6333	0	2275	456	0	2637	0	0
2024-04-03 12:16:45	881	93	89	885	173	180	868	450	0	1246	6328	2712	6328	0	2275	456	0	2640	0	0
2024-04-03 12:16:46	890	86	96	883	174	180	872	450	0	1246	6265	2685	6265	0	2230	456	0	2667	0	0
2024-04-03 12:16:50	888	88	89	886	175	182	878	455	0	1239	6293	2703	6293	0	2220	510	0	2661	0	0
2024-04-03 12:16:52	888	89	89	888	175	180	880	450	0	1309	6323	2709	6323	0	2270	510	0	2664	0	0

As mentioned in previous chapter, the unaddressed aspect of Ethernet dynamic link load estimation lies in the accurate estimation of the transmission frequency for each type of PDU. Therefore, after the data collection, various approaches are applied including probability distribution analysis and machine learning methodologies.

Probability Distribution Analysis

One approach to estimate dynamic link load involves the use of probability analysis. Several studies have already been conducted on the estimation of signal transmission frequency across various types of networks.

Haeri Kim, Wonsuk Yoo, and their colleagues [26] developed an average response time model based on mixed traffic consisting of periodic and aperiodic messages in CAN-FD and Automotive Ethernet networks. They verified their response time results through simulation. In their analytical model, they proposed that sporadic messages are sent according to a Poisson distribution. They then utilized OMNeT++ to simulate

message arrival behavior and concluded that the analytical model accurately represents the average response time in in-vehicle networks.

Tang C, Mou W, and their team [27] studied short message burst signals to evaluate receiving performance. In their probability model, they also employed a Poisson distribution to simulate sporadic signals and constructed an M/M/n/n queuing model to analyze system performance with a given call loss rate. They used a Monte-Carlo method to simulate signal arrival and processing times, confirming the theoretical analysis under a 0.5% call loss rate.

Hence, it seems that the Poisson process would serve as a suitable method for estimating the PDU frequency within each ECU gateway. The corresponding formula for the Poisson process is as follows:

$$P(n_{PDU} = k) = \frac{\lambda^k e^{-\lambda}}{k!} \tag{4.1}$$

where $P(n_{PDU} = k)$ is the probability of k PDUs sent in a fixed interval of time, λ is the average rate of occurrence.

Figure 4.7 illustrates the Poisson distribution corresponding to typical λ values, specifically 10, 200, and 3000, as estimated from the trace files.

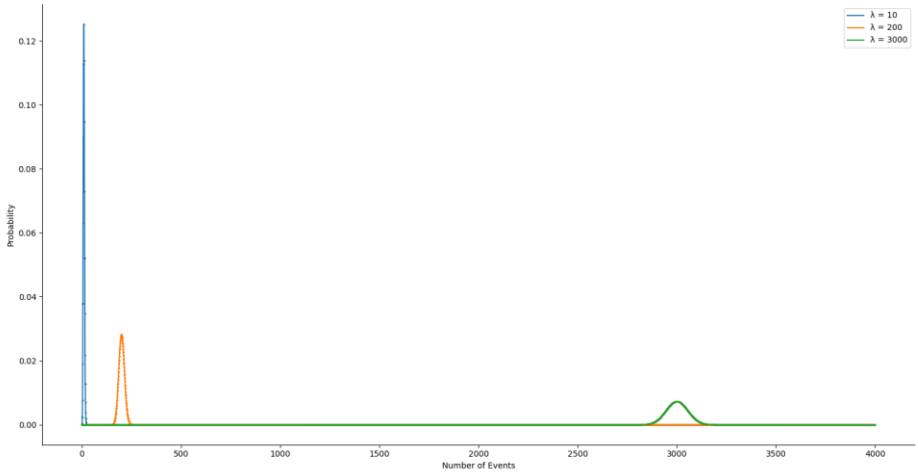


Figure Error! No text of specified style in document.**.20** Poisson distribution of common $\boldsymbol{\lambda}$ values

Before calculating the maximum possible value, it is essential to specify a confidence level. A confidence level is a range of values derived from sample data that is likely to contain the true population parameter, such as the mean, with a certain level of

confidence. This interval provides an estimate of the uncertainty associated with the sample statistic.

For instance, consider an automotive Ethernet link where the minimum transmission frequency is determined to be 2000 under a 95% confidence level. This implies that there is a 95% confidence that the true transmission frequency is less than 2000. According to calculations based on the Poisson distribution, this translates to an expectation of 0.12 cases per million transmission seconds where the true transmitted frequency exceeds the minimum threshold of 2000. The 95% confidence level indicates that the area under the distribution curve to the left of the critical value contributes 95% of the total area. The selection of a 95% confidence level is a common practice in statistical analysis due to its balance between precision and reliability [28].

By setting the confidence level to 0.95, it is possible to calculate the maximum number of PDUs transmitted over the link for which the confidence level is satisfied:

$$P(X \leq k) = \sum_{i=0}^{k} \frac{\lambda^i e^{-\lambda}}{i!} \geq 0.95 \tag{4.2}$$

For common λ values of 10, 200, 3000, the value of k such that $P(X \leq k) \geq 0.95$ is approximately 15, 224 and 3090.

The analysis reveals that the observed data closely aligns with the average rate. If the hypothesis that the transmission frequency adheres to a Poisson distribution is true, then utilizing a Poisson distribution could serve as an effective method for estimating the maximum link load at a given confidence level. However, a deeper examination of this method reveals a differing perspective.

In the first model developed by Haeri Kim et al., the authors employed OMNeT++ to simulate the delivery process within the network. They modeled sporadic message transmission by introducing random time differences using the parameter 'drift,' which was drawn from a uniform distribution, thus implying a constant transmission rate. Similarly, in the second model by Tang C et al., a Monte Carlo simulation based on the M/M/n/n queuing model was used, where arrivals were organized into a single queue governed by a Poisson process.

Given that both simulations rest on the assumption that packet arrivals follow a Poisson process, it is crucial to evaluate the validity of this assumption in automotive Ethernet context. The Poisson process presupposes that sporadic messages are sent at a

constant average rate. However, this assumption may be challenged by several factors in practical applications:

Burstiness of Traffic: In real-world scenarios, traffic often exhibits burstiness, with messages arriving in clusters rather than at a steady rate. This characteristic could result in significant deviations from the Poisson model, particularly if the deviations are larger than the average value.

Network Congestion: Fluctuations in message arrival rates due to varying levels of network congestion could undermine the assumption of a constant rate, potentially invalidating the Poisson model.

Adaptive Control Mechanisms: Modern in-vehicle networks often implement adaptive control mechanisms that dynamically adjust transmission rates based on network conditions. Such adjustments may not conform to the assumptions of the Poisson process.

Vern Paxson and Sally Floyd [29] discusses the limitations of using Poisson models for network traffic, particularly in wide area networks (WAN). In their study, they analyzed 24 traces of wide area TCP traffic, including TELNET and FTP traffic. The results indicate that TCP traffic exhibits much more burstiness than Poisson models predict. This burstiness has significant implications for congestion control and traffic performance.

Figure 4.5 and figure 4.6 show two sample segments of PDU sending frequency in one of the traces. It is obvious that the data arrival rate is varying significantly through time. Notice the straight lines indicate a vacant period. Therefore, the distribution is more of a varying rate Poisson distribution.

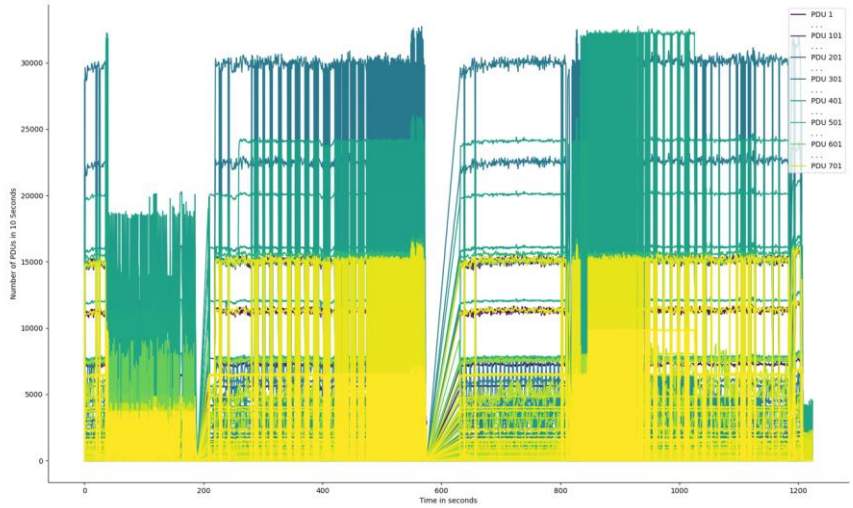


Figure Error! No text of specified style in document.**.21** PDU data segment on 2024.05.28

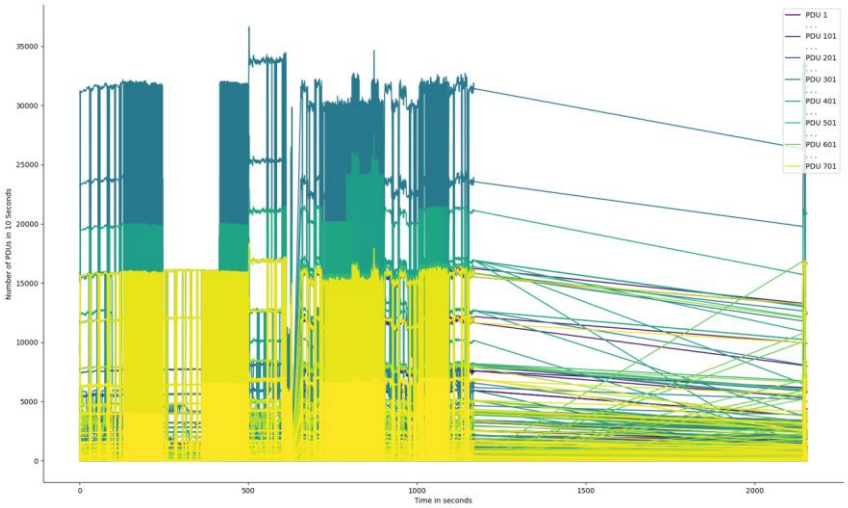


Figure Error! No text of specified style in document.**.22** PDU data segment on 2024.06.07

Hung, H. N., Lin, Y. B., and Luo, C. L. [30] derived a SMS arrival number of patterns for Chunghwa Telecom's SMSC. In this paper, the authors found that the volume of SMS arrivals to a mobile telecom network is typically very large, and these arrivals can be viewed as statistically independent. Additionally, SMS arrivals that exhibit significant variation over different times of the day. Therefore, the traffic can be modeled by a non-homogeneous Poisson process(NHPP) with different arrival rate function in different time period, which is estimated by a kernel-based fitting.

Thus, it can be suggested that the PDU frequency follows the following distribution of NHPP [30]:

$$P(n_{PDU}(t) = k) = \frac{\Lambda(t)^k e^{-\Lambda(t)}}{k!} \tag{4.3}$$

Where $\Lambda(t)$ is the integral of the intensity function $\lambda(t)$ over the interval $[0, t]$:

$$\Lambda(t) = \int_0^t \lambda(u)du \tag{4.4}$$

However, determining the appropriate intensity function poses a challenge. In the paper by Hung, H. N., Lin, Y. B., and Luo, C. L., the authors addressed this issue by employing a Nadaraya–Watson kernel weighted function to estimate the distribution of SMS arrivals.

The kernel density estimation is [30]:

$$K_\zeta(T', T) = \begin{cases} \frac{3}{4}\left[1 - \left(\frac{|T' - T|}{\zeta}\right)^2\right] & if \left|\frac{|T' - T|}{\zeta}\right| < 1 \\ 0 & else \end{cases} \tag{4.5}$$

Using this kernel function, the estimate of the arrival rate function $\widehat{r_x}(T)$ is obtained by [30]:

$$\lambda_x(T) = \frac{\sum K_\zeta(T', T)N(T')}{\sum K_\zeta(T', T)} \tag{4.6}$$

Which is essentially a convolution operation on the data. This indicates that the problem might be more effectively addressed using convolutional neural networks. Furthermore, this suggests that a more accurate estimation could be obtained by dividing the data into segments, each characterized by an approximately constant rate of occurrence..

Segmentation and Classification

Previous Works

Based on the previous analysis, accurately classifying different scenarios is of paramount importance. Individual use cases exhibit varying durations within the trace files, and even within the same scenario, the duration can differ significantly. For instance, when engaging cruise control on the highway, a vehicle might remain in this state for 10 or 20 minutes.

Given the absence of pre-segmented and pre-labeled data, and the fact that manual segmentation can be time-consuming, it is necessary for the model to perform this task using unlabeled data. This task fundamentally involves unsupervised time-series

segmentation and clustering. Each dataset will have a dimension of n obtained by downscaling the PDUs through Principal Component Analysis (PCA). Consequently, this problem should be approached as a multivariate, unsupervised time-series segmentation and clustering task.

There have already been some previous studies conducted in this area.

Gharghabi, S., Ding, Y., Yeh, C et al. [31] introduces FLOSS (Fast Low-cost Online Semantic Segmentation) for unsupervised semantic segmentation of time series data. In the paper, the authors used Matrix Profile (MP) representation and the STAMP (Scalable Time series Anytime Matrix Profile) algorithm to calculate the all-pairs similarity of subsequences in time series data. During the FLOSS algorithm, an Arc Curve is generated, which stores the value of how many similar pairs of subsequences are crossing that position. An Idealized Arc Curve is then removed from the Arc Curve to avoid edge effects through the equation [29].

$$CAC_i = min\left(\frac{IAC_i}{AC_i}, 1\right) \tag{4.7}$$

Lower CAC values indicate that the subsequence at that position has low similarity with subsequences at other positions because state change boundaries typically occur at points of low similarity.

FLOSS was tested on the PRCP dataset, the algorithm accurately identified regime changes in arterial blood pressure data during postural changes.

Yasuko Matsubara, Yasushi Sakurai, and Christos Faloutsos [32] used multi-level chain model (MLCM) and invented a coding cost function to find the optimal segmentation and clustering of time-series data. They used CutPointSearch to identify the optimal cut points to segment the data and RegimeSplit to estimate the parameters for the identified regimes as inner loop and AutoPlait to determine the best number of regimes and segments through a greedy approach. The algorithm continually splitting the data until the coding cost no longer decreases. The effectiveness of AutoPlait is validated through motion capture data, web click logs, and Google Trends data.

Nagano, M., Nakamura, T., Nagai, T et al. [33] proposes a method called HVGH for segmenting high-dimensional time series data. The proposed HVGH model integrates a variational autoencoder (VAE) with HDP-GP-HSMM. The VAE was used to extract features from high-dimension data and compress them into latent variables, while the HDP-GP-HSMM is used cluster the results.

Clustering Algorithm

For clustering data into distinct classes, several methodologies can be employed. The first is K-means clustering, a classical and widely recognized algorithm. This method partitions the data into K clusters by iteratively assigning each data point to the nearest cluster center and then updating the cluster centers to represent the mean of the assigned points. The algorithm converges by minimizing the within-cluster sum of squares, which is mathematically expressed as:

$$\mathcal{L} = \sum_{i=1}^{K} \sum_{x \in C_i} \|x - \mu_i\|^2 \tag{4.8}$$

Where C_i is the i th cluster and μ_i is the mean of the points in C_i.

The second approach is Spectral Clustering, which leverages the eigenvalues of a similarity matrix to reduce dimensionality before performing clustering in the lower-dimensional space. According to graph theory, the data points to be clustered can be represented as nodes in a graph, where clustering is analogous to partitioning the graph into subgraphs (clusters) such that each node has as many connections (edges) to its neighboring nodes within the same cluster as possible.

To achieve this, an adjacency matrix of the graph is first constructed using methods like K-nearest neighbors or a Gaussian similarity function, which captures the similarity between pairs of data points. Once the adjacency matrix is established a Laplacian matrix is computed:

$$\mathcal{L} = D^{-1/2} A D^{-1/2} \tag{4.9}$$

By selecting the eigenvectors with minimum eigenvalues and perform clustering algorithms like K-means, a clustered result can be derived.

The third method is the Dirichlet Process Gaussian Mixture Model (DPGMM), which is a non-parametric Bayesian approach that can automatically determine the number of clusters. The model combines Dirichlet Process and Gaussian mixture, where Dirichlet Process as a prior over the infinite mixture components.

$$e_i|\eta_i \sim \sum_{k=1}^{K} \pi_k \mathcal{N}(\mu_k, {\sigma_k}^2) \tag{4.10}$$

$$\eta_i \mid G \sim G \tag{4.11}$$

$$G \sim DP(\alpha, H) \tag{4.12}$$

Which means G is derived from Dirichlet Process, η_i is the parameters sampled from G and e_i is derived from the Gaussian Mixture Process through the given parameters.

Through estimating initial mean values through Dirichlet Process and fitting the values into different multivariant gaussian distribution, a clustering result can be derived.

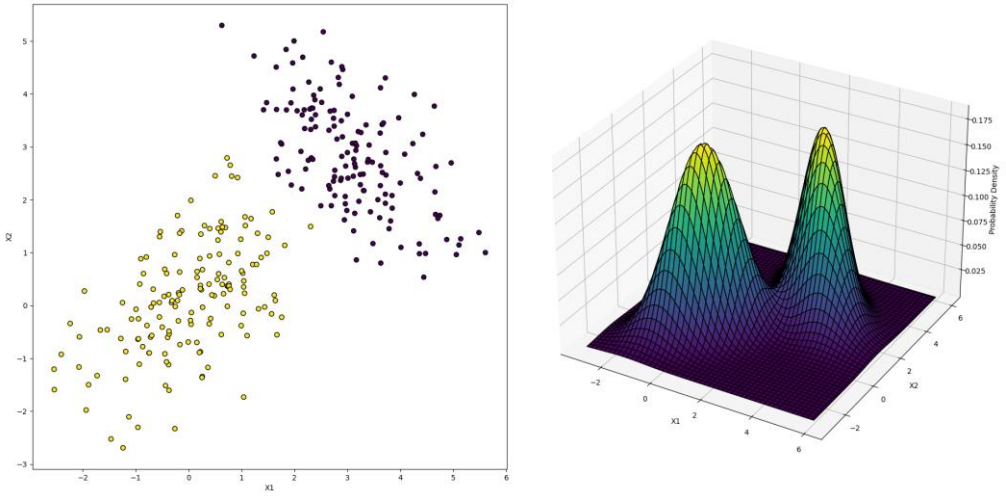


Figure Error! No text of specified style in document.**.23** Illustration of a gaussian mixture

A Gaussian mixture model (GMM) is well-suited for capturing the shape of complex distributions, making it an ideal choice for clustering multivariate samples without requiring a fixed number of clusters. Therefore, DPGMM is employed in this paper to cluster the data.

Procedure of a DPGMM is as follows:

Variable initialization:

r_{ik}: posterior probabilities for each data point i belonging to each component k with form:

$$r_{ik} = \frac{\pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_i \mid \mu_j, \Sigma_j)} \tag{4.13}$$

a) π_k: weight of component k, indicating the portion data is related to component k

b) $\mathcal{N}(x_i|\mu_k, \Sigma_k)$ Gaussian probability density function of component k for data point i

r_{ik} is initialized through a k-means clustering, where For each data point i:

$$r_{ik} = \begin{cases} 1 & , if\ point\ i\ \in\ C_k \\ 0 & else \end{cases} \tag{4.14}$$

W: a hyperparameter of the model and represents the initial guess of the covariance before any data is observed. It acts as a regularizer to ensure the updated covariances do not become degenerate.

Updating process:

r_{ik}: updated through taking the logarithm of the formular shown above. Taking logarithm helps to ensure that there are no underflow or explosion of the value:

$$\log r_{ik} = \log \pi_k + \log N\left(x_i \mid \mu_k, \Sigma_k\right) - \log \sum_{j=1}^{K} \pi_j N\left(x_i \mid \mu_j, \Sigma_j\right) \qquad (4.15)$$

π_k: updated through summing up the related to r_{ik} component k and normalizing to a total sum of 1 to estimate of how much data is attributed to each component:

$$\pi_k = \frac{N_k + \alpha - 1}{N + \sum_{j=1}^{K}(\alpha_j - 1)} \qquad (4.16)$$

a) N_k: sum of r_{ik} of component k

b) α: Dirichlet prior concentration parameter

μ_k: mean over weighted sum of data points:

$$\mu_k = \frac{\sum_{i=1}^{N} r_{ik}\, x_i + \lambda \mu_0}{N_k + \lambda} \qquad (4.17)$$

a) x_i: data points

b) μ_0: prior mean, prevents the means from being overly influenced by the data

c) λ: mean precision prior, adjust the precision

Σ_k: evaluate the regularized and Degree of Freedom normed spread of the data points assigned to component k

$$\Sigma_k = \frac{S_k + W^{-1}}{DoF + N_k + d + 1} \qquad (4.18)$$

a) S_k: scatter matrix for component k:

$$S_k = \sum_{i=1}^{N} r_{ik}(x_i - \overline{x_k})(x_i - \overline{x_k})^T \qquad (4.19)$$

where $\overline{x_k} = \frac{1}{N_k}\sum_{i=1}^{N} r_{ik} x_i$

b) W: prior covariance

c) d: number of features

Convergence:

The lower bound on the log-likelihood is computed to monitor convergence. The model iterates until the change in the lower bound is less than a specified threshold (`tol`):

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \log\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)\right) - \text{KL-divergence} \tag{4.20}$$

Representation Learning

Even with an appropriate classifier in place, directly segmenting and clustering time series data remains impractical. The length of time series data, which contains over 20000 time points, makes direct clustering overly complex and time-consuming. Furthermore, clustering without prior transformation is often ineffective because DPGMM clustering algorithm treat each time point as an individual object to be clustered although raw data typically does not adequately reveal the underlying patterns or state structures.

Therefore, it is crucial to first learn the representations of the data.

Principle Component Analysis

The first consideration is to reduce the second dimension of the data, namely the number of PDUs. The reason is that the number of PDUs is excessively large; when including both IPDUs and SOME-IP PDUs, the analysis involves over 800 possible PDU types. This results in prolonged computation times and excessive GPU memory consumption, as doubling the number of channels also doubles the number of convolution kernels, significantly increasing GPU memory usage. To address these issues, Principal Component Analysis (PCA) is employed to reduce the dimensionality of PDU types.

PCA is a widely used dimensionality reduction technique that simplifies data by transforming it into a new coordinate system, where the greatest variance in the data lies along the first coordinate (the first principal component), the second greatest variance along the second coordinate, and so on. The process of PCA involves the following steps:

Standardize the Data:

$$Z = \frac{X - \mu}{\sigma} \tag{4.21}$$

Compute the Covariance Matrix:

$$C = \frac{1}{n - 1} Z^T Z \tag{4.22}$$

Eigenvalue Decomposition:

$$C_{v_i} = \lambda_i v_i \tag{4.23}$$

Select the Top k Eigenvectors:

$$V_k = [v_1, v_2, \dots, v_k] \tag{4.24}$$

Transform the Data:

$$Y = ZV_k \tag{4.25}$$

Y is the transformed data in the new lower-dimensional space.

To determine the number of principal components to retain, the Cumulative Variance Explained is calculated. The explained variance ratio for each principal component measures the proportion of the dataset's total variance that is accounted for by that component. This ratio is calculated using the following formula:

$$Explained\ Variance\ Ratio = \frac{\lambda_i}{\sum_{k=1}^{p} \lambda_k} \tag{4.26}$$

Where λ_i stands for the eigenvalue of the i-th principal component and p is the total number of principle components. The cumulative variance explained up to the i-th principal component is the sum of the explained variance ratios of all principal components up to i.

$$Cumulative\ Variance\ Explained_i = \sum_{j=1}^{i} Explained\ Variance\ Ratio \tag{4.27}$$

It thus provides a measure of how much total variance is retained by the first i principal components.

Figure 4.8 presents a visualization of the most significant features extracted from the trace data. The x-axis represents time in seconds, while the y-axis displays the scaled values of the most significant features as identified through the analysis. The graph demonstrates the temporal evolution of these key features over the duration of the trace.

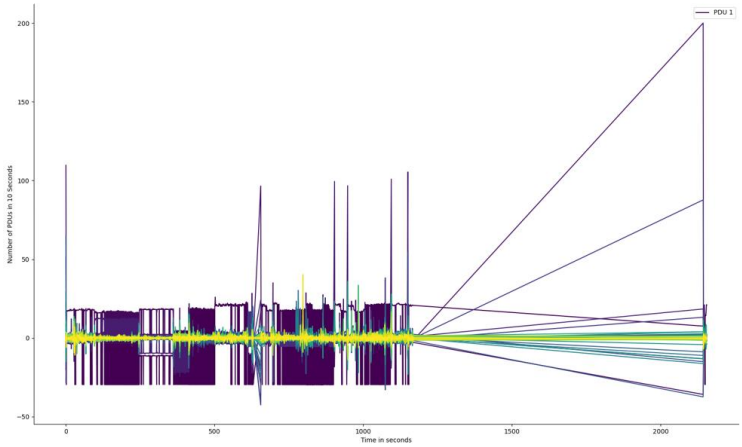


Figure Error! No text of specified style in document.**.24** Most significant features of trace

For the choice of threshold for cumulative variance explained, Alethea Rea and William Rea [34] explores the determination of the number of principal components to retain in multivariate time series PCA. They used heat maps and angle variation on financial data (FTSE 250 index). As a result, the authors suggest that cumulative which most significantly retain the components should be between 70% or 90%.

Based on the results of the PCA, the number of principal components was selected to be 49, ensuring that the cumulative variance explained is approximately 0.9. This dimensionality reduction through PCA will be employed in both the decompositional convolutional network and the Unsupervised Denoising Autoencoder (UDAE) to reduce the model size.

Figure 4.9 displays the Scree Plot of PCA. The x-axis represents the principal components, ordered from the first to the last, while the y-axis shows the explained variance ratio for each principal component.

The plot reveals that the first 49 principal components account for the significant portion of the total variance in the data. The steep drop in the explained variance ratio after the first few components indicates that these initial components capture most of the variance.

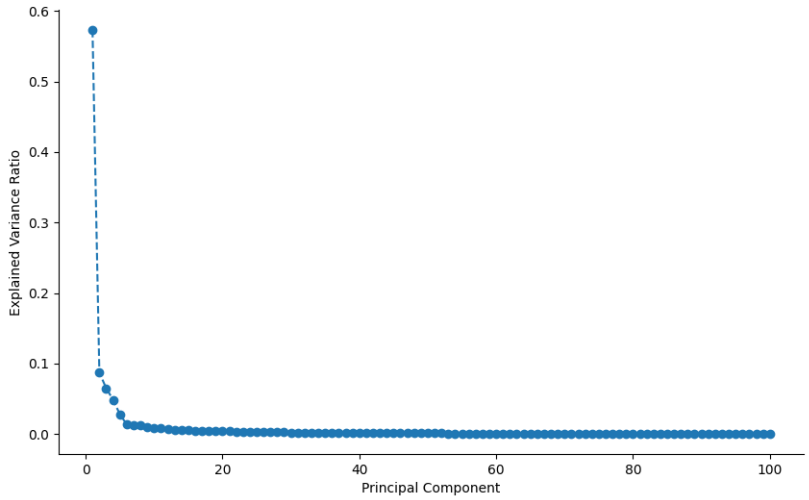


Figure Error! No text of specified style in document.**.25** Scree Plot of PCA

Causal Convolution Network

The initial approach for network selection in representation learning for time series data involves the use of a causal convolution network. As discussed in previous sections, convolutional networks have demonstrated exceptional performance in modeling the distribution of signal arrival time series. To align with the temporal characteristics of the data to be segmented, a causal convolution network is a logical and effective choice. This type of network utilizes convolutional filters that only take into account past data points, thereby preserving the causal structure inherent in time series data.

The procedure for implementing a causal convolution network is as follows:

For a given input sequence $\mathrm{X} = [\mathrm{x}_1, \mathrm{x}_2, \dots, \mathrm{x}_\mathrm{T}]$, a causal convolution operation with kernel W of length k and bias term b is defined as:

$$\mathrm{Y}_\mathrm{t} = \sum_{i=0}^{k-1} \mathrm{W}_i \cdot \mathrm{X}_{\mathrm{t}-\mathrm{i}} + \mathrm{b} \tag{4.28}$$

k: kernel size

W_i: i-th element of the kernel

$\mathrm{X}_{\mathrm{t}-\mathrm{i}}$: input at time step of $\mathrm{t} - \mathrm{i}$

The layers of the proposed Causal Convolution Network are:

Causal CNN: mentioned before. Notice that n convolution kernels are needed on n channels.

Adaptive Max Pooling: reduces the length of the time series to a fixed size. If the original length is L and the target size is P, the operation can be represented as:

$$Z_i = \max_{j \in window} (Y_{i+j}) \tag{4.29}$$

Linear Layer: mentioned before. Notice that n convolution kernels are needed on n channels.

$$O = ZW + b \tag{4.30}$$

All the layers combine to a single block, which repeats itself n times, which is defined in the parameter $depth$.

Figure 4.10 illustrates the latent variables resulting from the analysis, projected into the latent space. In this scatter plot, each point represents a data instance, with its position determined by the values of two latent variables. The different colors correspond to various combinations of latent variables, as indicated in the legend.

The figure visually captures the distribution and clustering of data points within the latent space, the clusters in the plot suggest that the latent variables effectively separate different underlying patterns or structures within the data.

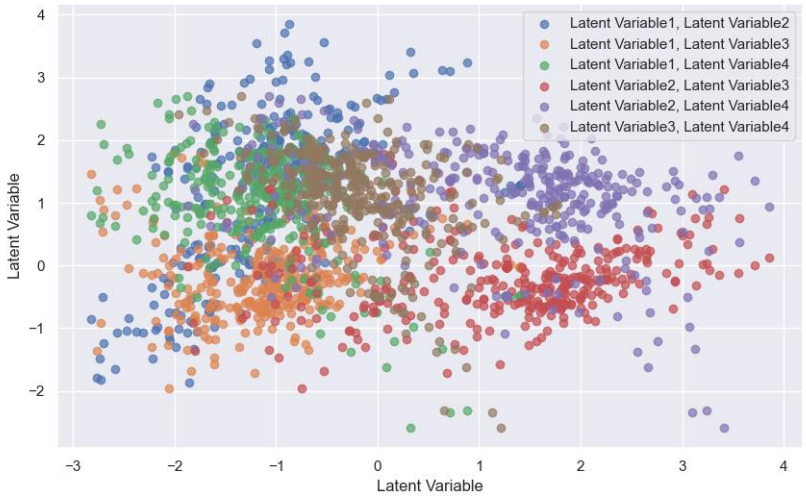


Figure Error! No text of specified style in document.**.26** Latent representations

1.1.11 Decomposition Deep Encoder Model

The second approach involves the use of a adapting a decomposition network. Zhichen Lai, Huan Li, Dalin Zhang, Yan Zhao, et al. [35] introduced decomposition network in E2Usd, which decomposes the input data into trend and seasonal components and then processing these components separately through convolutional

neural networks. To enhance the original architecture, an additional time embedding layer was incorporated to handle varying time stamps, enabling the model to better capture temporal dependencies and variations.

Layers:

Positional Embedding Layer

Decomposition Layer: Layer that splits the input time series into trend and seasonal components. This can be mathematically represented as:

a) $T(t)$: trend component.

b) $S(t)$: seasonal component.

Causal CNN: Notice that the trend component and seasonal component are separately convolved.

$$T'_t = \mathrm{CNN}_{\mathrm{trend}}\big(T(t)\big) \tag{4.31}$$

$$\mathrm{S}'_{\mathrm{t}} = \mathrm{CNN}_{\mathrm{seasonal}}\big(\mathrm{S(t)}\big) \tag{4.32}$$

Adaptive Max Pooling: The same.

Linear layer: Encoded values from the trend and seasonal components are concatenated and passed through another:

$$\mathrm{Latent} = \mathrm{Linear}\big(\big[\mathrm{T}_{\mathrm{embedding}}, \mathrm{S}_{\mathrm{embedding}}\big]\big) \tag{4.33}$$

The Trend Component is obtained using a moving average filter with a kernel size κ:

$$T_c[n,t] = \frac{1}{\kappa} \sum_{i=-\frac{\kappa-1}{2}}^{\frac{\kappa-1}{2}} \hat{x}[n, t+i] \tag{4.34}$$

The seasonal component is derived from the difference between the compressed time series and the trend component.

1.1.12 Unet Diffusion AutoEncoder

U-Net is a convolutional network architecture originally designed for biomedical image segmentation but is also effective for time series segmentation and representation learning. In 2022 K. Preechakul, N. Chatthee, S. Wizadwongsa and S. Suwajanakorn introduced [36] diffusion autoencoders to generate pictures. They used DAEs for representation learning by introducing a learnable encoder to discover high-level semantics and a DAE as the decoder for modeling stochastic variations. The approach

encodes images into a two-part latent code: one part capturing semantics and the other capturing stochastic details. Time series data can be viewed as a picture unfolded into segments, therefore making this powerful model applicable to learn high-level representations.

Like in the decomposition convolution network, UDAE learns a two-part latent code:

Semantic Subcode which Captures high-level, semantically meaningful information and Stochastic Subcode which encodes low-level stochastic variations.

Forward process:

Noise is added to the PDU trace data over time:

$$q(x_t|x_{t-1}) = N\left(\sqrt{1-\beta_t}x_{t-1}, \beta_t I\right) \tag{4.35}$$

Reverse Process:

Denoises the image:

$$p(x_{t-1}|x_t) = N(\mu_\theta(x_t, t), \sigma_t) \tag{4.36}$$

Semantic Encoder:

Encoder to summarize input PDU data into a descriptive vector:

$$z_{lat} = Enc(x_0) \tag{4.37}$$

Conditional Decoder:

With z_{lat} to be the latent subcode, the formula for the diffusion:

$$p_\theta(x_{0:T}|z_{lat}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t, z_{lat}) \tag{4.38}$$

Loss:

The deviation between reconstructed expectation and the original one:

$$\mathcal{L} = \sum_{t=1}^{T} E_{x_0,\epsilon_t}[|\epsilon_\theta(x_t, t, z_{sem}) - \epsilon_t|^2] \tag{4.39}$$

UNet architecture:

The UNet architecture is a type of autoencoder which typically consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. It employs a mechanism called lateral connection to directly connect the

input of one layer to the output of the same level in the expanding path. This architecture helps to retain important spatial information throughout the network, facilitating better reconstruction of the input data. Figure 4.11 illustrates the architecture of the proposed model, which integrates a Latent Encoder, Dirichlet DPGMM, and Conditional DDIM for processing and analyzing time series data. The process begins with time series data, which is fed into the model. The input data is first passed through a latent encoder, which compresses the input into a lower-dimensional latent space. The latent representation is then fed into a Conditional Denoising Diffusion Implicit Models(DDIM) module, which reconstructs the input data. The model calculates the reconstruction loss and improves the trainable parameters in each iteration.

After the model is trained, the latent representation is then processed by DPGMM to classify the data into different scenarios.

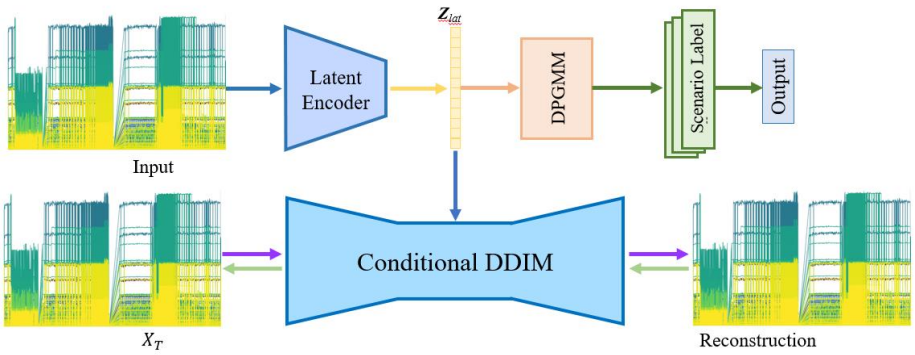


Figure Error! No text of specified style in document.**.27** Structure of proposed model: UDAE with DPGMM

To effectively interpret time series data, UDAE is modified by adding an additional attention layer is introduced before the first layer of the latent encoder. This attention layer enables the model to focus on relevant parts of the time series data.

Furthermore, time is converted into a relative time shift from the origin and is interpreted as a positional embedding, similar to the approach used in transformer models. This positional embedding is created by simply adding the normalized time difference to the input data. This modification allows the model to better capture and extract temporal dependencies and patterns within the time series data, improving its ability to interpret and process sequential information effectively.

The architecture depicted in Figure 4.12 represents a multi-layered network model. In the input stage, an attention mechanism is immediately applied following the data import, facilitated by an attention layer. This is followed by a sequence of residual blocks, each comprising a Group Normalization layer with SiLU activation function, a Convolutional Neural Network (CNN) layer and a Group Normalization layer, repeated N times.

In the intermediary section of the model, there is a singular attention block positioned between two residual blocks. The output stage mirrors the structure of the initial stage, consisting of N residual blocks, but concludes with a linear block. The latent representation within this architecture is generated by an encoder, which largely replicates the network's structure, differing only by the inclusion of a linear layer at its conclusion. During the diffusion process, the noise corresponding to each time step, along with the latent representation, is input into each residual block to produce the final output.

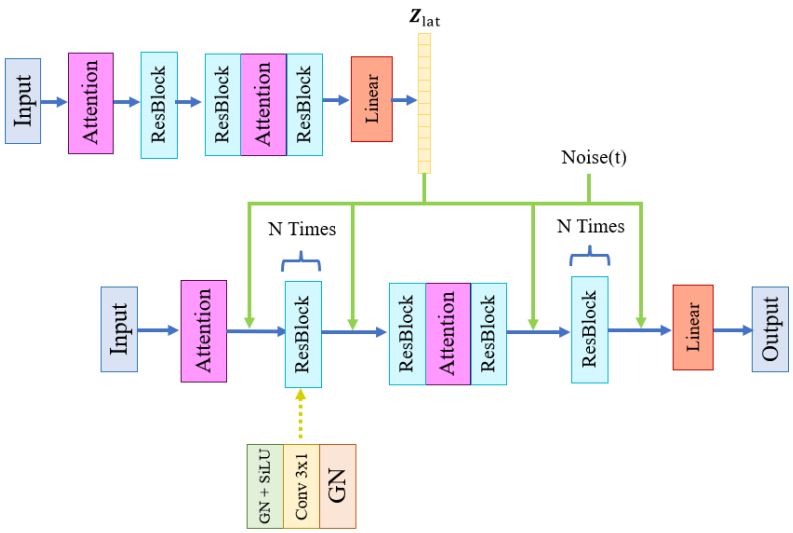


Figure Error! No text of specified style in document.**.28** Layered structure of the model

1.1.13 Loss Function

While the UDAE employs a reconstruction loss due to its inherent decoder structure, the other two encoders previously discussed do not incorporate a decoder component. As a result, the application of a reconstruction loss is not feasible for these models.

Cross Correlation loss function:

This is a classical loss functions, designed by measuring similarities between windows.

Given two time series segments X and Y, define their cross-correlation:

$$C_{XY}(l) = \sum_{t=1}^{T-l} X(t) \cdot Y(t+l) \tag{4.40}$$

Where T is the length of the time series segments.

The cross-correlation is then normalized to ensure that the result is scale-invariant. The normalized cross-correlation $NC_{XY}(l)$ is given by:

$$NCC_{XY}(l) = \frac{C_{XY}(l)}{\sqrt{\sum_{t=1}^{T-l} X(t)^2} \cdot \sqrt{\sum_{t=1}^{T-l} Y(t+l)^2}} \tag{4.41}$$

The Normalized Cross Correlation loss is computed as the negative of the maximum filtered normalized cross-correlation.

$$\mathcal{L}_{NCC}(X,Y) = -\max_{l}\left(NCC_{XY}(l)\right) \tag{4.42}$$

This loss function encourages to learn representations of the correct alignment and maximizes the representation similarity between windows.

The LSE-Loss

Chengyu Wang, Kui Wu, Tongqing Zhou, and Zhiping Cai. [37] proposed LSE-Loss to learn segmentation semantics.

$$\mathcal{L}_{LSE} = \mathcal{L}_{Intra} + \mathcal{L}_{Inter} \tag{4.43}$$

a) Intra-state loss:

The intra-state part $\mathcal{L}_{Intra}$ encourages the encoder to maximize the similarity between the representations of intra-state samples pushing them tight in the latent space. Therefore, the loss is calculated as the dot similarity between windows

$$\mathcal{L}_{Intra} = \alpha_1 \Sigma_{k=1}^{M} \Sigma_{i=1}^{N} \Sigma_{j=1,j<i}^{N} - log\left(\sigma\left(f(o_{ki})^{\intercal} f\left(o_{kj}\right)\right)\right) \tag{4.44}$$

Where $\alpha_1 = \frac{2}{(M*N*(N-1))}$ is a normalization factor.

σ is the sigmoid function.

f is the parameterized of encoder.

$o_{ki} = x_{t_k+i:t_k+i+w}$ denotes the i-th window in the k-th group.

b) Inter-state loss:

The inter-state loss aims to minimize the similarity between the representations of inter-state samples:

$$\mathcal{L}_{Inter} = \alpha_2 \Sigma_{k=1}^{M} \Sigma_{l=1,l<k}^{M} - log\left(\sigma(-c_k^{\intercal} c_l)\right) \quad (4.45)$$

where:

$\alpha 2 = \frac{2}{(M*(M-1))}$ is a normalization factor.

$c_k = \frac{1}{N} \Sigma_{i=1}^{N} f_\theta(o_{ki})$ is the embedding center for the k-th group.

For estimating the LSE Loss, a sampling procedure is unavoidable. To obtain inter-state samples, M non-consecutive windows are randomly placed on the given time series. Since they are non-consecutive, they are considered as inter-state samples. So M samples are obtained. The positions of these windows $\{t_k\}_{k=1}^{M}$ are drawn from a uniform distribution:

$$t_k \sim U(0, T - w - N) \quad (4.45)$$

where T is the length of the time series, w is the window size, and N is the number of consecutive windows. These windows are assigned a state, corresponding to all windows, the inter-state samples are collected: $\{x_{t_k:t_k+w}\}_{k=1}^{M}$

Intra-state Sampling is conducted afterwards. Each window selected before slides forward N times with a step size of 1. These sliding windows represent intra-state samples: $\{x_{t+i:t+w+i}\}_{i=1}^{N}$.

1.1.14 General Procedure

Figure 4.13 illustrates the structure framework for segmentation.

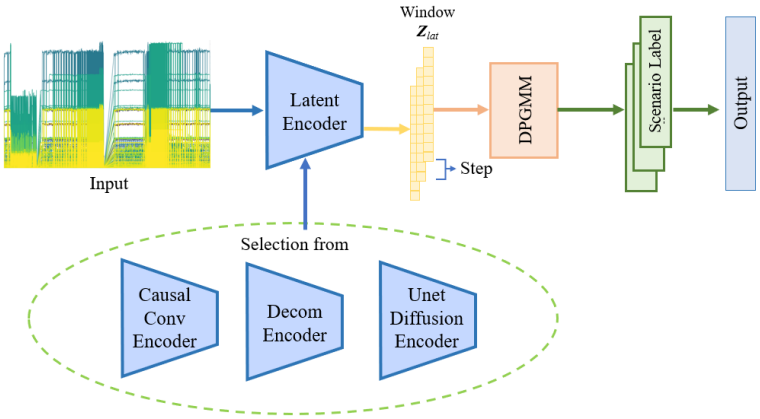


Figure Error! No text of specified style in document.**.29** General model structure

Through training process of the encoder, the latent representation of each window is learnt. After encoding through the trained encoder, the DPGMM will cluster the representations in each window into groups. By sliding the window at a certain step size, the classification value of at each point of time will be obtained. Sometimes when multiple values are received, a voting process will assign the most possible value to the time point.

Experiments

A series of detailed experiments were conducted to determine the optimal model for segmenting and clustering the trace files. The experimental procedure began with stacking all the segmented data files together, while inserting a zero sequence of size 300 between them. This insertion was critical in ensuring that time segments with no PDUs transmitted for more than 120 seconds were distinctly separated. By implementing this approach, the potential issue of the models mistakenly segmenting unrelated states into the same scenario was effectively mitigated.

Following the data preprocessing, the prepared datasets were input into various models to generate segmentation and classification outcomes. Each model was tested with a range of parameters to evaluate performance and identify the configuration that yielded the best results. This comprehensive approach allowed for a thorough comparison and facilitated the selection of the most effective model for accurately segmenting and clustering the trace files.

Causal Convolution Network

For the causal convolution network, various hyperparameters were systematically evaluated to optimize performance. These included testing different output channels,

corresponding to the dimensions of the latent variables, as well as varying window lengths and step sizes. The results, covering the entire time intervals of all conducted experiments, are presented below.

The results demonstrate that, for most of the given configurations, the causal convolution network effectively identified the zero patterns within the data. This indicates the network's capability to detect segments with no activity, thereby confirming its effectiveness in distinguishing between periods of transmission and inactivity. The successful identification of these zero patterns across different configurations highlights the robustness of the causal convolution network in handling temporal segmentation tasks under varying conditions.

The following figures illustrates the segmentation results, where different colors indicate different classes.

Experiment 1: Iteration number is 200, Sampled number of windows is 40, window size is 50, step is 5. The resulting Loss is 0.647

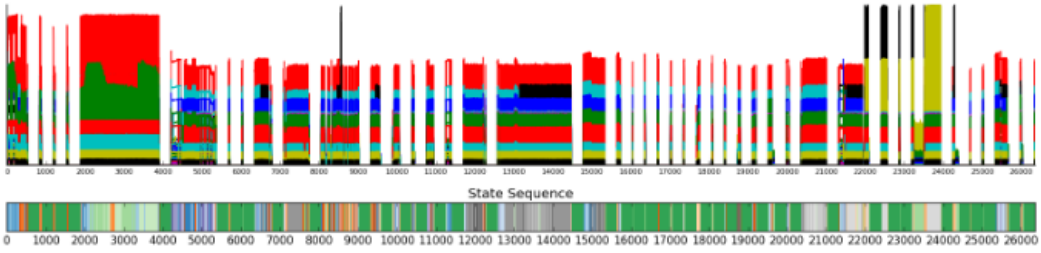


Figure Error! No text of specified style in document..**30** Total segmentation with window=50, step=5

Experiment 2: Iteration number is 200, Sampled number of windows is 40, window size is 100, step is 10. The resulting Loss is 0.683

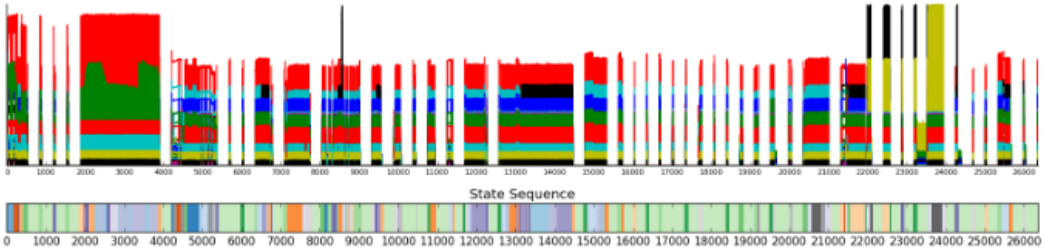


Figure Error! No text of specified style in document..**31** Total segmentation with window=100, step=10

Experiment 3: Iteration number is 200, Sampled number of windows is 40, window size is 200, step is 20. The resulting Loss is 0.689

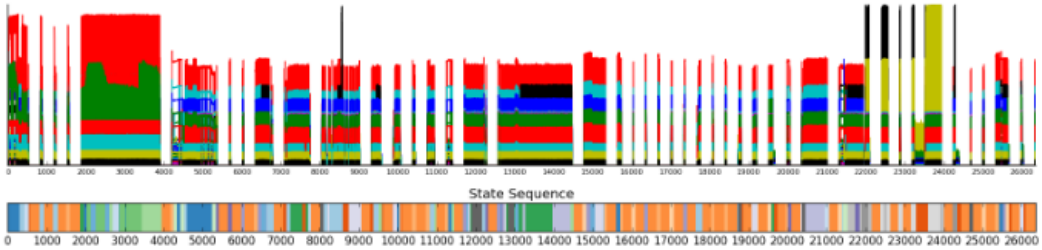


Figure Error! No text of specified style in document.**.32** Total segmentation with window=200, step=20

Experiment 4: Iteration number is 200, Sampled number of windows is 40, window size is 300, step is 30. The resulting Loss is 0.709

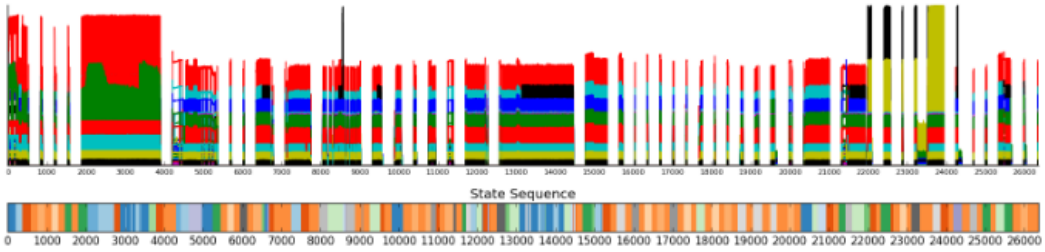


Figure Error! No text of specified style in document.**.33** Total segmentation with window=300, step=30

Experiment 4: Iteration number is 200, Sampled number of windows is 40, window size is 500, step is 50. The resulting Loss is 0.833

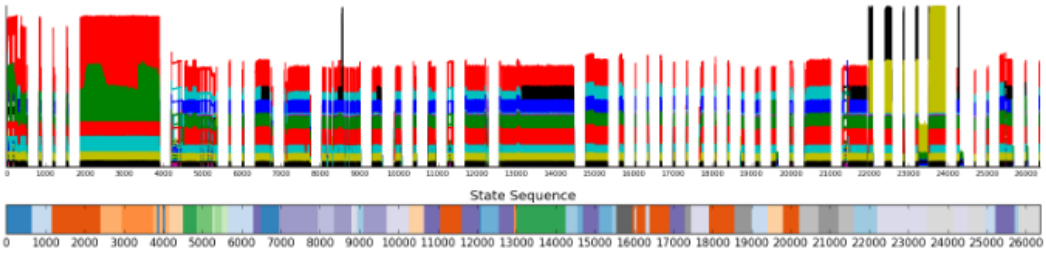


Figure Error! No text of specified style in document.**.34** Total segmentation with window=500, step=50

The results of previously separated individual time series according to the time information are presented in the following figures.

For segment 1:

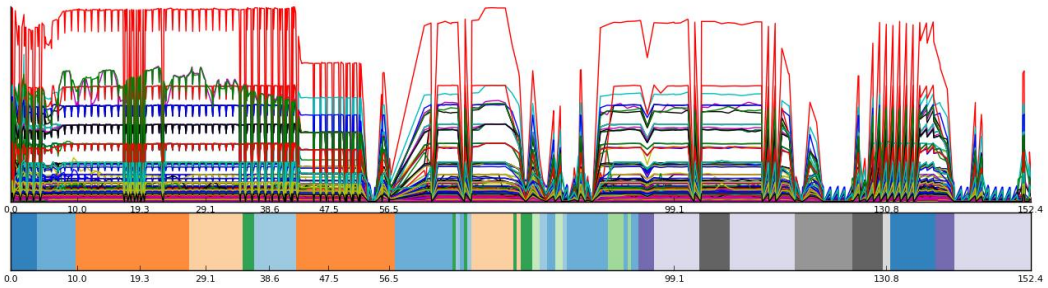


Figure Error! No text of specified style in document.**.35** Segmentation 1 with window=50, step=5

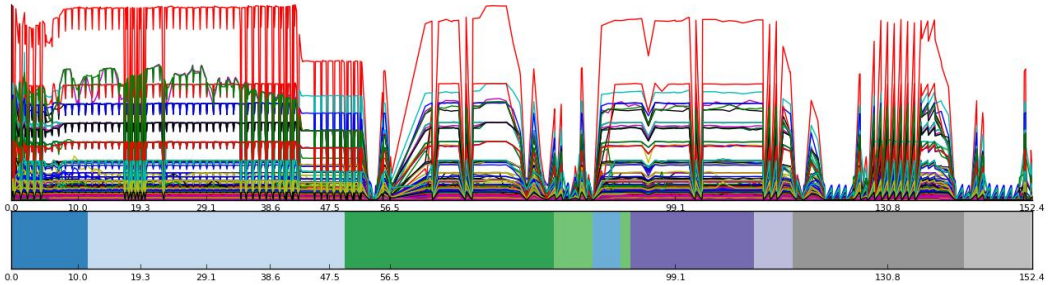


Figure Error! No text of specified style in document.**.36** Segmentation 1 with window=100, step=10

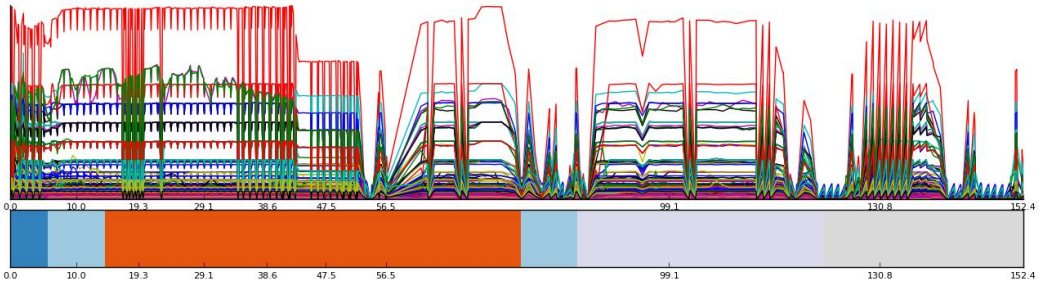


Figure Error! No text of specified style in document.**.37** Segmentation 1 with window=200, step=20

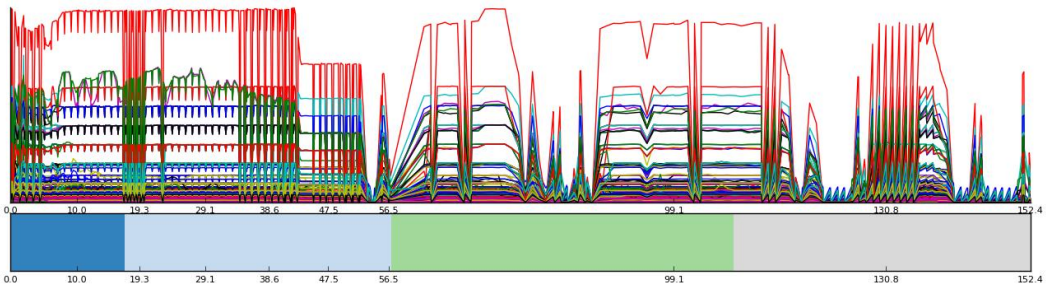


Figure Error! No text of specified style in document.**.38** Segmentation 1 with window=300, step=30

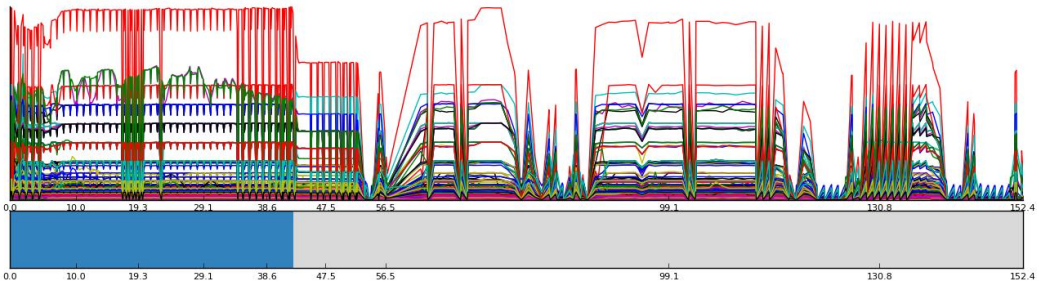


Figure Error! No text of specified style in document.**.39** Segmentation 1 with window=500, step=50

For segment 10:

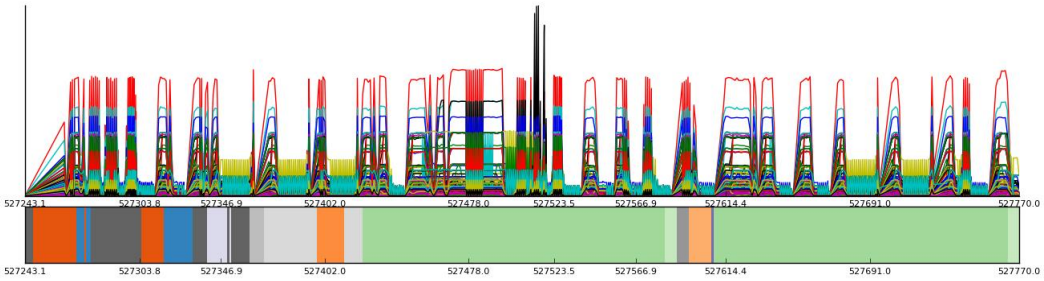


Figure Error! No text of specified style in document.**.40** Segmentation 10 with window=50, step=5

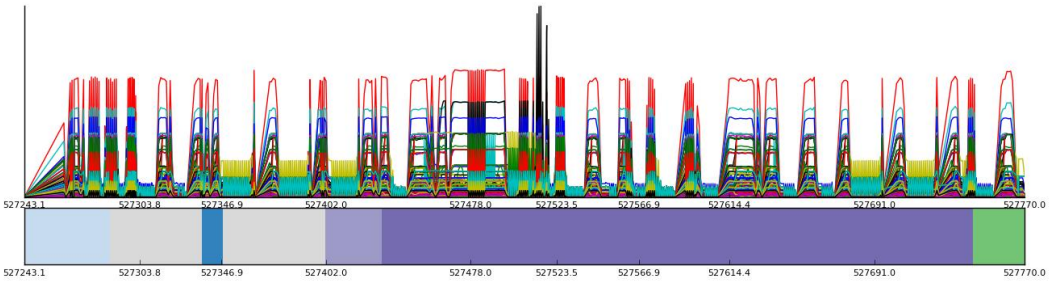


Figure Error! No text of specified style in document.**.41** Segmentation 10 with window=100, step=10

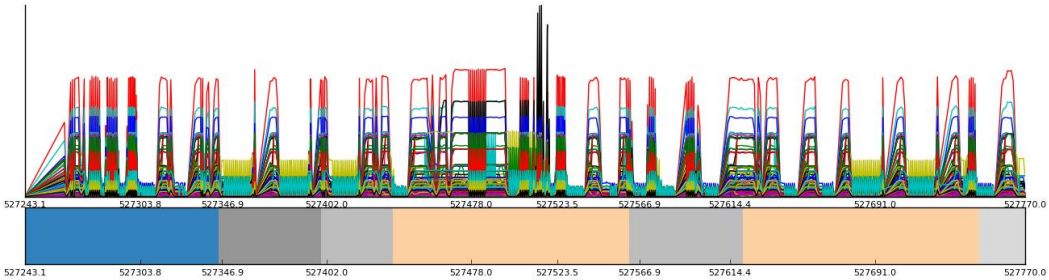


Figure Error! No text of specified style in document.**.42** Segmentation 10 with window=200, step=20

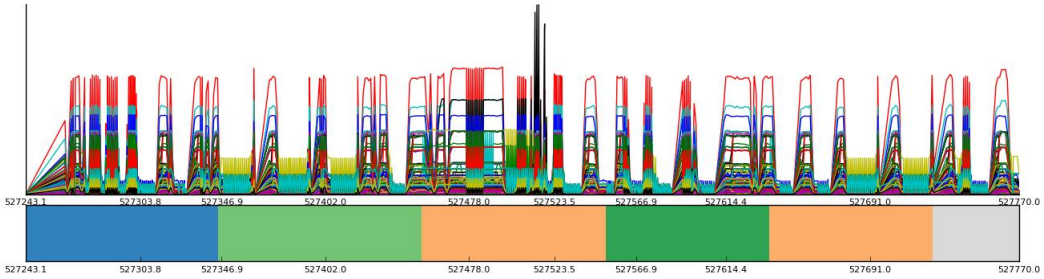


Figure Error! No text of specified style in document.**.43** Segmentation 10 with window=300, step=30

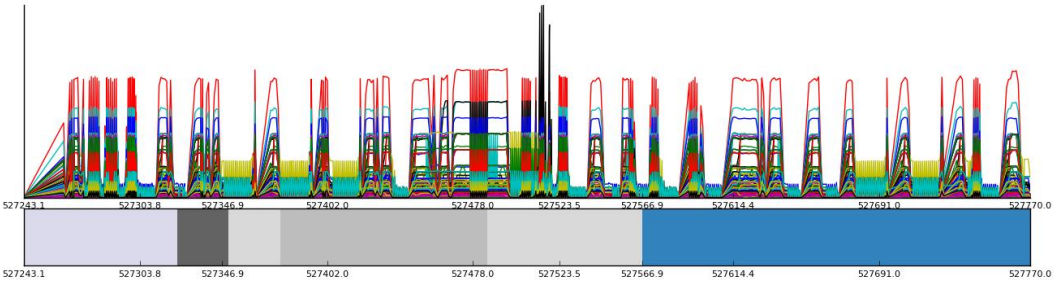


Figure Error! No text of specified style in document.**.44** Segmentation 10 with window=500, step=50

By analyzing the graphs and observing the segmentation results with the original trace data, it is evident that there is no single segmentation approach that achieves ideal results across all segments within a trace. In other words, no configuration is consistently accurate in segmenting every part of the trace. Some hyperparameters yield strong performance in the first segment but are less effective in subsequent segments, while others perform better in different parts of the trace. Despite this variability, it is observed that window sizes between 200 and 500 consistently produce more favorable segmentation outcomes, suggesting that this range of window lengths offers a generally better balance across different segments.

Decomposed Convolution Network

For the decomposed convolution network, the same experiments are conducted.

The following figures illustrates results from decomposed convolution network with different hyperparameters, different colours indicate different classes.

Experiment 1: Iteration number is 200, Sampled number of windows is 40, window size is 50, step is 5. The resulting Loss is 0.756

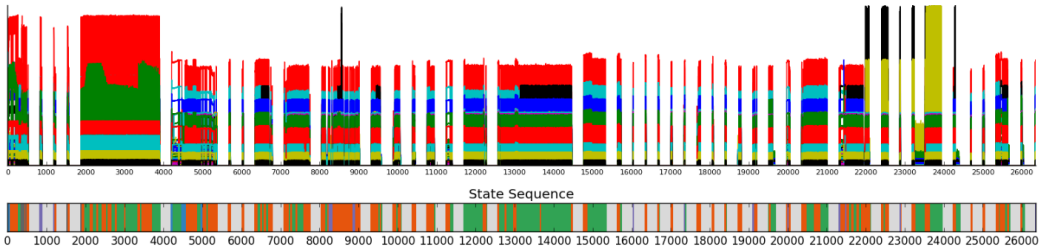


Figure Error! No text of specified style in document.**.45** Total segmentation with window=50, step=5

Experiment 1: Iteration number is 200, Sampled number of windows is 40, window size is 50, step is 5. The resulting Loss is 0.821

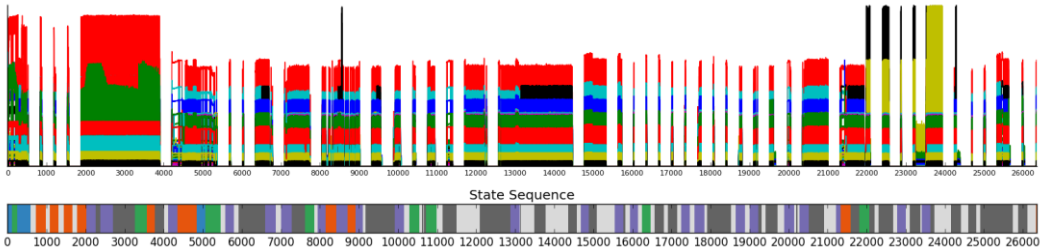


Figure Error! No text of specified style in document.**.46** Total segmentation with window=200, step=20

For segment 1:

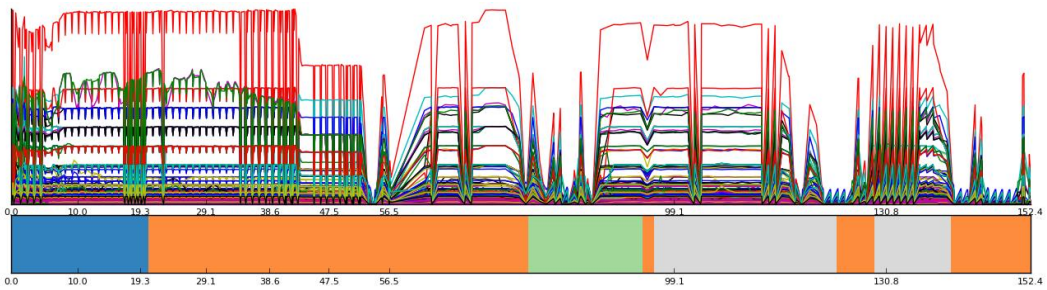


Figure Error! No text of specified style in document.**.47** Segmentation 1 with window=50, step=5

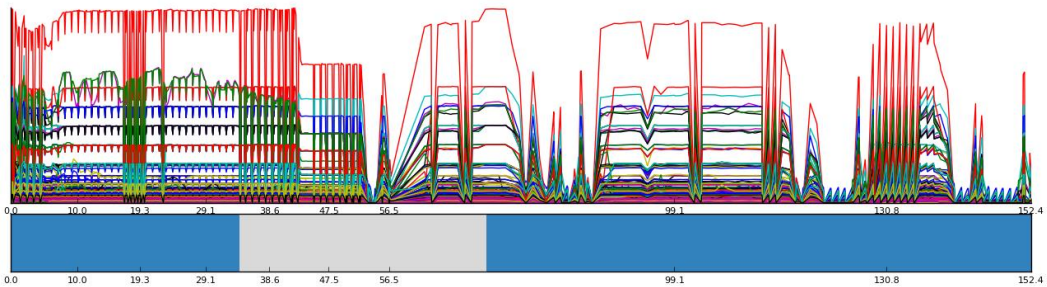


Figure Error! No text of specified style in document.**.48** Segmentation 1 with window=200, step=20

For segment 10:

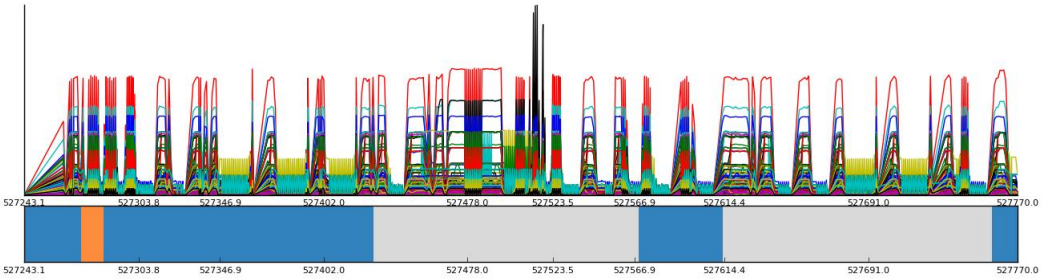


Figure Error! No text of specified style in document.**.49** Segmentation 10 with window=100, step=10

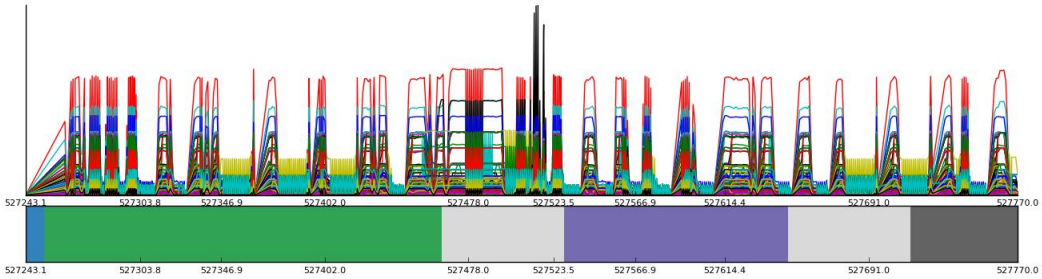


Figure Error! No text of specified style in document.**.50** Segmentation 10 with window=200, step=20

The results demonstrate that, for the first configuration, the decomposed convolution network effectively identifies the zero patterns within the data. Comparing the two results, the first hyperparameter configuration demonstrates superior performance relative to the other. Additionally, the decomposed convolutional network exhibits slightly better performance comparing to most results from the causal convolution network.

Unet Diffusion AutoEncoder

A single model was trained using a hyperparameter configuration of a window size of 128 and a step size of 10. In the downstream clustering task, implemented using DPGMM, two step sizes, 5 and 10, were evaluated. The step size of 5 was specifically assessed to obtain more precise information, potentially offering finer granularity in the clustering process. The following figures illustrate the results from the UDAE using different hyperparameters, with different colors indicating different classes.

Experiment 1: Iteration number is 284 epochs, window size is 128, step is 10. The resulting Loss is 0.202

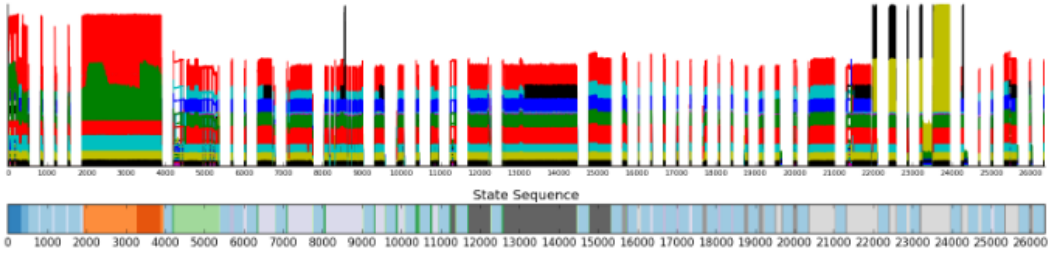


Figure Error! No text of specified style in document.**.51** Total segmentation with step=5

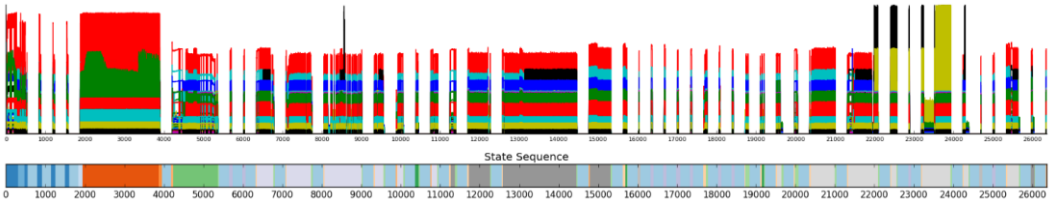


Figure Error! No text of specified style in document.**.52** Total segmentation with step=10

For segment 1:

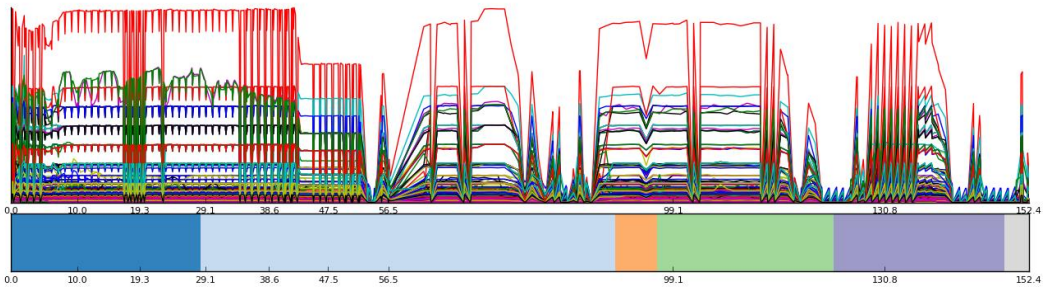


Figure Error! No text of specified style in document.**.53** Segmentation 1 with step=5

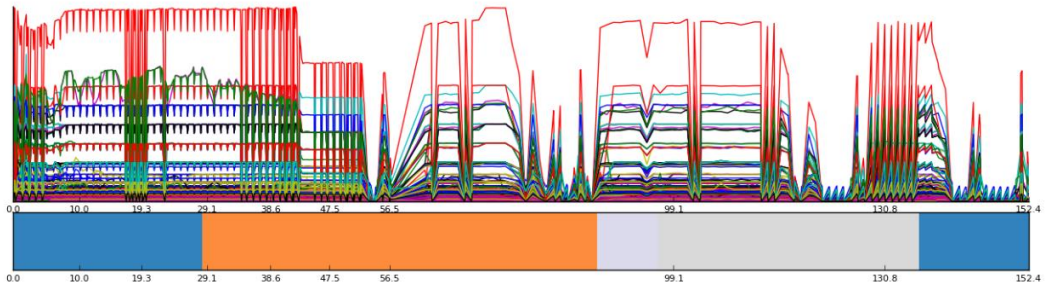


Figure Error! No text of specified style in document.**.54** Segmentation 1 with step=10

For segment 10:

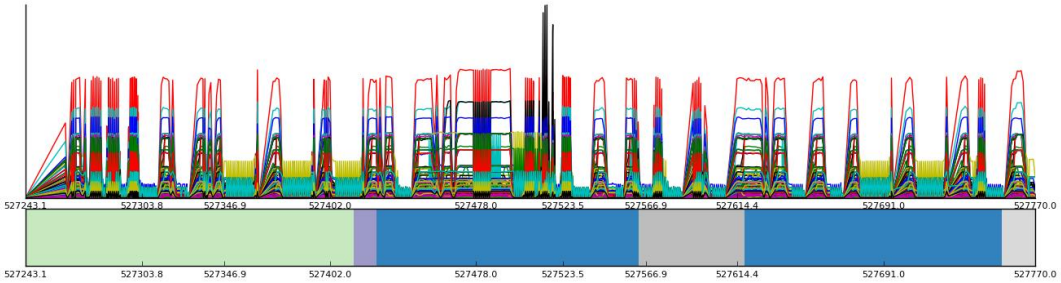


Figure Error! No text of specified style in document.**.55** Segmentation 10 with step=5

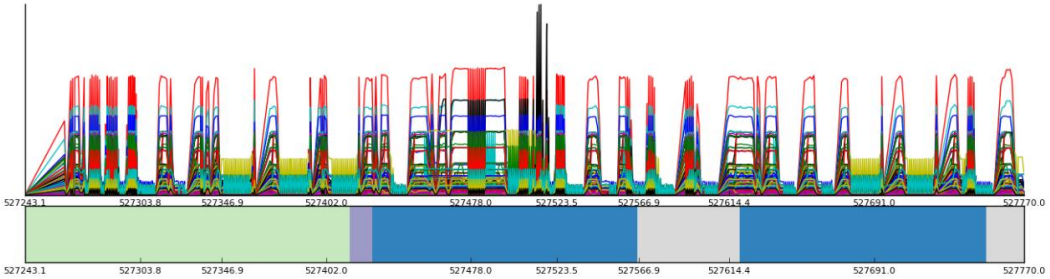


Figure Error! No text of specified style in document.**.56** Segmentation 10 with step=5

The results also demonstrate that the UDAE is effective in identifying zero patterns within the data. When observing the segmentation results with the original trace data, it is evident that UDAE performs well in capturing the underlying structure of the segments. In the downstream tasks, comparing the two configurations reveals that, for some segments, the results are nearly identical, indicating consistent performance. However, for other segments, there are noticeable differences, suggesting that the choice of step size and other hyperparameters can impact the segmentation and clustering outcomes in certain cases.

Voting and Merging Process

To combine the results previously obtained from different models, a voting and merging process is used.

First, classification result from causal convolution model with window size 200, 300, 500 together with the result from decompositional convolution model of window size 200 and UDAE of step 5 are selected.

Second, a weighted voting process will decide the most possible class for a certain time point among all the classification C suggestions:

$$WeightedVotes(v) = \sum_{j=0}^{n-1} \delta\left(v, C_j\right) \cdot w_j \tag{4.47}$$

$$C = \arg\max_{v} \text{WeightedVotes}\,(v) \tag{4.48}$$

The weights are chosen to be 2, 3, 2, 4, 4 following the order of models mentioned before.

Finally, a merging process will eliminate the too small segments that may occur.

For a certain segment $S(t_i, t_{i+1})$ and a certain threshold:

$$t_{split} = floor\big((t_{i+1} - t_i)/2\big) \qquad if\ T_s < Threshold \tag{4.49}$$

$$S_1' = S(t_{i-1}, t_i) \cup S(t_i, t_{split}) \tag{4.50}$$

$$S_2' = Sn(t_{i+1}, t_{i+2}) \cup S(t_{split}, t_{i+1}) \tag{4.51}$$

Where t_i denote the start index of the i -th segment.

The following figures illustrate the results after voting and merging of two segments, segment 0 and segment 10, with different colors indicating different classes. By observing the segmentation results with the original trace data, it is evident that the voted and merged result is having a better performance than any single result from a certain encoder.

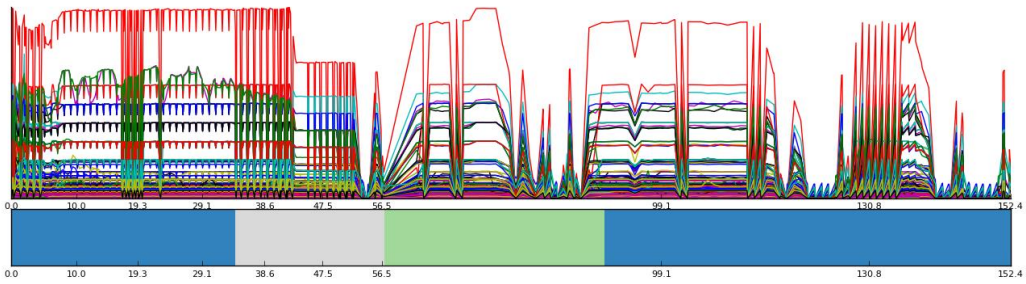


Figure Error! No text of specified style in document..**57** Voted and merged result of segment 1

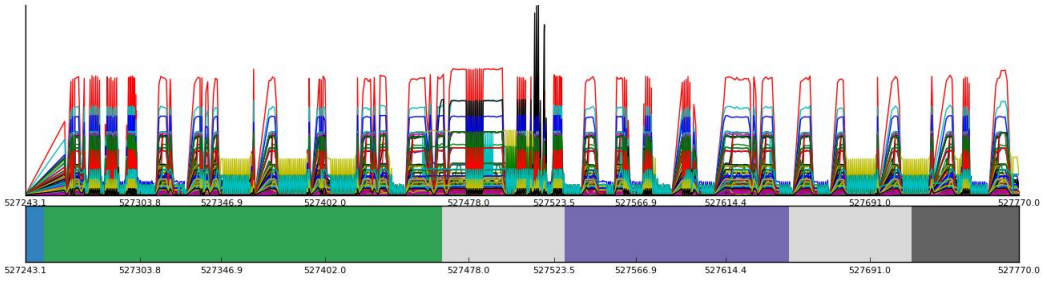


Figure Error! No text of specified style in document..**58** Voted and merged result of segment 10

Classification Results and Inspections

After conducting the experiments, the classification results were obtained from the trace files. Figure 4.43 provides insights into the performance of the models in categorizing the different segments within the data. 10 classes identified by the models are indicated in the legend located in the top right corner of the figure. The number of classes is determined by the self-clustering results of the DPGMM and is not predefined. The result of 10 classes also corresponds to the length of the trace file, making it a reasonable result.

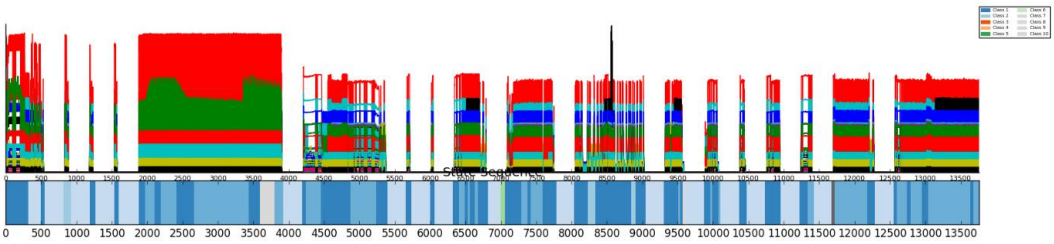


Figure Error! No text of specified style in document..**59** Overall classification result

The 10 classes detected distribute throughout the entire trace. To calculate the dynamic link load, the mean value of all the segments within each class was computed, with the first and last two time points excluded to eliminate edge effects. This approach ensures a more accurate representation of the link load by avoiding potential distortions caused by boundary conditions in the segmentation process. The procedure is shown in the formula below.

For segments $S(t_{i-1}, t_i)_{C,i}$:

$$\mu_{C,PDU} = \frac{\frac{1}{|S_C|}\sum_{i \in S_C} S(t_{i-1}, t_i)_{PDU}}{10} \tag{4.52}$$

Table 4.3 presents the mean transmission frequency per second of each PDU calculated in each class.

Table Error! No text of specified style in document..**5:** Mean value of PDU numbers sent per Class per Second

Average: (Class, PDU)	A	B	C	D	E	F	G	H	I	J	K	L	M
Class 1	46.12267	5.312551	5.339271	51.14575	4.872874	10.7045	52.7603	26.7259	0.03562	74.91296	375.9818	163.6296	375.9818
Class 2	40.06023	3.922028	4.592152	44.68194	1.99167	9.20158	45.829	23.9961	0.00027	67.02345	335.3556	142.6942	335.3556
Class 3	37.23076	3.506983	4.347521	40.32814	1.010421	8.80455	43.8177	23.9534	0	66.56578	333.2947	140.5071	333.2947

Class 4	44.95483	5.004247	5.049807	50.42239	4.984556	10.0918	50.4131	25.2297	0	70.61969	352.5089	151.0571	352.5089
Class 5	43.39006	4.70117	4.91462	48.20234	5.745029	9.69941	48.5309	27.6403	0.02105	76.59064	382.9322	163.2585	382.9322
Class 6	40.69005	3.955355	4.579968	43.90666	2.383484	9.19164	45.6932	24.1874	0.00083	67.70553	338.3767	144.816	338.3767
Class 7	36.5375	4.626786	4.623214	46.01429	0	9.2	46.0214	23	0	64.35893	322.2607	138.1054	322.2607
Class 8	31.07368	3.828947	3.821053	30.96842	0	7.62105	38.1894	19.0526	0	53.49474	267.1711	114.4974	267.1711
Class 9	58.8449	7.443878	6.785204	68.79388	0	13.9755	69.2469	37.5882	0	102.6071	516.2531	221.2342	516.2531
Class 10	49.0303	5.115152	5.109091	51.22424	0	10.2181	51.2363	25.5454	0	71.78182	358.5848	153.6758	358.584

As discussed in the previous section, all classes could be modeled as Poisson distributions. By specifying a 95% confidence level according to the previous section, an estimation of the number of PDUs sent per second in this scenario can be provided.

$$P\left(X_{C,PDU} \leq k\right) = \sum_{i=0}^{k} \frac{{\lambda_{C,PDU}}^{i} e^{-\lambda_{C,PDU}}}{i!} \geq 0.95 \tag{4.53}$$

For our case:

$$\lambda_{C,PDU} = \mu_{C,PDU} \tag{4.54}$$

Notice that some PDUs which sends at a low frequency will be estimated to 0 with Poisson process. In order to account for the worst case, these numbers will be adjusted to 1.

Table Error! No text of specified style in document..**6:** Threshold value of PDU numbers sent per Class per Second

Poisson: (Class, PDU)	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**
Class 1	58	9	9	63	9	16	65	35	1	89	408	185	408	1	141	27	1	173	1	1
Class 2	51	7	8	56	5	14	57	32	1	81	366	163	366	1	117	23	1	156	1	1
Class 3	48	7	8	51	3	14	55	32	1	80	364	160	364	1	116	20	1	148	1	1
Class 4	56	9	9	62	9	16	62	34	1	85	384	172	384	1	126	19	1	172	1	1
Class 5	54	9	9	60	10	15	60	37	1	91	415	185	415	1	114	26	1	156	1	1
Class 6	51	7	8	55	5	14	57	33	1	82	369	165	369	1	110	21	1	152	1	1
Class 7	47	8	8	57	1	14	57	31	1	78	352	158	352	1	119	23	1	158	1	1
Class 8	41	7	7	40	1	12	49	26	1	66	294	132	294	1	92	17	1	132	1	1
Class 9	72	12	11	83	1	20	83	48	1	120	554	246	554	1	207	28	1	230	1	1
Class 10	61	9	9	63	1	16	63	34	1	86	390	174	390	1	143	26	1	174	1	1

Thus, the dynamic loads are calculated to be:

$$Load_{FrameTrans} = \frac{N_{FrameWithGapPerBit} \cdot (1 + sFE)}{Baudrate} \cdot N_{PDUsSent} \tag{5.55}$$

Results and Discussions

Results

The transmission frequencies of dynamic PDUs are integrated into the transmission properties of each connection mentioned in Chapter Three. By combining both static and dynamic transmission frequencies, a model capable of estimating the total Ethernet link load is established.

The calculation results for specific tables are presented within the ARXML Visualizer. By selecting various scenarios, the ARXML Visualizer program computes the link load for each connection under the selected scenario.

Figure 5.1 illustrates the process of selecting different classes within the ARXML Visualizer.

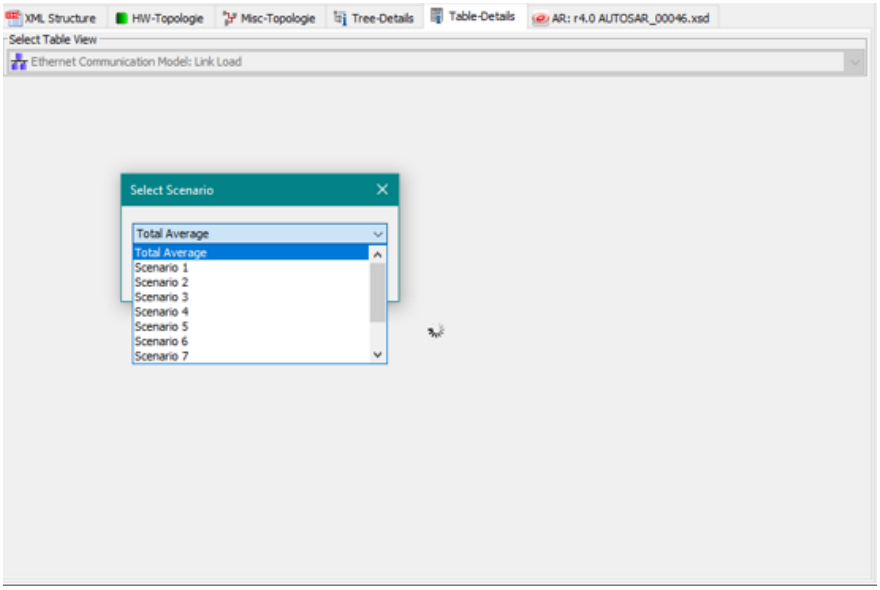


Figure Error! No text of specified style in document..**60** Selection of scenarios

The following tables show the link loads under the first three scenarios, including a table showing the link load considering over the entire trace. Other tables are listed in the Appendix.

Table Error! No text of specified style in document..**7:** Link load average over the entire trace

Start of link	End of Link	Baudrate	Load in Percent (%)	MAIN_1 in Percent (%)	MAIN_10 in Percent (%)	MAIN_100 in Percent (%)	MAIN_101 in Percent (%)	MAIN_110 in Percent (%)	MAIN_111 in Percent (%)	MAIN_120 in Percent (%)	MAIN_127 in Percent (%)
A	B	100000000	0.36472	0.00000	0.10038	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C	D	100000000	13.90278	0.00000	13.90206	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

E	B	100000000	1.35301	0.00000	0.02517	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
F	B	100000000	4.63426	0.00000	1.59314	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
G	B	100000000	4.17119	0.00000	0.56109	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
H	B	1000000000	0.47917	0.00000	0.16763	0.00000	0.00000	0.00000	0.00000	0.00000	0.00496
B	M	100000000	1.21202	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	E	100000000	0.01128	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
B	N	100000000	11.78054	0.00000	1.72147	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	I	100000000	4.52986	0.00000	0.07462	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	L	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	D	100000000	45.80091	0.04738	38.07726	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	A	100000000	0.54301	0.00000	0.22669	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
B	H	1000000000	4.79341	0.23920	1.83477	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	G	100000000	1.28382	0.00000	0.00288	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
B	J	100000000	24.60820	0.00000	24.60676	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	K	100000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	F	100000000	45.24258	0.01053	42.13731	0.00000	0.00000	0.00000	0.00000	0.00000	0.02557
B	O	100000000	0.02824	0.00000	0.00627	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
L	B	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
P	M	1000000000	0.12251	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
M	B	100000000	0.11538	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
M	R	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
M	P	1000000000	0.01745	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
M	Q	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
S	D	1000000000	1.65090	0.00489	0.87853	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
T	D	10000000000	0.15413	0.00000	0.15412	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	T	10000000000	0.00002	0.00002	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	B	100000000	8.08141	0.00226	7.15839	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
D	S	1000000000	0.80776	0.00008	0.71568	0.00000	0.00000	0.00000	0.00000	0.00000	0.00015
D	C	100000000	0.02624	0.00150	0.02398	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
U	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
V	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
W	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
X	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Y	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Z	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AB	AC	100000000	16.47276	0.00000	16.47204	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AC	J	100000000	0.13320	0.00150	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AC	AB	100000000	0.13320	0.00150	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AD	J	100000000	8.13864	0.00000	8.13792	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	AC	100000000	16.47276	0.00000	16.47204	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	B	100000000	0.13754	0.00150	0.13453	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
J	AD	100000000	0.00754	0.00000	0.00678	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	B	100000000	4.09728	0.00000	0.42685	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
N	U	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AP	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AG	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AF	1000000000	0.39852	0.00000	0.03942	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	AE	1000000000	0.29762	0.00000	0.00326	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
N	V	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AH	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	X	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	W	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AE	N	1000000000	0.44045	0.00000	0.14168	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AH	N	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AF	N	1000000000	0.75192	0.00000	0.03047	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	Y	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	B	100000000	0.02115	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00451
I	AM	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AI	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	AJ	1000000000	0.19584	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
I	AK	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	Z	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AL	1000000000	0.27262	0.00000	0.00746	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AJ	I	1000000000	0.19784	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AI	I	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AN	AL	1000000000	0.25818	0.00000	0.00746	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AL	AN	1000000000	0.00336	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AL	AO	1000000000	0.00177	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AL	I	1000000000	0.00513	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AO	AL	1000000000	0.01444	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AP	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

O	B	100000000	1.89054	0.00000	0.00442	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AG	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Q	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
R	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AM	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AK	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
K	B	100000000	2.35890	2.35890	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Figure 5.1 illustrates the table view of the average link load over the entire trace within the ARXML Visualizer interface.

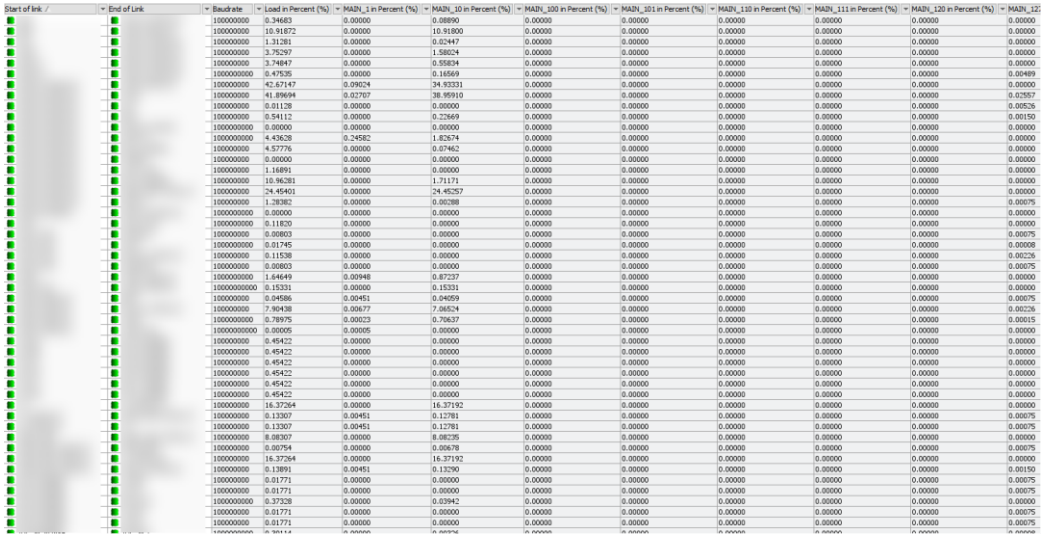


Figure Error! No text of specified style in document..**61** Link load average over the entire trace in ARXML Visualizer

Table Error! No text of specified style in document..**8:** Link load under scenario 1

Start of link	End of Link	Baudrate	Load in Percent (%)	MAIN_1 in Percent (%)	MAIN_10 in Percent (%)	MAIN_100 in Percent (%)	MAIN_101 in Percent (%)	MAIN_110 in Percent (%)	MAIN_111 in Percent (%)	MAIN_120 in Percent (%)	MAIN_127 in Percent (%)
A	B	100000000	0.36472	0.00000	0.10038	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C	D	100000000	12.95374	0.00000	12.95302	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
E	B	100000000	1.31240	0.00000	0.02517	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
F	B	100000000	4.43756	0.00000	1.48655	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
G	B	100000000	4.04117	0.00000	0.52119	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
H	B	1000000000	0.47085	0.00000	0.15931	0.00000	0.00000	0.00000	0.00000	0.00000	0.00496
B	M	100000000	1.17141	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	E	100000000	0.01128	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
B	N	100000000	11.49120	0.00000	1.61699	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	I	100000000	4.44525	0.00000	0.07123	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	L	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	D	100000000	43.16752	0.04587	35.57158	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	A	100000000	0.54301	0.00000	0.22669	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
B	H	1000000000	4.66033	0.23905	1.77127	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	G	100000000	1.28382	0.00000	0.00288	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
B	J	100000000	23.13456	0.00000	23.13312	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	K	100000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	F	100000000	41.98331	0.00902	38.87955	0.00000	0.00000	0.00000	0.00000	0.00000	0.02557
B	O	100000000	0.02824	0.00000	0.00627	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
L	B	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
P	M	1000000000	0.11845	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
M	B	100000000	0.11538	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
M	R	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
M	P	1000000000	0.01745	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008

M	Q	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
S	D	1000000000	1.57189	0.00474	0.81229	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
T	D	10000000000	0.14519	0.00000	0.14518	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	T	10000000000	0.00002	0.00002	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	B	100000000	7.65725	0.00226	6.73423	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
D	S	1000000000	0.76534	0.00008	0.67327	0.00000	0.00000	0.00000	0.00000	0.00000	0.00015
D	C	100000000	0.02624	0.00150	0.02398	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
U	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
V	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
W	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
X	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Y	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Z	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AB	AC	100000000	15.47737	0.00000	15.47665	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AC	J	100000000	0.13320	0.00150	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AC	AB	100000000	0.13320	0.00150	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AD	J	100000000	7.66038	0.00000	7.65966	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	AC	100000000	15.47737	0.00000	15.47665	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	B	100000000	0.13754	0.00150	0.13453	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
J	AD	100000000	0.00754	0.00000	0.00678	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	B	100000000	4.09728	0.00000	0.42685	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
N	U	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AP	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AG	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AF	1000000000	0.39852	0.00000	0.03942	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	AE	1000000000	0.29762	0.00000	0.00326	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
N	V	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AH	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	X	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	W	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AE	N	1000000000	0.42448	0.00000	0.13157	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AH	N	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AF	N	1000000000	0.73896	0.00000	0.03013	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	Y	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	B	100000000	0.02115	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00451
I	AM	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AI	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	AJ	1000000000	0.19584	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
I	AK	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	Z	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AL	1000000000	0.26822	0.00000	0.00712	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AJ	I	1000000000	0.19378	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AI	I	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AN	AL	1000000000	0.25378	0.00000	0.00712	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AL	AN	1000000000	0.00336	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AL	AO	1000000000	0.00177	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AL	I	1000000000	0.00513	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AO	AL	1000000000	0.01444	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AP	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
O	B	100000000	1.84994	0.00000	0.00442	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AG	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Q	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
R	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AM	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AK	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
K	B	100000000	2.35739	2.35739	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Table Error! No text of specified style in document.**.9:** Link load under scenario 2

Start of link	End of Link	Baudrate	Load in Percent (%)	MAIN_1 in Percent (%)	MAIN_10 in Percent (%)	MAIN_100 in Percent (%)	MAIN_101 in Percent (%)	MAIN_110 in Percent (%)	MAIN_111 in Percent (%)	MAIN_120 in Percent (%)	MAIN_127 in Percent (%)
A	B	100000000	0.36472	0.00000	0.10038	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C	D	100000000	12.99314	0.00000	12.99242	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
E	B	100000000	1.27781	0.00000	0.02517	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
F	B	100000000	4.37554	0.00000	1.47077	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
G	B	100000000	3.99218	0.00000	0.51844	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
H	B	1000000000	0.46996	0.00000	0.15842	0.00000	0.00000	0.00000	0.00000	0.00000	0.00496

B	M	100000000	1.13682	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	E	100000000	0.01128	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
B	N	100000000	11.33980	0.00000	1.60057	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	I	100000000	4.37267	0.00000	0.06784	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	L	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	D	100000000	42.88086	0.04211	35.37102	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	A	100000000	0.54301	0.00000	0.22669	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
B	H	1000000000	4.53596	0.23800	1.69623	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	G	100000000	1.28382	0.00000	0.00288	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
B	J	100000000	22.67569	0.00000	22.67425	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	K	100000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	F	100000000	42.08961	0.01203	38.98285	0.00000	0.00000	0.00000	0.00000	0.00000	0.02557
B	O	100000000	0.02824	0.00000	0.00627	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
L	B	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
P	M	1000000000	0.11499	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
M	B	100000000	0.11538	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
M	R	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
M	P	1000000000	0.01745	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
M	Q	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
S	D	1000000000	1.56160	0.00436	0.81061	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
T	D	10000000000	0.14296	0.00000	0.14295	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	T	10000000000	0.00002	0.00002	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	B	100000000	7.59848	0.00301	6.67471	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
D	S	1000000000	0.75947	0.00015	0.66731	0.00000	0.00000	0.00000	0.00000	0.00000	0.00015
D	C	100000000	0.02624	0.00150	0.02398	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
U	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
V	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
W	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
X	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Y	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Z	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AB	AC	100000000	15.15163	0.00000	15.15091	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AC	J	100000000	0.13320	0.00150	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AC	AB	100000000	0.13320	0.00150	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AD	J	100000000	7.52726	0.00000	7.52654	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	AC	100000000	15.15163	0.00000	15.15091	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	B	100000000	0.13754	0.00150	0.13453	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
J	AD	100000000	0.00754	0.00000	0.00678	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	B	100000000	4.09728	0.00000	0.42685	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
N	U	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AP	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AG	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AF	1000000000	0.39852	0.00000	0.03942	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	AE	1000000000	0.29762	0.00000	0.00326	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
N	V	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AH	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	X	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	W	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AE	N	1000000000	0.41791	0.00000	0.13026	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AH	N	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AF	N	1000000000	0.73039	0.00000	0.02979	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	Y	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	B	100000000	0.02115	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00451
I	AM	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AI	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	AJ	1000000000	0.19584	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
I	AK	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	Z	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AL	1000000000	0.26442	0.00000	0.00678	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AJ	I	1000000000	0.19032	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AI	I	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AN	AL	1000000000	0.24998	0.00000	0.00678	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AL	AN	1000000000	0.00336	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AL	AO	1000000000	0.00177	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AL	I	1000000000	0.00513	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AO	AL	1000000000	0.01444	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AP	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
O	B	100000000	1.81534	0.00000	0.00442	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AG	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Q	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
R	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

AM	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AK	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
K	B	100000000	2.35438	2.35438	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Table Error! No text of specified style in document.**.10:** Link load under scenario 3

Start of link	End of Link	Baudrate	Load in Percent (%)	MAIN_1 in Percent (%)	MAIN_10 in Percent (%)	MAIN_100 in Percent (%)	MAIN_101 in Percent (%)	MAIN_110 in Percent (%)	MAIN_111 in Percent (%)	MAIN_120 in Percent (%)	MAIN_127 in Percent (%)
A	B	100000000	0.36472	0.00000	0.10038	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C	D	100000000	14.05691	0.00000	14.05619	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
E	B	100000000	1.37105	0.00000	0.02517	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
F	B	100000000	4.60796	0.00000	1.56044	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
G	B	100000000	4.16245	0.00000	0.54596	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
H	B	1000000000	0.47631	0.00000	0.16477	0.00000	0.00000	0.00000	0.00000	0.00000	0.00496
B	M	100000000	1.23006	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	E	100000000	0.01128	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
B	N	100000000	11.80870	0.00000	1.68908	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	I	100000000	4.56765	0.00000	0.07632	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	L	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	D	100000000	46.60763	0.04662	38.85427	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	A	100000000	0.54301	0.00000	0.22669	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
B	H	1000000000	4.89628	0.23920	1.91823	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	G	100000000	1.28382	0.00000	0.00288	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
B	J	100000000	25.47155	0.00000	25.47011	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	K	100000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	F	100000000	46.07365	0.00902	42.96989	0.00000	0.00000	0.00000	0.00000	0.00000	0.02557
B	O	100000000	0.02824	0.00000	0.00627	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
L	B	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
P	M	1000000000	0.12432	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
M	B	100000000	0.11538	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
M	R	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
M	P	1000000000	0.01745	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
M	Q	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
S	D	1000000000	1.67667	0.00481	0.90133	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
T	D	10000000000	0.15808	0.00000	0.15807	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	T	10000000000	0.00002	0.00002	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	B	100000000	7.94743	0.00226	7.02440	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
D	S	1000000000	0.79436	0.00008	0.70228	0.00000	0.00000	0.00000	0.00000	0.00000	0.00015
D	C	100000000	0.02624	0.00150	0.02398	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
U	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
V	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
W	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
X	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Y	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Z	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AB	AC	100000000	16.99174	0.00000	16.99102	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AC	J	100000000	0.13320	0.00150	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AC	AB	100000000	0.13320	0.00150	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AD	J	100000000	8.48301	0.00000	8.48229	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	AC	100000000	16.99174	0.00000	16.99102	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	B	100000000	0.13754	0.00150	0.13453	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
J	AD	100000000	0.00754	0.00000	0.00678	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	B	100000000	4.09728	0.00000	0.42685	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
N	U	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AP	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AG	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AF	1000000000	0.39852	0.00000	0.03942	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	AE	1000000000	0.29762	0.00000	0.00326	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
N	V	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AH	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	X	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	W	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AE	N	1000000000	0.44005	0.00000	0.13827	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AH	N	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AF	N	1000000000	0.75514	0.00000	0.03064	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	Y	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	B	100000000	0.02115	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00451

I	AM	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AI	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	AJ	1000000000	0.19584	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
I	AK	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	Z	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AL	1000000000	0.27459	0.00000	0.00763	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AJ	I	1000000000	0.19965	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AI	I	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AN	AL	1000000000	0.26015	0.00000	0.00763	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AL	AN	1000000000	0.00336	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AL	AO	1000000000	0.00177	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AL	I	1000000000	0.00513	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AO	AL	1000000000	0.01444	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AP	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
O	B	100000000	1.90859	0.00000	0.00442	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AG	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Q	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
R	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AM	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AK	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
K	B	100000000	2.35814	2.35814	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Discussions

In this paper, the primary challenge in estimating the total Ethernet link load lies in accurately assessing dynamic link loads. Several methodologies were applied to real-world trace files, including probability distribution analysis, causal convolutional networks, decomposed convolutional networks, and the UDAE. While probability analysis, under the assumption of a Poisson distribution, initially appeared promising, further examination of the automotive Ethernet context revealed that the data followed a NHPP. This discovery complicated the estimation process, as determining an accurate arrival rate function $\hat{r_x}(T)$ is difficult.

For machine learning approaches, the objective was to segment the trace files and classify them into distinct scenarios, within which a Poisson distribution with a constant arrival rate could be assumed, thereby facilitating a more accurate estimation of dynamic link load. The classification procedure involved applying PCA to the input data, followed by using different encoders to learn the data's internal representations, and finally employing DPGMM for classification. All three encoders, causal convolutional networks, decomposed convolutional networks, and UDAE, demonstrated segmentation and classification capabilities with the trace files. However, no single encoder consistently outperformed the others across the entire trace file, leading to the introduction of a voting and merging procedure, which proved to be effective.

Several notable observations were made from the Ethernet link load results. In certain scenarios, the Ethernet link load exhibited a significant increase compared to the overall average across the entire trace. For instance, in the case of link BD, the link load increased by over 15%, rising from 45.24% to 60.32% in scenario 8. This finding

underscores the effectiveness of the segmentation and classification method, as it successfully identified scenarios that caused higher link loads, thereby enhancing the reliability and accuracy of the link load estimation.

Another key observation is that, for most links, cyclic PDUs are the predominant contributors to the link load. However, in a few links, the consideration of dynamic PDUs resulted in an unexpected load increase of up to 41%. For example, in the link BF, the link rises from 4.92813% to 45.24258%. One of such links is associated with the connection between the central infotainment computer and the in-exterior function controller, indicating that infotainment systems can cause substantial throughput bursts on Ethernet links.

Future Works

Several avenues for future research and development remain open. The segmentation method proposed in this study could be extended to evaluate how different vehicle topologies influence signal transmission behavior. Additionally, researchers could investigate whether vehicles of the same type exhibit varying behaviors in signal transmission, providing valuable insights into the differences and effectiveness of various network designs in automotive systems.

Beyond the use of Gaussian mixtures, exploring the application of negative binomial mixtures for certain signal cases may yield more accurate results. Since signal transmission rates can be modeled as event occurrence rates within a specific time interval, negative binomial mixtures could potentially offer a better fit. Several coding approaches were explored during the research, though they did not yield promising results. The corresponding code is included in the appendix, enabling future researchers to build upon the existing work.

Moreover, training additional UDAE models could improve segmentation results, leading to more precise classifications. In the estimation process, separating dual cyclic and cyclic-on-change PDUs from dynamic PDUs, rather than grouping them together, may result in more accurate load estimations.

Finally, developing a comprehensive table that displays the maximum values across all scenarios could be a valuable tool for future studies, providing a clear and concise summary of the peak load conditions under different configurations.

Conclusions

In this study, a comprehensive investigation of link load in automotive **Ethernet** networks was conducted. While there are few prior studies specifically focused on automotive **Ethernet**, existing research on CAN buses was adapted in this work to calculate static link load. A model based on ARXML files was established for this purpose, offering a more practical approach compared to traditional Excel-based NCD models. The model considered various PDU types, transport protocols, AUTOSAR specifications, and error rates to ensure accuracy. Static link load was automatically calculated using values derived from the corresponding ARXML files and protocols, with VLAN information also incorporated. A table view was developed in the ARXML Visualizer to present the static link loads, both overall and within individual VLANs.

For dynamic link load, empirical derivation was necessary. Data was collected from vehicles and was interpreted using an internal tool called "Bug Hunter." Custom scripts were created to gather PDU counts every 10 seconds from .mfx trace files, with SQLite used to expedite the process, because processing DataBases using SQL is much more faster than processing Excel files or Python arrays when the data amount is huge. Diagrams depicting PDU transmission frequency were generated to provide insights into signal transmission patterns. Although statistical methods for estimating a distribution were ultimately inapplicable, they offered valuable perspectives on segmentation and classification approaches, as well as the utility of convolutional networks for feature extraction.

The DPGMM segmentation algorithm, in combination with PCA analysis, proved effective, since together with an encoder, they successfully identify the zero values deliberately added in the trace data. Feature extraction through deep learning models also yielded positive results. Several models were evaluated, including the causal convolutional network, the decompositional convolutional network, and the UDAE, with experiments conducted across various hyperparameters. The results indicated that no single model could optimally segment and classify all trace segments. As a result, a voting and merging process was implemented, ultimately achieving a sufficiently accurate segmentation result.

Based on the segmentation and classification outcomes, a Poisson distribution model was proposed, and the minimum PDU frequency was estimated within a 95% confidence level. This information was then integrated into the link load model in the ARXML Visualizer, resulting in the creation of a comprehensive link load table that accounts for different scenarios as well as the overall average value.

Literature

[1] Z. Z. W. a. Z. F. Liu, Impact, Challenges and Prospect of Software-Defined Vehicles, *Automot. Innov,* Bd. 5, p. 180–194, 2022.

[2] G. S. V. P. a. M. L. V. Bandur, Making the Case for Centralized Automotive E/E Architectures, *IEEE Transactions on Vehicular Technology,* Bd. 70, Nr. 2, pp. 1230-1245, 2021.

[3] L. G. P. a. L. L. Lo Bello, A Perspective on Ethernet in Automotive Communications—Current Status and Future Trends, *Applied Sciences,* Bd. 13, Nr. 3, p. 1278, 2023.

[4] M. M. K. S. R. S. S. M. S. M. L. R. K. S. F. B. R. P. O. B. S. M. a. M. L. Friesa, An Overview of Costs for Vehicle Components , Fuels, Greenhouse Gas Emissions and Total Cost of Ownership Update 2017, *Environmental Science, Economics, Engineering,* 2018.

[5] Automotive Ethernet Cables, 2024. [Online]. Available: https://www.lapptannehill.com/wire-cable/automotive-wire-cable/automotive-Ethernet-cable.

[6] M. Staron, AUTOSAR (AUTomotive Open System ARchitecture), in *Automotive Software Architectures*, Springer, 2021, p. 97–136.

[7] A. a. S. P. Vetter, Custom Toolchain as Bridge between OEM Specific Needs and Standard Software, in *AUTOSAR 20th Anniversary 2003-2023*, AUTOSAR Community, 2023, p. 168–169.

[8] Layered Software Architecture R23-11, AUTOSAR Administritive, 2023.

[9] M. A. N. R. N. D. B. P. S. K. R. A. a. V. K. Y. Shajahan, AUTOSAR Classic vs. AUTOSAR Adaptive: A Comparative Analysis in Stack Development, *Engineering International,* Bd. 7, Nr. 2, pp. 161-178, 2019.

[10] Application Interfaces User Guide CP R22-11, AUTOSAR Administration, 2023.

[11] K. a. T. K. Matheus, Automotive Ethernet, Cambridge University Press, 2021.

[12] W. a. S. R. Zimmermann, Bussysteme in der Fahrzeugtechnik. Protokolle, Standards und Softwarearchitektur, Wiesbaden: Springer Vieweg, 2014.

[13] System Template AUTOSAR CP R23-11, AUTOSAR Administration, 2023.

[14] T. a. T. M. Streichert, Elektrik/Elektronik- Architekturen im Kraftfahrzeug, Berlin und Heidelberg: Springer, 2012.

[15] N. e. a. Islam, Quality of Service Analysis of Ethernet Network Based on Packet Size, *Journal of Computer and communications,* Bd. 04, pp. 63-72, 2016.

[16] L. V. a. D. H. H. -T. Lim, Challenges in a future IP/Ethernet-based in-car network for real-time applications, in *48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, San Diego, CA, USA, 2011.

[17] S. H. a. H. Kim, On AUTOSAR TCP/IP Performance in In-Vehicle Network Environments, *IEEE Communications Magazine,* Bd. 54, Nr. 12, pp. 168-173, 2016.

[18] T. a. L. O. Nemenz, Erstellung einer Simulation zur Analyse und Bewertung der Busleistungsfähigkeit anhand der Baureihe 223 (S-Klasse), Göppingen: Bachelorarbeit, Hochschule Esslingen, 2019.

[19] Specification of UDP Network Management AUTOSAR CP R22-11, AUTOSAR Administritive, 2022.

[20] SOME/IP Service Discovery Protocol Specification AUTOSAR FO R22-11, AUTOSAR Administritive, 2022.

[21] SOME/IP Protocol Specification AUTOSAR R22-11, AUTOSAR Administritive, 2022.

[22] W. Goralski, The illustrated network: how TCP/IP works in a modern network, 2017: Goralski, Walter, Morgan Kaufmann.

[23] Specification of Socket Adaptor AUTOSAR CP R22-11, AUTOSAR Administritive, 2022.

[24] T. Schäuffele Jörg and Zurawka, Automotive Software Engineering. Grundlagen, Prozesse, Methoden und Werkzeuge effizient einsetzen. 6. Auflage. ATZ / MTZ-Fachbuch, Wiesbaden: Springer Vieweg, 2016.

[25] A. a. R. E. Kostrzewa, Achieving safety and performance with reconfiguration protocol for Ethernet TSN in automotive systems, *Journal of Systems Architecture,* p. 118, 2021.

[26] H. W. Y. S. H. a. J.-M. C. Kim, In-vehicle network average response time analysis for CAN-FD and automotive Ethernet, *IEEE Transactions on Vehicular Technology,* Bd. 6, Nr. 72, pp. 6916-6932, 2023.

[27] M. W. L. J. L. Z. a. S. G. Tang C, Simulation Study of Short Message Burst Signals Based on Poisson Distribution, in *China Satellite Navigation Conference (CSNC) 2020 Proceeding*, Singapore, 2020.

[28] G. Cumming, Understanding the new statistics: Effect sizes, confidence intervals, and meta-analys, Routledge, 2013.

[29] V. &. F. S. Paxson, Wide area traffic: the failure of Poisson modeling, *IEEE/ACM Transactions on networking,* Bd. 3, Nr. 3, pp. 226-244, 1995.

[30] H. N. L. Y. B. &. L. C. L. Hung, Deriving the distributions for the numbers of short message arrivals, *Wireless Communications and Mobile Computing,* Bd. 14, Nr. 4, pp. 450-459, 2014.

[31] S. D. Y. Y. C. K. K. U. L. &. K. E. Gharghabi, Matrix Profile VIII: Domain Agnostic Online Semantic Segmentation at Superhuman Performance Levels, in *IEEE International Conference on Data Mining*, LA, 2017.

[32] Y. S. a. C. F. Yasuko Matsubara, AutoPlait: automatic mining of co-evolving time sequence, in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, New York, 2014.

[33] M. N. T. N. T. M. D. K. I. &. T. W. Nagano, HVGH: Unsupervised Segmentation for High-Dimensional Time Series Using Deep Neural Compression and Statistical Generative Model, *Frontiers in robotics and AI,* Bd. 6, pp. 115-116, 2019.

[34] A. a. W. S. R. Principle Rea, How Many Components should be Retained from a Multivariate Time Series PCA, *Methodology,* 2016.

[35] H. L. D. Z. Y. Z. W. Q. a. C. S. J. Zhichen Lai, E2Usd: Efficient-yet-effective Unsupervised State Detection for Multivariate Time Series, in *Proceedings of the ACM on Web Conference*, New York, 2024.

[36] N. C. S. W. a. S. S. K. Preechakul, Diffusion Autoencoders: Toward a Meaningful and Decodable Representation, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, LA, 2022.

[37] K. W. T. Z. a. Z. C. .. a. 1. 1. A. 1. (. 2. 1. p. h. Chengyu Wang, Time2State: An Unsupervised Framework for Inferring the Latent States in Time Series Data, *Proc. ACM Manag. Dat,* Bd. 1, Nr. 1, p. 18, 2023.

Appendix

Some of the codes are emitted and not putted in the Appendix

A.1 Source Code for Static Link Load Calculation

Link.java:

```
package autosar.model.EthernetCom;

import autosar.model.classic.communicationCluster.CouplingPortConnection;
import autosar.model.classic.communicationCluster.physicalChannel.NetworkPhysical
       ChannelOrVlan;
import autosar.model.classic.couplingElement.couplingPort.VlanMembership;

import java.math.BigDecimal;
import java.util.*;
import java.util.stream.Collectors;

import static autosar.schema.Ar4SchemaDefinition.*;

public class Link {

    private final CouplingPortConnection connection;

    private final String direction;

    BigDecimal baudrate;

    HashMap<NetworkPhysicalChannelOrVlan, BigDecimal> loadPerVlan = new
HashMap<>();

    public Link(CouplingPortConnection connection, String direction) {
        this.connection = connection;
        this.direction = direction;
        if(connection.getFirstPort().getPhysicalLayerLeaf() != null){
            switch (connection.getFirstPort().getPhysicalLayerLeaf().getValue()){
                case "1000BASE-T": //Ethernet Standard (IEEE 802.3ab) to support
1Gbit/s over 4 twisted pairs.
                case "1000BASE-T1": //Ethernet Standard (IEEE 802 3bp) to support
1Gbit/s over a single twisted pair cable.
                    baudrate = BigDecimal.valueOf(1000000000); break;
                case "100BASE-T1": //Ethernet Standard (IEEE 802.3bw) to support
100Mbit/s over a single twisted pair cable. 100BASE-T1 is the IEEE Standardized
version of BroadRReach.
                case "100BASE-TX": //Ethernet Standard (IEEE 802.3u) to support
100Mbit/s over two twisted pairs.
                    baudrate = BigDecimal.valueOf(100000000); break;
                case "10BASE-T1S": baud rate = BigDecimal.valueOf(10000000); break;
//Physical layer interface  (10Mbit/s, 2 pairs). Used for automotive.
                case "IEEE802-11p": break; //Ethernet Standard (IEEE 802.11p) to
support wireless communication in vehicular environments
            }
        }
        else if(connection.getFirstPort().getMacLayerLeaf() != null){
            switch (connection.getFirstPort().getMacLayerLeaf().getValue()){
```

```
                case "X-MII": baud rate = BigDecimal.valueOf(100000000); break;
//Mac layer interface (data) bandwith class 10-100Mbit/s (e.g.RMII; RvMII, SMII,
RvMID
                case "XG-MII": baud rate = BigDecimal.valueOf(1000000000); break;
//Mac layer interface (data) bandwith class 1Gbit/s (e.g. GMII, RGMII, SGMII,
RGMII, USGMII)
                case "XXG-MII":  baudrate = new BigDecimal("10000000000"); break;
//Mac layer interface (data) bandwith class 10Gbit/s
            }
        }
        else{
            baudrate =
BigDecimal.valueOf(connection.getParentAutosarObject(ETHERNET_CLUSTER).getXmlLeafBy
ExactPath(ETHERNET_CLUSTER_VARIANTS, ETHERNET_CLUSTER_CONDITIONAL,
BAUDRATE).getIntegerValue());
        }
        LinkedList<VlanMembership> list = (LinkedList<VlanMembership>)
connection.getFirstPort().getVlanMemberships();
        list.get(0).getVlan();
        Collection<VlanMembership> firstPortVlanMemberships =
connection.getFirstPort().getVlanMemberships();
        Set<NetworkPhysicalChannelOrVlan> firstPortVlanSet =
firstPortVlanMemberships.stream()
                .map(VlanMembership::getVlan)
                .collect(Collectors.toCollection(HashSet::new));
        Map<NetworkPhysicalChannelOrVlan, BigDecimal> firstPortVlanMap;
        firstPortVlanMap = firstPortVlanSet.stream().collect(Collectors.toMap(key -
> key, value -> BigDecimal.ZERO));
        loadPerVlan.putAll(firstPortVlanMap);
        Collection<VlanMembership> secondPortVlanMemberships =
connection.getSecondPort().getVlanMemberships();
        HashSet<NetworkPhysicalChannelOrVlan> secondPortVlanSet =
secondPortVlanMemberships.stream()
                .map(VlanMembership::getVlan)
                .collect(Collectors.toCollection(HashSet::new));
        Map<NetworkPhysicalChannelOrVlan, BigDecimal> secondPortVlanMap;
        secondPortVlanMap = secondPortVlanSet.stream().collect(Collectors.toMap(key
-> key, value -> BigDecimal.ZERO));
        loadPerVlan.putAll(secondPortVlanMap);
    }

    public BigDecimal getLoad() {
        return loadPerVlan.values().stream()
                .reduce(BigDecimal.ZERO, BigDecimal::add);
    }

    public float getLoadPercent(){
        BigDecimal loadPercentBig = this.loadPerVlan.values().stream()
                .reduce(BigDecimal.ZERO,
BigDecimal::add).multiply(BigDecimal.valueOf(100));
        return loadPercentBig.floatValue();
    }

    public void addLoad(BigDecimal load, NetworkPhysicalChannelOrVlan vlan){
        this.loadPerVlan.merge(vlan, load.divide(baudrate), BigDecimal::add);
    }

    public BigDecimal getBaudrate(){
        return baudrate;
    }

    public CouplingPortConnection getConnection() {
```

```
        return connection;
    }

    public String getDirection() {
        return direction;
    }

    public HashMap<NetworkPhysicalChannelOrVlan, BigDecimal> getloadPerVlan(){
        return loadPerVlan;
    }
}
```

TranmissionProperties.java:

```
package autosar.model.EthernetCom;

import
autosar.model.classic.communicationCluster.physicalChannel.NetworkPhysicalChannelOr
Vlan;
import autosar.model.classic.pdu.GeneralPurposePdu;
import autosar.model.classic.pdu.ISignalIPdu;
import autosar.model.classic.pdu.NmPdu;
import autosar.model.classic.pdu.SecureIPdu;
import autosar.model.classic.pdu.iPduTimingSpecification.PduTiming;

public class TransmissionProperties {

    private final CommunicationEndpoint baseData;
    private final NetworkPhysicalChannelOrVlan vlan;


    public TransmissionProperties(CommunicationEndpoint sender,
CommunicationEndpoint receiver) {
        if(sender != null)
        {
            baseData = sender;
        }
        else
        {
            baseData = receiver;
        }
        if(baseData.getConnector() != null){
            vlan = baseData.getConnector().getNetworkPhysicalChannelOrVlan();
        }
        else{
            vlan = baseData.getSocket().getVlan();
        }
    }

    public int getFrameSize(){
        int pduSize = baseData.getPduId().getPdu().getLengthInByte();
        if(baseData.getPduId().getPdu() instanceof SecureIPdu){
            pduSize = pduSize + 16;
        }
        else if (baseData.getPduId().getPdu() instanceof ISignalIPdu) {
            pduSize = pduSize + 8;
        }
        else if (baseData.getPduId().getPdu() instanceof NmPdu) {
            //Figure 7-1  AUTOSAR_SWS_UDPNetworkManagement
        }
        else if (baseData.getPduId().getPdu() instanceof GeneralPurposePdu) {
            if(baseData.getPduId().getPdu().getCategoryString().equals("SD")){
```

```
                //XCP_Book_V1.5_EN.pdf
            }

if(baseData.getPduId().getPdu().getCategoryString().equals("SOMEIP_SEGMENTED_IPDU")
){
                pduSize = pduSize + 16;//SOME/IP Protocol Specification Figure 4.19
            }
            if(baseData.getPduId().getPdu().getCategoryString().equals("DoIP")){
                pduSize = pduSize + 8;//SWS_DoIP_00004
            }
            if(baseData.getPduId().getPdu().getCategoryString().equals("XCP")){
                pduSize = pduSize + 8;//XCP_Book_V1.5_EN.pdf
            }
        }
        int transportHeaderSize = 0;
        if(baseData.getSocket().getTpConfig().getProtocolType().equals("UDP")){
            transportHeaderSize = 8;// UDP
        }
        else{
            transportHeaderSize = 30;// TCP with time stamp
        }
        int frameSize = pduSize + 20 + transportHeaderSize + 8 + 30 + 12;//ip
header, transportation header, SoAd PDU header, frame header and inter frame gap
        return frameSize;
    }

    public double getPduPerSec() {
    double pduPerSec = 0;

    PduTiming pduTiming = baseData.getPduId().getPdu().getPduTiming();
    Double cycleTime;
    if (pduTiming != null) {
        cycleTime = pduTiming.getCycleTime();
        if (cycleTime != null) pduPerSec = 1 / cycleTime;
        else {
            if(baseData.getPduId().getPdu().getName() != null){
                pduPerSec = getPduNumbers(baseData.getPduId().getPdu().getName(),
getVlan());
            }
        }
    } else {
        if(baseData.getPduId().getPdu().getName() != null){
            pduPerSec = getPduNumbers(baseData.getPduId().getPdu().getName(),
getVlan());
        }
    }
    return pduPerSec;
}

public NetworkPhysicalChannelOrVlan getVlan() {
    return vlan;
}

public double getPduNumbers(String PduName, NetworkPhysicalChannelOrVlan vlan) {
    int index = vlan.getName().indexOf("_");
    PduName = PduName + "_" + vlan.getName().substring(index + 1);
    if (PDUNumberMap.get(PduName) != null){
        return PDUNumberMap.get(PduName);
    }
    else{
        return 0;
```

```
    }
}
```

Part of EthernetComModel.java

```
public HashMap<CouplingPortConnection, HashMap<String, Link>> getLinks(){
    HashMap<String, Double> PDUNumberMap = createPDUNumberMap(ClassFilePath);
    HashMap<CouplingPortConnection, HashMap<String, Link>> linksConnection= new
HashMap<>();
    for(EndpointConnection endpointConnection : endpointConnections){
        endpointConnection.setTransmissionProperties();
        EndpointRoute route = endpointConnection.getRoute();
        NetworkPhysicalChannelOrVlan vlan =
endpointConnection.getTransmissionProperties().getVlan();
        if(route != null && route.isUsed()){
            List<Link> routeLinks = route.getLinks();
            BigDecimal load =
BigDecimal.valueOf(endpointConnection.getTransmissionProperties().getFrameSize() *
8
                    *
endpointConnection.getTransmissionProperties().getPduPerSec());
            for(Link link : routeLinks){
                if(linksConnection.containsKey(link.getConnection())){
                    HashMap<String, Link> linkDirection =
linksConnection.get(link.getConnection());
                    if(linkDirection.containsKey(link.getDirection())){
                        linkDirection.get(link.getDirection()).addLoad(load, vlan);
                    }
                    else{
                        link.addLoad(load, vlan);
                        linkDirection.put(link.getDirection(), link);
                    }
                }
                else {
                    link.addLoad(load, vlan);
                    HashMap<String, Link> linkDirection = new HashMap<>();
                    linkDirection.put(link.getDirection(), link);
                    linksConnection.put(link.getConnection(), linkDirection);
                }
            }
        }
    }
    return linksConnection;
}

public HashMap<String, Double> createPDUNumberMap(String ClassFilePath)
{
    HashMap<String, Double> PDUNumberMap = new HashMap<>();
    try (BufferedReader br = new BufferedReader(new FileReader(ClassFilePath))) {
        String line;
        while ((line = br.readLine()) != null) {
            String[] parts = line.split("\\s+");
            if (parts.length == 2) {
                String key = parts[0];
                double value = Double.parseDouble(parts[1]);
                PDUNumberMap.put(key, value);
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
```

```
    return PDUNumberMap;
}
```

Part of EcuInterface.java

```
public int getMaximumTransmissionUnit()
{
    XmlLeaf ret= getXmlLeafByPath(MAXIMUM_TRANSMISSION_UNIT);
    if(ret == null)
    {
       IdentifiableAutosarObject ne =
getIdentifiableAutosarObjectByRefPath("*NETWORK-ENDPOINT-REF");
    }
    return ret.getIntegerValue();
}
```

A.2 Source Code for Table Establishment

EthernetComModelTable5.java

```
package visualizer.functions.tableDetails.views.network;

import autosar.model.AutosarModelDatabase;
import autosar.model.classic.communicationCluster.CouplingPortConnection;
import
autosar.model.classic.communicationCluster.physicalChannel.NetworkPhysicalChannelOr
Vlan;
import autosar.model.EthernetCom.*;
import autosar.schema.Ar4SchemaDefinition;
import visualizer.functions.tableDetails.views.AbstractTableView;
import visualizer.model.xml.XmlItem;
import xml.query.ArxmlQuery;

import java.math.BigDecimal;
import java.util.*;

public class EthernetComModelTable5 extends AbstractTableView
{
    Collection<NetworkPhysicalChannelOrVlan> vlans = new
TreeSet<>(Comparator.comparing(NetworkPhysicalChannelOrVlan::getShortName));

    public EthernetComModelTable5()
    {
        setIcon1("Ethernet");
        setIcon2("mapping");
        setTitle("Ethernet Communication Model: Static Link Load");
    }

    @Override
    protected void CalculateTableData(final AutosarModelDatabase autosarModelDb) {
String[] text = new String[]{"Total Average", "Scenario 1", "Scenario 2", "Scenario
3", "Scenario 4","Scenario 5", "Scenario 6", "Scenario 7", "Scenario 8", "Scenario
9","Scenario 10"};
String Scenario = ViewController.getInstance().showCombo(text, "Select Scenario");
if (Scenario == null) {
    Scenario = "Total Average";
}
```

```
String ClassFilePath = "C:\\";
switch(Scenario) {
    case "Total Average":
        ClassFilePath =
"C:\\Users\\YANZCHE\\Desktop\\Time2State\\poisson_values_total.txt";
        break;

    case "Scenario 1":
        ClassFilePath =
"C:\\Users\\YANZCHE\\Desktop\\Time2State\\poisson_values_2.txt";
        break;

    case "Scenario 2":
        ClassFilePath =
"C:\\Users\\YANZCHE\\Desktop\\Time2State\\poisson_values_3.txt";
        break;

    case "Scenario 3":
        ClassFilePath =
"C:\\Users\\YANZCHE\\Desktop\\Time2State\\poisson_values_4.txt";
        break;

    case "Scenario 4":
        ClassFilePath =
"C:\\Users\\YANZCHE\\Desktop\\Time2State\\poisson_values_5.txt";
        break;

    case "Scenario 5":
        ClassFilePath =
"C:\\Users\\YANZCHE\\Desktop\\Time2State\\poisson_values_6.txt";
        break;

    case "Scenario 6":
        ClassFilePath =
"C:\\Users\\YANZCHE\\Desktop\\Time2State\\poisson_values_7.txt";
        break;

    case "Scenario 7":
        ClassFilePath =
"C:\\Users\\YANZCHE\\Desktop\\Time2State\\poisson_values_8.txt";
        break;

    case "Scenario 8":
        ClassFilePath =
"C:\\Users\\YANZCHE\\Desktop\\Time2State\\poisson_values_9.txt";
        break;

    case "Scenario 9":
        ClassFilePath =
"C:\\Users\\YANZCHE\\Desktop\\Time2State\\poisson_values_10.txt";
        break;

    case "Scenario 10":
        ClassFilePath =
"C:\\Users\\YANZCHE\\Desktop\\Time2State\\poisson_values_11.txt";
        break;

    default:
        ClassFilePath =
"C:\\Users\\YANZCHE\\Desktop\\Time2State\\poisson_values_total.txt";
        break;
}
```

```
EthernetComModel comModel = autosarModelDb.getEhternetComModel();
HashMap<CouplingPortConnection, HashMap<String, Link>> linksConnection =
comModel.getLinks(ClassFilePath);
        for (CouplingPortConnection connection : linksConnection.keySet()) {
            HashMap<String, Link> interLink= linksConnection.get(connection);
            for (Link link : interLink.values()){
                newLine();
                HashMap<NetworkPhysicalChannelOrVlan, BigDecimal> loadPerVlan =
link.getloadPerVlan();
                if(link.getDirection().equals("OUT")){
                    if(link.getConnection().getFirstPort().getSwitch() == null){
                        if(link.getConnection().getSecondPort().getSwitch() ==
null){

addCell(link.getConnection().getFirstPort().getComController().getEcu());

addCell(link.getConnection().getSecondPort().getComController().getEcu());
                            addCell(link.getBaudrate());
                            addCell(String.format("%.05f", link.getLoadPercent()));
                            for (NetworkPhysicalChannelOrVlan vlan : vlans) {
                                if(loadPerVlan.containsKey(vlan)){

addCell(loadPerVlan.get(vlan).multiply(BigDecimal.valueOf(100)));
                                }
                                else{
                                    addCell(String.format("%.05f",
BigDecimal.valueOf(0)));
                                }
                            }
                        }
                        else {

addCell(link.getConnection().getFirstPort().getComController().getEcu());

addCell(link.getConnection().getSecondPort().getSwitch());
                            addCell(link.getBaudrate());
                            addCell(String.format("%.05f", link.getLoadPercent()));
                            for (NetworkPhysicalChannelOrVlan vlan : vlans) {
                                if(loadPerVlan.containsKey(vlan)){
                                    addCell(String.format("%.05f",
loadPerVlan.get(vlan).multiply(BigDecimal.valueOf(100))));
                                }
                                else{
                                    addCell(String.format("%.05f",
BigDecimal.valueOf(0)));
                                }
                            }
                        }
                    }
                    else {
                        if(link.getConnection().getSecondPort().getSwitch() ==
null){

addCell(link.getConnection().getFirstPort().getSwitch());

addCell(link.getConnection().getSecondPort().getComController().getEcu());
                            addCell(link.getBaudrate());
                            addCell(String.format("%.05f", link.getLoadPercent()));
                            for (NetworkPhysicalChannelOrVlan vlan : vlans) {
                                if(loadPerVlan.containsKey(vlan)){
                                    addCell(String.format("%.05f",
```

```
loadPerVlan.get(vlan).multiply(BigDecimal.valueOf(100))));
                                    }
                                    else{
                                        addCell(String.format("%.05f",
BigDecimal.valueOf(0)));
                                    }
                                }
                            }
                            else {

addCell(link.getConnection().getFirstPort().getSwitch());

addCell(link.getConnection().getSecondPort().getSwitch());
                                addCell(link.getBaudrate());
                                addCell(String.format("%.05f", link.getLoadPercent()));
                                for (NetworkPhysicalChannelOrVlan vlan : vlans) {
                                    if(loadPerVlan.containsKey(vlan)){
                                        addCell(String.format("%.05f",
loadPerVlan.get(vlan).multiply(BigDecimal.valueOf(100))));
                                    }
                                    else{
                                        addCell(String.format("%.05f",
BigDecimal.valueOf(0)));
                                    }
                                }
                            }
                        }
                    }
                    else {
                        if(link.getConnection().getFirstPort().getSwitch() == null){
                            if(link.getConnection().getSecondPort().getSwitch() ==
null){

addCell(link.getConnection().getSecondPort().getComController().getEcu());

addCell(link.getConnection().getFirstPort().getComController().getEcu());
                                addCell(link.getBaudrate());
                                addCell(String.format("%.05f", link.getLoadPercent()));
                                for (NetworkPhysicalChannelOrVlan vlan : vlans) {
                                    if(loadPerVlan.containsKey(vlan)){
                                        addCell(String.format("%.05f",
loadPerVlan.get(vlan).multiply(BigDecimal.valueOf(100))));
                                    }
                                    else{
                                        addCell(String.format("%.05f",
BigDecimal.valueOf(0)));
                                    }
                                }
                            }
                            else {

addCell(link.getConnection().getSecondPort().getSwitch());

addCell(link.getConnection().getFirstPort().getComController().getEcu());
                                addCell(link.getBaudrate());
                                addCell(String.format("%.05f", link.getLoadPercent()));
                                for (NetworkPhysicalChannelOrVlan vlan : vlans) {
                                    if(loadPerVlan.containsKey(vlan)){
                                        addCell(String.format("%.05f",
loadPerVlan.get(vlan).multiply(BigDecimal.valueOf(100))));
                                    }
                                    else{
```

```
                                    addCell(String.format("%.05f",
BigDecimal.valueOf(0)));
                                }
                            }
                        }
                    }
                    else {
                        if(link.getConnection().getSecondPort().getSwitch() ==
null){

addCell(link.getConnection().getSecondPort().getComController().getEcu());

addCell(link.getConnection().getFirstPort().getSwitch());
                            addCell(link.getBaudrate());
                            addCell(String.format("%.05f", link.getLoadPercent()));
                            for (NetworkPhysicalChannelOrVlan vlan : vlans) {
                                if(loadPerVlan.containsKey(vlan)){
                                    addCell(String.format("%.05f",
loadPerVlan.get(vlan).multiply(BigDecimal.valueOf(100))));
                                }
                                else{
                                    addCell(String.format("%.05f",
BigDecimal.valueOf(0)));
                                }
                            }
                        }
                        else{

addCell(link.getConnection().getSecondPort().getSwitch());

addCell(link.getConnection().getFirstPort().getSwitch());
                            addCell(link.getBaudrate());
                            addCell(String.format("%.05f", link.getLoadPercent()));
                            for (NetworkPhysicalChannelOrVlan vlan : vlans) {
                                if(loadPerVlan.containsKey(vlan)){
                                    addCell(String.format("%.05f",
loadPerVlan.get(vlan).multiply(BigDecimal.valueOf(100))));
                                }
                                else{
                                    addCell(String.format("%.05f",
BigDecimal.valueOf(0)));
                                }
                            }
                        }
                    }
                }
            }
        }
    }

    @Override
    public boolean isValid(final AutosarModelDatabase modelDb, final boolean
isCompare)
    {
        Collection<XmlItem> couplingElements =
ArxmlQuery.querySearchXmlItems(modelDb.getDataModel().getXmlDb(), "<" +
Ar4SchemaDefinition.COUPLING_ELEMENT + ">", null);
        return !couplingElements.isEmpty();
    }

    @Override
    public void initialiseTableHeaders(final AutosarModelDatabase autosarModelDb)
```

```
    {
        List<String> headers = new ArrayList<>();
        headers.add("Start of link");
        headers.add("End of Link");
        headers.add("Baudrate");
        headers.add("Load in Percent (%)");

vlans.addAll(autosarModelDb.getAutosarObjects(NetworkPhysicalChannelOrVlan.class));
        for(NetworkPhysicalChannelOrVlan vlan : vlans){
            headers.add(vlan.getShortName() + " in Percent (%)");
        }
        setTableHeaders(headers);
    }

}
```

A.3 Part of the Code from the UDAE

```
import math
from dataclasses import dataclass
from numbers import Number
from typing import NamedTuple, Tuple, Union

import numpy as np
import torch as th
from torch import nn
import torch.nn.functional as F
from choices import *
from config_base import BaseConfig
from .blocks import *

from .nn import (conv_nd, linear, normalization, timestep_embedding,
                 torch_checkpoint, zero_module)


class DiFFUNetConfig(BaseConfig):
    in_channels: int = 3
    model_channels: int = 64
    out_channels: int = 3
    num_res_blocks: int = 2
    num_input_res_blocks: int = None
    embed_channels: int = 512
    time_embed_channels: int = None
    dropout: float = 0.1
    channel_mult: Tuple[int] = (1, 2, 4, 8)
    input_channel_mult: Tuple[int] = None
    conv_resample: bool = True
    dims: int = 1
    use_checkpoint: bool = False
    num_heads: int = 1
    resnet_two_cond: bool = False
    resnet_cond_channels: int = None
    attn_checkpoint: bool = False

    def make_model(self):
        return DiFFUNetModel(self)
```

```
class DiFFUNetModel(nn.Module):
    def __init__(self, conf: DiFFUNetConfig):
        super().__init__()
        self.conf = conf

        if conf.num_heads_upsample == -1:
            self.num_heads_upsample = conf.num_heads

        self.dtype = th.float32

        self.time_emb_channels = conf.time_embed_channels or conf.model_channels
        self.time_embed = nn.Sequential(
            linear(self.time_emb_channels, conf.embed_channels),
            nn.SiLU(),
            linear(conf.embed_channels, conf.embed_channels),
        )


        ch = input_ch = int(conf.channel_mult[0] * conf.model_channels)
        self.input_blocks = nn.ModuleList([
            TimestepEmbedSequential(
                conv_nd(conf.dims, conf.in_channels, ch, 3, padding=1))
        ])

        kwargs = dict(
            use_zero_module=conf.resnet_use_zero_module,
            cond_emb_channels=conf.resnet_cond_channels,
        )

        self._feature_size = ch
        input_block_chans = [[] for _ in range(len(conf.channel_mult))]
        input_block_chans[0].append(ch)
        self.input_num_blocks = [0 for _ in range(len(conf.channel_mult))]
        self.input_num_blocks[0] = 1
        self.output_num_blocks = [0 for _ in range(len(conf.channel_mult))]
        ds = 1
        self.input_blocks.append(AttentionBlock(
                            ch,
                            use_checkpoint=conf.use_checkpoint,
                            num_heads=conf.num_heads,
                            num_head_channels=conf.num_head_channels,
                        ))

        for level, mult in enumerate(conf.input_channel_mult
                                     or conf.channel_mult):
            for _ in range(conf.num_input_res_blocks or conf.num_res_blocks):
                layers = [
                    ResBlockConfig(
                        ch,
                        conf.embed_channels,
                        conf.dropout,
                        out_channels=int(mult * conf.model_channels),
                        dims=conf.dims,
                        use_checkpoint=conf.use_checkpoint,
                        **kwargs,
                    ).make_model()
                ]
                ch = int(mult * conf.model_channels)
                self.input_blocks.append(TimestepEmbedSequential(*layers))
                self._feature_size += ch
                input_block_chans[level].append(ch)
                self.input_num_blocks[level] += 1
```

```
        if level != len(conf.channel_mult) - 1:
            out_ch = ch
            self.input_blocks.append(
                TimestepEmbedSequential(
                    ResBlockConfig(
                        ch,
                        conf.embed_channels,
                        conf.dropout,
                        out_channels=out_ch,
                        dims=conf.dims,
                        use_checkpoint=conf.use_checkpoint,
                        down=True,
                        **kwargs,
                    ).make_model() if conf.
                    resblock_updown else Downsample(ch,
                                                    conf.conv_resample,
                                                    dims=conf.dims,
                                                    out_channels=out_ch)))
            ch = out_ch
            input_block_chans[level + 1].append(ch)
            self.input_num_blocks[level + 1] += 1
            ds *= 2
            self._feature_size += ch

    self.middle_block = TimestepEmbedSequential(
        ResBlockConfig(
            ch,
            conf.embed_channels,
            conf.dropout,
            dims=conf.dims,
            use_checkpoint=conf.use_checkpoint,
            **kwargs,
        ).make_model(),
        AttentionBlock(
            ch,
            use_checkpoint=conf.use_checkpoint or conf.attn_checkpoint,
            num_heads=conf.num_heads,
            num_head_channels=conf.num_head_channels,
        ),
        ResBlockConfig(
            ch,
            conf.embed_channels,
            conf.dropout,
            dims=conf.dims,
            use_checkpoint=conf.use_checkpoint,
            **kwargs,
        ).make_model(),
    )
    self._feature_size += ch

    self.output_blocks = nn.ModuleList([])
    for level, mult in list(enumerate(conf.channel_mult))[::-1]:
        for i in range(conf.num_res_blocks + 1):
            try:
                ich = input_block_chans[level].pop()
            except IndexError:
                ich = 0
            layers = [
                ResBlockConfig(
                    channels=ch + ich,
                    emb_channels=conf.embed_channels,
                    dropout=conf.dropout,
```

```
                        out_channels=int(conf.model_channels * mult),
                        dims=conf.dims,
                        use_checkpoint=conf.use_checkpoint,
                        has_lateral=True if ich > 0 else False,
                        lateral_channels=None,
                        **kwargs,
                    ).make_model()
                ]
                ch = int(conf.model_channels * mult)
                    out_ch = ch
                    layers.append(
                        ResBlockConfig(
                            ch,
                            conf.embed_channels,
                            conf.dropout,
                            out_channels=out_ch,
                            dims=conf.dims,
                            use_checkpoint=conf.use_checkpoint,
                            up=True,
                            **kwargs,
                        ).make_model() if (
                            conf.resblock_updown
                        ) else Upsample(ch,
                                        conf.conv_resample,
                                        dims=conf.dims,
                                        out_channels=out_ch))
                    ds //= 2
                self.output_blocks.append(TimestepEmbedSequential(*layers))
                self.output_num_blocks[level] += 1
                self._feature_size += ch
        self.out = nn.Sequential(
                normalization(ch),
                nn.SiLU(),
                zero_module(
                    conv_nd(conf.dims,
                            input_ch,
                            conf.out_channels,
                            3,
                            padding=1)),

    def forward(self, x, t, y=None, **kwargs):
        assert (y is not None) == (
            self.conf.num_classes is not None
        ),
        hs = [[] for _ in range(len(self.conf.channel_mult))]
        emb = self.time_embed(timestep_embedding(t, self.time_emb_channels))

        if self.conf.num_classes is not None:
            raise NotImplementedError()
        h = x.type(self.dtype)
        k = 0
        for i in range(len(self.input_num_blocks)):
            for j in range(self.input_num_blocks[i]):
                h = self.input_blocks[k](h, emb=emb)
                hs[i].append(h)
                k += 1
        assert k == len(self.input_blocks)

        h = self.middle_block(h, emb=emb)
        k = 0
        for i in range(len(self.output_num_blocks)):
            for j in range(self.output_num_blocks[i]):
```

```
                try:
                    lateral = hs[-i - 1].pop()
                except IndexError:
                    lateral = None
                h = self.output_blocks[k](h, emb=emb, lateral=lateral)
                k += 1

        h = h.type(x.dtype)
        pred = self.out(h)
        return Return(pred=pred)


class Return(NamedTuple):
    pred: th.Tensor

class DiffusionEncoderConfig(BaseConfig)
    in_channels: int
    model_channels: int
    out_hid_channels: int
    out_channels: int
    num_res_blocks: int
    dropout: float = 0
    channel_mult: Tuple[int] = (1, 2, 4, 8)
    use_time_condition: bool = True
    conv_resample: bool = True
    use_checkpoint: bool = False
    num_heads: int = 1
    num_head_channels: int = -1
    resblock_updown: bool = False

    def make_model(self):
        return DiFFEncoderModel(self)


class DiFFEncoderModel(nn.Module):
    def __init__(self, conf: DiFFEncoderConfig):
        super().__init__()
        self.conf = conf
        self.dtype = th.float32
        if conf.use_time_condition:
            time_embed_dim = conf.model_channels * 4
            self.time_embed = nn.Sequential(
                linear(conf.model_channels, time_embed_dim),
                nn.SiLU(),
                linear(time_embed_dim, time_embed_dim),
            )
        else:
            time_embed_dim = None

        ch = int(conf.channel_mult[0] * conf.model_channels)
        self.input_blocks = nn.ModuleList([
            TimestepEmbedSequential(
                conv_nd(1, conf.in_channels, ch, 3, padding=1))
        ])
        self._feature_size = ch
        input_block_chans = [ch]
        ds = 1
        self.input_blocks.append(AttentionBlock(
                            ch,
                            use_checkpoint=conf.use_checkpoint,
                            num_heads=conf.num_heads,
                            num_head_channels=conf.num_head_channels,
```

```
                    use_new_attention_order=conf.
                    use_new_attention_order,
                ))
for level, mult in enumerate(conf.channel_mult):
    for _ in range(conf.num_res_blocks):
        layers = [
            ResBlockConfig(
                ch,
                time_embed_dim,
                conf.dropout,
                out_channels=int(mult * conf.model_channels),
                dims=conf.dims,
                use_condition=conf.use_time_condition,
                use_checkpoint=conf.use_checkpoint,
            ).make_model()
        ]
        ch = int(mult * conf.model_channels)
        self.input_blocks.append(TimestepEmbedSequential(*layers))
        self._feature_size += ch
        input_block_chans.append(ch)
    if level != len(conf.channel_mult) - 1:
        out_ch = ch
        self.input_blocks.append(
            TimestepEmbedSequential(
                ResBlockConfig(
                    ch,
                    time_embed_dim,
                    conf.dropout,
                    out_channels=out_ch,
                    dims=conf.dims,
                    use_condition=conf.use_time_condition,
                    use_checkpoint=conf.use_checkpoint,
                    down=True,
                ).make_model() if (
                    conf.resblock_updown
                ) else Downsample(ch,
                                  conf.conv_resample,
                                  1,
                                  out_channels=out_ch)))
        ch = out_ch
        input_block_chans.append(ch)
        ds *= 2
        self._feature_size += ch

self.middle_block = TimestepEmbedSequential(
    ResBlockConfig(
        ch,
        time_embed_dim,
        conf.dropout,
        dims=conf.dims,
        use_condition=conf.use_time_condition,
        use_checkpoint=conf.use_checkpoint,
    ).make_model(),
    AttentionBlock(
        ch,
        use_checkpoint=conf.use_checkpoint,
        num_heads=conf.num_heads,
        num_head_channels=conf.num_head_channels,
        use_new_attention_order=conf.use_new_attention_order,
    ),
    ResBlockConfig(
        ch,
```

```
                time_embed_dim,
                conf.dropout,
                dims=conf.dims,
                use_condition=conf.use_time_condition,
                use_checkpoint=conf.use_checkpoint,
            ).make_model(),
        )
        self._feature_size += ch
          self.out = nn.Sequential(
                normalization(ch),
                nn.SiLU(),
                nn.AdaptiveAvgPool1d((1)),
                conv_nd(conf.dims, ch, conf.out_channels, 1),
                nn.Flatten(),
            )
        else:
            raise NotImplementedError(f"Unexpected {conf.pool} pooling")

    def forward(self, x, t=None, return_2d_feature=False):
        if self.conf.use_time_condition:
            emb = self.time_embed(timestep_embedding(t, self.model_channels))
        else:
            emb = None

        results = []
        h = x.type(self.dtype)
        for i, module in enumerate(self.input_blocks):
            if i == 1:
                h = module(h )
            else:
                h = module(h, emb=emb)
        h = self.middle_block(h, emb=emb)
        h = h.type(x.dtype)
        h = self.out(h)
        return h

    def forward_flatten(self, x):
        h = self.out(x)
        return h


class SuperResModel(DiFFUNetModel):
    def __init__(self, image_size, in_channels, *args, **kwargs):
        super().__init__(image_size, in_channels * 2, *args, **kwargs)

    def forward(self, x, timesteps, low_res=None, **kwargs):
        _, _, new_height, new_width = x.shape
        upsampled = F.interpolate(low_res, (new_height, new_width),
                                  mode="bilinear")
        x = th.cat([x, upsampled], dim=1)

        return super().forward(x, timesteps, **kwargs)
```

A.4 Code for Voting and Merging Process

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from TSpy.view import plot_mts, visualize_trace, visualize_tmerged_trace,
plot_mts_l
import matplotlib.pyplot as plt
```

```
import traceback
import os
import sys
sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))
from scipy.stats import mode
from collections import Counter


import pandas as pd
import numpy as np
import pca



data_path = "C:\\Users\\YANZCHE\\Desktop\\\CANoeBR223\\"
files = [f for f in os.listdir(data_path) if f.endswith(".npy")]
data_path1 = "C:\\Ipdu_stastics\\X2E_npy\\data.npy"
data1 = np.load(data_path1)


time_path_data = "C:\\Ipdu_stastics\\X2E_npy\\pdu_time.npy"
datatime = np.load(time_path_data)
data_arrays = [np.load(os.path.join(data_path, f)) for f in files]

new_data_list = []
m = len(data1[0])
for i in range(len(datatime) - 1):
    new_data_list.append(data1[i])
    if datatime[i+1] - datatime[i] > 120:
        zero_block = np.zeros((300, m))
        new_data_list.append(zero_block)

new_data_list.append(data1[-1])

data4 = np.vstack(new_data_list)


try:
    files = [f for f in os.listdir(data_path) if f.endswith(".npy")]
except PermissionError:
    print(f"Permission denied: '{data_path}'")
    raise

data_arrays = []
for f in files:
    file_path = os.path.join(data_path, f)
    try:
        data = np.load(file_path)
        data_arrays.append(data)
    except PermissionError:
        print(f"Permission denied: '{file_path}'")
        raise
indices_to_remove = [i for i in range(len(datatime) - 1) if datatime[i + 1] -
datatime[i] > 120]
min_length = len(datatime)


modified_data_arrays = []
for data in data_arrays:
    for idx in sorted(indices_to_remove, reverse=True):
        if idx < len(data) - 300:
            data = np.delete(data, np.s_[-300:])
```

```
    modified_data_arrays.append(data)
try:
    datatime = np.load(time_path_data)
except PermissionError:
    print(f"Permission denied: '{time_path_data}'")
    raise
summed_data = np.sum(modified_data_arrays, axis=0)
voted_data = []
modified_data_arrays = np.vstack(modified_data_arrays)
weights =[2, 3, 3, 4]
for i in range(modified_data_arrays.shape[1]):
    weighted_votes = Counter()
    for j in range(modified_data_arrays.shape[0]):
        value = modified_data_arrays[j, i]
        weighted_votes[value] += weights[j]
    voted_data.append(weighted_votes.most_common(1)[0][0])
voted_data = np.array(voted_data)


min_time_interval = 120
segments = []
start_idx = 0
summed_data = voted_data

for i in range(1, len(summed_data)):
    if summed_data[i] != summed_data[start_idx]:
        segments.append((start_idx, i - 1))
        start_idx = i

segments.append((start_idx, len(summed_data) - 1))

new_summed_data = np.copy(summed_data)
for i, (start, end) in enumerate(segments):
    segment_time = datatime[end] - datatime[start]
    if segment_time < min_time_interval and start > 0 and end < len(summed_data) -
1:
        mid_point = (start + end) // 2
        new_summed_data[start:mid_point + 1] = new_summed_data[start - 1]
        new_summed_data[mid_point + 1:end + 1] = new_summed_data[end + 1]
        if i + 1 < len(segments):
            next_start, next_end = segments[i + 1]
            new_next_start = max(start - 1, 0)
            segments[i + 1] = (new_next_start, next_end)


unique_values = np.unique(voted_data)
unique_values = np.unique(new_summed_data)

merged_segments = np.vstack(new_summed_data)
plt.style.use('classic')
np.save("0classified_combine.npy", merged_segments)
plot_mts_1(data4, merged_segments)
plt.savefig('0classified_combine.png')
visualize_tmerged_trace(data1, merged_segments, indices_to_remove, datatime)
```

A.5 Link Load Tables from Classes 6 to 10

Link load average under scenario 4

Start of link	End of Link	Baudrate	Load in Percent (%)	MAIN_1 in Percent (%)	MAIN_10 in Percent (%)	MAIN_100 in Percent (%)	MAIN_101 in Percent (%)	MAIN_110 in Percent (%)	MAIN_111 in Percent (%)	MAIN_120 in Percent (%)	MAIN_127 in Percent (%)
A	B	100000000	0.36472	0.00000	0.10038	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C	D	100000000	13.16790	0.00000	13.16718	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
E	B	100000000	1.35000	0.00000	0.02517	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
F	B	100000000	4.73807	0.00000	1.65919	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
G	B	100000000	4.23647	0.00000	0.58861	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
H	B	1000000000	0.48467	0.00000	0.17313	0.00000	0.00000	0.00000	0.00000	0.00000	0.00496
B	M	100000000	1.20901	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	E	100000000	0.01128	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
B	N	100000000	11.84211	0.00000	1.78889	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	I	100000000	4.52384	0.00000	0.07462	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	L	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	D	100000000	44.53864	0.06768	36.78400	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	A	100000000	0.54301	0.00000	0.22669	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
B	H	1000000000	4.90965	0.24100	1.94267	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	G	100000000	1.28382	0.00000	0.00288	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
B	J	100000000	24.49659	0.00000	24.49515	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	K	100000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	F	100000000	42.72621	0.03309	39.59838	0.00000	0.00000	0.00000	0.00000	0.00000	0.02557
B	O	100000000	0.02824	0.00000	0.00627	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
L	B	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
P	M	1000000000	0.12221	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
M	B	100000000	0.11538	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
M	R	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
M	P	1000000000	0.01745	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
M	Q	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
S	D	1000000000	1.60909	0.00692	0.83362	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
T	D	10000000000	0.15304	0.00000	0.15303	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	T	10000000000	0.00002	0.00002	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	B	100000000	8.35399	0.00376	7.42946	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
D	S	1000000000	0.83502	0.00023	0.74279	0.00000	0.00000	0.00000	0.00000	0.00000	0.00015
D	C	100000000	0.02624	0.00150	0.02398	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
U	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
V	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
W	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
X	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Y	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Z	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AB	AC	100000000	16.57232	0.00000	16.57160	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AC	J	100000000	0.13320	0.00150	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AC	AB	100000000	0.13320	0.00150	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AD	J	100000000	7.92747	0.00000	7.92675	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	AC	100000000	16.57232	0.00000	16.57160	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	B	100000000	0.13754	0.00150	0.13453	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
J	AD	100000000	0.00754	0.00000	0.00678	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	B	100000000	4.09728	0.00000	0.42685	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
N	U	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AP	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AG	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AF	1000000000	0.39852	0.00000	0.03942	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	AE	1000000000	0.29762	0.00000	0.00326	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
N	V	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AH	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	X	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	W	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AE	N	1000000000	0.44554	0.00000	0.14842	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AH	N	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AF	N	1000000000	0.75299	0.00000	0.03047	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	Y	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	B	100000000	0.02115	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00451
I	AM	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AI	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	AJ	1000000000	0.19584	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
I	AK	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	Z	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AL	1000000000	0.27232	0.00000	0.00746	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AJ	I	1000000000	0.19754	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AI	I	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AN	AL	1000000000	0.25788	0.00000	0.00746	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AL	AN	1000000000	0.00336	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

AL	AO	1000000000	0.00177	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AL	I	1000000000	0.00513	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AO	AL	1000000000	0.01444	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AP	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
O	B	100000000	1.88754	0.00000	0.00442	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AG	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Q	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
R	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AM	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AK	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
K	B	100000000	2.38070	2.38070	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Link load average under scenario 5

Start of link	End of Link	Baudrate	Load in Percent (%)	MAIN_1 in Percent (%)	MAIN_10 in Percent (%)	MAIN_100 in Percent (%)	MAIN_101 in Percent (%)	MAIN_110 in Percent (%)	MAIN_111 in Percent (%)	MAIN_120 in Percent (%)	MAIN_127 in Percent (%)
A	B	100000000	0.36472	0.00000	0.10038	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C	D	100000000	13.14686	0.00000	13.14614	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
E	B	100000000	1.31691	0.00000	0.02517	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
F	B	100000000	4.44434	0.00000	1.49756	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
G	B	100000000	4.04108	0.00000	0.52532	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
H	B	1000000000	0.47167	0.00000	0.16013	0.00000	0.00000	0.00000	0.00000	0.00000	0.00496
B	M	100000000	1.17592	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	E	100000000	0.01128	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
B	N	100000000	11.50249	0.00000	1.62800	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	I	100000000	4.45427	0.00000	0.07123	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	L	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	D	100000000	43.61941	0.04512	36.02844	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	A	100000000	0.54301	0.00000	0.22669	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
B	H	1000000000	4.66170	0.23875	1.77193	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	G	100000000	1.28382	0.00000	0.00288	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
B	J	100000000	23.41965	0.00000	23.41820	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	K	100000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	F	100000000	42.71195	0.01053	39.60669	0.00000	0.00000	0.00000	0.00000	0.00000	0.02557
B	O	100000000	0.02824	0.00000	0.00627	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
L	B	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
P	M	1000000000	0.11890	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
M	B	100000000	0.11538	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
M	R	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
M	P	1000000000	0.01745	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
M	Q	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
S	D	1000000000	1.58851	0.00466	0.82941	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
T	D	10000000000	0.14611	0.00000	0.14611	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	T	10000000000	0.00002	0.00002	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	B	100000000	7.69991	0.00226	6.77688	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
D	S	1000000000	0.76961	0.00008	0.67753	0.00000	0.00000	0.00000	0.00000	0.00000	0.00015
D	C	100000000	0.02624	0.00150	0.02398	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
U	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
V	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
W	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
X	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Y	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Z	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AB	AC	100000000	15.64052	0.00000	15.63980	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AC	J	100000000	0.13320	0.00150	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AC	AB	100000000	0.13320	0.00150	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AD	J	100000000	7.78232	0.00000	7.78160	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	AC	100000000	15.64052	0.00000	15.63980	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	B	100000000	0.13754	0.00150	0.13453	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
J	AD	100000000	0.00754	0.00000	0.00678	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	B	100000000	4.09728	0.00000	0.42685	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
N	U	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AP	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AG	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AF	1000000000	0.39852	0.00000	0.03942	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	AE	1000000000	0.29762	0.00000	0.00326	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
N	V	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075

N	AH	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	X	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	W	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AE	N	1000000000	0.42603	0.00000	0.13267	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AH	N	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AF	N	1000000000	0.73854	0.00000	0.03013	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	Y	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	B	100000000	0.02115	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00451
I	AM	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AI	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	AJ	1000000000	0.19584	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
I	AK	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	Z	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AL	1000000000	0.26867	0.00000	0.00712	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AJ	I	1000000000	0.19423	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AI	I	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AN	AL	1000000000	0.25423	0.00000	0.00712	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AL	AN	1000000000	0.00336	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AL	AO	1000000000	0.00177	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AL	I	1000000000	0.00513	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AO	AL	1000000000	0.01444	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AP	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
O	B	100000000	1.85445	0.00000	0.00442	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AG	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Q	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
R	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AM	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AK	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
K	B	100000000	2.35664	2.35664	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Link load average under scenario 6

Start of link	End of Link	Baudrate	Load in Percent (%)	MAIN_1 in Percent (%)	MAIN_10 in Percent (%)	MAIN_100 in Percent (%)	MAIN_101 in Percent (%)	MAIN_110 in Percent (%)	MAIN_111 in Percent (%)	MAIN_120 in Percent (%)	MAIN_127 in Percent (%)
A	B	100000000	0.36472	0.00000	0.10038	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C	D	100000000	12.59178	0.00000	12.59106	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
E	B	100000000	1.25224	0.00000	0.02517	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
F	B	100000000	4.30578	0.00000	1.44114	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
G	B	100000000	3.93554	0.00000	0.50193	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
H	B	1000000000	0.46699	0.00000	0.15546	0.00000	0.00000	0.00000	0.00000	0.00000	0.00496
B	M	100000000	1.11125	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	E	100000000	0.01128	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
B	N	100000000	11.29475	0.00000	1.57158	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	I	100000000	4.32493	0.00000	0.07123	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	L	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	D	100000000	42.02992	0.03910	34.53012	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	A	100000000	0.54301	0.00000	0.22669	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
B	H	1000000000	4.48932	0.23769	1.67301	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	G	100000000	1.28382	0.00000	0.00288	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
B	J	100000000	22.44470	0.00000	22.44326	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	K	100000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	F	100000000	41.27315	0.00902	38.16939	0.00000	0.00000	0.00000	0.00000	0.00000	0.02557
B	O	100000000	0.02824	0.00000	0.00627	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
L	B	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
P	M	1000000000	0.11244	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
M	B	100000000	0.11538	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
M	R	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
M	P	1000000000	0.01745	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
M	Q	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
S	D	1000000000	1.55432	0.00406	0.80433	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
T	D	10000000000	0.13919	0.00000	0.13918	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	T	10000000000	0.00002	0.00002	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	B	100000000	7.47012	0.00226	6.54709	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
D	S	1000000000	0.74663	0.00008	0.65455	0.00000	0.00000	0.00000	0.00000	0.00000	0.00015
D	C	100000000	0.02624	0.00150	0.02398	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
U	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
V	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

W	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
X	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Y	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Z	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AB	AC	100000000	14.99499	0.00000	14.99427	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AC	J	100000000	0.13320	0.00150	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AC	AB	100000000	0.13320	0.00150	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AD	J	100000000	7.45291	0.00000	7.45219	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	AC	100000000	14.99499	0.00000	14.99427	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	B	100000000	0.13754	0.00150	0.13453	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
J	AD	100000000	0.00754	0.00000	0.00678	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	B	100000000	4.09728	0.00000	0.42685	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
N	U	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AP	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AG	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AF	1000000000	0.39852	0.00000	0.03942	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	AE	1000000000	0.29762	0.00000	0.00326	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
N	V	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AH	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	X	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	W	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AE	N	1000000000	0.41377	0.00000	0.12703	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AH	N	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AF	N	1000000000	0.73003	0.00000	0.03013	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	Y	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	B	100000000	0.02115	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00451
I	AM	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AI	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	AJ	1000000000	0.19584	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
I	AK	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	Z	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AL	1000000000	0.26220	0.00000	0.00712	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AJ	I	1000000000	0.18776	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AI	I	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AN	AL	1000000000	0.24776	0.00000	0.00712	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AL	AN	1000000000	0.00336	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AL	AO	1000000000	0.00177	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AL	I	1000000000	0.00513	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AO	AL	1000000000	0.01444	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AP	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
O	B	100000000	1.78978	0.00000	0.00442	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AG	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Q	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
R	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AM	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AK	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
K	B	100000000	2.35062	2.35062	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Link load average under scenario 7

Start of link	End of Link	Baudrate	Load in Percent (%)	MAIN_1 in Percent (%)	MAIN_10 in Percent (%)	MAIN_100 in Percent (%)	MAIN_101 in Percent (%)	MAIN_110 in Percent (%)	MAIN_111 in Percent (%)	MAIN_120 in Percent (%)	MAIN_127 in Percent (%)
A	B	100000000	0.36472	0.00000	0.10038	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C	D	100000000	10.99106	0.00000	10.99034	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
E	B	100000000	1.26277	0.00000	0.02517	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
F	B	100000000	3.99029	0.00000	1.22287	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
G	B	100000000	3.75852	0.00000	0.42212	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
H	B	1000000000	0.44985	0.00000	0.13831	0.00000	0.00000	0.00000	0.00000	0.00000	0.00496
B	M	100000000	1.12178	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	E	100000000	0.01128	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
B	N	100000000	10.83552	0.00000	1.35244	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	I	100000000	4.33411	0.00000	0.05936	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	L	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	D	100000000	37.32508	0.03760	30.02628	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	A	100000000	0.54301	0.00000	0.22669	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
B	H	1000000000	4.32777	0.23739	1.56990	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	G	100000000	1.28382	0.00000	0.00288	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075

B	J	100000000	19.74397	0.00000	19.74252	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	K	100000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	F	100000000	35.28301	0.00902	32.17925	0.00000	0.00000	0.00000	0.00000	0.00000	0.02557
B	O	100000000	0.02824	0.00000	0.00627	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
L	B	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
P	M	1000000000	0.11349	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
M	B	100000000	0.11538	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
M	R	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
M	P	1000000000	0.01745	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
M	Q	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
S	D	1000000000	1.41167	0.00391	0.68179	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
T	D	10000000000	0.12241	0.00000	0.12241	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	T	10000000000	0.00002	0.00002	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	B	100000000	6.58618	0.00226	5.66316	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
D	S	1000000000	0.65824	0.00008	0.56616	0.00000	0.00000	0.00000	0.00000	0.00000	0.00015
D	C	100000000	0.02624	0.00150	0.02398	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
U	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
V	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
W	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
X	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Y	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Z	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AB	AC	100000000	13.09454	0.00000	13.09382	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AC	J	100000000	0.13320	0.00150	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AC	AB	100000000	0.13320	0.00150	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AD	J	100000000	6.65262	0.00000	6.65190	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	AC	100000000	13.09454	0.00000	13.09382	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	B	100000000	0.13754	0.00150	0.13453	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
J	AD	100000000	0.00754	0.00000	0.00678	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	B	100000000	4.09728	0.00000	0.42685	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
N	U	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AP	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AG	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AF	1000000000	0.39852	0.00000	0.03942	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	AE	1000000000	0.29762	0.00000	0.00326	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
N	V	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AH	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	X	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	W	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AE	N	1000000000	0.38899	0.00000	0.10630	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AH	N	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AF	N	1000000000	0.70889	0.00000	0.02895	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	Y	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	B	100000000	0.02115	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00451
I	AM	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AI	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	AJ	1000000000	0.19584	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
I	AK	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	Z	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AL	1000000000	0.26207	0.00000	0.00594	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AJ	I	1000000000	0.18882	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AI	I	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AN	AL	1000000000	0.24763	0.00000	0.00594	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AL	AN	1000000000	0.00336	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AL	AO	1000000000	0.00177	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AL	I	1000000000	0.00513	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AO	AL	1000000000	0.01444	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AP	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
O	B	100000000	1.80030	0.00000	0.00442	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AG	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Q	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
R	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AM	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AK	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
K	B	100000000	2.34912	2.34912	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Link load average under scenario 8

Start of link	End of Link	Baudrate	Load in Percent (%)	MAIN_1 in Percent (%)	MAIN_10 in Percent (%)	MAIN_100 in Percent (%)	MAIN_101 in Percent (%)	MAIN_110 in Percent (%)	MAIN_111 in Percent (%)	MAIN_120 in Percent (%)	MAIN_127 in Percent (%)
A	B	100000000	0.36472	0.00000	0.10038	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C	D	100000000	18.88584	0.00000	18.88512	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
E	B	100000000	1.68840	0.00000	0.02517	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
F	B	100000000	5.87666	0.00000	2.18274	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
G	B	100000000	5.04277	0.00000	0.77988	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
H	B	1000000000	0.52547	0.00000	0.21393	0.00000	0.00000	0.00000	0.00000	0.00000	0.00496
B	M	100000000	1.54741	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	E	100000000	0.01128	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
B	N	100000000	13.69641	0.00000	2.30624	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	I	100000000	5.22608	0.00000	0.10006	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	L	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	D	100000000	60.32185	0.06843	51.69574	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	A	100000000	0.54301	0.00000	0.22669	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
B	H	1000000000	5.75490	0.24311	2.27284	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	G	100000000	1.28382	0.00000	0.00288	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
B	J	100000000	33.13816	0.00000	33.13672	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	K	100000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	F	100000000	62.52483	0.01354	59.41656	0.00000	0.00000	0.00000	0.00000	0.00000	0.02557
B	O	100000000	0.02824	0.00000	0.00627	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
L	B	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
P	M	1000000000	0.15605	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
M	B	100000000	0.11538	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
M	R	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
M	P	1000000000	0.01745	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
M	Q	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
S	D	1000000000	2.10099	0.00699	1.23837	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
T	D	10000000000	0.20450	0.00000	0.20449	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	T	10000000000	0.00002	0.00002	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	B	100000000	10.45221	0.00301	9.52844	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
D	S	1000000000	1.04484	0.00015	0.95269	0.00000	0.00000	0.00000	0.00000	0.00000	0.00015
D	C	100000000	0.02624	0.00150	0.02398	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
U	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
V	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
W	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
X	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Y	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Z	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AB	AC	100000000	22.27637	0.00000	22.27564	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AC	J	100000000	0.13320	0.00150	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AC	AB	100000000	0.13320	0.00150	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AD	J	100000000	10.86499	0.00000	10.86427	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	AC	100000000	22.27637	0.00000	22.27564	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	B	100000000	0.13754	0.00150	0.13453	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
J	AD	100000000	0.00754	0.00000	0.00678	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	B	100000000	4.09728	0.00000	0.42685	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
N	U	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AP	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AG	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AF	1000000000	0.39852	0.00000	0.03942	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	AE	1000000000	0.29762	0.00000	0.00326	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
N	V	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AH	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	X	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	W	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AE	N	1000000000	0.54136	0.00000	0.19761	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AH	N	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AF	N	1000000000	0.84261	0.00000	0.03302	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	Y	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	B	100000000	0.02115	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00451
I	AM	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AI	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	AJ	1000000000	0.19584	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
I	AK	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	Z	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AL	1000000000	0.30870	0.00000	0.01001	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AJ	I	1000000000	0.23138	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AI	I	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AN	AL	1000000000	0.29426	0.00000	0.01001	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AL	AN	1000000000	0.00336	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

AL	AO	1000000000	0.00177	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AL	I	1000000000	0.00513	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AO	AL	1000000000	0.01444	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AP	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
O	B	100000000	2.22594	0.00000	0.00442	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AG	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Q	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
R	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AM	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AK	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
K	B	100000000	2.38070	2.38070	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Link load average under scenario 9

Start of link	End of Link	Baudrate	Load in Percent (%)	MAIN_1 in Percent (%)	MAIN_10 in Percent (%)	MAIN_100 in Percent (%)	MAIN_101 in Percent (%)	MAIN_110 in Percent (%)	MAIN_111 in Percent (%)	MAIN_120 in Percent (%)	MAIN_127 in Percent (%)
A	B	100000000	0.36472	0.00000	0.10038	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C	D	100000000	13.72307	0.00000	13.72235	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
E	B	100000000	1.31089	0.00000	0.02517	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
F	B	100000000	4.58178	0.00000	1.58277	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
G	B	100000000	4.12220	0.00000	0.55421	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
H	B	1000000000	0.47813	0.00000	0.16659	0.00000	0.00000	0.00000	0.00000	0.00000	0.00496
B	M	100000000	1.16990	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	E	100000000	0.01128	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
B	N	100000000	11.75878	0.00000	1.71174	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	I	100000000	4.44902	0.00000	0.07802	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	L	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	D	100000000	45.37577	0.04061	37.65288	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	A	100000000	0.54301	0.00000	0.22669	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
B	H	1000000000	4.70574	0.23800	1.77838	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	G	100000000	1.28382	0.00000	0.00288	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
B	J	100000000	24.39697	0.00000	24.39553	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	K	100000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	F	100000000	45.17630	0.00902	42.07254	0.00000	0.00000	0.00000	0.00000	0.00000	0.02557
B	O	100000000	0.02824	0.00000	0.00627	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
L	B	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
P	M	1000000000	0.11830	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
M	B	100000000	0.11538	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
M	R	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
M	P	1000000000	0.01745	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
M	Q	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
S	D	1000000000	1.65626	0.00421	0.88396	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
T	D	10000000000	0.15114	0.00000	0.15113	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	T	10000000000	0.00002	0.00002	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	B	100000000	8.04048	0.00226	7.11746	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
D	S	1000000000	0.80367	0.00008	0.71159	0.00000	0.00000	0.00000	0.00000	0.00000	0.00015
D	C	100000000	0.02624	0.00150	0.02398	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
U	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
V	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
W	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
X	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Y	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Z	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AB	AC	100000000	16.32948	0.00000	16.32876	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AC	J	100000000	0.13320	0.00150	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AC	AB	100000000	0.13320	0.00150	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AD	J	100000000	8.07069	0.00000	8.06997	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	AC	100000000	16.32948	0.00000	16.32876	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	B	100000000	0.13754	0.00150	0.13453	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
J	AD	100000000	0.00754	0.00000	0.00678	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	B	100000000	4.09728	0.00000	0.42685	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
N	U	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AP	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AG	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AF	1000000000	0.39852	0.00000	0.03942	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	AE	1000000000	0.29762	0.00000	0.00326	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
N	V	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075

N	AH	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	X	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	W	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AE	N	1000000000	0.43734	0.00000	0.14036	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AH	N	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AF	N	1000000000	0.75286	0.00000	0.03081	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	Y	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	B	100000000	0.02115	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00451
I	AM	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AI	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	AJ	1000000000	0.19584	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
I	AK	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	Z	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AL	1000000000	0.26874	0.00000	0.00780	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AJ	I	1000000000	0.19363	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AI	I	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AN	AL	1000000000	0.25430	0.00000	0.00780	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AL	AN	1000000000	0.00336	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AL	AO	1000000000	0.00177	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AL	I	1000000000	0.00513	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AO	AL	1000000000	0.01444	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AP	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
O	B	100000000	1.84843	0.00000	0.00442	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AG	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Q	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
R	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AM	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AK	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
K	B	100000000	2.35213	2.35213	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Link load average under scenario 10

Start of link	End of Link	Baudrate	Load in Percent (%)	MAIN_1 in Percent (%)	MAIN_10 in Percent (%)	MAIN_100 in Percent (%)	MAIN_101 in Percent (%)	MAIN_110 in Percent (%)	MAIN_111 in Percent (%)	MAIN_120 in Percent (%)	MAIN_127 in Percent (%)
A	B	100000000	0.36472	0.00000	0.10038	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C	D	100000000	2.05773	0.00000	2.05701	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
E	B	100000000	0.67771	0.00000	0.02517	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
F	B	100000000	1.71610	0.00000	0.10536	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
G	B	100000000	2.19817	0.00000	0.01757	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
H	B	1000000000	0.36300	0.00000	0.05146	0.00000	0.00000	0.00000	0.00000	0.00000	0.00496
B	M	100000000	0.53672	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	E	100000000	0.01128	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
B	N	100000000	7.24524	0.00000	0.24300	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	I	100000000	3.10464	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	L	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	D	100000000	11.18057	0.00000	5.56849	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	A	100000000	0.54301	0.00000	0.22669	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
B	H	1000000000	2.53233	0.23093	0.71769	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	G	100000000	1.28382	0.00000	0.00288	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
B	J	100000000	4.40601	0.00000	4.40457	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	K	100000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
B	F	100000000	4.92813	0.00000	1.83339	0.00000	0.00000	0.00000	0.00000	0.00000	0.02557
B	O	100000000	0.02824	0.00000	0.00627	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
L	B	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
P	M	1000000000	0.05498	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
M	B	100000000	0.11538	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
M	R	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
M	P	1000000000	0.01745	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
M	Q	100000000	0.00803	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
S	D	1000000000	0.62023	0.00000	0.05917	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
T	D	10000000000	0.02935	0.00000	0.02934	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	T	10000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
D	B	100000000	2.09682	0.00000	1.17605	0.00000	0.00000	0.00000	0.00000	0.00000	0.00226
D	S	1000000000	0.20945	0.00000	0.11745	0.00000	0.00000	0.00000	0.00000	0.00000	0.00015

D	C	100000000	0.01643	0.00000	0.01568	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
U	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
V	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
W	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
X	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Y	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Z	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AB	AC	100000000	2.62772	0.00000	2.62700	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AC	J	100000000	0.13170	0.00000	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AC	AB	100000000	0.13170	0.00000	0.13094	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AD	J	100000000	1.78149	0.00000	1.78077	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	AC	100000000	2.62772	0.00000	2.62700	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
J	B	100000000	0.13603	0.00000	0.13453	0.00000	0.00000	0.00000	0.00000	0.00000	0.00150
J	AD	100000000	0.00754	0.00000	0.00678	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	B	100000000	4.09728	0.00000	0.42685	0.00000	0.00000	0.00000	0.00000	0.00000	0.00526
N	U	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AP	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AG	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AF	1000000000	0.39852	0.00000	0.03942	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	AE	1000000000	0.29762	0.00000	0.00326	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
N	V	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	AH	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
N	X	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
N	W	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
AE	N	1000000000	0.20081	0.00000	0.00129	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AH	N	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AF	N	1000000000	0.53804	0.00000	0.02301	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	Y	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	B	100000000	0.02115	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00451
I	AM	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AI	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
I	AJ	1000000000	0.19584	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
I	AK	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	Z	100000000	0.00075	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00075
I	AL	1000000000	0.19762	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AJ	I	1000000000	0.13031	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AI	I	1000000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AN	AL	1000000000	0.18318	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AL	AN	1000000000	0.00336	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AL	AO	1000000000	0.00177	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AL	I	1000000000	0.00513	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008
AO	AL	1000000000	0.01444	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AP	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
O	B	100000000	1.21525	0.00000	0.00442	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AG	N	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Q	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
R	M	100000000	0.03105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AM	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
AK	I	100000000	0.45422	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
K	B	100000000	2.30926	2.30926	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

A.6 SQLite Data Processing code

```
import os
import sqlite3
import pandas as pd


folder_path =
"C:\Ipdu_stastics\Original_X2E\SomeIp_X2E_297_01096_20240403_134331_20240409_071902
"

output_folder_path = "C:\\pdu_stats_output"

if not os.path.exists(output_folder_path):
    os.makedirs(output_folder_path)
```

```
files = [f for f in os.listdir(folder_path) if f.endswith(".db")]


temp_db = os.path.join(output_folder_path, "temp.db")
if os.path.exists(temp_db):
    os.remove(temp_db)

conn_temp = sqlite3.connect(temp_db)
cur_temp = conn_temp.cursor()
cur_temp.execute('''
CREATE TABLE ipdu_statistics (
    start_time TEXT,
    ipdu_name TEXT,
    vlan INTEGER,
    frequency INTEGER,
    pdu_id INTEGER
)
''')
conn_temp.commit()


for file in files:
    db_file_path = os.path.join(folder_path, file)
    try:
        conn = sqlite3.connect(db_file_path)
        cur = conn.cursor()
        cur.execute('''DELETE FROM ipdu_statistics WHERE substr(datetime_interval,
1, 4) <> substr(datetime_interval, instr(datetime_interval, ' - ') + 3, 4)''')
        cur.execute("ATTACH DATABASE ? AS temp_db", (temp_db,))
        cur.execute('''
        INSERT INTO temp_db.ipdu_statistics (start_time, ipdu_name, vlan,
frequency, pdu_id)
        SELECT
            substr(datetime_interval, instr(datetime_interval, ' ') - 4, 4) || '-'
||
            substr(datetime_interval, 1, instr(datetime_interval, '/') - 1) || '-'
||
            substr(datetime_interval, instr(datetime_interval, '/') + 1,
instr(datetime_interval, ' ') - instr(datetime_interval, '/') - 6) ||
            '' || substr(datetime_interval, instr(datetime_interval, ' '),
instr(datetime_interval, ' - ') - instr(datetime_interval, ' ') - 3) AS start_time,
            ipdu_name, vlan, frequency, pdu_id
        FROM ipdu_statistics
        ''')
        cur.execute('''
            CREATE TABLE temp_db.ipdu_statistics_sorted AS
            SELECT * FROM temp_db.ipdu_statistics
            ORDER BY start_time
        ''')

        cur.execute('DROP TABLE temp_db.ipdu_statistics')
        cur.execute('ALTER TABLE temp_db.ipdu_statistics_sorted RENAME TO
ipdu_statistics')

        conn.commit()
        conn.close()
    except sqlite3.OperationalError as e:
        print(f"unable to open db {db_file_path}: {e}")

all_data_df = pd.read_sql_query("SELECT * FROM ipdu_statistics LIMIT 5", conn_temp)
```

```
print(all_data_df)

distinct_dates_query = '''
SELECT DISTINCT substr(start_time, 1, instr(start_time, ' ')-1)
     as date
FROM ipdu_statistics
ORDER BY date
'''
dates_df = pd.read_sql_query(distinct_dates_query, conn_temp)
distinct_dates = dates_df['date'].tolist()

for date in distinct_dates:
    date_db = os.path.join(output_folder_path, f"{date}.db")
    if os.path.exists(date_db):
        os.remove(date_db)
    date_conn = sqlite3.connect(date_db)
    date_cur = date_conn.cursor()
    date_cur.execute('''
    CREATE TABLE ipdu_statistics (
    start_time TEXT,
    ipdu_name TEXT,
    vlan INTEGER,
    frequency INTEGER,
    pdu_id INTEGER
    )''')
    cur_temp.execute("ATTACH DATABASE ? AS date_db", (date_db,))
    cur_temp.execute(f'''
                INSERT INTO date_db.ipdu_statistics (start_time, ipdu_name, vlan,
frequency, pdu_id)
                SELECT start_time, ipdu_name, vlan, frequency, pdu_id
                FROM ipdu_statistics
                WHERE substr(start_time, 1, instr(start_time, ' ')-1) = ?
            ''', (date,))
    date_conn.commit()
    conn_temp.commit()

    cur_temp.execute("DETACH DATABASE date_db")
    print(f"Database for date {date} created")
    all_data_df = pd.read_sql_query("SELECT * FROM ipdu_statistics LIMIT 5",
date_conn)
    print(all_data_df)
    date_conn.close()

conn_temp.close()

os.remove(temp_db)
```

A.7 Part of the Code for Causal CNN

```
from Trace2Scen.Trace2Scen import Trace2Scen
from Trace2Scen.diffuss import  Diffu
from Trace2Scen.adapers import CausalConv_LSE_Adaper, LSTM_LSE_Adaper
from Trace2Scen.clustering import DPGMM
from Trace2Scen.default_params import *
import pandas as pd
from sklearn.preprocessing import StandardScaler
from TSpy.view import plot_mts, visualize_trace
import matplotlib.pyplot as plt
import traceback
import os
```

```
import sys
sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))
from Trace2Scen.clustering import PoissonMM

win_size = 100
step = 5
import pandas as pd
import numpy as np
import pca
def str_to_datetime(time_str):
    return datetime.datetime.strptime(time_str, "%Y-%m-%d %I:%M:%S")

def convert_to_relative_time(initial_time, target_time):
    target_time_con = np.vectorize(str_to_datetime)(target_time)
    initial_time_con = np.vectorize(str_to_datetime)(initial_time)
    target_time_o = str_to_datetime(target_time)
    initial_time_o = str_to_datetime(initial_time)
    time_difference_con = target_time_con - initial_time_con
    seconds_difference_con = time_difference_con.total_seconds()
    target_time_only = target_time_o.time()
    initial_time_only = initial_time_o.time()
    target_time_converted = datetime.datetime.combine(datetime.date(1, 1, 1),
target_time_only)
    initial_time_converted = datetime.datetime.combine(datetime.date(1, 1, 1),
initial_time_only)
    time_difference = target_time_converted - initial_time_converted
    seconds_difference = time_difference.total_seconds()
    if seconds_difference < 0:
        adjusted_time_str = (str_to_datetime(target_time) +
datetime.timedelta(hours=12)).strftime("%Y-%m-%d %I:%M:%S")
        target_time_con = np.vectorize(str_to_datetime)(target_time)
        seconds_difference_con = (target_time_con -
initial_time_con).total_seconds()
    return seconds_difference_con

folder_path =
"C:\\Ipdu_stastics\\X2E_npy\\SomeIp_X2E_297_01096_20240528_110735_20240621_102829"
    files1 = [f for f in os.listdir(folder_path) if f.endswith(".npy")]
    folder_path2 =
"C:\\Ipdu_stastics\\X2E_npy\\SomeIp_X2E_297_01096_20240403_134331_20240409_071902"
    files2 = [f for f in os.listdir(folder_path2) if f.endswith(".npy")]
    all_X = []
    pdu_time = []
    for file in files1:
        if file.endswith("names.npy"):
            all_pdu_names = np.load(os.path.join(folder_path, file),
allow_pickle=True)
        elif file.endswith("0.npy"):
            X = np.load(os.path.join(folder_path, file), allow_pickle=True)
            data = np.vstack(X)
            df = pd.DataFrame(data, columns=['Timestamp', 'Signal', 'Quantity1',
'Quantity2', 'Quantity3'])
            df['Timestamp'] = pd.to_datetime(df['Timestamp'],
format='%Y-%m-%d %I:%M:%S %p')
            df['Signal_Quantity1'] = df['Signal'] + '_' + df['Quantity1']
            df_pivot = df.pivot_table(index='Timestamp',
columns='Signal_Quantity1', values=['Quantity2'], aggfunc='first')
            df_pivot = df_pivot.fillna(0)
            all_X.append(df_pivot)
    for file in files2:
        if file.endswith("names.npy"):
```

```
            all_pdu_names = np.load(os.path.join(folder_path, file),
allow_pickle=True)
        elif file.endswith("0.npy"):
            X = np.load(os.path.join(folder_path2, file), allow_pickle=True)
            data = np.vstack(X)
            df = pd.DataFrame(data, columns=['Timestamp', 'Signal', 'Quantity1',
'Quantity2', 'Quantity3'])
            df['Timestamp'] = pd.to_datetime(df['Timestamp'],
format='%Y-%m-%d %I:%M:%S %p')
            df['Signal_Quantity1'] = df['Signal'] + '_' + df['Quantity1']
            df_pivot = df.pivot_table(index='Timestamp',
columns='Signal_Quantity1', values=['Quantity2'], aggfunc='first')
            df_pivot = df_pivot.fillna(0)
            all_X.append(df_pivot)
    combined_df = pd.concat(all_X, axis=0,
join='outer').sort_values(by='Timestamp')
    multiindex_columns = combined_df.columns
    data_time= combined_df.index.tolist()
    combined_df = combined_df.reset_index(drop=True)
    combined_df = combined_df.fillna(0)
    data = combined_df.to_numpy().astype(int)
    first_time = pdu_time[0]
    datatime = [convert_to_relative_time(first_time, time_str) for time_str in
data_time]
new_data_list = []
new_data_list2 = []
m = len(data[0])
n = len(data[0])
for i in range(len(datatime) - 1):
    new_data_list.append(data[i])
    new_data_list2.append(data[i])
    if datatime[i+1] - datatime[i] > 120:
        zero_block = np.zeros((300, m))
        new_data_list.append(zero_block)
        zero_block = np.zeros((300, n))
        new_data_list2.append(zero_block)

new_data_list.append(data[-1])

data1 = np.vstack(new_data_list)

data1 = data1

new_data_list2.append(data[-1])

data3 = np.vstack(new_data_list2)
data2 = data1/np.max(data1)

params_LSE['in_channels'] = data2.shape[1]
params_LSE['out_channels'] = 50
params_LSE['nb_steps'] = 300
params_LSE['win_size'] = win_size

t2s = Trace2Scen(win_size, step, CausalConv_LSE_Adaper(params_LSE), DPGMM(n_states
= 50), params_LSE)
try:
    t2s.fit(data2, win_size, step)
except Exception as e:
    print("An exception occurred: ", str(e))
    traceback.print_exc()
```

```
plt.style.use('classic')
np.save("diffclassified_result1101.npy", t2s.state_seq)

import numpy as np
from TSpy.label import reorder_label
from TSpy.utils import calculate_scalar_velocity_list

class Trace2Scen:
    def __init__(self, win_size, step, encoder, clustering_component,
verbose=False):
        if win_size%2 != 0:
            raise ValueError()

        self.__win_size = win_size
        self.__step = step
        self.__offset = int(win_size/2)
        self.__encoder = encoder
        self.__clustering_component = clustering_component

    def fit(self, X, win_size, step):
        self.__length = X.shape[0]
        self.fit_encoder(X)
        self.__encode(X, win_size, step)
        latent_variables = self.__embeddings
        self.__cluster()
        self.__assign_label()
        return self, latent_variables

    def predict(self, X, win_size, step):
        self.__length = X.shape[0]
        self.__step = step
        self.__encode(X, win_size, step)
        self.__cluster()
        self.__assign_label()
        return self

    def set_step(self, step):
        self.__step = step

    def set_clustering_component(self, clustering_obj):
        self.__clustering_component = clustering_obj
        return self

    def fit_encoder(self, X):
        self.__encoder.fit(X)
        return self

    def predict_without_encode(self, X, win_size, step):
        self.__cluster()
        self.__assign_label()
        return self

    def __encode(self, X, win_size, step):
        self.__embeddings = self.__encoder.encode(X, win_size, step)

    def __cluster(self):
        self.__embedding_label =
reorder_label(self.__clustering_component.fit(self.__embeddings))

    def __assign_label(self):
        hight = len(set(self.__embedding_label)
        weight_vector = np.ones(shape=(2*self.__offset)).flatten()
```

```
        self.__state_seq = self.__embedding_label
        vote_matrix = np.zeros((self.__length,hight))
        i = 0
        for l in self.__embedding_label:
            vote_matrix[i:i+self.__win_size,l]+= weight_vector
            i+=self.__step
        self.__state_seq = np.array([np.argmax(row) for row in vote_matrix])

    def embeddings(self):
        return self.__embeddings

    def state_seq(self):
        return self.__state_seq

    def embedding_label(self):
        return self.__embedding_label

class CausalConv_LSE_Adaper(BasicEncoderClass):
    def _set_parmas(self, params):
        self.hyperparameters = params
        self.encoder = encoders.CausalConv_LSE(**self.hyperparameters)

    def fit(self, X):
        _, dim = X.shape
        X = np.transpose(np.array(X[:,:], dtype=float)).reshape(1, dim, -1)
        X = all_normalize(X)
        self.encoder.fit(X, save_memory=True, verbose=False)

    def encode(self, X, win_size, step):
        _, dim = X.shape
        X = np.transpose(np.array(X[:,:], dtype=float)).reshape(1, dim, -1)
        X = all_normalize(X)
        embeddings = self.encoder.encode_window(X, win_size=win_size, step=step)
        return embeddings

class CausalConv_LSE(BasicEncoder):
    def __init__(self, win_size, batch_size, nb_steps, lr,
                 channels, depth, reduced_size, out_channels, kernel_size,
                 in_channels, cuda, gpu, M, N, win_type):
        self.network = self.__create_network(in_channels, channels, depth,
reduced_size,
                                     out_channels, kernel_size, cuda, gpu)

        self.win_type = win_type
        self.architecture = ''
        self.cuda = cuda
        self.gpu = gpu
        self.batch_size = batch_size
        self.nb_steps = nb_steps
        self.lr = lr
        self.in_channels = in_channels
        self.out_channels = out_channels
        self.loss = losses.LSE_loss.LSELoss(
            win_size, M, N, win_type
        )
        self.optimizer = torch.optim.Adam(self.network.parameters(), lr=lr)
        self.loss_list = []

    def __create_network(self, in_channels, channels, depth, reduced_size,
                         out_channels, kernel_size, cuda, gpu):
        network = networks.causal_cnn.CausalCNNEncoder(
            in_channels, channels, depth, reduced_size, out_channels,
```

```
            kernel_size
        )
        network.double()
        if cuda:
            network.cuda(gpu)
        return network

    def fit(self, X, y=None, save_memory=False, verbose=False):
        train = torch.from_numpy(X)
        if self.cuda:
            train = train.cuda(self.gpu)

        train_torch_dataset = utils.Dataset(X)
        train_generator = torch.utils.data.DataLoader(
            train_torch_dataset, batch_size=self.batch_size, shuffle=True
        )

        i = 0
        epochs = 0
        while i < self.nb_steps:
            if verbose:
                print('Epoch: ', epochs + 1)
            for batch in train_generator:
                if self.cuda:
                    batch = batch.cuda(self.gpu)
                self.optimizer.zero_grad()
                loss = self.loss(batch, self.network, save_memory=save_memory)
                loss.backward()
                self.optimizer.step()
                i += 1
                if i >= self.nb_steps:
                    break
            epochs += 1
            if i == 100:
                pass

        return self.network

class QKVAttention(torch.nn.Module):
    def __init__(self, n_heads):
        super().__init__()
        self.n_heads = n_heads

    def forward(self, qkv):
        bs, width, length = qkv.shape
        assert width % (3 * self.n_heads) == 0
        ch = width // (3 * self.n_heads)
        q, k, v = qkv.chunk(3, dim=1)
        scale = 1 / math.sqrt(math.sqrt(ch))
        weight = torch.einsum(
            "bct,bcs->bts",
            (q * scale).view(bs * self.n_heads, ch, length),
            (k * scale).view(bs * self.n_heads, ch, length),
        )
        weight = torch.softmax(weight.float(), dim=-1).type(weight.dtype)
        a = torch.einsum("bts,bcs->bct", weight,
                         v.reshape(bs * self.n_heads, ch, length))
        return a.reshape(bs, -1, length)

class AttentionBlock(torch.nn.Module):
    def __init__(
        self,
```

```
            channels,
        ):
            super().__init__()
            self.num_heads = 1
            self.norm = torch.nn.GroupNorm(min(32, channels), channels)
            self.qkv = torch.nn.Conv1d(
                channels, channels * 3, 1)
            self.attention = QKVAttention(self.num_heads)

        def forward(self, x):
            b, c, *spatial = x.shape
            x = x.reshape(b, c, -1)
            qkv = self.qkv(self.norm(x))
            h = self.attention(qkv)
            h = self.proj_out(h)
            return (x + h).reshape(b, c, *spatial)


    class CausalConvolutionBlock(torch.nn.Module):
        def __init__(self, in_channels, out_channels, kernel_size, dilation,
                     final=False):
            super(CausalConvolutionBlock, self).__init__()
            padding = (kernel_size - 1) * dilation
            conv1 = torch.nn.utils.weight_norm(torch.nn.Conv1d(
                in_channels, out_channels, kernel_size,
                padding=padding, dilation=dilation
            ))
            chomp1 = Chomp1d(padding)
            relu1 = torch.nn.LeakyReLU()
            conv2 = torch.nn.utils.weight_norm(torch.nn.Conv1d(
                out_channels, out_channels, kernel_size,
                padding=padding, dilation=dilation
            ))
            chomp2 = Chomp1d(padding)
            relu2 = torch.nn.LeakyReLU()
            self.causal = torch.nn.Sequential(
                conv1, chomp1, relu1, conv2, chomp2, relu2
            )

            self.upordownsample = torch.nn.Conv1d(
                in_channels, out_channels, 1
            ) if in_channels != out_channels else None
            self.relu = torch.nn.LeakyReLU() if final else None

        def forward(self, x):
            out_causal = self.causal(x)
            res = x if self.upordownsample is None else self.upordownsample(x)
            if self.relu is None:
                return out_causal + res
            else:
                return self.relu(out_causal + res)


    class CausalCNN(torch.nn.Module):
        def __init__(self, in_channels, channels, depth, out_channels,
                     kernel_size):
            super(CausalCNN, self).__init__()

            layers = []
            dilation_size = 1
            layers += AttentionBlock(
                                in_channels,
```

```
                    )
        for i in range(depth):
            in_channels_block = in_channels if i == 0 else channels
            layers += [CausalConvolutionBlock(
                in_channels_block, in_channels, kernel_size, dilation_size
            )]
            dilation_size *= 2
        layers += [CausalConvolutionBlock(
            channels, out_channels, kernel_size, dilation_size
        )]

        self.network = torch.nn.Sequential(*layers)

    def forward(self, x):
        return self.network(x)
```

A.8 Unfinished Code for Negative Binomial Mixture

```
import numpy as np
from scipy import linalg
from sklearn.utils import check_array
from sklearn.utils._param_validation import StrOptions
from sklearn.utils.extmath import row_norms
from sklearn.mixture._base import BaseMixture
from scipy.special import betaln, digamma, gammaln

def _check_shape(array, shape, name):
    array = np.atleast_1d(array)
    if array.shape != shape:
        raise ValueError(
            "The parameter '{}' has the wrong shape. Expected {}, but got
{}".format(
                name, shape, array.shape
            )
        )
    return array

def _log_dirichlet_norm(dirichlet_concentration):
    return gammaln(np.sum(dirichlet_concentration)) -
np.sum(gammaln(dirichlet_concentration))

def _estimate_negbinom_parameters(X, resp):
    nk = resp.sum(axis=0) + 10 * np.finfo(resp.dtype).eps
    means = np.dot(resp.T, X) / nk[:, np.newaxis]
    variances = np.dot(resp.T, X**2) / nk[:, np.newaxis] - means**2
    return nk, means, variances

def _estimate_log_negbinom_prob(X, means, variances):
    n_samples, n_features = X.shape
    n_components, _ = means.shape

    log_prob = np.empty((n_samples, n_components))
    for k in range(n_components):
        mean_k = means[k]
        var_k = variances[k]

        r = (mean_k**2) / np.abs(var_k - mean_k)
        p = mean_k / np.abs(var_k)

        # Ensure r and p are valid
```

```
        r = np.maximum(r, 1e-6)
        p = np.clip(p, 1e-6, 1 - 1e-6)

        # Compute the log probability
        with np.errstate(divide='ignore', invalid='ignore'):
            log_prob[:, k] = (
                gammaln(X + r) - gammaln(r) - gammaln(X + 1)
                + r * np.log(1 - p) + X * np.log(p)
            ).sum(axis=1)

        # Replace any NaN values with a large negative number
        log_prob[:, k] = np.nan_to_num(log_prob[:, k], nan=-1e10)

    return log_prob



def _estimate_log_negbinom_prob(X, means, variances):
    r = (means**2) / np.abs(variances - means)
    p = means / np.abs(variances)
    X = np.clip(X, 1e-10, None)
    means = np.clip(means, 1e-10, None)
    variances = np.clip(variances, 1e-10, None)

    log_prob = (
        gammaln(X + r) - gammaln(r) - gammaln(X + 1)
        + r * np.log(1 - p) + X * np.log(p)
    )
    return log_prob

def _estimate_negbinom_parameters(X, resp):
    nk = resp.sum(axis=0) + 10 * np.finfo(resp.dtype).eps
    means = np.dot(resp.T, X) / nk[:, np.newaxis]
    variances = np.dot(resp.T, X**2) / nk[:, np.newaxis] - means**2
    return nk, means, variances


class BayesianNegativeBinomialMixture(BaseMixture):
    def __init__(
        self,
        *,
        n_components=1,
        tol=1e-3,
        max_iter=100,
        n_init=1,
        init_params="kmeans",
        weight_concentration_prior_type="dirichlet_process",
        weight_concentration_prior=None,
        mean_prior=None,
        variance_prior=None,
        random_state=None,
        warm_start=False,
        verbose=0,
        verbose_interval=10,
    ):
        super().__init__(
            n_components=n_components,
            tol=tol,
            reg_covar=1e-6,
            max_iter=max_iter,
            n_init=n_init,
            init_params=init_params,
```

```
            random_state=random_state,
            warm_start=warm_start,
            verbose=verbose,
            verbose_interval=verbose_interval,
        )
        self.weight_concentration_prior_type = weight_concentration_prior_type
        self.weight_concentration_prior = weight_concentration_prior
        self.mean_prior = mean_prior
        self.variance_prior = variance_prior

    def _check_parameters(self, X):
        self._check_weights_parameters()
        self._check_means_variances_parameters(X)

    def _check_weights_parameters(self):
        if self.weight_concentration_prior is None:
            self.weight_concentration_prior_ = 1.0 / self.n_components
        else:
            self.weight_concentration_prior_ = self.weight_concentration_prior

    def _check_means_variances_parameters(self, X)
        _, n_features = X.shape
        if self.mean_prior is None:
            self.mean_prior_ = X.mean(axis=0)
        else:
            self.mean_prior_ = check_array(
                self.mean_prior, dtype=[np.float64, np.float32], ensure_2d=False
            )
            _check_shape(self.mean_prior_, (n_features,), "mean_prior")

        if self.variance_prior is None:
            self.variance_prior_ = X.var(axis=0)
        else:
            self.variance_prior_ = check_array(
                self.variance_prior, dtype=[np.float64, np.float32],
ensure_2d=False
            )
            _check_shape(self.variance_prior_, (n_features,), "variance_prior")

    def _initialize(self, X, resp):
        nk, xk, = _estimate_negbinom_parameters(X, resp)
        self._estimate_weights(nk)
        self._estimate_means_variances(nk, xk)

    def _estimate_weights(self, nk):
        if self.weight_concentration_prior_type == "dirichlet_process":
            self.weight_concentration_ = (
                1.0 + nk,
                (
                    self.weight_concentration_prior_
                    + np.hstack((np.cumsum(nk[::-1])[-2::-1], 0))
                ),
            )
        else:
            self.weight_concentration_ = self.weight_concentration_prior_ + nk

    def _estimate_means_variances(self, nk, xk):
        self.means_ = (self.mean_prior_ + nk[:, np.newaxis] * xk) / (1 + nk[:,
np.newaxis])
        self.variances_ = (self.variance_prior_ + nk[:, np.newaxis] * (xk**2)) / (1
+ nk[:, np.newaxis])
```

```
    def _m_step(self, X, log_resp):
        nk, xk = _estimate_negbinom_parameters(X, np.exp(log_resp))
        self._estimate_weights(nk)
        self._estimate_means_variances(nk, xk)

    def _estimate_log_weights(self):
        if self.weight_concentration_prior_type == "dirichlet_process":
            digamma_sum = digamma(
                self.weight_concentration_[0] + self.weight_concentration_[1]
            )
            digamma_a = digamma(self.weight_concentration_[0])
            digamma_b = digamma(self.weight_concentration_[1])
            return (
                digamma_a
                - digamma_sum
                + np.hstack((0, np.cumsum(digamma_b - digamma_sum)[:-1]))
            )
        else:
            return digamma(self.weight_concentration_) - digamma(
                np.sum(self.weight_concentration_)
            )

    def _estimate_log_prob(self, X):
        return _estimate_log_negbinom_prob(X, self.means_, self.variances_)

    def _compute_lower_bound(self, log_resp, log_prob_norm):
        """Estimate the lower bound of the model."""
        log_norm_weight = _log_dirichlet_norm(self.weight_concentration_)
        return (
            -np.sum(np.exp(log_resp) * log_resp)
            - log_norm_weight
        )

    def _get_parameters(self):
        return (
            self.weight_concentration_,
            self.means_,
            self.variances_,
        )

    def _set_parameters(self, params):
        (
            self.weight_concentration_,
            self.means_,
            self.variances_,
        ) = params

        if self.weight_concentration_prior_type == "dirichlet_process":
            weight_dirichlet_sum = (
                self.weight_concentration_[0] + self.weight_concentration_[1]
            )
            tmp = self.weight_concentration_[1] / weight_dirichlet_sum
            self.weights_ = (
                self.weight_concentration_[0]
                / weight_dirichlet_sum
                * np.hstack((1, np.cumprod(tmp[:-1])))
            )
            self.weights_ /= np.sum(self.weights_)
        else:
            self.weights_ = self.weight_concentration_ /
np.sum(self.weight_concentration_)
```

```
    def fit(self, X, y=None):
        X = check_array(X, dtype=[np.float64, np.float32], ensure_min_samples=2)
        self._check_parameters(X)
        random_state = self.random_state

        best_lower_bound = -np.inf
        for init in range(self.n_init):
            self._initialize_parameters(X, random_state)
            lower_bound = -np.inf
            for n_iter in range(self.max_iter):
                prev_lower_bound = lower_bound

                log_prob_norm, log_resp = self._e_step(X)
                self._m_step(X, log_resp)

                lower_bound = self._compute_lower_bound(log_resp, log_prob_norm)
                change = lower_bound - prev_lower_bound

                if abs(change) < self.tol:
                    break

            if lower_bound > best_lower_bound:
                best_params = self._get_parameters()
                best_lower_bound = lower_bound

        self._set_parameters(best_params)
        self.converged_ = True if lower_bound > best_lower_bound else False


        return self

import numpy as np
import pymc3 as pm
import theano.tensor as tt


new_data = np.random.randn(10, 2)

with pm.Model() as new_model:
    alpha = pm.Gamma('alpha', 1., 1.)
    beta = pm.Beta('beta', 1., alpha, shape=initial_centers.shape[0])
    phi = pm.Deterministic('phi', beta * tt.concatenate([[1], tt.cumprod(1 -
beta)[:-1]]))
    mu = pm.Normal('mu', mu=initial_centers, sd=1, shape=initial_centers.shape)
    sd = pm.HalfNormal('sd', sd=1, shape=initial_centers.shape)

    new_obs = pm.NormalMixture('new_obs', w=phi, mu=mu, sd=sd, observed=new_data)

    ppc_new = pm.sample_posterior_predictive(trace, var_names=['new_obs'],
samples=500, model=new_model)

new_predictions = ppc_new['new_obs']

final_predictions = new_predictions.mean(axis=0).argmax(axis=1)

print(final_predictions)
```