University of Sussex

A University of Sussex PhD thesis

This thesis is protected by copyright which belongs to the author

It cannot be reproduced or quoted extensively from without first obtaining permission in writing from the author

The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given

Please visit Figshare for more information and further details

https://sussex.figshare.com/Theses

UNIVERSITY OF SUSSEX

In-Vehicle Network Monitoring with Network Tomography

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

School of Engineering and Informatics

Author:
Amani Ibraheem

Supervisor:
Dr. Zhengguo Sheng

July, 2023

Declaration

I hereby declare that this thesis has not been and will not be, submitted in whole or in part to another University for the award of any other degree.

Amani Ibraheem

Acknowledgements

First and foremost, I would like to express my deepest appreciation to my supervisor Dr. Zhengguo Sheng for his dedicated support and guidance. Your feedback allowed me to deepen and refine my research, and the results presented in my thesis would be impossible without your invaluable supervision. I have been extremely lucky to have a supervisor who cared so much about my work, and who responded to my questions and queries so promptly. I would also like to extend my deepest gratitude to my second supervisor, Prof. George Parisis for his insightful comments and kind support. Our meetings and conversations were vital in inspiring me to think outside the box. I am also grateful for the opportunity given to me during my secondment at Nanyang Technological University, Singapore. I cannot forget the generosity and kind hospitality of Prof. Guan Yong Liang and the COSMO lab team. This gratitude extends to SEEDS, an EU-funded project which enabled such an opportunity and opened doors for great collaborations.

A special thanks go to my small and big family. Especially my husband, Abdulwahab. No words can express my gratitude to you. Your unwavering support and sacrifices have overwhelmed me, and I cannot thank you enough. To my lovely sons, Mohammad and Mohannad, you fill my heart with joy and laughter, you are the light in my life. Thanks for being by my side all these years!

To my amazing parents, thanks for your patience, love and support. My dad, Mohammad, you always inspire me to be the best version of myself, I aspire to be like you one day. My mum, Aishah, I have always been astonished by your strength and kind soul. It is because of you that I am able to weather any storm. To my sister and brothers, I cannot thank you enough for your continuous support and kind words. Maher, thanks for the time you spent with me, you have been a great loving brother and companion. My aunt, Asma, I am very grateful for your continuous caring and support.

Finally, I would like to thank all my friends in Brighton, your friendship is a treasure I will always cherish.

In loving memory of my brother "Mohannad", I miss you so much,

you are always in my heart . . .

To the memory of my loving grandparents, "Ali" and "Sharifah", I will always remember our great times together . . .

"*Anyone who stops learning is old, whether at twenty or eighty. Anyone who keeps learning stays young. The greatest thing in life is to keep your mind young.*"

Henry Ford

Abstract

With the advances in Connected and Autonomous Vehicles (CAVs), the in-vehicle network becomes more complex and harder to manage. In addition, as the vehicle becomes part of Internet-of-Vehicles (IoVs), it is now more exposed to the outside world than ever. Current in-vehicle networks are vulnerable to cyber-security threats, including IP-based attacks. Monitoring the in-vehicle network is thus one of the crucial tasks that deserves careful consideration. However, the closed-in system of the in-vehicle network entails difficulty in accessing the internal elements of the network. This hinders the monitoring process from obtaining insights about the internal performance of the network. As a result, monitoring every part of the network becomes intractable. Our focus in this thesis is to investigate monitoring solutions that do not require contribution from the internal components of the network. To this end, our first contribution is that we propose, for the first time in literature, to use network tomography as a monitoring approach for in-vehicle networks. An attractive feature of network tomography is that it exploits the end-to-end measurements to infer the performance of an internal network without the need to access any internal component. This feature is well-suited for closed-in systems such as in-vehicle networks. One challenge in applying network tomography in an in-vehicle network, however, is that the in-vehicle network must be fully identifiable. The second contribution of this thesis is to address this challenge by leveraging the advances in deep learning and proposing to complement network tomography with deep learning-based tomography. Furthermore, to facilitate the monitoring and management process, the third contribution is to propose a new in-vehicle network topology that is fully identifiable, redundant and enabled with Software-Defined Networking (SDN) functionalities. Such topology is aligned with the next-generation Electronic and Electrical (E/E) architecture that is shifting towards centralisation. The proposed monitoring approach can be applied to such topology to infer the overall performance of the network. In addition, with the SDN paradigm and redundancy support, the network can intelligently cope with failures by locating the failed component and re-routing the traffic to the redundant alternative. Hence, achieving self-healing in-vehicle network without external intervention.

Contents

List of Tables

List of Figures

List of Abbreviations

ABS	Anti-lock Braking System
ACK	Acknowledgement
ADAS	Advanced Driver Assistance System
AE	Automotive Ethernet
AFDX	Avionics Full Duplex Switched Ethernet
ASC	Automatic Stability Control
ATS	Asynchronous Traffic Shaper
AVB	Audio Video Bridging
AVTP	Audio-Video Transport Protocol
BE	Best-Effort
BGD	Batch Gradient Descent
BNT	Binary Network Tomography
C-V2X	Cellular Vehicle-to-Everything
CAN	Controller Area Network
CAV	Connected and Autonomous Vehicle
CBS	Credit Based Shaper
CDCU	Cross-Domain Control Unit
CDT	Control Data Traffic
CNC	Central Network Controller
CNN	Convolutional Neural Network
COIDS	Clock Offset based Intrusion Detection System
CPS	Cyber-Physical System

CQF	Cyclic Queuing and Forwarding
CRC	Cyclic Redundancy Check
CSMA/CD	Carrier-Sense Multiple Access with Collision Detection
CSMA/CD+AMP	Carrier-Sense Multiple Access with Collision Detection and Arbitration on Message Priority
CUC	Centralised User Configuration
DCU	Domain Control Unit
DetNet	Deterministic Networking
DNN	Deep Neural Network
DNT	Delay Network Tomography
DoIP	Diagnostics over Internet Protocol
DoS	Denial-of-Service
E/E	Electrical/Electronic
ECU	Electronic Control Unit
EM	Expectation Maximization
EMC	Electromagnetic Compatibility
EOF	End of Frame
ESP	Electronic Stability Program
EV	Electric Vehicle
FCS	Frame Check Sequence
FRL	Fully Restructurable Link
FQTSS	Forwarding and Queuing Enhancements for Time-Sensitive Streams
GAN	Generative Adversarial Network
gPTP	generalized Precision Time Protocol
HMI	Human Machine Interface
HPCP	High-Performance Computing Platform
IDS	Intrusion Detection System

IFS	Interframe Space
IoT	Internet-of-Thing
IoV	Internet-of-Vehicle
IP	Internet Protocol
IPV	Internal Priority Value
IRT	Identifiability then Redundancy Transformation
LAN	Local Area Network
LIN	Local Interconnect Network
LSTM	Long Short-Term Memory
LVDS	Low-Voltage Differential Signalling
MAC	Media Access Control
MAPE	Mean Absolute Percentage Error
MBGD	Mini-Batch Gradient Descent
MitM	Man-in-the-Middle
MOST	Media Oriented Systems Transport
MSE	Mean-Squared Error
NBI	Northbound Interface
NNDE	Neural Network Delay Estimation
NNDT	Neural Network Delay Tomography
OBU	On-board Unit
OEM	Original Equipment Manufacturer
OTIDS	Offset ratio and Time interval based Intrusion Detection System
PDF	Probability Density Function
POF	Plastic Optical Fiber
PTP	Precision Time Protocol
PRL	Partially Restructurable Link

PS	Peristaltic Shaper
QoS	Quality-of-Service
RBD	Ring Break Diagnosis
RC	Rate-Constraint
ReLU	Rectified Linear Unit
RIT	Redundancy then Identifiability Transformation
ROC	Receiver Operating Characteristic
RQA	Recurrence Quantification Analysis
RSU	Road Side Unit
RTR	Remote Transmission Request
SAE	Society of Automotive Engineers
SBI	Southbound Interface
SDN	Software-Defined Network
SD-WAN	Software-Defined Wide Area Network
SDV	Software-Defined Vehicle
SGD	Stochastic Gradient Descent
SOF	Start of Frame
SR	Stream Reservation
SRP	Stream Reservation Protocol
SVM	Support Vector Machine
TAS	Traffic Aware Shaper
TDMA	Time Division Multiple Access
TSN	Time-Sensitive Networking
TT	Time-Triggered
TTEthernet	Time-Triggered Ethernet
UTP	Unshielded Twisted Pair
V2I	Vehicle-to-Infrastructure

V2P	Vehicle-to-Pedestrian
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-Everything
VANET	Vehicular Ad hoc Network
VLAN	Virtual Local Area Network
ZCU	Zone Control Unit

List of Notations

Network and Graph Theory

$G = (V, E)$	In-vehicle network as a graph G containing pair of two sets: vertices $V(G)$ and edges $E(G)$
G_i	Identifiable network topology
G_r	Redundant network topology
G_{ir}	Identifiable and redundant network topology
$G + \{v_i v_j\}$	Network G when link between two vertices $v_i, v_j \in V(G)$ is added
$G - \{v_i v_j\}$	Network G when link between two vertices $v_i, v_j \in V(G)$ is deleted
$V(G)$	Set of vertices (nodes) in network G
$E(G)$	Set of edges (links) in network G
$\mathcal{E}(G) \subset V(G)$	Set of edge nodes in G
$\mathcal{E}_m(G) \subseteq \mathcal{E}(G)$	Set of monitoring edge nodes in G
$\mathcal{R}(G) \subset V(G)$	Set of intermediate nodes in G
$\mathcal{R}_{3^+} \subseteq \mathcal{R}(G)$	Set of internal nodes having node degree larger than three
$\mathcal{R}_{3^-} \subseteq \mathcal{R}(G)$	Set of internal nodes having node degree less than three
$\mathcal{P}(G)$	Set of all possible paths in network G
$\mathcal{P}_m(G) \subseteq \mathcal{P}(G)$	Set of measured paths in network G
$\mathcal{S} \subset G$	Partial network of G
$\mathcal{D}(G)$	Set of domains in an in-vehicle network G
$\mathcal{C}(G)$	Set of graph components in G

$\mathcal{B}(G)$	Set of bridges in G
$\mathcal{I}(G) \subset E(G)$	Set of internal links in G
$\mathcal{T}(G) \subseteq E(G)$	Set of external links in G
$\mathcal{W}(G)$	Set of internal nodes in G having node degree larger than three and are neighbours to more than one internal node
$\mathcal{N}(v_i)$	Set of neighbours directly connected to node $v_i \in V(G)$
$\mathcal{N}_{\mathcal{R}}(v_i) \subseteq \mathcal{N}(v_i)$	Set of internal neighbours in $\mathcal{R}$ directly connected to node $v_i \in V(G)$
$B(G)$	Set of CAN buses in G
$p_i \in \mathcal{P}(G)$	Single path between two edge nodes in G
$p_{v_i,v_j} \in \mathcal{P}(G)$	Path connecting two non-adjacent nodes $v_i, v_j \in \mathcal{E}(G)$
$e_i = v_i v_j$	Link $e_i \in E(G)$ connecting two adjacent nodes $v_i, v_j \in V(G)$
$d(v_i)$	Degree of node $v_i \in V(G)$
$d(v_i)_g$	Degree of node $v_i \in V(g)$
$p_i(v_i, v_j) \in \mathcal{P}(G)$	i^{th} path connecting edge nodes $v_i, v_j \in \mathcal{E}(G)$
s_{v_i,v_j}	Segment connecting two non-adjacent nodes $v_i, v_j \in V(G)$ with $v_i \vee v_j \in \mathcal{R}(G)$
$b_i \in B(G)$	Single CAN bus
$d(v_i, v_j)$	Distance between nodes $v_i, v_j \in V(G)$
C_n	A cycle with n nodes where $n \geq 3$
$v_h(e_i)$	Endpoint (head) of link $e_i \in E(G)$
$v_t(e_i)$	Endpoint (tail) of link $e_i \in E(G)$
$g_{\mathcal{B}} \subset g$	Subgraph of g that only contains bridges of g, $\mathcal{B}(g)$
$c_i \in \mathbb{Z}^+$	Priority level of CAN node $v_i \in \mathcal{E}(G)$

<u>Numbers and Cardinalities</u>

$\eta_G := \Big\lvert V(G) \Big\rvert$	Total number of nodes in network G

$\lambda_G := \Big\lvert \mathcal{R}(G) \Big\rvert$	Total number of intermediate nodes in network G
$\gamma_G := \Big\lvert E(G) \Big\rvert$	Total number of links in network G
$\kappa_G := \Big\lvert \mathcal{P}_m(G) \Big\rvert$	Total number of measured paths for network G
$\sigma := \Big\lvert \mathcal{R}_{3-} \Big\rvert$	Total number of internal nodes having node degree less than three
$\zeta_{v_i} := \Big\lvert \mathcal{N}_{\mathcal{R}}(v_i) \Big\rvert$	Total number of internal nodes that are neighbours to $v_i \in \mathcal{R}_{3+}$
$\psi := \Big\lvert R_{3+} \Big\rvert$	Number of internal nodes with node degree larger than three
$\omega := \Big\lvert \mathcal{W} \Big\rvert$	Total number of internal nodes having node degree larger than three and are neighbours to more than one internal node
ϑ	Total number of partial networks
l_G	Number of uniquely identifiable links in G
$\varphi_{v_i} := d(v_i) - 3$	Number of remaining links incident to node $v_i \in V(G)$ after disconnecting it from three links
N_{v_i,v_j}	Total number of internally disjoint paths between $v_i, v_j \in \mathcal{E}(G)$
χ_η	Number of added nodes in the transformed topology
χ_γ	Number of added links in the transformed topology

Network Tomography

$\boldsymbol{A}$	Measurement matrix
$\boldsymbol{y}$	Vector of end-to-end measurements
$\boldsymbol{x}$	Vector of link-level measurements
y_{v_i,v_j}	End-to-end (path-level) measurement between node v_i and v_j
$y_i \in \boldsymbol{y}$	End-to-end (path-level) measurement of path $p_i \in \mathcal{P}_m(G)$
$x_i \in \boldsymbol{x}$	Link-level metric of link $e_i \in E(G)$

$\bar{y_i}$	Actual end-to-end performance of path $p_i \in \mathcal{P}_m(G)$
$\bar{x_i}$	Actual link-level performance of link $e_i \in E(G)$
$r(\boldsymbol{A})$	Rank of measurement matrix $\boldsymbol{A}$
$\mathcal{H}_{p_i} := \left\lvert \bar{y_i} - y_i \right\rvert$	Difference between measured and actual performance of i^{th} path $p_i \in \mathcal{P}_m(G)$
$\mathcal{H}_{e_i} := \left\lvert \bar{x_i} - x_i \right\rvert$	Difference between measured and actual performance of i^{th} link $e_i \in E(G)$
δ_{p_i}	Predefined threshold value for i^{th} path $p_i \in \mathcal{P}_m(G)$
δ_{e_i}	Predefined threshold value for i^{th} link $e_i \in E(G)$
$S(G) \in \{0, 1\}$	Status of network G, 0 is normal and 1 is anomalous
$S_d(G) \in \{\textbf{false}, \textbf{true}\}$	Identifiability status of network G, false means *unidentifiable* and true means *identifiable*
$S_r(G) \in \{\textbf{false}, \textbf{true}\}$	Redundancy status of network G, false means *non-redundant* and true means *redundant*
$S(p_i) \in \{0, 1\}$	Status of i^{th} path $p_i \in \mathcal{P}(G)$
$S(e_j) \in \{0, 1\}$	Status of j^{th} link $e_j \in E(G)$
$a_{ij} \in \{0, 1\}$	The element of the measurement matrix $\boldsymbol{A}$ at the i^{th} row (for $p_i \in \mathcal{P}_m(G)$) and j^{th} column (for $e_j \in E(G)$)
a_{e_j}	Minimum link-level delay of j^{th} link $e_j \in E(G)$
b_{e_j}	Maximum link-level delay of j^{th} link $e_j \in E(G)$
a_{p_i}	Minimum path-level delay of i^{th} path $p_i \in \mathcal{P}(G)$
b_{p_i}	Maximum path-level delay of i^{th} path $p_i \in \mathcal{P}(G)$
t_{trans}	Transmission time
t_{recv}	Reception time

Deep Learning

$\boldsymbol{\theta}$	Input vector to the DNN
$\boldsymbol{W}$	Matrix of weights used in DNN
$\boldsymbol{\beta}$	Bias vector used in DNN

$\boldsymbol{h}_i$	Output of the i^{th} hidden layer of a DNN
$f(z)$	Activation function at the hidden layers of DNN
$f_o(z)$	Activation function at the output layer of DNN
L	Loss function of estimated and actual value
α	Learning rate

Units

m	Meter
s	Second
ms	Millisecond
μs	Microsecond
bps	Bits per second
Kbps	Kilo bits per second
Mbps	Mega bits per second

Publications

1. A. Ibraheem, "Cross Network Slicing in Vehicular Networks". In Intelligent Technologies for Internet of Vehicles. Cham: Springer International Publishing, pp. 151-189, 2021.

2. A. Ibraheem, Z. Sheng, G. Parisis, and D. Tian, "Neural Network based Partial Tomography for In-Vehicle Network Monitoring", In proceeding of IEEE International Conference on Communications Workshops (ICC Workshops) pp. 1-6, 2021, Canada.

3. A. Ibraheem, Z. Sheng, G. Parisis, and D. Tian, "In-Vehicle Network Delay Tomography", In proceeding of IEEE Global Communications Conference (GLOBECOM), pp. 5528-5533, 2022, Brazil.

4. A. Ibraheem, Z. Sheng, G. Parisis, J. Zhou, and D. Tian, "Internal Network Monitoring with DNN and Network Tomography for In-Vehicle Networks", In proceeding of IEEE International Conference on Unmanned Systems (ICUS), pp. 928-933, 2022, China.

5. A. Ibraheem, Z. Sheng, G. Parisis, and D. Tian, "Network Tomography-based Anomaly Detection and Localisation in Centralised In-Vehicle Network", In proceeding of IEEE International Conference on Omni Layer Intelligent Systems (COINS) 2023, Germany.

6. A. Ibraheem, Z. Sheng, and G. Parisis, "New Identifiable and Redundant SDN-based Measurement for In-Vehicle Networks", IEEE Transactions on Vehicular Technology (submitted).

1 Introduction

"Until I began to learn to draw, I was never much interested in looking at art."

Richard P. Feynman

1.1 Overview

Nowadays, with the advances of communications and Internet-of-Things (IoTs), the vehicle becomes part of this connected system by forming another branch of IoTs, i.e., Internet-of-Vehicles (IoVs) [1]. In this system, the vehicle can communicate with different objects from the outside world. Despite the benefits of connecting the vehicle either to other vehicles or to the infrastructure, this comes at the cost of exposing the vehicle to cyber-security threats.

Generally, vehicle connectivity can be broadly categorised into inter-vehicle [2] and intra-vehicle [3] communications. Inter-vehicle communications connect the vehicle with other vehicles on the road while intra-vehicle communications connect the components inside the vehicle itself, this is generally what is referred to as an in-vehicle network. Ultimately, security threats target the intra-vehicle communication, i.e., in-vehicle network, to cause failure in some, or all, of the in-vehicle network. For example, adversaries can gain access to the vehicle either remotely [4] (e.g., through the radio interface) or physically [5] (e.g., by physically connecting to the On-board Unit (OBU)) with the intention of tampering with one or more of the vehicle's systems/components. Such components are parts of a larger communication system that forms the in-vehicle network.

The conventional design of in-vehicle networks does not consider the security aspect of the vehicle [6]. This is because traditional vehicles were not as open to the outside world as today's vehicles. Hence, securing the vehicle's network was not as necessary as it is today. For over thirty years now, Controller Area Network (CAN) has been the dominant communication protocol used by the automotive industry for in-vehicle networks [7]. For today's in-vehicle networks, CAN (with a maximum bandwidth of only 1 Mbps [8], [9]) cannot cope with the increasing bandwidth demand imposed by Advanced Driver Assistance System (ADAS) and infotainment systems [10]. In addition, it suffers a lack of authentication and authorisation mechanisms [11]. Automotive Ethernet (AE) has been recently used to compensate for the limited bandwidth in CAN.

To keep up with the rapidly advancing pace of vehicular communications, the research community is working hard to provide solutions to secure vehicle connectivity. This starts with monitoring solutions that provide an overall insight into the vehicle's operation. Unlike traditional vehicles, next-generation vehicles will consist of heterogeneous communication protocols such as CAN, Local Interconnect Network (LIN), Media Oriented Systems Transport (MOST), FlexRay, and AE. These different protocols need to coexist and therefore require coordination and monitoring of all the involved components to ensure appropriate functioning of the network. Additionally, for critical systems such as in-vehicle networks, it is imperative to ensure service continuity. For this, a fail-operational mechanism should be introduced to the in-vehicle system, so that, in case of any failure, in one or more of the system components, the operation is not disturbed, especially for the critical systems of the vehicle. Note that fail-operational behaviour is different from fail-safe behaviour. A fail-safe system ensures a safe shutdown of the system in case of any failure, while a fail-operational system ensures that safety-critical operations are functioning properly during any failure [12].

A monitoring solution based on *network tomography* [13], [14] is one of the appropriate solutions that this thesis proposes for in-vehicle networks. One of the attractive features of network tomography, that makes it the best fit for in-vehicle networks, is that there is no need to access all elements of the network. By only monitoring a subset of the network, the performance of the other subset can then be inferred using network tomography. For in-vehicle networks with limited capabilities, such a monitoring approach is efficient as it does not require heavy monitoring of all elements in the network. In addition, due to the proprietary nature of the in-vehicle ecosystem, it is difficult to access the internal elements in the in-vehicle networks. Therefore, network tomography allows monitoring of the overall network without worrying about accessing or modifying the internal components of the in-vehicle network.

1.2 Challenges and Motivation

Vehicles are one of the critical Cyber-Physical Systems (CPSs) [15] that need to be monitored to detect issues related to both performance failures and cyber-security threats [16]. However, monitoring the internal network is not always possible. This is because the internal elements of the in-vehicle network are difficult to access due to proprietary closed-in devices provided by Original Equipment Manufacturers (OEMs) [17], [18]. In addition, monitoring every part of the network can overburden it and may perturb the existing traffic where such disturbance can result in serious consequences, especially for safety- and latency-critical applications. For instance, as the monitoring traffic has to co-exist with the normal traffic in the in-vehicle network, this additional traffic can affect the performance of the existing operational traffic; the resource consumption and delivery time can increase due to this additional traffic and its prioritisation level.

For these reasons, alternative monitoring solutions that do not require contribution from internal elements should be investigated. Network tomography [13], [14] provides an efficient solution to monitor in-vehicle networks due to its advantages of requiring limited probes that can only be sent between end nodes, thus eliminating the need to involve internal elements of the network during the monitoring process, in addition to reducing the monitoring overhead. In particular, with network tomography, only the end-to-end network performance can be measured while the remaining (internal network performance) can be inferred from such measurements (provided by the accessible end devices).

Even though network tomography is a powerful monitoring mechanism, there are certain constraints that need to be addressed when applying this approach to in-vehicle networks. For instance, the constraint related to the in-vehicle network scenario that internal network devices cannot be accessed leads to a limited number of nodes that can act as monitors, where such nodes can only exist at the edge. Therefore, the application of network tomography needs to be investigated under the condition that only a limited number of (edge) nodes are accessible. Other constraints are related to the in-vehicle network topology, and the availability of adequate and independent end-to-end measurements [19], [20]. Such constraints affect the full identifiability of the network[1], which means that if, for instance, the measurements are

[1]Network identifiability is a concept in network tomography for which the performance of *all* links can be *uniquely* determined only if the network is fully identifiable (see Chapter 3 for more details).

not enough or the topology is not identifiable, then network tomography cannot provide fine-grained insights about the overall network performance. Hence, it is important to carefully consider these constraints when applying network tomography to in-vehicle networks so that the benefits of network tomography can be thoroughly exploited.

Another challenge in in-vehicle networks is the lack of fail-operational behaviour in in-vehicle networks [21]. The main reason for this is that current in-vehicle networks do not utilise any redundancy measures. For this, a new topology needs to be developed to support fail-operational behaviour with redundant networks [22], [23]. Software-Defined Network (SDN) [24] is a suitable paradigm that can further facilitate the process of achieving a fail-operational network [10]. It can significantly help in the management process of different components of the in-vehicle network, including the redundant part that can be activated once a failure is detected in the network.

1.3 Research Questions and Objectives

In the following subsections, we list the research questions and objectives this thesis is aiming to address.

1.3.1 Research Questions

This thesis aims to answer the following questions:

1. Can network tomography be applied in in-vehicle network scenarios?
2. If network tomography is not applicable to in-vehicle network scenarios, then what are the constraints that hinder this application?
3. How to overcome the constraints imposed by network tomography measurement matrix[2]?
4. How to design an overall monitoring system that can detect anomalies, including failures and security threats, and minimise their effect on the functional behaviour of the network?

1.3.2 Research Objectives

The main objectives of seeking answers to the research questions mentioned above are as follows:

[2]A measurement matrix in network tomography is a matrix with rows representing the end-to-end measurement paths and columns corresponding to the links that are traversed by such paths.

- The first objective is to investigate network tomography capabilities in monitoring the application of in-vehicle networks by studying the specific characteristics of in-vehicle networks.
- The second objective is to implement and test network tomography as a monitoring mechanism for different in-vehicle network architectures.
- The third objective is to understand if there is any limitation that may hinder the application of network tomography into the in-vehicle network domain, and look for possible solutions to tackle these issues.
- The fourth objective is to devise a solution to achieve a fail-operational in-vehicle network so that its critical functional operation is not affected by failures. For this SDN can help in maintaining and managing the network without external intervention.
- Finally, the ultimate goal is to come up with a complete and robust monitoring solution that benefits from the proposed approaches, based on network tomography, to efficiently and accurately detect any anomalies, in addition to mitigating their effect on the overall network performance.

In this thesis, we investigate monitoring solutions for modern in-vehicle networks, with a focus on applying network tomography as a new monitoring approach that can provide information about the overall network performance. In addition, we explore solutions to achieve fail-operational behaviour using network tomography and SDN. In particular, this thesis's focus is to study the application of network tomography in in-vehicle network scenarios, including different topologies based on Electrical/Electronic (E/E) architectures. In addition, this thesis considers the integration of SDN as a means to control the overall network and facilitate the entire monitoring process.

1.4 Contributions

This thesis presents a novel monitoring approach for in-vehicle networks leveraging network tomography-based mechanisms. Our main contributions presented in this thesis are listed below:

- Chapter 3: The first contribution of this thesis is to investigate the application of network tomography in in-vehicle networks. Different topologies based on different E/E architectures are considered in this chapter. In addition, ground theoretical analysis of identifiability and monitor conditions are studied, where topological and monitoring conditions required to fully utilise network tomography solutions have been derived. This chapter has further investigated the usability of network tomography in an in-vehicle network scenario, where the performance of the in-vehicle network under the existence of tomographic-based monitoring traffic is evaluated and compared the results with one of the state-of-the-art monitoring solutions.

- Chapter 4: Based on the findings of our first study described in the first contribution, we conclude that not all in-vehicle network topologies are identifiable, this chapter thus proposes new solutions to tackle this problem. In particular, a partial network tomography algorithm has been proposed. In addition, deep learning-based solutions have been considered to help improve the measurement matrix used in network tomography. A deep neural network is used for this purpose, where the available measurements are used to estimate the performances of other unmeasured parts of the network. The estimated performances are then added to the available measurements to form a full-rank measurement matrix that can be used to fully utilise network tomography to infer the performance of the remaining, internal, network.

- Chapter 5: Applying the proposed monitoring approaches to detect and locate anomalies within the vehicle network is examined in this chapter. This chapter presented three types of monitoring approaches: Binary Network Tomography (BNT), Delay Network Tomography (DNT), and Deep Neural Network (DNN)-based tomography. The first two approaches are based on algebraic tomography, whereas the last is based on deep learning. Evaluation results show that all approaches can detect anomalies resulting from Denial-of-Service (DoS) attack, with BNT outperforming other approaches in accurately locating the anomaly with no false positive or false negative.

- Chapter 6: The promising results of BNT presented in Chapter 5 enable the proposal of a novel in-vehicle network topology in this chapter. Because BNT is an algebraic approach, it requires having a full-rank measurement matrix, such matrix can be guaranteed if the topology is identifiable. Therefore, this chapter proposed a topology that has three main properties. First, the proposed topology is identifiable so that the measurement matrix is full-rank. Second, it is redundant. This property enables the fail-operational behaviour where the redundant paths can be used in case the original paths are compromised. The third property is that the topology is SDN-enabled. Such property can substantially help in network management, including the mitigation of anomalous behaviour to ensure a fail-operational network. Moreover, a number of transformation algorithms are devised in this chapter. The goal of such algorithms is to help in the transition process towards a topology that satisfies both identifiability and redundancy, without the need to build the network from scratch. Evaluation of the proposed algorithms shows that the transformed topologies achieved a minimum number of added weights.

1.5 Roadmap

The remainder of this thesis is structured as follows. The following chapter (i.e., Chapter 2) discusses the technical background of vehicular communications, with a detailed description of in-vehicle networks and the advances in E/E architectures. The monitoring challenges of in-vehicle networks are discussed in this chapter as well. In addition, this chapter reviews the existing monitoring solutions for in-vehicle networks. Moreover, network tomography and its different features and variants are introduced in this chapter.

The applicability of network tomography in in-vehicle networks is studied in detail in Chapter 3. Different in-vehicle networking topologies have been investigated in this chapter. Theoretical and technical analyses pertaining to network identifiability for in-vehicle networks are presented, where they establish the foundation of the subsequent chapters.

Chapter 4 presents partial network tomography and deep learning-based solutions to tackle the issue of having a measurement matrix that is deficient in rank. In addition, it evaluates the proposed approaches in inferring the path-level as well as link-level performance.

Chapter 5 studies how network tomography can be used as an anomaly detection and localisation approach for in-vehicle networks. Three proposed approaches (BNT, DNT, and DNN-based tomography) are described and evaluated in this chapter.

Chapter 6 proposes a new in-vehicle network topology that is identifiable, redundant and SDN-enabled. Furthermore, different transformation algorithms are derived based on extensive theoretical analysis. Additionally, a complete monitoring framework for the in-vehicle network is proposed in this chapter.

Lastly, Chapter 7 concludes this thesis and gives a summary of the presented research work. In addition, it highlights some of the interesting research directions drawn from the findings of the research carried on in this thesis.

2 Background

"The greatest enemy of knowledge is not ignorance, it is the illusion of knowledge."

Stephen Hawking

2.1 Vehicular Communications

Broadly speaking, vehicular communications are categorised into two ways of communication. First is external communications where the vehicle can connect to other external objects outside the vehicle such as other vehicles, Road Side Units (RSUs), etc. The second is internal communications that happen inside the vehicle itself. The focus of this thesis is on internal communications that form the overall in-vehicle network. Figure 2.1 shows an overview of vehicular communications. Details regarding external and internal communications are discussed next.

2.1.1 External Communication

In external communications, vehicles are part of a large, external communication system where the vehicle can be connected to anything, Vehicle-to-Everything (V2X). This can further be classified into three types of communications: Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), and Vehicle-to-Pedestrian (V2P) communications. In V2V communications, the vehicle can connect to other vehicles on the road, forming what is known as Vehicular Ad hoc Network (VANET) [25]. On the other hand, V2I communications allow the vehicle to communicate with the surrounding infrastructure such as RSUs (V2R), Networks (V2N) and Grids (V2G). In V2P, the vehicle can connect to a pedestrian smartphone to exchange safety messages [26]. In V2V, the connection between vehicles is based on wireless

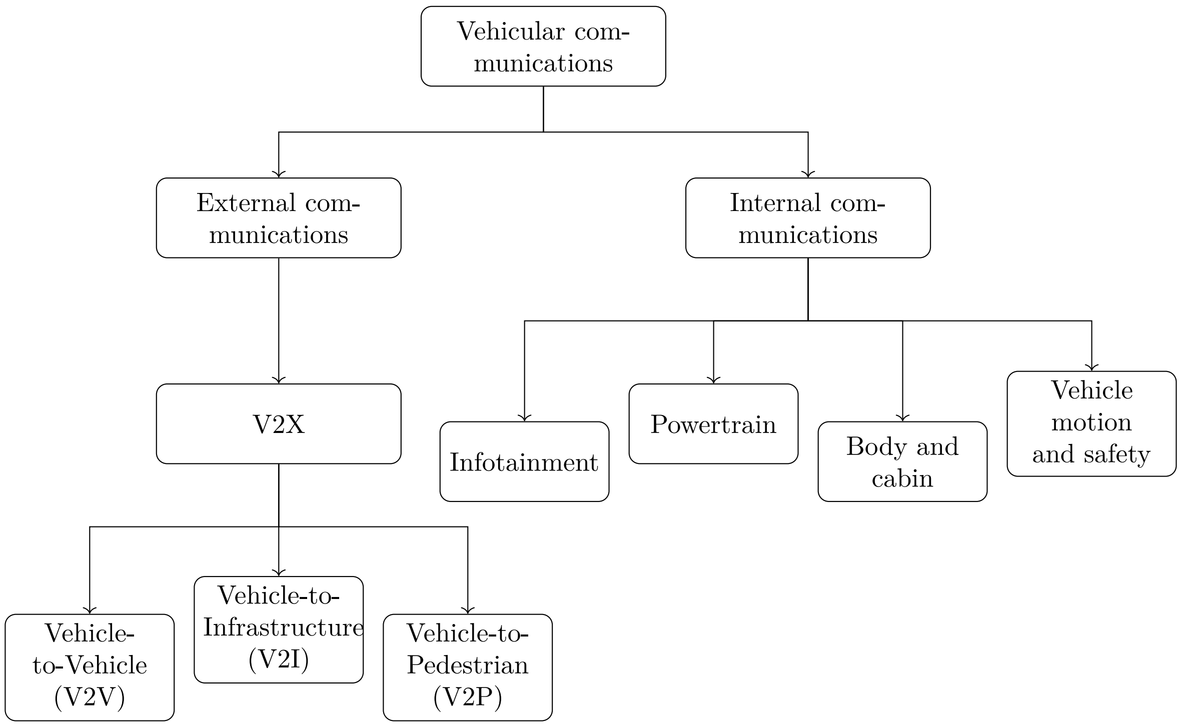


Figure 2.1: Overview of vehicular communications

communications such as Dedicated Short-Range Communication (DSRC). A Cellular Vehicle-to-Everything (C-V2X) is another recent 3GPP protocol that can be used for V2X.

2.1.2 Internal Communications

Internal communications encompass all the communications that take place inside the vehicle to connect different Electronic Control Units (ECUs). As seen in Figure 2.1, this type includes the communications in four main domains of the vehicle: *infotainment*, *powertrain*, *body and cabin*, and *vehicle motion and safety* [27], [28]. In the following, we briefly describe each of these domains. In addition, Table 2.1 shows a summary of network requirements for each domain in the in-vehicle network [29].

2.1.2.1 Vehicle's Domains

Infotainment domain is also referred to as *telematics domain.* This domain includes several functions related to multimedia and Human Machine Interface (HMI) such as rear seat entertainment, navigation system and remote diagnostics [28]. Connecting the vehicle to the outside world also happens in this domain. Infotainment systems require high bandwidth, low latency and jitter.

Control of the engine and power transmission are the main two functions of the **powertrain domain**. It includes several automobile components such as the engine, transmission, shaft, and wheel [29]. The Powertrain domain requires frequent communication with other domains [28]. Network requirements for powertrain systems are more stringent, especially for latency and jitter.

Body and cabin domain (can be referred to by simply *body domain*) is composed of comfort-controlling components such as doors opening/closing, windows rolling, seat adjustment, mirrors adjustment, dashboards control, lights setting, etc. [28], [29].

On the contrary to other domains, network requirements in the body domain are more relaxed. **Vehicle motion and safety** corresponds to the chassis components (e.g., Antilock Braking System (ABS), Electronic Stability Program (ESP), Automatic Stability Control (ASC)) and other components related to safety such as ADAS. As in the powertrain domain, network requirements in the vehicle motion and safety domain are strict.

Table 2.1: Functional domains and their network requirements in the in-vehicle network.

Requirement	Infotainment	Powertrain	Body and cabin	Vehicle motion and safety
Bandwidth	High	Low	Low	Low
Latency	Low	Low	Low	Low
Reliability	Medium	High	Medium	High

There are several communication protocols used in in-vehicle networks. The well-known ones are: CAN, LIN, FlexRay, MOST, and AE. Table 2.2 shows a summary of all communication protocols used in the in-vehicle network [23], [30]. In the following, we describe each of these protocols with a more focus on CAN and AE.

Table 2.2: Summary of communication protocols used in the in-vehicle network.

	CAN	LIN	FlexRay	MOST	AE
Bandwidth	1Mbps	0.02Mbps	20Mbps	150Mbps	100Mbps per link
Number of nodes	30	16	22	64	Limited by number of switch ports
Network length	40m	40m	24m	1280m	15m per link
Messaging	Cyclic-frames	Cyclic-frames	Cyclic-frames	Cyclic-frames/ streams	Frames
Payload	8Bytes	8Bytes	256Bytes	384Bytes	1500Bytes
MAC	Non-destructive arbitration	Polling	TDMA	Time-triggered	Full-duplex, contention-less
Cost	Low	Very low	High	High	High
Topology	Bus	Bus	Bus, star, hybrid	Ring, star	Star, tree
Security threat	High	Low	Medium	Medium to high	High
Availability	Many	Many	Few	One	Multiple vendors and growing
Cabling	UTP	1-wire	UTP	Optical, UTP	UTP
Error detection	Strong	Weak	Strong	Strong	Strong
Error correction	Re-transmit	None	None	Relies on higher layer protocols	Relies on higher layer protocols

Table 2.2: Summary of communication protocols used in the in-vehicle network.

	CAN	LIN	FlexRay	MOST	AE
Applications	General bus	Switches, doors, seats	Safety-critical x-by-wire	Infotainment	Infotainment, diagnostics, backbone and safety
Reference to OSI model	Layer 1 and 2	Layer 1 and 2	Layer 1 and 2	All layers	All layers

2.1.2.2 Controller Area Network (CAN)

Due to its robustness and low cost, Controller Area Network (CAN) has been the most widely used protocol for automotive networks for over a decade [8]. CAN follows a bus-based topology as shown in Figure 2.2 and it is designed using Unshielded Twisted Pair (UTP) cabling. According to ISO 11898 standard, the maximum allowed length of CAN is $40m$ with a maximum number of 30 nodes that can transmit and receive messages as cyclic frames. Referring to the OSI model, CAN only uses three layers: physical layer, data link layer, and application layer. The maximum bandwidth of CAN bus is 1 Mbps with up to 8 bytes of maximum payload. Although Bosch was the first to design CAN, CAN is available today from many vendors.

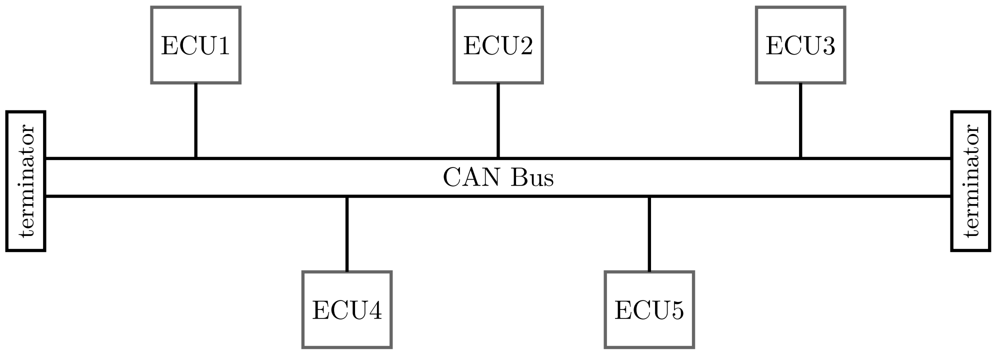


Figure 2.2: CAN bus network.

CAN supports four types of message frames: *data frame*, *remote frame*, *error frame*, and *overload frame* [9].

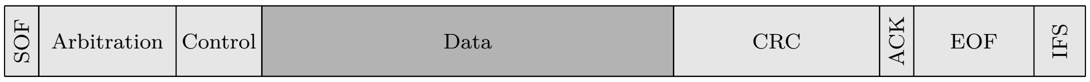


Figure 2.3: Fields of CAN message frame.

- **Data frame:** This is the frame transmitted by the sending ECU and consumed by one or more of the receiving ECUs. Data frames have higher priority than remote frames.
- **Remote frame:** This frame can be transmitted by a requesting ECU that wishes to request a certain message from the source providing such message. This frame is followed by a data frame containing the requested message.
- **Error frame:** It is a frame signalled by any ECU at any time during data or remote frame transmission to indicate an error condition.
- **Overload frame:** This frame adds a delay between two data or remote frames. The default distance between any consecutive frames is 3 bit times.

Figure 2.3 shows the fields of a CAN message frame. Note that there are two message frames: standard with 11-bits identifier (CAN 2.0A) and extended with 29-bits identifier (CAN 2.0B). The standard CAN supports 2048 messages (2^{11}), while extended CAN supports 536+ a million messages (2^{29}).

As shown in Figure 2.3, CAN frame has eight main fields:

1. **Start of Frame (SOF):** Marks the start of data or remote frame. It is a 1-bit field.
2. **Arbitration:** It is 12-bits field in standard CAN and 32-bits field in extended CAN, and consists of two main components. First is the message identifier (ID), which is used in the arbitration process and is 11-bits size in the standard frame and 29-bits size in the extended frame. Second is Remote Transmission Request (RTR) which is used to distinguish between data frame and remote frame. The RTR is 1-bit size and in the case of a remote frame its value is 1 (recessive), otherwise, it is 0 (dominant).
3. **Control:** The control field determines the size of data and length of message ID[1]. The size of this field is 6-bits.
4. **Data:** This field involves the actual data the message carries. The maximum payload in CAN message is 8 bytes, so this field is 64-bits. The field size is 0 in the case of a remote frame, as it does not contain any data.

[1]Occasionally, we say CAN ID to refer to CAN message ID.

5. **Cyclic Redundancy Check (CRC):** This is the checksum field and it is 16-bits size.
6. **Acknowledgement (ACK):** It is the acknowledgement of the checksum check. It is a 2-bits field.
7. **End of Frame (EOF):** This marks the end of data or remote frame, and it is 7-bits size, all are recessive bits.
8. **Interframe Space (IFS):** It is a 3-bits field and represents the minimum intermission time between any two frames. During this time, no node is allowed to transmit either data or a remote frame. Transmission of an overload frame, on the other hand, is allowed during this period.

Media Access Control and Arbitration Process CAN follows a Carrier-Sense Multiple Access with Collision Detection (CSMA/CD) scheme for its Media Access Control (MAC) to control its bus access when different nodes try to transmit simultaneously. In particular, CAN is a Carrier-Sense Multiple Access with Collision Detection and Arbitration on Message Priority (CSMA/CD+AMP) [8]. This means that CAN uses the arbitration field for this task, specifically the CAN ID. The lower the value of CAN ID, the higher its priority level. In the arbitration process, when more than one node is transmitting at the same time, the ID value is checked to permit the node with the highest priority to access the bus and start transmitting. An example of this process is shown in Figure 2.4. Suppose there are two ECUs: ECU_1 and ECU_2 with identifiers $ID_{ECU_1} = 113$ and $ID_{ECU_2} = 112$ which translated into binaries 00001110001 and 00001110000, respectively. The beginning bits in these two identifiers are the same until the last bit at index 10 which differs by one. Because 0 in this index for ECU_2 is dominant bit, and in fact $112 < 113$, then ECU_2 wins the arbitration process (as shown in Figure 2.4 where bits appear on the CAN bus) and transmits its message, while ECU_1 backs off until ECU_2 finishes transmitting, then ECU_1 can start transmitting its message. This way the arbitration in CAN is called *non-destructive arbitration* because frames are not destroyed in case of a collision, instead they will be re-transmitted.

In terms of error handling, CAN has a strong error detection mechanism using CRC where each receiving node compares the calculated CRC with the one transmitted and if an error is detected, an error message will be sent, and the frame will be re-transmitted [9], [30].

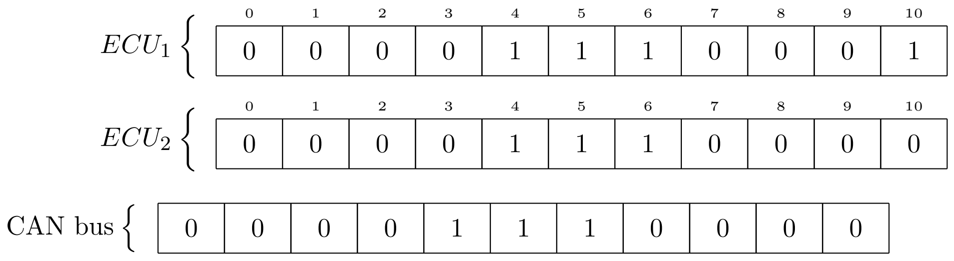


Figure 2.4: Example of an arbitration process in CAN.

From a security point of view, one of the major limitations in CAN is that it does not support any authentication or authorisation mechanisms (e.g., addressing in CAN is based on message ID with no information about the sender or receiver of the message [31]), which puts it at risk of cyber-security threats.

2.1.2.3 Local Interconnect Network (LIN)

Some functionalities inside the vehicle do not require fast networks. For example, doors lock/unlock, windows roll up/down, seat controls, etc. For these functionalities, a simple network is sufficient. Local Interconnect Network (LIN) was designed for this purpose. It is much slower than CAN (maximum rate of LIN is 20 Kbps), yet much cheaper and simpler. LIN uses a bus-based topology with master-slave architecture and uses a single wire of $40m$ maximum length with one master node and up to 15 slaves [23]. Like CAN, it supports a maximum payload of 8 bytes. However, LIN can have only up to 16 nodes capable of transmitting and receiving cyclic frames. In LIN, frames are transmitted in a polling fashion where the master node selects the node to transmit based on a schedule hosted by the master node. The master node first transmits the message header with ID value, the payload is then transmitted by the source node of the specified ID, so there are no collisions in LIN as nodes do not compete to access the bus like the case with CAN. One of the disadvantages in LIN is that it does not support strong error detection methods [30]. Similar to CAN, it uses physical, data link and application layers of the OSI model, and is available from several vendors. Unlike CAN, the security threat of LIN is lower. This is due to the difficulty of performing attacks on LIN. In order to succeed in the attack, two things should be done: first, injecting a false response with fake data into the LIN bus and second, such injection should happen at a specific time after the header part is transmitted by the master node. In this case, it is difficult to inject the false response because it might be met with the correct one. Moreover, it is not possible to manipulate the schedule at the master node, as this requires physical access to the master node [32].

2.1.2.4 FlexRay

FlexRay was designed to satisfy needs imposed by *x-by-wire* applications such as drive-by-wire, steer-by-wire, brake-by-wire, and shift-by-wire [30]. FlexRay is ten times faster than CAN with up to 10 Mbps bandwidth per channel. In addition, it allows up to 256 bytes of maximum payload [30]. As for the topology, FlexRay is more flexible than CAN and LIN in which it can support point-to-point, linear bus, and active as well as passive star topologies [30]. Like CAN, the cabling used in FlexRay is UTP with maximum length of $24m$ and up to 22 nodes in a bus topology. FlexRay supports two channels, in which the second one acts as a backup in case the first channel fails [23]. FlexRay uses Time Division Multiple Access (TDMA) for its MAC [23] and messages in FlexRay are cyclic-frames [30]. Errors in FlexRay frames can be detected using CRC and there is no way that the faulty frame can be re-transmitted. For FlexRay to operate, it needs the three layers of the OSI model just like CAN and LIN. Compared with other protocols, the cost of the FlexRay network is high [23] with only a few vendors' support. Cyber-security threats against FlexRay are possible, though not as easily as with CAN [23].

2.1.2.5 Media Oriented Systems Transport (MOST)

As the name suggests, MOST was designed specifically for infotainment systems. It has built-in capability for audio and video streaming. Unlike other protocols, MOST uses all layers of the OSI model. It typically follows a ring-based topology, and it can also be structured as a star topology. There are three types of MOST: MOST25, MOST50, and MOST150 supporting bandwidth of 25 Mbps, 50 Mbps, and 150 Mbps, respectively. The maximum network length of MOST is $1280m$ where it can run on UTP (such as in MOST50) [23] in addition to Plastic Optical Fiber (POF) (such as in MOST150) which alleviates issues related to Electromagnetic Compatibility (EMC) [30]. MOST supports bandwidth of up to 150 Mbps, which is much higher than previous protocols. A single MOST ring can have up to 64 nodes, and it can transmit up to 384 bytes [23], [33]. Like FlexRay, messaging in MOST is based on TDMA. MOST supports strong error detection using Ring Break Diagnosis (RBD) [33] where higher layer protocols handle the correction part [30]. Due to the fibre optic cabling used in MOST, it is considered one of the most expensive protocols. At the moment, MOST is not widely available; there is only one vendor, which is MOST cooperation[2]. Moreover, the level of security threats in MOST is medium to high due to the fact that it is mostly used for infotainment applications which normally have some sort of communication with the external world.

2.1.2.6 Automotive Ethernet (AE)

Due to the constantly increasing demands of high bandwidth and low latency required by new applications in in-vehicle networks, such as ADAS applications, the conventional communication protocols mentioned above cannot meet such stringent requirements. Therefore, a new communication protocol is needed to handle such strict requirements.

Ethernet is a well-known leading protocol for Local Area Network (LAN), and for a span of over 40 years, it proved its ability to keep up with the market demands by continuously developing new features, increasing data rates, and adding more and more flexible media options [30]. Diagnostics over Internet Protocol (DoIP) was the first application to use Ethernet in the automotive network. Nowadays, Ethernet can be found in infotainment applications as well as in ADAS, and it is expected that future in-vehicle networking architectures will follow flat and Ethernet-based topologies [34].

It is worth mentioning that there are two types of Ethernet: *Shared Ethernet* and *Switched Ethernet*. In shared Ethernet, the topology is a linear bus and the bus is shared among all connected devices, hence collisions are not rare in this type. On the other hand, switched Ethernet uses more intelligent devices, i.e., switches, instead of bus/hubs and the topology is structured in a star-like topology. In addition, switched Ethernet uses full-duplex transmission. Therefore, unlike in shared Ethernet, switched Ethernet eliminates the concerns over the possibility of collision occurrences [30]. For modern applications, including automotive networks, the type of Ethernet used is switched Ethernet.

Ethernet uses all layers of the OSI model, and it supports up to 100 Mbps bandwidth per link and number of connected nodes is only limited by the number of ports in the Ethernet switch. One of Ethernet's features is that the network can be structured in any topology including star, star of stars, and tree topologies. The other attractive feature of Ethernet is that it uses UTP cabling instead of the more expensive alternatives such as optical fibre, coaxial cables, and Low-Voltage Differential Signalling (LVDS). The maximum length of automotive Ethernet

[2]https://www.mostcooperation.com/specifications/

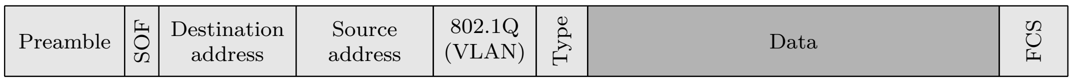


Figure 2.5: Fields of Ethernet message frame.

links is $15m$. There are multiple Ethernet vendors such as NXP[3], Marvell[4], and the list is still growing. However, currently, no many manufacturers use Ethernet. This is mainly due to two reasons. First, automotive Ethernet is relatively expensive compared with CAN. This results in lower demand and many vendors. Second, adopting Ethernet requires adding more hardware devices: Ethernet switches.

Figure 2.5 shows the format of the Ethernet frame. In the following, we briefly describe each field [30]:

1. **Preamble:** It is a 7-bytes field of alternating $1s$ and $0s$ signalling that a frame is coming and allowing for synchronisation between transmitter and receiver.

2. **Start of Frame (SOF):** It is a one byte field of $1s$ and $0s$ with the last two bits being 1 (i.e., 10101011). This indicates that the actual frame is coming.

3. **Destination address:** As the name indicates, this field determines the destination address of the frame, and it is a 6-bytes field (3 bytes organisation number and 3 bytes device number).

4. **Source address:** Similar to the destination address, except it indicates the address of the frame's transmitter.

5. **802.1Q (VLAN/Frame Priority) Tag:** This field is 4-bytes long, and it is an optional field. It can be used to instantiate a Virtual Local Area Network (VLAN) in addition to specifying a frame priority level.

6. **Type:** This is 2-bytes field and is widely known as *Ethertype*. It is used to indicate the type of data being transmitted in the payload.

7. **Data:** This is the field that carries the actual data to be transmitted, and it can be up to 1500 bytes long and a minimum of 46 bytes.

8. **Frame Check Sequence (FCS):** this is a 4-bytes field with 32 bits of CRC calculated by the sender. Once the receiver receives the frame, it calculates the CRC value using

[3]https://www.nxp.com/products/interfaces/ethernet-:MC_1436432488692
[4]https://www.marvell.com/products/automotive.html

the same formula the sender used. If there is any mismatch, then the frame is faulty and will be discarded.

Standard Ethernet in its current form cannot be used in automotive networks, as it does not provide adequate Quality-of-Service (QoS) required by automotive applications. Therefore, to use Ethernet in the automotive domain, it first has to be adapted in order to be able to satisfy the stringent requirements of automotive applications. For this reason, two main standards are working to extend the standard Ethernet to be suitable for usage in in-vehicle network applications. These standards are: *Time-Triggered Ethernet (TTEthernet)* and *Time-Sensitive Networking (TSN)*. Below, we describe each of these standards.

TTEthernet In 2011, Society of Automotive Engineers (SAE) standardised TTEthernet (AS6802) [35], [36]. TTEthernet is known to be used in industrial and avionics applications (instead of Avionics Full Duplex Switched Ethernet (AFDX)) [35], and it uses full-duplex links. TTEthernet supports strict network requirements through the use of three main traffic types: *Time-Triggered (TT)*, *Rate-Constraint (RC)*, and *Best-Effort (BE)* [35]–[37].

In TT, timing slots are assigned to each node in the network using offline scheduling. This way (that TTEthernet uses TDMA in its MAC for TT traffic), TTEthernet enables a deterministic behaviour of the network [36]. TT traffic has higher priority than the other two types (RC and BE). Brake-by-wire and steer-by-wire are two system examples that can benefit from using TT traffic. Unlike TT, RC and BE are event-triggered traffic [36]. The main goal of RC traffic is to limit the transmission rate so that each application can guarantee a maximum bandwidth [37]. RC can be used for applications with stringent bandwidth requirements and less timing constraint, such as multimedia and safety-critical applications [37]. On the other hand, BE traffic has the lowest priority level among all three types, and it uses the available, remaining, bandwidth that is not used by TT and RC. BE traffic does not have guaranteed performance, thus, it can be used by applications that do not necessitate QoS requirements, an example of such applications is Internet Protocol (IP) traffic with no QoS demands.

The other type of adapted AE is TSN. TSN was originally derived from Audio Video Bridging (AVB) [35], [38], [39]. Therefore, it is worth discussing AVB first, next will describe TSN.

AVB AVB was developed by IEEE AVB Task Group[5] and it was first developed to provide synchronised audio/video streaming services, then it deemed useful in automotive networks for transmitting latency-critical traffic [23]. AVB developed three main standards [37]:

1. **Timing and Synchronisation for Time-Sensitive Applications (IEEE 802.1AS)**: this standard uses generalized Precision Time Protocol (gPTP) protocol, which is based on IEEE1588 Precision Time Protocol (PTP) [35], to achieve nodes' synchronisation.

2. **Stream Reservation Protocol (SRP) (IEEE 802.1Qat)**: when this standard is used, network resources, buffers and queues, for instance, are reserved within network switches along the path between source and destination [37], [40].

3. **Forwarding and Queuing Enhancements for Time-Sensitive Streams (FQTSS) (IEEE 802.1Qav)**: this standard is used for traffic shaping. It uses Credit Based Shaper (CBS) [39] as it is essential for queuing algorithms [41]. The standard works by isolating latency-critical traffic from non latency-critical ones, and categorises them into two different classes [37].

In addition, AVB defines two QoS Stream Reservation (SR) classes [35], [37], [40][6]:

- **SR class A:** this class guarantees maximum latency of $2ms$ over seven hops [37] with default transmission period of $125\mu s$ [30].

- **SR class B:** it provides maximum latency of $50ms$ over seven hops [37] with default transmission period of $250\mu s$ [30].

These classes are assigned two priority levels 3 and 2 for SR class A and SR class B, respectively [30], with class A having a higher priority level than class B [40]. AVB also defines BE traffic class that is non-SR traffic [35].

TSN TSN has evolved from AVB and it is developed by IEEE TSN Task Group[7]. The goal of TSN is to meet hard real-time requirements, and thus they developed an additional traffic class called Control Data Traffic (CDT). TSN improved AVB by upgrading synchronisation standard IEEE 802.1AS into IEEE 802.1AS-Rev as well as adding new standards. The new standards defined by TSN are as follows:

[5]`https://grouper.ieee.org/groups/802/1/pages/avbridges.html`

[6]Note that there are some works on extending these two classes into more customised class such as AVB_ST [42] where new SR class is added on top of class A and class B in AVB network.

[7]`https://1.ieee802.org/tsn/`

- **Timing and Synchronisation for Time-Sensitive Applications - Revised (IEEE 802.1AS-Rev)**: enhanced version of synchronisation approach.
- **Enhancement for Scheduled Traffic (IEEE 802.1Qbv)**: scheduling is provided by this standard using Traffic Aware Shaper (TAS). The idea is that critical traffic uses time-aware windows for scheduling, where a guard band is used to block lower-priority traffic from interrupting the critical traffic [41].
- **Frame Preemption (IEEE 802.1Qbu)**: the idea of this standard is to allow high-priority traffic to interrupt lower-priority traffic in order to transmit critical traffic in time [43].
- **Frame Replication and Elimination for Reliability (IEEE 802.1CB)**: this standard aims to improve the reliability of TSN by numbering and replicating packets, then identifying the redundant packets and evicting them at the destination [41].
- **Path Control and Reservation (IEEE 802.1Qca)**: this allows for bridged networks to follow other routing protocols than the shortest path routing [44].
- **Stream Reservation Protocol (SRP) Enhancements and Performance Improvements (IEEE 802.1Qcc)**: enhancement of stream reservation protocol.
- **Cyclic Queuing and Forwarding (CQF) (IEEE 802.1Qch)**: uses Peristaltic Shaper (PS) to synchronise the process queuing and dequeuing [44].
- **Per-Stream Filtering and Policing (IEEE 802.1Qci)**: using this standard, streams can be identified by their header fields and assigned an Internal Priority Value (IPV) that determines the traffic class. The stream then can be further filtered into different sub-streams based on their, e.g., priority levels [39]. Hence, this standard can be used to block streams satisfying certain properties, which can be useful for securing the network [45].
- **Time-Sensitive Networking for Fronthaul (IEEE 802.1CM)**: to provide fronthaul networks with bridged Ethernet capability [44].
- **Interspersing Express Traffic (IEEE 802.3br)**: it is related to frame preemption to tackle traffic priority problem where low priority traffic intercepts higher priority traffic [44].

- **Asynchronous Traffic Shaper (ATS) (IEEE 802.1Qcr)**: this is a new standard, and it aims to improve the overall latency to have better performance than CQF [46].

TSN network is composed of five main constituents [47]:

1. End devices: these are the transmitter and receiver of TSN traffic. Sometimes, they are referred to as *talker* and *listener*.
2. Bridges: these are the Ethernet switches.
3. TSN flow: it is latency-critical traffic transmitted between transmitter and receiver.
4. Central Network Controller (CNC): this component controls the TSN network by setting the schedule for the TSN traffic.
5. Centralised User Configuration (CUC): it provides communication between end devices and the CNC. Generally, the CUC asks the CNC for TSN flows by providing their communication requirements.

Given the aforementioned description of TSN, it can be seen that it is the perfect candidate for automotive networks with strict timing and resource requirements [10].

It is worth noting that there is another similar, deterministic, communication protocol developed by IETF (Internet Engineering Task Force) called Deterministic Networking (DetNet)[8]. The main difference between TSN and DetNet is that TSN is developed for layer 2 of the OSI model while DetNet aims to expand this to higher layers, i.e., layer 3 [41]. DetNet is still new and the development of this protocol is ongoing.

In terms of error detection and correction, Ethernet has a strong detection mechanism, while correcting the detected errors can only be done at the higher layers [30]. Furthermore, because Ethernet is a more mature protocol and well-known in the IT world, it poses high risks of cyber-security threats as adversaries are aware of different attacks experienced and performed on computer networks. Such attacks can easily be extended to automotive Ethernet. Therefore, since Ethernet has started to be integrated into the in-vehicle network, it is more crucial now to investigate new solutions to secure this network effectively in order to ensure drivers' and passengers' safety as well as vehicle components' integrity.

[8]https://datatracker.ietf.org/wg/detnet/about/

2.2 In-Vehicle Networking Architectures

The overall in-vehicle network topology is highly dependent on the E/E architecture used in the vehicle. Such architectures define the integration between hardware (i.e., ECUs), communication networks and software applications into one, consolidated system.

Initially, the evolution of E/E architectures started back in the 1970s [48] where vehicle's components (ECUs) were mainly connected in a **point-to-point** fashion. This architecture, however, suffered a massive wiring complication. To deal with this limitation, in 1983, Bosch developed the first **fieldbus** communication, i.e., CAN, which significantly improved the wiring harness of the in-vehicle network. Typically, the in-vehicle network consisted of different control systems, where each system had its own fieldbus. The issue with fieldbus communication is that the interaction between different systems is not possible as each has its own communication protocol that is different from the others. For this, a **central-gateway** architecture was introduced. The central-gateway connects these different systems together. The main role of the gateway in this architecture is limited to regulating and translating traffic from different systems that might use different communication protocols. In addition, all ECUs are connected to this gateway, which means that the gateway incurs heavy load from all these ECUs. Two recent architectures that came to minimise the load on the gateway and to further improve the wiring cost are **domain-based** and **zonal-based** architectures. In domain-based architecture, ECUs are grouped based on their functionality in one domain. The zonal-based architecture, on the other hand, groups ECUs based on their physical locations in the vehicle. In the following, we describe, in detail, the fieldbus, central-gateway, domain-based, and zonal-based architectures.

2.2.1 Fieldbus Architecture

Fieldbus architectures are the first E/E architectures, and they follow a linear bus topology. Examples of communication protocols that use this architecture are CAN, LIN, and FlexRay. The topology of this architecture looks like the one shown in Figure 2.6. This architecture is considered *decentralised*, where there is no central entity that can control the overall in-vehicle network. Instead, each ECU is responsible for the sending and receiving of network traffic, in addition to detecting errors in the received message.

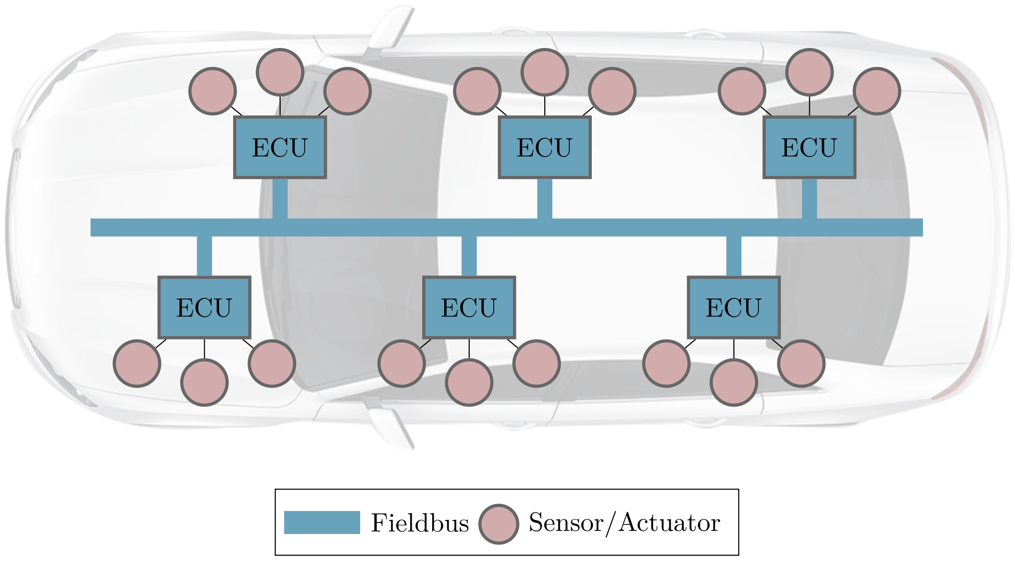


Figure 2.6: Fieldbus architecture.

2.2.2 Central-Gateway Architecture

Central-gateway architecture is sometimes referred to as *distributed E/E architecture* [49]. It is in fact the traditional E/E architecture [23], [49], [50] that was introduced to support the co-ordination between different subsystems that use different communication protocols. Figure 2.7 depicts an overview of the central-gateway architecture. As shown, the gateway connects different communication protocols such as CAN, LIN, and FlexRay. Mainly, the goal of the gateway is to facilitate communication between different systems by translating the traffic from the sender communication protocol to the receiver protocol.

This architecture can be found in cars manufactured until around 2019 [49]. Examples of cars that use this architecture are the Volkswagen Passat [23], Audi [51], and BMW 7 series [52].

However, with the increasing functionalities added to the vehicle, limitations of this architecture started to emerge. The main issue is that the gateway became a source of bottleneck where all traffic, both internal (between ECUs of the same protocol) and cross-traffic (between different protocols), have to go through this gateway for processing and regulation [53]. Therefore, a new architecture, domain-based, was introduced to deal with this issue as explained next.

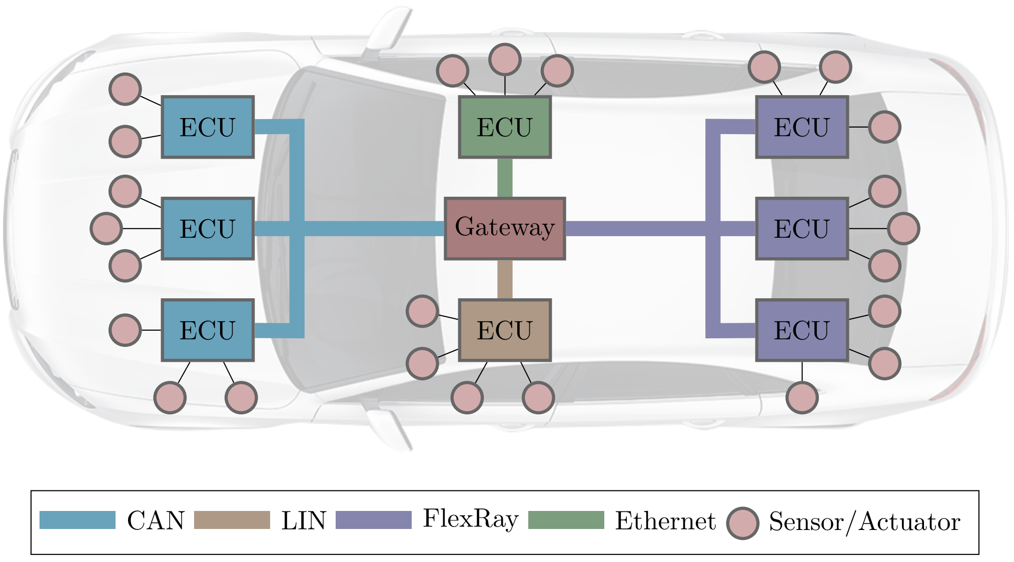


Figure 2.7: Central-gateway architecture.

2.2.3 Domain-based Architecture

This architecture eliminates the extra loads incurred on the gateway in the central-gateway architecture. The concept of this architecture is to distribute the load to other units called *Domain Control Units (DCUs)*[9]. These units are capable of handling complex operations [43]. An overview of this architecture is depicted in Figure 2.8. As can be seen, this architecture combines the ECUs based on their functionalities into one domain. Generally, there are four domains, as explained in the vehicle's domains (Chapter 2.1.2.1). Each domain is controlled by a DCU, and a backbone network, e.g., Ethernet, interconnects different DCUs [50]. This architecture is concerned with logical optimisation of the vehicle through functionalities [54]. For instance, all ECUs related to the infotainment system are grouped together into one domain and managed by this domain's DCU. This further improves the wiring harness and reduces the number of ECUs.

Domain-based architecture is a centralised architecture, and it is the current architecture used by many car manufacturers. However, domain-based architecture suffers from two challenges [43]. First, some functions require intensive communications between DCUs, which makes the boundaries among these DCUs ambiguous. A second challenge is that for future E/E architectures with increasing high-end functionalities, current bandwidths cannot accommodate

[9]Can also be called *Functional Domain Controllers (FDCs)*, or simply *Domain Controllers (DCs)*.

such intensive interactions. Therefore, to tackle these challenges, this architecture can be extended to *cross-domain architecture* [43]. In cross-domain architecture, more than one domain function can be integrated into one unit called Cross-Domain Control Unit (CDCU) [43], [55]. Then the processing of complex functions is done in such CDCUs [55]. For instance, chassis and powertrain functionalities can be consolidated into one vehicle motion control CDCU [43], [56]. Looking at Figure 2.8, in cross-domain architecture, the DCUs can be replaced with CDCUs.

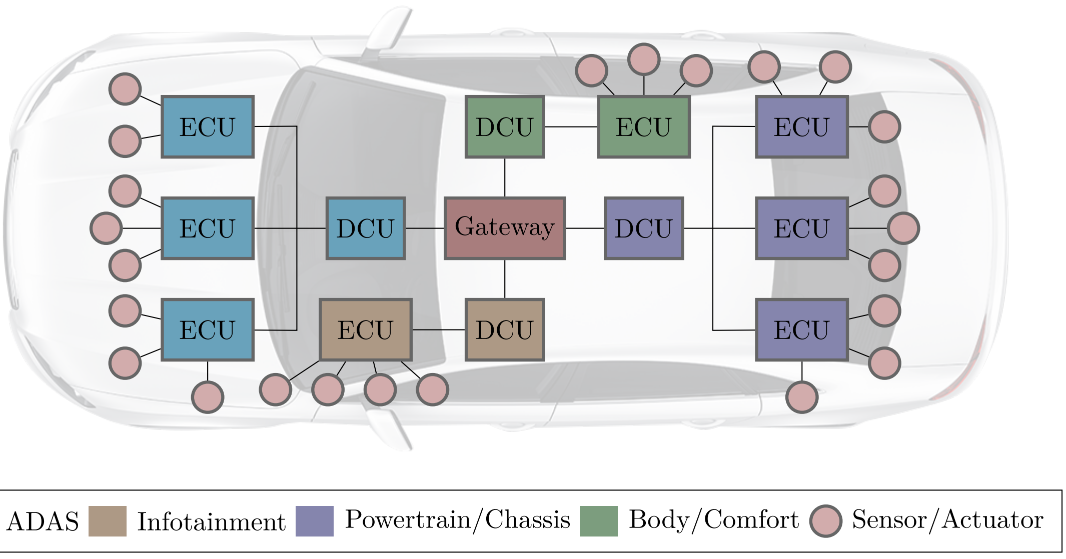


Figure 2.8: Domain-based architecture.

With more complex functionalities added to the vehicle system, domain and cross-domain control units become overloaded with local processing of such functionalities that even require further bandwidth to transfer the processed data between ECUs [43], [53]. Moreover, the increasing number of complex functionalities become more and more dependent on other functionalities, which adds more wiring complications [43].

2.2.4 Zonal-based Architecture

Zonal-based architecture proposed to overcome the limitations that arose in domain-based architecture due to the increasing number and complexity of vehicle functionalities. Zonal-based architecture is also known as "*Vehicle-Centralised Architecture*" [43]. In this architecture, the vehicle is divided into topological zones where each zone is controlled by a Zone Control Unit (ZCU)[10] [43], [50], [53]. Then, the ECUs are clustered based on their physical locations in

[10] Others call it a *Zonal Gateway*, *Zone I/O Controller* for controlling the zone input and output, or *Zone ECU (ZECU)*.

the vehicle, and each ECU is connected to the closest ZCU. All the ZCUs are interconnected via Ethernet backbone network or mesh network [50]. As shown in Figure 2.9, this architecture reduces the wiring and number of ECUs significantly as it enables centralised computing by utilising a powerful compute unit, i.e., *server*. The central server takes all the complex processing and computations burden, hence it can be a very powerful unit or a cluster of powerful units [55]. Zonal-based architecture is the enabler for SDNs [50] where hardware-specific functions can be replaced with software components, hence dramatically minimising the hardware and wiring cost.

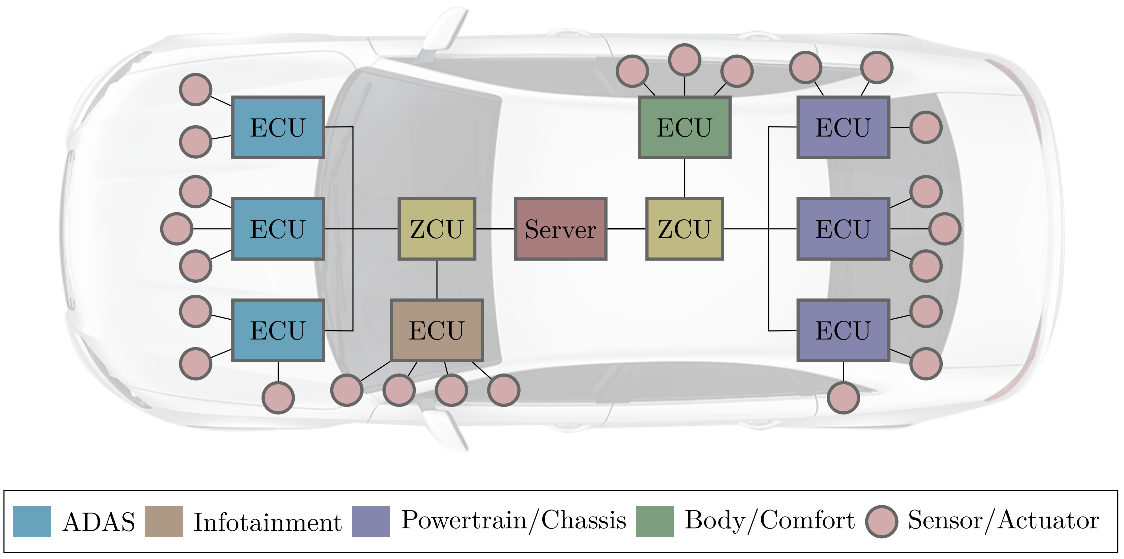


Figure 2.9: Zonal-based architecture

Unlike domain control units in domain-based architecture, zone control units do not implement complex functions, they are generally responsible for implementing the zone's basic functions [55] in addition to receiving and forwarding data related to the corresponding zone to the server which performs the complex functions. Further, these complex functions can be offloaded to the cloud for further processing [43]. ZCUs can also support gateway and switching functionalities and act as a smart junction box [57].

Future E/E architectures are shifting towards more centralised architectures that adopt the structure of having fewer number of very powerful ECUs instead of the current large number of resource-constrained ECUs [48], [50]. Such powerful units are referred to as *Vehicle Computers* [50], and they are considered *High-Performance Computing Platforms (HPCPs)* [58] that are far more powerful than the current ECUs [21], [59]. Both domain-based and zonal-based architectures are centralised architectures that evolved as a result of the ongoing transformation of current E/E architectures into centralised, cross-domain architectures [43]. As domain-based architecture is concerned with the logical optimisation of the vehicle by functionalities, zonal-based architecture is concerned with the physical optimisation of the vehicle by locations [54]. Benefiting from both architectures, it is expected that in the future, both domain-based and zonal-based architectures will coexist as needed to achieve the optimal structure and functionality with minimal wiring cost [54].

2.3 Security and Monitoring Challenges in In-Vehicle Networks

Until recently, monitoring the in-vehicle network was not seen as a task worth investigating. The main reason is that, in the past, security was not a concern [6], [23] because the vehicle components were only internally connected with no access from or connection to the external world as the case today. Nowadays, with the increasing advances in Connected and Autonomous Vehicles (CAVs), and IoVs, the vehicle becomes more connected to the outside world and is able to exchange information with other vehicles on the road (inter-vehicle communications) as well as with the surrounding objects, such as RSUs, base stations, pedestrians, etc. Such connections expose the vehicle to cybersecurity threats.

The following section discusses different attack surfaces of in-vehicle networks. Next, different types of attacks and analysis of their severity level are described. Then, the challenges of monitoring in-vehicle networks and the existing solutions are highlighted.

2.3.1 Attack Surfaces

Figure 2.10 shows examples of different attack surfaces that adversaries can exploit to gain access to and tamper with the vehicle system [5], [60], [61].

One of the attack surfaces is a keyless access system. For this, authors in [62] and [63] showed how the keyless access system can be hacked. The problem with this system is that the attack can occur remotely, for instance, the car can be unlocked using a smartphone application that sends the signal to the infotainment system [61], allowing adversaries to intercept the signal and hack the system. In addition, for charging the Electric Vehicle (EV), it needs to be connected to

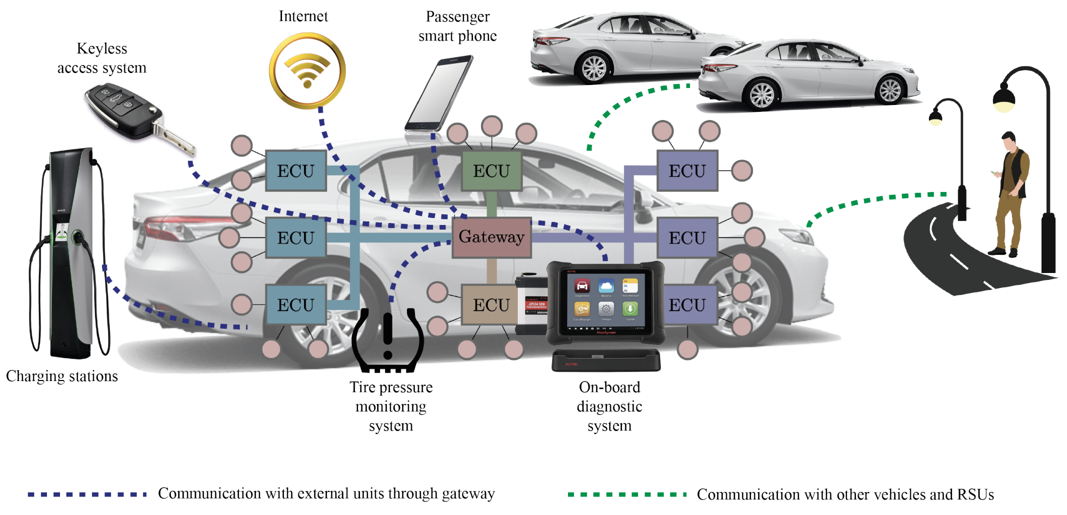

Internet
Passenger smart phone
Keyless access system
ECU
ECU
ECU
ECU
Gateway
ECU
ECU
ECU
ECU
Charging stations
Tire pressure monitoring system
On-board diagnostic system
Communication with external units through gateway
Communication with other vehicles and RSUs

Figure 2.10: Attack surfaces on vehicle systems.

the charging station for long hours with payment information being exchanged with the station. This gives plenty of time for adversaries to gain access to such private information through the charging plug interface [61].

Tire pressure monitoring system has its own vulnerabilities as well [4], [61]. It suffers from two critical issues. First, there are no authentication mechanisms for the messages exchanged between the system and the in-vehicle network. This can be exploited by adversaries as shown in [64] where they could eavesdrop on the communication of a moving vehicle and inject messages that trigger the system alarm. This is a general problem in CANs (which is the communication protocol used in tire pressure monitoring systems) where they do not support authentication or authorisation mechanisms [11]. The second issue is that because each tire has its own fixed identification key that is being communicated alongside the tire pressure messages to the in-vehicle network, these messages can easily be traced, and the car can be identified [61], [64].

Moreover, the onboard diagnostics system is susceptible to both remote and physical attacks [65]–[67]. Furthermore, connecting the vehicle to the Internet through the infotainment and telematics systems, provides another attack surface to exploit by hackers all over the Internet. Not to mention that passengers' and drivers' smartphones can also be exploited by hackers to tamper with the vehicle through smartphone car applications [68], [69]. Different sensors in the vehicle can also be exploited by adversaries to perform different types of attacks [70].

On the other hand, vehicle connectivity with other vehicles and the surrounding infrastructure (using external communications) also poses a great risk of security threats [61], [71].

2.3.2 Attack Types

Different attack types can be performed by hackers based on the communication protocol they target. The following are examples of the attack types that in-vehicle networks are susceptible to [72], [73]:

- **Eavesdropping attacks**
  In this attack, adversaries can intercept the communication between ECUs in the vehicle and listen to the information being communicated between them.
- **Impersonation attacks**
  Adversaries perform this attack by planting a malicious node into the in-vehicle network. The malicious node impersonates one of the existing, *legitimate*, nodes and manipulates its functionality to impair its operation.
- **DoS attacks**
  In DoS attacks, the attacker frequently transmits a large number of forged request messages with the goal of flooding the network so that the service will not be available to other users/nodes (or ECUs in the scenario of in-vehicle networks) in the network.
- **Fuzzy attacks**
  Fuzzy attacks occur when a malicious node transmits random messages with legitimate IDs but fake data values. This attack results in unexpected and unwanted behaviour which can cause serious harm.
- **Replay attacks**
  Adversaries who gained control over one of the ECUs can collect messages transmitted in the network, so they can replay them with the aim of causing disruption to the network.
- **Spoofing attacks**
  In this type, a malicious node sends messages with a false ID which is the same as one of the legitimate IDs. Then the receiving node cannot distinguish if the received message is valid or not.

Table 2.3 shows security threats of in-vehicle networks and the severity of each attack [23], [72]. It is important to note that the severity level of these attacks is subjective as it depends on different factors such as the sophistication of the attack, whether the attack is remote or physical, and if the attack targets a single or multiple vehicle's components. For instance, the eavesdropping attack by itself is relatively harmless compared with other attacks. However, such attack can be a starting point and enabler for more harmful attacks such as the impersonation attack.

In this thesis, we focus on CAN and AE as the communication protocols for in-vehicle networks, and investigate the proposed approach to detecting DoS attack.

Table 2.3: Security threats and their severity on the in-vehicle network.

Attack	Vulnerable network	Target	Consequences	Risk
Eavesdropping	CAN	In-vehicle communications, drive pattern, vehicle location, camera records	Privacy breaches	Low
Impersonation	CAN, AE	Immobilisation, remote key access, odometer reading, DVD player, vehicle speed	Financial loss, involuntary manoeuvre, life threat to users	High
DoS	CAN, AE	Vehicle speed, movement direction	Paralysing traffic, life threat to all road users	High
Fuzzy	CAN	Steering command, vehicle speed, brake interference	Involuntary manoeuvre, financial loss, life threat to all road users	High

Table 2.3: Security threats and their severity on the in-vehicle network.

Attack	Vulnerable network	Target	Consequences	Risk
Replay	CAN, AE	Immobilisation, remote key access, odometer reading, DVD player	Financial loss	Low to medium
Spoofing	CAN, AE	In-vehicle communications	Financial loss, communication disruption	Low to medium

As illustrated in Table 2.3, CAN is the most vulnerable protocol in in-vehicle networks. Despite that it has been around for a while now, it is still the least secured protocol. Several issues with CAN make it vulnerable to most types of attacks. For instance, the lack of authentication and authorisation mechanisms facilitates the spoofing attacks. In addition, the fact that CAN messages are broadcast to all ECUs connected to the CAN bus, makes it easy to perform eavesdropping attacks. Further, the arbitration process helps DoS attacks to take place on CAN by simply injecting frequent and high-priority messages [74]. Replay attacks, on the other hand, are driven by the lack of timestamp information of messages exchanged in the CAN network. To this extent, it is necessary to secure the vehicle against attacks as well as to ensure its appropriate functionality under abnormal behaviours.

The following highlights the importance of monitoring the in-vehicle networks and provide description of a complete monitoring system that the manufacturers should adopt for their in-vehicle networks.

2.3.3 In-Vehicle Network Monitoring

As discussed earlier, modern in-vehicle networks require considering security aspects [21]. In fact, security should be one of the main design principles of modern vehicles and car manufacturers should adopt the security-by-design norm [75].

However, with necessary security measures in place, there is still a chance that the vehicle can get compromised one way or another by, for instance, new attacks that have not been known to the system before, and thus no measures are readily available against such attacks. For this reason, the in-vehicle network should have a system that can detect such attacks. This can be achieved by developing a robust monitoring system. A typical monitoring system should consist of three main phases:

1. **Detection phase**: this phase involves monitoring the network to detect any abnormal behaviour.

2. **Localisation phase**: this phase is the subsequent phase to the detection phase, in which the source of the detected abnormality is located. Note that this phase can also be achieved through the detection phase, hence having a single phase for both detection and localisation (this has been illustrated in one of the proposed tomographic approaches, i.e., DNT, in Chapter 5).

3. **Mitigation phase**: this phase is crucial for safety-critical applications such as in-vehicle networks. In this phase, the malfunctioning behaviour should be mitigated so that it results in minimum loss and ensures the safety of drivers and passengers.

2.3.4 Existing Solutions for Monitoring In-Vehicle Networks

As current in-vehicle networks lack sophisticated monitoring approaches, the research community are investigating different monitoring solutions to be incorporated into the in-vehicle network. In the following, we discuss these solutions and categorise them into four categories: frequency and time-based solutions, physical specification-based solutions, information theory-based solutions, and AI-based solutions.

2.3.4.1 Frequency and Time-based Solutions

Measuring the frequency and time intervals of CAN messages is a good indicator of whether the network is experiencing anomalous behaviour or not. For example, authors in [76] proposed an IDS to detect injection attacks based on the message frequency of different CAN IDs exploiting the fact that each message ID has a certain time interval. The procedure starts by checking the ID of the incoming message and measuring its time interval based on the last message received. The IDS trigger an alarm if the measured time interval is less than the normal value.

To reduce the false alerts, they used a scoring strategy. For instance, to detect DoS attacks, for all CAN messages they set a single threshold of $0.2ms$ time interval, if the time interval of the received message is less than this value then the IDS determines this as DoS behaviour and increases the score value by 1. This approach achieved 100% accuracy if the scoring threshold is 4. This means that the IDS classify the message as a DoS attack if there are 4 (or more) consecutive instances of this message with a time interval less than $0.2ms$. Similar approaches proposed in [77] and [78]. Authors in [77] used inter-signal wait times of CAN messages as a measure to detect injection attacks, including DoS attacks. While authors in [78] used message rates in addition to time intervals to detect injection attacks.

Another one of the state-of-the-art monitoring solutions for in-vehicle networks is proposed by Lee in [79] and is called Offset ratio and Time interval based Intrusion Detection System (OTIDS). The idea of OTIDS is that it plugs a monitoring ECU for the sole purpose of monitoring the CAN bus using remote frames (see Chapter 2.1.2.2). It periodically requests messages provided by all CAN nodes and measures their offset ratio and time intervals. They evaluated their approach to detecting three types of attacks: DoS, fuzzy, and impersonation attacks. For each attack type, they measured a different metric. For example, to detect DoS attacks, they measured the ratio of instant replies of remote frames, to detect fuzzy attacks, they used a correlation coefficient between offsets and time intervals, while average response time was used to detect impersonation attacks. A limitation of this approach is that if the number of unique CAN IDs is high, then it incurs extra burden on the network and consumes a large amount of bus bandwidth by frequently communicating request/reply frames for all CAN IDs. Moreover, a delay-based monitoring approach was proposed by Wang et al. [80]. The idea in this approach is to connect one monitor to the CAN bus through two wires. The approach comprises three main steps: measuring normal delay profiles by timestamping CAN messages, learning the threshold delay value for each CAN message ID, and the final step is the detection step based on the measured delay difference.

The above approaches consider only the CAN part as a subsystem of the whole in-vehicle network. In contrast, Waszecki et al. [81] studied the monitoring aspect of CAN in the overall E/E architecture. They proposed a monitoring algorithm based on network calculus and a leaky-bucket algorithm to compute the arrival curve, which is then used to detect message-based attacks that result in violation of the predefined communication parameters.

2.3.4.2 Physical Specification-based Solutions

Other approaches focus on monitoring the physical specifications of the network. For instance, Clock Offset based Intrusion Detection System (COIDS) [82]. As the name suggests, this approach is based on monitoring the clock offset of each ECU. It follows three main steps: first, a baseline of each ECU's normal clock profile is constructed using active learning such as in [83], second, to detect anomalies, the cumulative sum of deviations from the normal behaviour is derived using cumulative sum method [84], last, the exact time of the attack is specified using sequential change-point detection. As with OTIDS, they evaluated their proposed approach to detect DoS, fuzzy, and impersonation attacks. COIDS also requires plugging a monitoring ECU into the CAN bus to monitor the network.

Older approaches suggested the use of monitoring sensors [85], [86]. A number of sensors were introduced such as *formality sensor* that checks for formal correctness of each CAN message, *location sensor* to check if the message belongs to the correct domain, *type sensor* that checks for the message type correctness, *range sensor* that checks for the validity of data range in the payload, *frequency sensor*, to check if the periodicity of CAN messages are within the upper and lower bounds specified, *protocol sensor* which monitors the messages with respect to the protocol specifications, *plausibility sensor* to check for message semantics, within a single domain, and decide whether it is realistic or not, and *consistency sensor* that checks for message semantics across all domains.

A local outlier factor-based Intrusion Detection System (IDS) was proposed by Ning et al. [87] to detect spoofing attacks and one type of DoS attacks called *bus-off attacks* [88]. Their approach leverages the fact that the voltage waveform on the CAN bus is hard to fake since messages sent by different ECUs are assigned unique voltage waveforms. They used this property to detect the attacker node. They compared their approach with Support Vector Machine (SVM) algorithm and found that their approach achieved a better detection rate with a lower false detection rate.

Other approaches to detect spoofing attacks are [7] and [89]. In [7], authors proposed *CopyCAN*; an anti-spoofing IDS. The main concept of CopyCAN is that it uses the CAN fault confinement technique to determine when an ECU has been disconnected from the network because it enters a bus-off state. On the other hand, [89] used a sliding window with Recurrence Quantification Analysis (RQA) to detect spoofing attacks on CAN. They claimed that the benefit of their approach is that it does not require any processing of the CAN frame fields, including the arbitration and payload fields. Instead, it only needs CAN messages' arrival times.

2.3.4.3 Information Theory-based Solutions

Researchers also tried information theory-based monitoring approaches, such as [90]–[92]. Authors in [90] suggested measuring the entropy of the CAN network to construct the normal behaviour of the network. Then, any deviation from the normal behaviour is considered an anomaly. Optimised by sliding window, another entropy-based approach was proposed by Wu et al. in [91]. For the sliding window, they used a fixed number of CAN messages and adopted a heuristic algorithm based on simulated annealing to get the optimal sliding window parameters. After the IDs are extracted based on sliding window size, entropy analysis will take place. This is the first, offline, step. The second step is the online detection step, which determines attack messages with IDs that are not within the correct range of the calculated entropy value. Baldini [92] extended the approach in [91] by considering more types of attacks and entropy measures. Another information theory approach based on measuring the hamming distance was proposed in [93]. By calculating the hamming distance between subsequent payloads of the same message ID, they were able to detect fuzzy and replay attacks on CAN.

2.3.4.4 AI-based Solutions

With the contemporary advances in Artificial Intelligence (AI) and Machine Learning (ML), more researchers are focusing on employing these techniques to detect anomalies on the vehicle network [94]–[100]. Authors in [94] proposed *GIDS*; a Generative Adversarial Network (GAN)-based anomaly detection for CAN. The approach first converts CAN data into images, then, using generator and discriminator models, the generator generates fake images while the discriminator tries to distinguish these fake images. They used two discriminators to detect both known and unknown attacks. GIDS was tested on different types of attacks, including DoS and fuzzy attacks. A DNN-based IDS was proposed by Zhang et al. in [95] to detect replay and spoofing attacks on CAN. A transfer learning-based approach called *CANTransfer* was introduced by Tariq et al. in [96] where they used convolutional Long Short-Term Memory (LSTM) network to train with normal data and known attacks, then using one-shot transfer learning [101], they retrain the model to be able to detect new attacks. With transfer learning [102], they tackle the challenges of finding large datasets to train with.

Another deep learning-based approach using Convolutional Neural Network (CNN) was proposed by authors in [97]. They exploit the sequential nature of CAN messages to detect attacks on the CAN network. This approach could only detect attacks that the model was trained on, it could not detect new attacks unknown to the model. To tackle this issue, they proposed a self-supervised anomaly detection approach to detect attacks that the model has not seen before [98]. In [98], they also addressed the problem of not having enough anomalous datasets to train with, and they proposed to use normal data contaminated with some noise ratios. They generated such datasets using LSTM network. In addition, a domain adversarial neural network was used in [100] to detect attacks on CAN. This approach requires accessing the CAN message details, including all fields. However, they claim that their approach could detect variant attacks (of known attacks) that the model has not been trained on. A recent approach, [103], used Deep-SVDD (Support Vector Data Description) to obtain voltage fingerprints for each CAN ID, which is then used to detect malicious frames and determine their source. Different ML-based models were evaluated in [104] and it was found that the ones that use time-series data achieve better accuracy.

In addition, another monitoring solution for CAN is proposed in [105] where authors used n-gram analysis to detect three types of attacks on CAN: replay, fuzzing, and DoS attacks. However, their approach could not distinguish between single message replay and payload fuzzy attacks. In addition, it cannot detect Man-in-the-Middle (MitM) attacks where the attacker can have full control over the medium.

It is clear that all the aforementioned monitoring approaches focused on CAN networks and failed to consider other communication protocols used in the automotive networks, especially the newcomer that is expected to prevail in the automotive network, i.e., AE. The first study to consider automotive Ethernet is by Jeong et al. [99]. In particular, they focused on detecting injection attacks for Audio-Video Transport Protocol (AVTP) streams. Their approach utilised a machine learning algorithm based on CNN and it could detect continuous replay attacks with high recall value.

The aforementioned studies are mostly concerned with one or two aspects of the monitoring steps, i.e., anomaly detection and/or localisation. Anomaly mitigation, on the other hand, is ignored by these studies. A very limited number of studies consider anomaly mitigation, such as [106] and [55]. Kwon et al. [106] proposed a mitigation framework through ECU reconfiguration.

For instance, the ECU mode can be switched to safe mode. On the other hand, Bandur et al. [55] proposed a mitigation mechanism for centralised domain-based architecture. In particular, they proposed a decentralised mitigation approach where the functions implemented by the failed domain controller can be implemented by the ECU in the same domain. A brief summary of the monitoring solutions proposed for in-vehicle networks is shown in Table 2.4. For complete surveys on the topic, readers can refer to [107] and [16]. And to [108] for more AI-based IDSs.

Table 2.4: Summary of existing monitoring approaches. *AL* stands for *Anomaly Localisation* and *AM* stands for *Anomaly Mitigation*.

Approach	Method	AL	AM
OTIDS [79]	Time interval monitoring of remote frames	No	No
COIDS [82]	Clock offset monitoring	No	No
Wang et al. [80]	Delay-based monitoring	Yes	No
Müter et al. [85], [86]	Monitoring sensors	No	No
Ning et al. [87]	Voltage waveform	Yes	No
CopyCAN [7]	CAN fault confinement	Yes	No
G. Baldini [89]	RQA	No	No
Müter et al. [90]	Entropy-based monitoring	No	No
Wu et al. [91]	Sliding window entropy	No	No
G. Baldini [92]	Information-based entropy	No	No
Stabili et al. [93]	Hamming distance measure	No	No
Song et al. [76]	CAN message frequency	No	No
Moore et al. [77]	Inter-signal weight time	No	No
Young et al. [78]	Message rate and message interval	No	No
Seo et al. [94]	GAN	No	No
Zhang et al. [95]	DNN	Yes	No
CANTransfer [96]	Transfer learning with LSTM network	No	No
Song et al. [97]	CNN	No	No
Song et al. [98]	Self-supervised learning approach	No	No
Wei et al. [100]	Domain adversarial neural network	No	No

Table 2.4: Summary of existing monitoring approaches. *AL* stands for *Anomaly Localisation* and *AM* stands for *Anomaly Mitigation*.

Approach	Method	AL	AM
Deng et al. [103]	Deep-SVDD	Yes	No
Stabili et al. [105]	n-gram analysis	No	No
Waszecki et al. [81]	Network calculus	No	No
Kwon et al. [106]	ECU reconfiguration	No	Yes
Bandur et al. [55]	Enabling ECUs to implement DC functions	No	Yes

There are many limitations in the existing monitoring approaches for in-vehicle networks. First, they mostly only focused on a single communication protocol, mainly CAN, and ignored the fact that other protocols (such as AE) exist. Consequently, they fail to consider a very important factor, which is the cross-communication between different subsystems. Such communication also deserves a high level of monitoring solutions [109]. Third, most existing approaches rely heavily on ML-based algorithms, which neglect the time and resources needed to train such algorithms. In addition, it has been doubted that such algorithms would be suitable for in-vehicle networks [110] due to the fact that they still have some level of false positive/negative rates, which makes them even inapplicable for IDSs [111]. Finally, and more importantly, the existing solutions do not consider a complete monitoring solution where all three steps (detection, localisation, and mitigation) should have to be considered.

2.4 Network Tomography

Network tomography is a monitoring tool that can monitor the network and provide useful insights about its performance, traffic density, and topology. In the following, we further describe network tomography, its categories, measurement types, metrics and applications.

2.4.1 Background

The term *network tomography* was first popularised by Vardi [112] back in 1996. The naming emerged due to the similarity with tomography in medical imaging [13]. In general, network tomography can be defined as the process of inferring network parameters using measurement traffic sent along a limited subset of the network. Network tomography scope, so far, has been limited to large-scale computer networks and the Internet [13], [14], [113], [114].

There are three main components of network tomography: vector of end-to-end measurements $\boldsymbol{y}$, vector of link-level measurements $\boldsymbol{x}$, and measurement matrix $\boldsymbol{A}$ (see Table 2.5). Then network tomography problem is formulated as a system of linear equations as follows

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}. \tag{2.1}$$

Table 2.5: Chapter 2 notations and their descriptions.

Notation	Description
Network & Graph Theory	
$G = (V, E)$	Network represented as graph G
$V(G)$	Set of vertices (nodes) in network G
$E(G)$	Set of edges (links) in network G
$\mathcal{I}(G) \subset E(G)$	Set of internal links in G (see Definition 6.7.1)
$\mathcal{T}(G) \subset E(G)$	Set of external links in G (see Definition 6.7.1)
$v_iv_j \in E(G)$	A link connecting nodes v_i and v_j
$G + \{v_iv_j\}$	Adding link v_iv_j to network G
$d(v_i, v_j)$	Distance between nodes $v_i, v_j \in V(G)$
$\mathcal{C}(G)$	Set of graph components in G
C_n	A cycle with n nodes where $n \geq 3$
$v_h(e_i)$	Endpoint (head) of i^{th} link $e_i \in E(G)$
$v_t(e_i)$	Endpoint (tail) of i^{th} link $e_i \in E(G)$
Network Tomography	
$\boldsymbol{y}$	Vector of end-to-end measurements
$\boldsymbol{x}$	Vector of link-level measurements
$\boldsymbol{A}$	Measurement matrix
$y_{v_i,v_j} \in \boldsymbol{y}$	End-to-end (path-level) measurement between nodes v_i and v_j

Table 2.5: Chapter 2 notations and their descriptions.

Notation	Description
Network Tomography	
$x_i \in \boldsymbol{x}$	Link-level metric of link $e_i \in E(G)$
Deep Neural Network	
$\boldsymbol{\theta}$	Input vector
$\boldsymbol{\beta}$	Bias vector
$\boldsymbol{W}$	Matrix of weights
$\boldsymbol{h}_i$	Vector of i^{th} hidden layer
$f(z)$	Hidden layer activation function
$f_o(z)$	Output function
$\hat{\boldsymbol{y}}$	Vector of output layer
L	Loss function
α	Learning rate

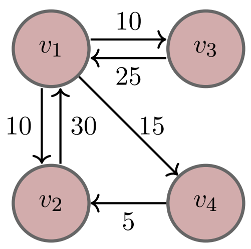
Depending on the problem at hand, either one (or more) of these three components is unknown, and the goal is to infer its value(s) given the known components.

2.4.2 Categories

Three different categories of network tomography have been studied in the literature based on the problem that network tomography is trying to solve. These categories are path-level inference, link-level inference, and topology inference. Below is a brief description of each one of these categories.

2.4.2.1 Path-level Inference

In this problem, the link-level measurements $\boldsymbol{x}$ (and likely the network topology) are known and used to infer the unknown path-level performance $\boldsymbol{y}$. Therefore, in this category, the link-level performance between internal network elements are measured to infer the path-level performance. Vardi's work [112] falls into this category, where the goal was to estimate the origin-destination traffic intensities between different end nodes in the network. An example of this category is shown in Figure 2.11. The network G shown on the left consists of four nodes $(v_1, \dots, v_4)$, traffic counts between these nodes are known and hence used in the table shown on the right to compute the origin-destination traffic matrix.

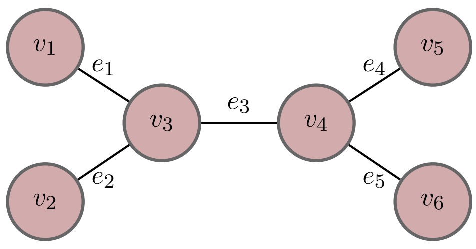


	v_1	v_2	v_3	v_4	y_i
v_1	0	10	10	15	35
v_2	30	0	0	0	30
v_3	25	0	0	0	25
v_4	0	5	0	0	5
y_j	55	15	10	15	95

Figure 2.11: Example of origin-destination traffic intensity inference.

2.4.2.2 Link-level Inference

This category often assumes that the measurement matrix $\boldsymbol{A}$ is known, as well as the path-level performance $\boldsymbol{y}$ along selected paths in the network. Then the goal is to find the link-level performance $\boldsymbol{x}$. Figure 2.12 shows an example of this category. The topology shown in this figure consists of six nodes $(v_1, \dots, v_6)$ and five links $(e_1, \dots, e_5)$. Let y_{v_i,v_j} be the end-to-end (path-level) measurements[11] of the path between v_i and v_j, then as shown in the figure, by measuring five paths, then solving the system in (2.1), the link-level measurement $x_i \in \boldsymbol{x}$ for each link e_i could be inferred. Multiple number of studies consider this problem such as [115]–[117], [20] and [118]. Similarly, this thesis uses this type of tomography given the constraints of in-vehicle networks (internal elements are inaccessible).



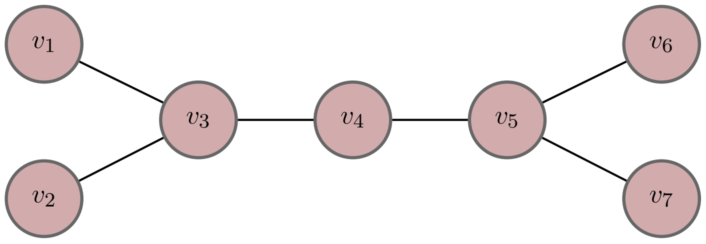
$$
\begin{aligned}
y_{v_1,v_2} &= 25 & \qquad x_1 &= 17.5\\
y_{v_1,v_5} &= 30 & \qquad x_2 &= 7.5\\
y_{v_2,v_5} &= 20 & \qquad x_3 &= 10\\
y_{v_2,v_6} &= 35 & \qquad x_4 &= 2.5\\
y_{v_5,v_6} &= 20 & \qquad x_5 &= 17.5
\end{aligned}
$$

Figure 2.12: Example of link-level inference.

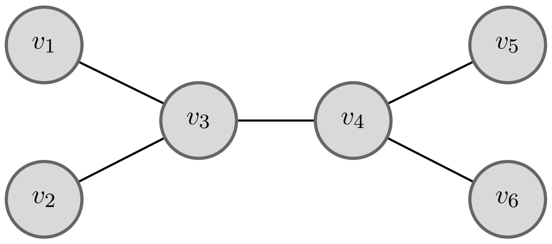
[11] Note that the terms *end-to-end measurements* and *path-level measurements* are used interchangeably in this thesis.

2.4.2.3 Topology Inference

Topology inference is another category of network problems that network tomography can be used to solve. This problem assumes that the network topology is unknown, and the goal is to infer it using network tomography. However, network tomography can only infer the *logical* topology of the network, where in some cases this logical topology does not match the physical topology as shown in the example illustrated in Figure 2.13. *Traceroute* can be used to infer the logical topology where the end-to-end performance, e.g, time-to-live, can be measured [119]. In addition to [119], other studies have also focused on this category of network tomography such as [120], [121], and [122].



(a) Physical topology



(b) Logical topology

Figure 2.13: Example of logical topology inference.

2.4.3 Measurement Types

As mentioned earlier, network tomography requires some sort of either end-to-end or link-level measurements (depending on the nature of the problem being addressed). These measurements can be taken in two ways: active or passive, or sometimes a combination of both. The following defines these two types of measurements.

2.4.3.1 Active Measurements

Active measurements use certain probe messages designed and sent between selected nodes for the purpose of monitoring the network. Examples of such measurements' techniques are *ping*, *pathchar* (pchar), *traceroute*, and *clink* [114]. This type requires a certain number of monitors[12] where the probes are sent between them. Most network tomography-related literature focuses on using this type of measurement [20], [117], [123]–[125].

2.4.3.2 Passive Measurements

In passive measurements, the existing traffic is measured by, for example, mirroring or sampling methods. Hence, unlike the case with active measurements, there is no need to design specific monitoring probes. In contrast to active measurements, fewer number of studies used passive measurements [126]–[129].

Each one of these types has its pros and cons. For example, active measurements require a certain number of monitors to ensure full identifiability of the network, in addition, such monitors need to be placed properly within the network. Moreover, inserting monitoring probes into the network might add extra burden and can affect the performance of original traffic [126]. On the positive side, however, with the proper design of probes and placement of monitors, this type can ensure full identifiability and give accurate results.

On the contrary, passive measurements eliminate the hustle of finding monitors and the process of placing them [130]. In addition, using existing traffic can provide more realistic and accurate measurements than the inserted probes, which, usually, behave differently to the actual traffic [126]. However, because passive measurements use existing traffic, the existing routing does not ensure full network identifiability. Moreover, different traffic in the network might use different network parameters, and thus cannot be used in network tomography to characterise the overall network performance. For instance, to measure the network delay, if existing traffic has different parameters, then the measurements will be different based on the used parameters, hence the inferred delay will not be accurate.

[12]A monitor is a node that participates in the monitoring process by sending and/or receiving monitoring messages.

2.4.4 Metrics

Different network monitoring techniques can be used to measure different parameters such as delay, packet loss/success rate, binary status, and bandwidth consumption. The network tomography model should specify which metric it measures. In the following, we focus on delay, loss/success rate, and binary status metrics by giving a brief description of each.

2.4.4.1 Delay

With network tomography, one can measure the delay of the network to infer the internal link-level delay or the end-to-end delay. The delay can be measured by a timestamp assigned to the probe message. For inferring the internal network delay performance, the probe message is timestamped at the source node and the receiving time at the receiving node is recorded. Numerous studies consider using network tomography to measure the network delay performance [131]–[142].

2.4.4.2 Packet Success/Loss Rate

Another metric that network tomography can measure is the success (or loss) packet rate. The success/loss rate metric is not additive like the delay metric, but it can be transformed into an additive metric by taking the natural logarithm of the measurements. Several studies used network tomography to measure the network packet success (or loss) rate [125], [143]–[145]. It is worth noting that packet success/loss metrics can greatly benefit from passive measurements if these measurements are independent and can fully identify the network. This is because, with this metric, only counts of the sent/received packets are needed, then the rate (or percentage) is computed, and these are not affected by the different network parameters used, unlike the case with the delay metric discussed above.

2.4.4.3 Binary Status

A type of network tomography called *binary tomography* measures the state of the network. The state is determined as either *good* (*normal*) or *bad* (*anomalous*) using binary values 0 and 1, respectively. The system in (2.1) then uses boolean matrix multiplication to determine the state of links, or paths, depending on the nature of the problem. This category usually requires using other metrics, e.g., the delay metric, as a way to characterise the status. The first to introduce this category was Duffield in [146]. Many other studies then further investigated and used binary tomography to determine faults or abnormal behaviour of the network [147]–[151]. Binary tomography can also be combined with other tomography approaches, such as delay or packet loss rate [152].

2.4.5 Solution Approaches for Network Tomography Problem

In order to solve the network tomography problem (2.1), different approaches can be used based on the type of problem and the availability of measurements and monitors. Broadly speaking, there are three main approaches investigated so far: algebraic, statistical, and machine learning-based approaches.

2.4.5.1 Algebraic Approaches

The typical algebraic approach [153] requires having a full-rank matrix $\boldsymbol{A}$ (see the following chapter for more details). Sometimes this is not possible, for instance, if some part of the network is inaccessible or cannot perform monitoring tasks. For this, other algebraic approaches can be adopted, such as compressed sensing [125], [154]–[156], and network coding [157]–[159].

2.4.5.2 Statistical Approaches

To tackle the problem of not having enough measurements, researchers are trying to solve network tomography using different statistical approaches [124] including the first tomography paper by Vardi [112] which utilised statistical moment methods and Expectation Maximization (EM)-based algorithm. Also, authors in [127] used the EM algorithm. Some other studies used Bayesian inference [160], [161], while others used different statistical approaches [134], [162], [163].

2.4.5.3 Machine Learning-based Approaches

Recently, like any other field, machine and deep learning made their way into the networking realm. This includes network tomography problems. One example is by Yang et al. [164]. In their study, they used artificial neural networks to solve network tomography in space optical networks. In addition, authors in [165] developed a neural network-based network tomography framework called *NeuTomography*. Similarly, [166] used a neural networks-based approach for network tomography in overlay networks based on Software-Defined Wide Area Network (SD-WAN). Moreover, the neural network has been used for dynamic routing and partial topology knowledge [167].

For critical systems such as in-vehicle networks, statistical approaches are not recommended since they do not provide exact deterministic values, instead, they give probabilistic values which are not always accurate [168]. Similarly, machine learning-based solutions are not efficient to be used for in-vehicle networks due to the high computational and memory requirements that in-vehicle devices with their resource-constrained capabilities do not provide. In addition, such solutions cannot completely eliminate false alerts, where having false positive or false negative results is intolerable for highly critical applications as is the case with in-vehicle networks. Therefore, algebraic approaches are preferable for such scenarios.

2.4.6 Applications

Network tomography has been utilised in different fields, mainly in computer networks and the Internet. Examples of specific applications that benefit from network tomography include traffic load balancing [169], [170], anomaly detection [171], [172], anomaly localisation [173], [174], and network slicing applications [166].

Beyond computer networks and the Internet, network tomography has been used in the transportation sector for urban traffic monitoring [175], smart cities' traffic monitoring [176], locating congestions in VANET [177], and for allocating power to EVs charging stations [178].

Differently, this thesis investigates network tomography as a monitoring application for in-vehicle networks.

2.5 Deep Neural Networks (DNNs)

Deep Neural Network (DNN) [179] is one branch of several machine learning algorithms. Generally, there are four types of learning: *supervised learning*, *unsupervised learning*, *semi-supervised learning*, and *reinforcement learning* [180]. In supervised learning, the dataset used for training is labelled. In contrast, unsupervised learning uses unlabelled datasets, whereas semi-supervised learning uses a combination of both supervised and unsupervised learning, where the datasets used in this type include labelled and unlabelled entries. Reinforcement learning [181] is different from the other types, where the idea is to deploy an agent in an environment so that it explores and learns from its actions. The agent then learns by either receiving rewards or punishments based on the correctness of the action it takes.

The DNN used in this thesis is based on supervised learning. As shown in Figure 2.14, the neural network structure consists of three main layers [179], [182]: *input layer*, *hidden layer*, and *output layer*. Each layer consists of a certain number of nodes called *neurons*. If there is more than one hidden layer, then the network is called *a deep neural network* [179], [182].

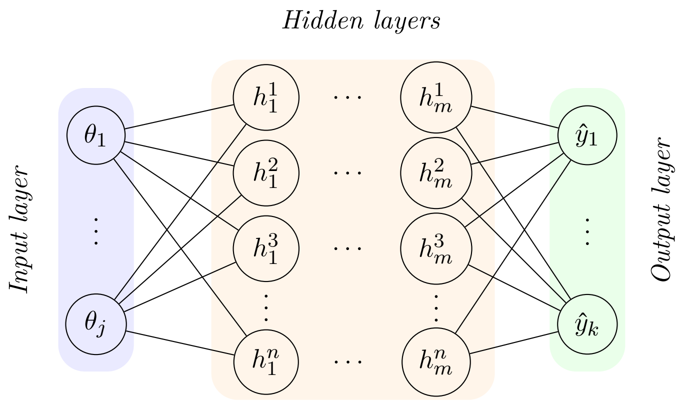


Figure 2.14: Deep neural network structure.

Training the neural network is done in two steps: *feed forward* and *backpropagation* [182], [183]. The description of each step is explained in the subsequent sections.

2.5.1 Feedforward

In the feedforward step, the input vector $\boldsymbol{\theta}$ is fed to the first layer of the neural network (see Figure 2.14). At the first hidden layer, the input will be multiplied by a weight matrix $\boldsymbol{W}$ and added to a bias value $\beta_1 \in \boldsymbol{\beta}$. The result then will be fired by a function $f(z)$ called *an activation function*. The activation function usually used for the hidden layers is *Rectified Linear Unit (ReLU)* [184]. The role of this function is to introduce some nonlinearity to the input. ReLU is calculated as

$$f(z) = \max(z, 0). \tag{2.2}$$

The output of the first hidden layer then will be a vector $\boldsymbol{h}_i$, where $i = 1$, and is computed as

$$\boldsymbol{h}_1 = f\big(\boldsymbol{W}_1^T \boldsymbol{\theta} + \beta_1\big). \tag{2.3}$$

Similarly, the output of the first hidden layer $\boldsymbol{h}_1$ will be feed-forwarded to the subsequent hidden layers. The output of any subsequent hidden layer $\boldsymbol{h}_{i+1}$ then will be

$$\begin{gathered}\boldsymbol{h}_{i+1} = f(\boldsymbol{W}_{i+1}^T \boldsymbol{h}_i + \beta_i), \\ \forall i \in \{1, \ldots, m-1\}.\end{gathered} \tag{2.4}$$

where m is the number of hidden layers in the neural network.

This process continues in the same fashion until the output layer. At the output layer, usually, a different activation function $f_o(z)$ is used depending on the type of problem [185]. For instance, a *sigmoid* function [186] is often used in binary classification problems [187], while the *softmax* function [188] is used for multi-classification problems [189], and the *linear* function [185] is usually used for regression problems [190].

The output layer then produces a vector of estimated values $\hat{\boldsymbol{y}}$ as

$$\hat{\boldsymbol{y}} = f_o(\boldsymbol{W}_{m+1}^T \boldsymbol{h}_m + \beta_m). \tag{2.5}$$

Next, the backpropagation step will take place as explained next.

2.5.2 Backpropagation

At the output layer, the estimated value will be compared against the actual value using a loss function L. Such a function is used to compute the error value between the actual and the estimated values. There are many loss functions that one can use [191]. The well-known one is Mean-Squared Error (MSE) [179] and it is computed as

$$L = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2, \tag{2.6}$$

where N is the number of samples in a batch, as the neural network is trained on mini-batches [192]. Compared with Stochastic Gradient Descent (SGD) [193] and Batch Gradient Descent (BGD) [194], Mini-Batch Gradient Descent (MBGD) is more efficient [195]. In SGD, only one entry of the dataset is used to update the model weights. This is considered computationally expensive. On the other hand, in BGD, the whole dataset is used. This is computationally efficient, however, it suffers from poor performance due to under-fitting issues. MBGD uses a combination of both, hence it is more appropriate in terms of both performance accuracy and computational consumption [194], [195] .

At the backpropagation step, the goal is to minimise (2.6) so that the estimated value is close to the actual value [183]. To achieve this goal, the error will be backpropagated into the previous layers in order to update the weights so that the loss function L is minimised. This is done using gradient descent that takes the partial derivative of the loss function with respect to each weight. In particular, for each weight w_{ij} between neurons i and j, calculate $\frac{\partial L}{\partial w_{ij}}$ [182]. For a batch of size N, the partial derivative of L with respect to w_{ij} is

$$\frac{\partial L}{\partial w_{ij}} = \frac{1}{N}\sum_{n=1}^{N}\frac{\partial L_n}{\partial w_{ij}}. \tag{2.7}$$

With learning rate α, the updated weight w_{ij}^t at iteration t then will be

$$w_{ij}^t = w_{ij}^{t-1} - \alpha\frac{\partial L}{\partial w_{ij}^{t-1}} \quad . \tag{2.8}$$

Training DNN goes through multiple iterations, where each iteration follows the above two steps (feedforward and backpropagation) [179], [182].

In this thesis, we leverage DNN to tackle the problem of rank-deficient matrix $\boldsymbol{A}$.

2.6 Graph Theory Preliminaries

Graph theory is a mathematical branch of studying graphs which represent networks of connected objects [196], [197]. Graph theory has been useful for representing different real-world phenomena and structures. This includes communication networks. Similarly, graph theory is used to represent and model the in-vehicle networks studied throughout this thesis.

For the sake of clarity, the following graph properties are necessary to define (see Table 2.5 for the notations' descriptions):

- **Graph connectivity:** A graph G is **connected** if there is at least one path connecting any two vertices in G. In other words, $d(v_i, v_j) \neq \infty$ for all $v_i, v_j \in V(G)$, where $d(v_i, v_j)$ is the distance between v_i and v_j. If, for some $v_i, v_j \in V(G)$, $d(v_i, v_j) = \infty$, then G is **disconnected,** where each connected subgraph is called **component** and in this case $\mathcal{C}(G) \neq \emptyset$, where $\mathcal{C}(G)$ is the set of components in disconnected graph G.

- **Mutligraph:** A graph G is multigraph if it contains more than one link between some nodes $v_i, v_j \in V(G)$.
- **Cyclic graph:** A connected graph G is called a cyclic graph if it contains at least one cycle C_n, where $n := \Big|V(C)\Big|$ and $n \geq 3$. A path connecting such cycle contains n edges with $e_1 = v_1v_2$, $e_2 = v_2v_3$, $e_{n-1} = v_{n-1}v_n$ and $e_n = v_nv_1$.
- **Self-loop:** A graph G has self-loop when there is a link $e_i \in E(G)$ with $v_h(e_i) = v_t(e_i)$, where $v_h(e_i)$ and $v_t(e_i)$ represent the endpoints of the link $e_i \in E(G)$.
- **Simple graph:** A graph G that has no self-loops and has at most one link between any two nodes in $V(G)$ is called a simple graph.
- **Acyclic graph:** A connected graph G is called an acyclic graph if it is simple and does not contain any cycle. It is also called **maximally acyclic** graph because $G + \{v_iv_j\}$ results in a cycle for any two non-adjacent vertices $v_i, v_j \in V(G)$.
- **k-edge-connected graph:** A graph G is k-edge-connected if it stays connected whenever any fewer than k links are removed.
- **Simple paths:** Simple paths are paths that do not traverse repeating nodes or edges.
- **Disjoint paths:** These are paths that do not have common links, i.e., we say that two paths p_i and p_j are disjoint if $p_i \cap p_j = \emptyset$.
- **Internally disjoint paths:** These are paths that do not have common internal links, i.e., we say that two paths p_i and p_j are internally disjoint if $p_i \cap p_j \not\subseteq \mathcal{I}(G)$ and $p_i \cap p_j = \{e_k, e_l\}$ where both $e_k, e_l \in \mathcal{T}(G)$. $\mathcal{I}(G) \subset E(G)$ and $\mathcal{T}(G) \subseteq E(G)$ are sets of internal and external links, respectively (see Definition 6.7.1).

2.7 Tools Used for Implementation and Analysis

The research experiments implemented in this thesis are all based on simulations. Several simulation and implementation tools, such as OMNeT++, Mininet, MATLAB, and Ryu SDN controller, have been used. In the following, we give a brief background of these tools.

2.7.1 OMNeT++

OMNeT++ [198] is an open-source discrete event-based simulator that is primarily used for designing and modelling networked systems. It was developed to help the academic and research communities to have a powerful simulation tool that can be used to model, test, and evaluate different systems. OMNeT++ can be used on all main platforms: MacOS, Linux, and Windows. Several features of OMNeT++ make it stand out among other simulation tools. For instance, it supports the reusability of components. This feature eliminates the need to implement new modules, instead, one module can be implemented once and used in different scenarios. Another feature is the support of topology description language called *NED language*. With NED, the user can model the structure of the network with all the involved components and interactions between them. Further, NED files can be translated into XML format and vice versa. Additionally, it has powerful visualisation and debugging capabilities available in its own Integrated Development Environment (IDE). The editor supports two-way model representation so that the user can edit the network in either a graphical or source code view. Simulations with OMNeT++ are highly scalable and one can experiment with different input parameters to represent the different behaviours of the system. Furthermore, result analysis and result visualisation are other built-in features of OMNeT++ with the ability to use different data processing and filtering steps. Parallel and batch simulation supports are further characteristics of OMNeT++ which can facilitate speeding up the simulation process.

This thesis uses OMNeT++ to simulate different in-vehicle networking architectures and study and evaluate the proposed monitoring approach on such architectures.

2.7.2 Mininet

Mininet is a virtual network emulator for emulating SDN networks [199]. Mininet supports six attractive characteristics of designing and developing any SDN network. First, it is flexible in that new network topologies and features can be implemented in software. Second, Mininet source code developed for a network prototype can be easily used in a real network without any modifications. Third, it supports interactivity with real-time management of the emulated network. Fourth, it is scalable, making it possible to add hundreds to thousands of network elements. Fifth, it mimics realistic real-time network environments with application and protocol stacks to be available to use. Finally, network prototypes developed in Mininet can be shared between collaborators and researchers, allowing further experimentation and alteration of the prototype.

In this thesis, Mininet is used to emulate SDN-enabled in-vehicle networks with Ryu as the SDN controller (see next).

2.7.3 Ryu SDN Controller

Ryu is a component-based SDN controller [200]. It supports different predefined software components where each component can be modified, extended, and composed to develop new controller applications. Besides OpenFlow protocol, Ryu provides support for other southbound protocols such as Netconf and OF-config. Ryu is publicly available to use under the Apache 2.0 license. Although other SDN controllers are also available (e.g., POX, FloodLight, and OpenDaylight), Ryu is the best controller [201], [202], especially in terms of latency requirements [202] which makes it the ideal option to use for delay-sensitive applications like the ones in vehicular communications.

Ryu controller is used in this thesis as the SDN controller for the SDN-enabled in-vehicle network.

2.7.4 MATLAB

MATLAB (MATrix LABoratory) [203] is a powerful programming and computing tool that is used by researchers and instructors. It includes a computation, programming, and visualisation environment with advanced data structures and a built-in editor and debugger. It has different toolboxes for applied science and engineering such as signal processing, symbolic computation, control theory, optimisation, and simulation.

MATLAB, in this thesis, is used for network tomography computations as well as to simulate different network topologies and to implement a number of the proposed algorithms.

Additionally, for data analysis and deep learning models, tools such as Pandas, Keras and TensorFlow have been used.

2.7.5 Pandas

This is a Python library for data analysis and manipulations [204]. Pandas provides several functionalities that are important for analysing different types of data. For example, it enables data access using labels in addition to the traditional method with integer-based indexing. It also allows slicing and extracting data with the same access methods (either using integer-index or label-index data access). Automatic alignment of data, e.g., time series datasets is another functionality that Pandas support in addition to supporting different ways of handling missing data from the dataset (e.g., dropping or replacing missing data entries). Other functionalities of Pandas include hierarchical indexing, pivoting, reshaping, grouping and aggregating data based on user-defined criteria, as well as combining and joining different datasets. Additionally, it allows integration with other libraries such as Matplotlib [205] for producing data plots.

2.7.6 TensorFlow and Keras

TensorFlow is another Python library for machine and deep learning [205]. It is mainly used to build and train machine and deep learning models. Further, it supports different system platforms including Windows, Mac, and Linux.

Keras is a higher layer of TensorFlow that provides high-level building blocks to facilitate the process of creating any deep neural network model [206]. Keras supports many key features such as the ability to run the same code on CPU or GPU, the support for different neural networks like convolutional neural networks, and recurrent neural networks, as well as a combination of both. With Keras, one can build any network structure with multi-input and multi-output models, layer sharing, model sharing, etc.

2.8 Summary

In this chapter, we first discussed vehicular communications in general, with a focus on in-vehicle networks. We described the well-known communication protocols used in automotive networks, covering a detailed description of CAN and AE. Next, different in-vehicle networking architectures based on E/E architectures were described, starting from the first E/E architecture, fieldbus, to the next-generation centralised zonal-based architecture. In-vehicle networks' limitations related to security were then discussed, in addition to the challenges faced in monitoring the network. In particular, attack surfaces on in-vehicle networks were highlighted with brief descriptions of different attack types that can target in-vehicle networks, in addition to defining the main three steps of a typical monitoring system: anomaly detection, localisation, and mitigation. A detailed discussion of current research efforts related to monitoring the in-vehicle networks was also mentioned in this chapter. Moreover, the concept of network tomography was introduced in addition to a description of its different categories, measurement types, metrics, solutions approaches as well as different applications that can benefit from it. DNN's structure and training have also been described. In addition, some of the graph theory preliminaries have been stated as they are necessary for the understanding of the in-vehicle networks analysis and modelling. Finally, a brief background about the implementation, simulation, and analysis tools used in this thesis was highlighted.

The following chapter introduces network tomography, for the first time, as a monitoring approach for in-vehicle networks.

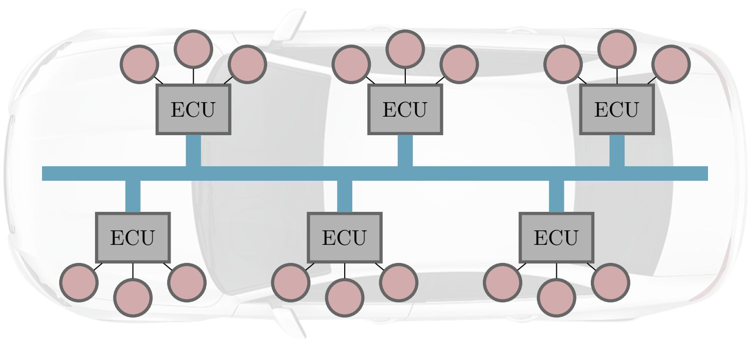
3 In-Vehicle Network Tomography

"The best way to predict the future is to create it."

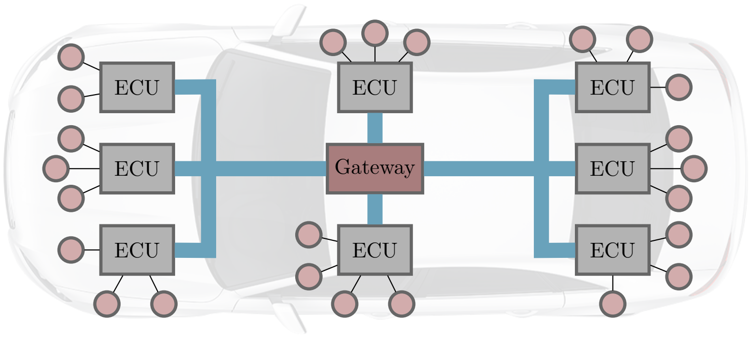
Abraham Lincoln

3.1 Overview

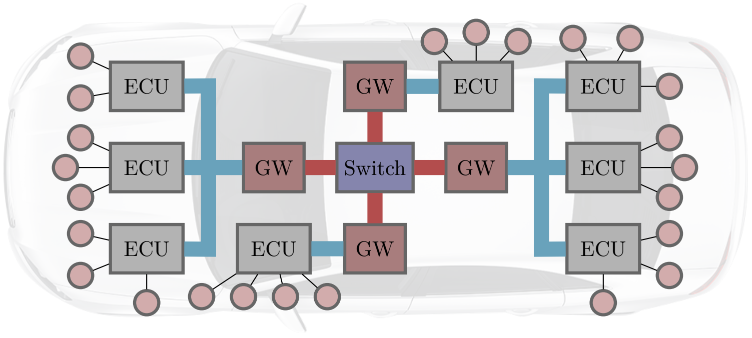
Network tomography is a powerful monitoring tool that can be used to infer network performance by monitoring just a small subset of the whole network. In this chapter, we study the applicability of this approach in in-vehicle networks by considering different E/E architectures. In particular, we investigate the use of network tomography in an in-vehicle network by analysing network identifiability of three main architectures: fieldbus, central-gateway, and Ethernet-based architectures (see Figure 3.1).



(a) Fieldbus architecture



(b) Central-gateway architecture



(c) Ethernet-based architecture

🟦 CAN bus 🟥 Ethernet ● Sensor/Actuator

Figure 3.1: Three main E/E architectures investigated for network tomography application. ***GW*** stands for gateway.

3.2 Motivation

Although Controller Area Network (CAN) [8] is the most dominant communication protocol used in the automotive industry, it suffers from two major shortcomings. First, it lacks authentication and authorisation mechanisms which negatively affect network security [11]. Second, it cannot meet today's high bandwidth demands imposed by new E/E architectures and ADAS applications [48]. To tackle the first issue, a central-gateway architecture was introduced to separate different subsystems and regulate traffic at the gateway [48]. On the other hand, to compensate for the limited bandwidth in CAN, Ethernet has become an essential part of the in-vehicle network, mainly as a backbone to connect different system domains [207], and for bandwidth-hungry applications.

The new E/E architectures result in more closed-in vehicle networks, which lead to difficulty in direct monitoring of the internal network's components. This issue can be addressed by investigating new solutions based on monitoring the network from the edges, where no participation is required from the internal components. Such solutions need not only to provide end-to-end measurements but should also be able to infer the internal network performance. To this end, we propose, for the first time in literature, to employ network tomography as a monitoring approach for in-vehicle networks.

Network tomography is one of the network monitoring approaches that is based on mathematical modelling of the network and its performance metrics. It was first studied by Vardi [112] to estimate the origin-destination traffic matrix. As mentioned in the last chapter, network tomography can be divided into three categories: (i) link-level parameter estimation, (ii) origin-destination traffic matrix estimation, and (iii) topology inference. Because it is difficult to access the internal elements of in-vehicle networks, we focus on the first category, where the end-to-end measurements (*path-level*) are used to infer the metric of *link-level* internal performance.

3.3 System Model and Problem Formulation

3.3.1 In-Vehicle Network Model

In-vehicle network topology is assumed to be known as it is naturally a fixed topology that does not change or can easily be modified. Graph theory conventions defined in [196] are followed in this thesis to represent the in-vehicle network and its characteristics. Table 3.1 shows the notations used in this chapter and their descriptions.

Table 3.1: Chapter 3 notations and their descriptions.

Notation	Description
Network & Graph Theory	
$G = (V, E)$	Network represented as graph G
$V(G)$	Set of vertices (nodes) in network G
$E(G)$	Set of edges (links) in network G
$\mathcal{E}(G) \subset V(G)$	Set of edge nodes in G (see Definition 3.3.1)
$\mathcal{E}_m(G) \subset \mathcal{E}(G)$	Set of monitoring edge nodes in G
$\mathcal{R}(G) \subset V(G)$	Set of intermediate nodes in G (see Definition 3.3.1)
$\mathcal{P}(G)$	Set of all possible paths in network G
$\mathcal{P}_m(G) \subseteq \mathcal{P}(G)$	Set of measured paths in network G
$B(G)$	Set of CAN buses in G
$d(v_i)$	Degree of node $v_i \in V(G)$
$p_{v_i,v_j} \in \mathcal{P}(G)$	Path connecting two non-adjacent nodes $v_i, v_j \in \mathcal{E}(G)$
$p_i \in \mathcal{P}(G)$	i^{th} path connecting two non-adjacent nodes $v_i, v_j \in \mathcal{E}(G)$
s_{v_i,v_j}	Segment connecting two non-adjacent nodes $v_i, v_j \in V(G)$ with $v_i \vee v_j \in \mathcal{R}(G)$
$e_i \in E(G)$	i^{th} link in network G
$v_iv_j \in E(G)$	A link connecting nodes v_i and v_j
$b_i \in B(G)$	The i^{th} CAN bus in G
Network Tomography	
$\boldsymbol{y}$	Vector of end-to-end measurements
$\boldsymbol{x}$	Vector of link-level measurements
$\boldsymbol{A}$	Measurement matrix
$y_i \in \boldsymbol{y}$	End-to-end measurement of i^{th} path $p_i \in \mathcal{P}_m(G)$
$x_i \in \boldsymbol{x}$	Link-level metric of i^{th} link $e_i \in E(G)$
$a_{ij} \in \{0, 1\}$	The element of the measurement matrix $\boldsymbol{A}$ at the i^{th} row (for $p_i \in \mathcal{P}_m(G)$) and j^{th} column (for $e_j \in E(G)$)
$r(\boldsymbol{A})$	Rank of the measurement matrix $\boldsymbol{A}$

Table 3.1: Chapter 3 notations and their descriptions.

Notation	Description
Numbers & Cardinalities	
$\eta_G := \Big\lvert V(G) \Big\rvert$	Total number of nodes in network G
$\gamma := \Big\lvert E(G) \Big\rvert$	Total number of links in network G
$\kappa := \Big\lvert \mathcal{P}_m(G) \Big\rvert$	Total number of measured paths in G
l_G	Number of uniquely identifiable links in G
$c_i \in \mathbb{Z}^+$	Priority level of CAN node $v_i \in \mathcal{E}(G)$

The in-vehicle network is mapped into a graph $G = (V, E)$ with a pair of two sets, $V(G)$ and $E(G)$ as sets of vertices (nodes) and edges (links)[1], respectively. Nodes in $V(G)$ are interconnected through the set of links $E(G)$ with each link $e_i = v_i v_j$ connecting two *adjacent* nodes $v_i, v_j \in V(G)$.

Based on node degree, $d(v_i)$, which is defined as the number of links node $v_i \in V(G)$ is incident to, the following definition further defines two sets of nodes.

Definition 3.3.1. Given an in-vehicle network G, sets of edge nodes $\mathcal{E}(G) \subset V(G)$ and intermediate nodes $\mathcal{R}(G) \subset V(G)$ are defined as[2]

- $\mathcal{E}(G) := \{v_i \in V(G) : d(v_i) = 1\}$ and
- $\mathcal{R}(G) := \{v_i \in V(G) : d(v_i) \geq 2\}$,

where $\mathcal{E}(G) \cup \mathcal{R}(G) = V(G)$ and $\mathcal{E}(G) \cap \mathcal{R}(G) = \emptyset$.

Let $\eta_G := \Big\lvert V(G) \Big\rvert$ be the total number of nodes in G, and let $p_{v_i,v_j} \in \mathcal{P}(G)$ be a path between any two edge nodes $v_i, v_j \in \mathcal{E}(G)$, and it is represented as a set of links such path traverses[3]. Further, let $\mathcal{P}(G)$ be the set of all possible paths and let $\mathcal{P}_m(G) \subseteq \mathcal{P}(G)$ be a set of measured (*monitored*) paths in the in-vehicle network G. Note that paths in $\mathcal{P}(G)$ are *simple* paths, they do not include cycles with repeating nodes. A segment s_{v_i,v_j} connects two *non-adjacent* nodes, $v_i, v_j \in V(G)$, where at least one of these nodes is an intermediate node, $v_i \vee v_j \in \mathcal{R}(G)$.

[1]In this thesis, we use the terms *vertices* and *nodes*, *edges* and *links* interchangeably.
[2]When the context is clear, we sometimes drop the network G and simply say $\mathcal{E}$, $\mathcal{R}$, etc.
[3]When suitable, we sometimes refer to $p_{v_i,v_j} \in \mathcal{P}(G)$ as simply $p_i \in \mathcal{P}(G)$.

Example 3.3.1. *Figure 3.2 shows an example of an in-vehicle network mapped into a graph* $G = (V, E)$ *with the sets of nodes* $V(G) = \{v_1, v_2, \ldots, v_{10}\}$ *and links* $E(G) = \{e_1, e_2, \ldots, e_9\}$*. In this example,* $\mathcal{E}(G) = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_{10}\}$ *and* $\mathcal{R}(G) = \{v_8, v_9\}$*. The set of all paths in this example is*

$$\begin{aligned}\mathcal{P}(G) =\{&p_{v_1,v_2}, p_{v_1,v_3}, p_{v_1,v_4}, p_{v_1,v_5},\\ &p_{v_1,v_6}, p_{v_1,v_7}, p_{v_1,v_{10}}, p_{v_2,v_3},\\ &p_{v_2,v_4}, p_{v_2,v_5}, p_{v_2,v_6}, p_{v_2,v_7},\\ &p_{v_2,v_{10}}, p_{v_3,v_4}, p_{v_3,v_5}, p_{v_3,v_6},\\ &p_{v_3,v_7}, p_{v_3,v_{10}}, p_{v_4,v_5}, p_{v_4,v_6},\\ &p_{v_4,v_7}, p_{v_4,v_{10}}, p_{v_5,v_6}, p_{v_5,v_7},\\ &p_{v_5,v_{10}}, p_{v_6,v_7}, p_{v_6,v_{10}}, p_{v_7,v_{10}}\}.\end{aligned}$$

Examples of segments in this network include s_{v_1,v_8}*,* s_{v_7,v_8} *and* $s_{v_9,v_{10}}$*.*

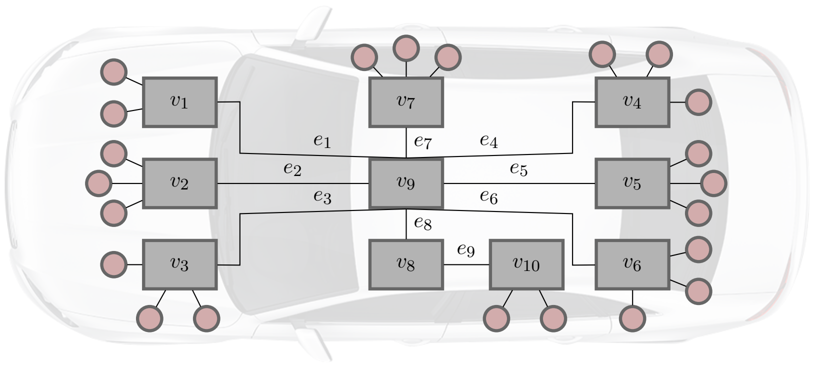


Figure 3.2: Example of an in-vehicle network model.

The in-vehicle network model discussed above can represent any in-vehicle network that is based on either central-gateway or Ethernet-based architecture with a set of CAN buses $B(G)$ of at least two, i.e., $\left|B(G)\right| \geq 2$. For a single CAN (i.e., fieldbus architecture), on the other hand, the network is represented as $G = (V, \{b_i\})$, where $b_i \in B(G)$ is the CAN bus, and since it is a single CAN, $i = 1$. Note that in this case, $\mathcal{E}(G) = V(G)$ and $\mathcal{R}(G) = \emptyset$.

3.3.2 Problem Statement

Given an in-vehicle network G, our objective is to determine the network performance by monitoring G. The issue with G is that it is a closed-in system where it is difficult to access the internal network components, i.e., links in $E(G)$ and intermediate nodes in $\mathcal{R}(G)$. This limitation makes it hard to directly monitor every single part of the network.

Let $\mathcal{E}_m(G) \subseteq \mathcal{E}(G)$ be a subset of nodes that can be directly accessed, and let $V(G) \backslash \mathcal{E}_m(G)$ be the set of remaining nodes. Each monitoring node $v_i \in \mathcal{E}_m(G)$ provides its end-to-end measurement value $y_i \in \boldsymbol{y}$. Then, the aim is to infer the performance of $e_i, \forall e_i \in E(G)$ given only the measurements set $\boldsymbol{y}$.

3.4 Network Tomography for In-Vehicle Networks

This section discusses network tomography for in-vehicle networks, analyses its requirements and derives the necessary conditions needed for network tomography to be applied in in-vehicle network scenarios.

3.4.1 Network Tomography Problem Formulation

Assuming that a path $p_i \in \mathcal{P}_m(G)$, $i \in \{1, \ldots, \left|\mathcal{P}_m(G)\right|\}$ is measured, then the measurement for such path is represented by $y_i \in \boldsymbol{y}$. Such measurement can be the end-to-end delay, packet loss rate, or any other metric discussed in Chapter 2.4.4. Let the set of all measurements of paths in $\mathcal{P}_m(G) \subseteq \mathcal{P}(G)$ be $\boldsymbol{y}$ and let $\kappa_G := \left|\mathcal{P}_m(G)\right|$ be the total number of measured paths. For each link $e_i \in E(G), i \in \{1, \ldots, \left|E(G)\right|\}$, let its measurement be $x_i \in \boldsymbol{x}$, then the set of measurements for all links in $E(G)$ is represented by $\boldsymbol{x}$, and let $\gamma_G := \left|E(G)\right|$ be the total number of links in network[4] G. The system of network tomography therefore can be formulated as

$$\boldsymbol{y} = \boldsymbol{A} \otimes \boldsymbol{x}, \tag{3.1}$$

where $\boldsymbol{y} = [y_1, y_2, \ldots, y_{\kappa_G}]^T$ is a vector in $\mathbb{R}^\kappa$ of path-level measurements, $\boldsymbol{A}$ is a $\kappa_G \times \gamma_G$ measurement matrix, and $\boldsymbol{x} = [x_1, x_2, \ldots, x_{\gamma_G}]^T$ is a vector in $\mathbb{R}^\gamma$ of link-level metrics. Entries of the measurement matrix $\boldsymbol{A}$ can be either binary values if the routing in the in-vehicle network is deterministic, or probabilities in case the routing is non-deterministic. Mostly in this thesis, we assume that the in-vehicle network G supports only deterministic routing with a single path in use between any nodes in the network, therefore the entries of $\boldsymbol{A}$ are binary values with $a_{ij} = 1$ if path p_i traverses link e_j, and $a_{ij} = 0$ otherwise. The operation $\otimes$ depends on the

[4]When suitable, we drop the network subscript G and simply say γ, κ, η, etc.

problem type. If the problem is additive (e.g., delay or packet success/loss rate tomography) then $\otimes$ is for matrix multiplication. For boolean problems such as binary tomography, $\otimes$ is boolean matrix multiplication, i.e., $y_i = \vee_j (a_{ij} \wedge x_j)$.

3.4.2 Network Identifiability

In network tomography, the term "*identifiability*" [208] determines the applicability of algebraic tomography and, hence, is worth defining.

Definition 3.4.1. A link $e_i \in E(G)$, $i \in \{1, 2, \ldots, \gamma_G\}$ is *identifiable* if its associated metric, $x_i \in \boldsymbol{x}$, can be uniquely determined from the end-to-end, path-level, measurements $\boldsymbol{y}$ by solving the system in (3.1) for $\boldsymbol{x}$.

Definition 3.4.2. Let l_G be the number of uniquely identifiable links in G, then based on the above definition, we classify network-wide identifiability into three main levels:

- **Fully-identifiable network**: an in-vehicle network G is *fully-identifiable* if the link-level metrics for *all* links in $E(G)$ are uniquely determined by solving (3.1). In this case, $l_G = \gamma_G$, hence, a fully-identifiable network can also be called γ_G*-identifiable network*.
- l_G**-identifiable network**: we say that an in-vehicle network G is l_G*-identifiable* if the *maximum* number of links that can be uniquely identified is l_G, where $l_G < \gamma_G$.
- **Unidentifiable network**: if *no* link metrics for any link in $E(G)$ can be uniquely determined by solving (3.1), then we say that G is *unidentifiable*. In this case $l_G = 0$.

Two main factors contribute to determining network identifiability level. First is the number of measurements, and second is whether the measurements are linearly independent or not. Below is the definition of linearly independent measurements.

Definition 3.4.3. A set of measurement paths $\mathcal{P}_m(G)$ is *linearly independent* if none of its elements is a linear combination of others. Otherwise, if at least one path $p_i \in \mathcal{P}_m(G)$ is a linear combination of one or more other paths, then the measurement paths are *linearly dependent*.

In order to have a fully-identifiable network G, the measurement matrix $\boldsymbol{A}$ should be a full-rank matrix with $r(\boldsymbol{A}) = \gamma_G$.

Remark 3.4.1. To achieve a full-rank measurement matrix $\boldsymbol{A}$, the following conditions need to be satisfied:

1. The number of end-to-end measurements is equal to the number of links in the network. In other words, $\kappa_G = \gamma_G$.

2. All the available measurements are linearly independent (see Definition 3.4.3).

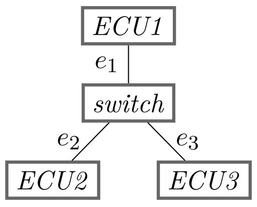
Assuming G is symmetric, then the maximum number of *distinct* paths between edge nodes in $\mathcal{E}(G)$ can be computed by

$$\left|\mathcal{P}(G)\right| = \binom{\left|\mathcal{E}(G)\right|}{2} = \frac{\left|\mathcal{E}(G)\right|\left(\left|\mathcal{E}(G)\right| - 1\right)}{2}. \tag{3.2}$$

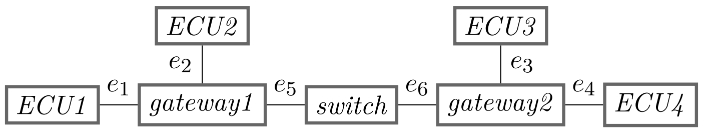
The rank of measurement matrix $\boldsymbol{A}$, $r(\boldsymbol{A})$, is affected by two elements: the network topology and the paths used in $\mathcal{P}_m(G)$. The latter depends on the number of monitors and their placement. These elements are further discussed below with a discussion of identifiability in in-vehicle networks.

3.4.2.1 Identifiability based on Topology

As will be shown in the following examples, satisfying $r(\boldsymbol{A}) = \gamma_G$ and hence, having fully identifiable network G highly depends on the network topology.



(a) Topology example forming full-rank measurement matrix



(b) Topology example forming rank-deficient measurement matrix

Figure 3.3: In-vehicle network topologies forming full-rank vs rank-deficient measurement matrices.

Example 3.4.1. *Consider the network topology shown in Figure 3.3(a) with* $\gamma_G = 3$*. Let* $\mathcal{P}_m(G) = \{p_1, p_2, p_3\}$ *where* $p_1 = \{e_1, e_2\}$*,* $p_2 = \{e_1, e_3\}$*, and* $p_3 = \{e_2, e_3\}$*. Then the measurement matrix will be*

$$\boldsymbol{A} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}. \tag{3.3}$$

The matrix in (3.3) *has* $r(\boldsymbol{A}) = 3$ *and thus, it is a full-rank matrix (because* $r(\boldsymbol{A}) = \gamma_G$*). This matrix can be used to solve* (3.1) *to uniquely identify* $x_i \in \boldsymbol{x}, \forall e_i \in E(G)$*.*

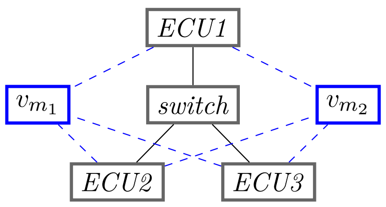
Example 3.4.2. *Now consider the topology shown in Figure 3.3(b). This topology has* $\gamma_G = 6$ *links. Let* $\mathcal{P}_m(G) = \{p_1, p_2, p_3, p_4, p_5, p_6\}$ *where* $p_1 = \{e_1, e_2\}$*,* $p_2 = \{e_1, e_5, e_6, e_3\}$*,* $p_3 = \{e_1, e_5, e_6, e_4\}$*,* $p_4 = \{e_2, e_5, e_6, e_3\}$*,* $p_5 = \{e_2, e_5, e_6, e_4\}$*, and* $p_6 = \{e_3, e_4\}$*. The measurement matrix using these paths is*

$$\boldsymbol{A} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}. \tag{3.4}$$

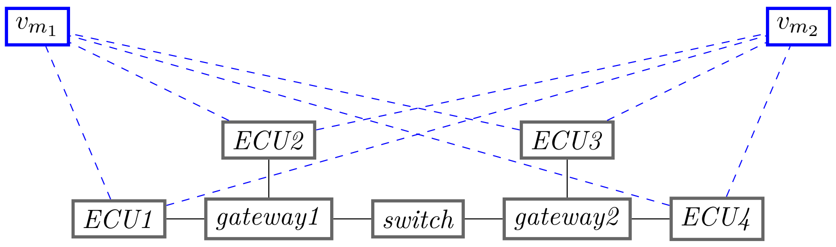
The rank of $\boldsymbol{A}$ *in* (3.4) *is* $r(\boldsymbol{A}) = 5$*, hence,* $r(\boldsymbol{A}) < \gamma_G$*. This is due to the two links* e_5 *and* e_6 *always appearing together whenever any path in* $\mathcal{P}_m(G)$ *traverses any of them. Therefore, the link-level metrics for these two links cannot be uniquely identified.*

It is clear from Example 3.4.2 and as stated in [115] and [19] that the topology can affect network identifiability. Authors in [19] studied the necessary and sufficient conditions for network topologies to be fully identifiable with two or more monitors. They proved that it is impossible to identify all $\gamma_G \geq 2$ links using only two monitors. And they stated that the extended graph G_{ex} of the network G should be 3-vertex-connected[5] in order to identify all link-level metrics in G (Theorem III.3 [19]). The extended graph is formed by adding two virtual nodes v_{m_1} and v_{m_2} to G and connecting all monitors in G to these additional nodes (i.e., adding $2\left|\mathcal{E}_m(G)\right|$ links). Recall that only edge nodes $\mathcal{E}(G)$ are accessible (monitors) in the in-vehicle network,

[5]A graph G is k-vertex-connected if it has $\eta_G > k$ and it remains connected whenever fewer than k vertices are deleted.



(a) Extended graph of Figure 3.3(a)



(b) Extended graph of Figure 3.3(b)

Figure 3.4: Extended graphs for the topology examples shown in Figure 3.3.

thus, the additional virtual nodes can only be connected to the edge nodes in G (see Figure 3.4 for the extended graphs of the examples shown in Figure 3.3).

Although the condition that the extended graph G_{ex} should be 3-vertex-connected is satisfied for the topology shown in Figure 3.3(b) (see Figure 3.4(b) for its extended graph), we have shown in Example 3.4.2 that links e_5 and e_6 cannot be uniquely identified. Therefore, in the following theorem, we state the necessary topological condition for identifying all links' metrics in an in-vehicle network G where only edge nodes in $\mathcal{E}(G)$ are accessible and can act as monitors.

Theorem 3.4.1. *In order to identify all links' metrics of in-vehicle network G with $\gamma \geq 2$, the necessary topological condition is that $d(v_i) \geq 3, \forall v_i \in \mathcal{R}(G)$.*

Proof. We prove this theorem by contradiction.

Assuming that network G is identifiable if $d(v_i) < 3, \exists v_i \in \mathcal{R}(G)$. This means that $\boldsymbol{A}$ should have full rank, *i.e.*, $r(\boldsymbol{A}) = \gamma_G$. Let $V(G) = \{v_1, v_2, v_3\}$ where each node v_i is directly connected to v_{i+1}. Let $E(G) = \{e_1, e_2\}$ with $e_1 = v_1v_2$ and $e_2 = v_2v_3$, hence $\gamma_G = 2$. Then by (3.2), the maximum number of possible paths in G is $\left|\mathcal{P}(G)\right| = 1$, then the highest rank $\boldsymbol{A}$ can have is $r(\boldsymbol{A}) = 1 < \gamma_G$, thus $r(\boldsymbol{A}) \neq \gamma_G$ which is a contradiction. $\square$

Testing the above theorem on the network G shown in Figure 3.3(b) where $\gamma_G = 6$, we can see that (let $v_1 =$ *switch*) $d(v_1) < 3$. Hence, e_5 and e_6 are not identifiable, and we can only identify their concatenated metrics for segment s_{v_2,v_3} as $x_{s_{v_2,v_3}} = x_{e_5} + x_{e_6}$, where $v_2 =$

gateway1 and $v_3 = gateway2$. In this case, G is l_G-*identifiable* with $l_G = 4 < \gamma_G = 6$, hence this topology is 4-*identifiable* network (see Definition 3.4.2).

3.4.2.2 Identifiability based on Monitors

Two other aspects can determine $r(\boldsymbol{A})$: number of monitors and their placement. From Theorem III.1 in [19], we know it is not possible to have only two monitors to identify all $\gamma_G \geq 2$ links in G. Therefore, more than two monitors are required. Moreover, monitors should be properly placed such that all links in $E(G)$ can be uniquely identified.

Lemma 3.4.1. *In order to fully identify in-vehicle network G, given that the topological condition in Theorem 3.4.1 is satisfied, each link $e_i \in E(G)$, $i \in \{1, 2, \ldots, \gamma_G\}$ should be traversed by at least one measurement path $p_i \in \mathcal{P}_m(G)$, $i \in \{1, 2, \ldots, \kappa_G\}$.*

Proof. By definition, identifying in-vehicle network G means to uniquely determine the link-level metrics for all links in $E(G)$ using measurements in $\boldsymbol{y}$ (see Definition 3.4.1 and Definition 3.4.2). If no paths in $\mathcal{P}_m(G)$ traversed link $e_i \in E(G)$, $i \in \{1, 2, \ldots, \gamma_G\}$, then the associated metric of e_i, x_i, cannot be uniquely determined from $\boldsymbol{y}$. □

Let $\mathcal{E}_m(G) \subseteq \mathcal{E}(G)$ be the set of monitoring edge nodes in an in-vehicle network G, then monitor placement condition in the following theorem should be satisfied in order to fully identify G.

Theorem 3.4.2. *An in-vehicle network G with $\gamma_G \geq 2$ that satisfies the condition in Theorem 3.4.1 is fully identifiable if and only if $\mathcal{E}(G) = \mathcal{E}_m(G)$. That is $\forall v_i \in \mathcal{E}(G)$ with $i \in \{1, 2, \ldots, \eta_G\}$, v_i is a monitoring node.*

Proof. We prove this by contradiction.

Assume that in-vehicle network G with $\gamma_G \geq 2$ is identifiable, hence, $r(\boldsymbol{A}) = \gamma_G$. Let $v_i \in \mathcal{E}(G) \backslash \mathcal{E}_m(G)$ be a non-monitor edge node. Then $\left|\mathcal{E}_m(G)\right| < \left|\mathcal{E}(G)\right|$ and the link incident to node v_i will not be traversed by any paths in $\mathcal{P}_m(G)$ (see Lemma 3.4.1) and by rewriting (3.2) as

$$\left|\mathcal{P}_m(G)\right| = \frac{\left|\mathcal{E}_m(G)\right|\left(\left|\mathcal{E}_m(G)\right| - 1\right)}{2},$$

where $\left|\mathcal{E}_m(G)\right| = \left|\mathcal{E}(G)\right| - 1$, we get $\left|\mathcal{P}_m(G)\right| < \gamma_G$. Thus, $r(\boldsymbol{A}) < \gamma_G$ which contradicts that $r(\boldsymbol{A}) = \gamma_G$. □

From the above theorem, we can quantify the minimum number of monitors required to achieve a fully identifiable network.

Corollary 3.4.1. *Given that the topological condition in Theorem 3.4.1 is satisfied, the minimum number of monitors required to uniquely identify all $\gamma_G \geq 2$ links in G is $\left|\mathcal{E}(G)\right|$.*

3.4.2.3 Identifiability in In-Vehicle Networks

CAN network is a special case, where it can be viewed as a network of a single link that connects multiple ECUs (see fieldbus architecture shown in Figure 3.1(a)). As described in Chapter 2, each CAN node is assigned a unique ID; this ID gives each CAN node a priority level in which such node uses this value in the arbitration process (see the following definition). The lowest ID value indicates the highest priority level.

Definition 3.4.4. Given a single CAN network G, let $\mathcal{E}(G) = V(G) = \{v_1, v_2, \ldots, v_{\eta_G}\}$ and $c_1, c_2, \ldots, c_{\eta_G} \geq 0$, $c_i \in \mathbb{Z}^+$, be the set of edge nodes and their associated IDs, respectively. If two nodes v_i and v_j with respective IDs $c_i = 0$ and $c_j = 1$ start transmitting simultaneously, then node v_i wins the arbitration process and access the CAN bus to transmit, while v_j backs off until v_i finishes transmitting. Similarly, if all nodes in $V(G)$ are trying to transmit at the same time, then we say that $v_i \in V(G)$ wins the arbitration process and access the CAN bus b_n if $\min\{c_1, c_2, \ldots c_{\eta_G}\} = c_i$.

It is worth mentioning that CAN networks can be seen as asymmetric networks, this means that the delay (primarily composed of access and transmission delay components) in one direction is different from the opposite direction. This is due to the priority level assigned to each ECU.

Lemma 3.4.2. *Link delay in CAN network G has different values in opposite directions if two nodes transmit simultaneously.*

Proof. Consider CAN network G with two nodes v_1 and v_2. Let c_1 be the ID for v_1 and c_2 be the ID for v_2, where $c_2 < c_1$. Assume at time t that both v_1 and v_2 are trying to transmit. By Definition 3.4.4 we know that v_2 will win the arbitration process and start transmitting before v_1. After v_2 finishes transmitting, v_1 can start its transmission process. Then, the access delay on link b_1 (the CAN bus) from v_2 to v_1 is less than its access delay from v_1 to v_2 because a message from v_2 will access the bus before v_1's messages. Hence, messages transmitted by v_1

will be delayed by messages transmitted by v_2. Thus, $\overrightarrow{x_{b_1}} < \overleftarrow{x_{b_1}}$ where $\overrightarrow{x_{b_1}}$ is the delay of link b_1 from v_2 to v_1 and $\overleftarrow{x_{b_1}}$ is the delay of link b_1 from v_1 to v_2. □

The above lemma indicates that the CAN network can be seen as an asymmetric network and, based on Corollary 2.7 in [209], asymmetric networks are not identifiable unless the nodes incident to each link are monitoring nodes. Although CAN can be asymmetric, the following theorem is true.

Theorem 3.4.3. *Any single CAN network G can be abstracted as $\left|V(G)\right| = 2$ and it is always identifiable.*

Proof. We first prove the case when G is asymmetric. In the CAN network, all nodes have a degree of one, i.e., $d(v_i) = 1$, $\forall v_i \in V(G)$, thus $\mathcal{E}(G) = V(G)$. According to Theorem 3.4.2, both nodes in G should be monitors as they are already in $\mathcal{E}(G)$. And because there is only one link b_1 in CAN, $\gamma_G = 1$, the condition that the nodes incident to the link should be monitoring nodes (Corollary 2.7 in [209]) is fulfilled. Therefore, if the two nodes are transmitting at the same time, then G is asymmetric and both link directions could be measured.

If G is symmetric (link delay in opposite directions are the same), meaning that the two nodes are not competing to access the bus b_1, then G is identifiable according to Corollary 2.5 in [209]. □

This can also be true for a single CAN network with $\left|V(G)\right| > 2$. We can assume that CAN always includes two nodes; one is the source node and the second is the receiving node. This is a valid assumption in CAN as the priority level assigned to nodes can also be seen as traffic property. Therefore, we can view CAN as a network of two nodes and one link. Hence, CAN is always identifiable.

Corollary 3.4.2. *Any single CAN network G with $\left|V(G)\right| > 1$ is identifiable using only two monitors.*

Based on Lemma 3.4.2, an in-vehicle network G that is composed of multiple CANs connected via a central-gateway as shown in Figure 3.1(b) can be asymmetric if two or more nodes transmit simultaneously. However, G can be symmetric if CAN nodes used as monitors are carefully configured to avoid engaging in the arbitration process with other monitors. This can be achieved by changing the transmission times of different CAN monitoring nodes.

Corollary 3.4.3. *An in-vehicle network G with $\left|B(G)\right| > 2$, where all CANs are connected via a central-gateway $g \in \mathcal{R}(G)$, is identifiable if G is symmetric. The minimum number of required monitors is $\left|B(G)\right|$, where each monitor is placed in one CAN network.*

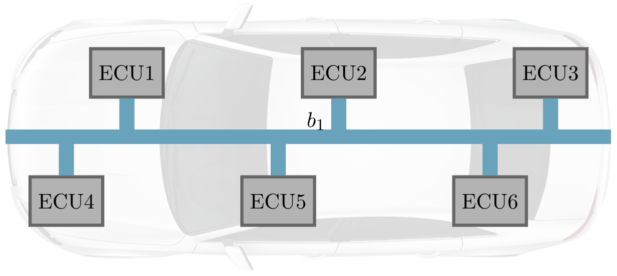
From the above analysis, we can conclude that fieldbus and central-gateway architectures are always identifiable given that the required number and placement of monitors are satisfied. Other architectures, however, cannot be guaranteed to be identifiable. For example, the Ethernet-based architecture shown in Figure 3.1(c) is unidentifiable because it does not satisfy the topological condition in Theorem 3.4.1. As seen, the gateways have a degree less than 3 (remember that CAN includes a single link, i.e., the CAN bus that the gateway is connected to). However, it can be l_G-identifiable, if we connect some (or all) gateways to extra CANs. If each gateway is connected to another CAN, then similar to central-gateway architecture, Ethernet-based network G shown in Figure 3.1(c) can be fully identifiable with $\left|B(G)\right|$ monitors, assuming that all nodes in $\mathcal{E}(G)$ are CAN nodes. Additionally, if exists in the network, each edge Ethernet node should be a monitor.

3.5 Evaluating Network Tomography in In-Vehicle Networks

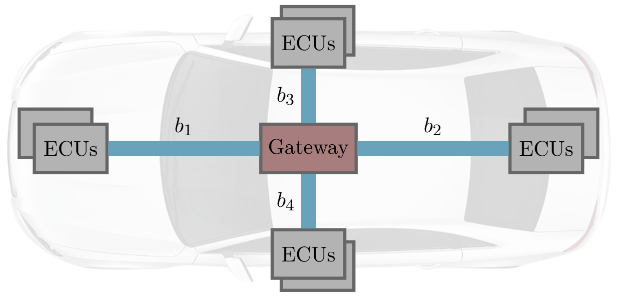
This section evaluates the above theoretical analysis by applying network tomography to in-vehicle networks. To do this, we use simulation-based experiments using OMNeT++ simulator [198]. In addition, we compare the performance of the network tomography-based monitoring approach with OTIDS [79]. The following gives a more detailed description of the experiments' setup and the results obtained from such experiments.

3.5.1 Simulation

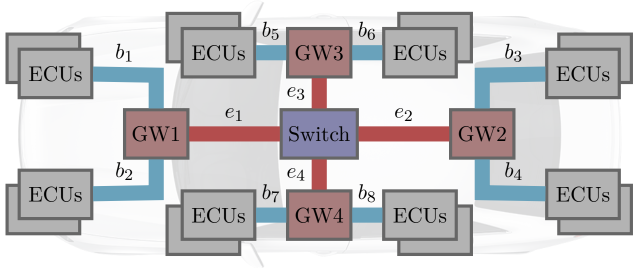
Figure 3.5 shows an overview of the three different in-vehicle network architectures that we simulated. Figure 3.5(a) is a single CAN network G_1 with $\eta_{G_1} = 30$. In addition, central-gateway and Ethernet-based in-vehicle networks shown in Figure 3.5(b) and Figure 3.5(c), respectively, were also simulated. These network architectures resemble the new E/E architectures that support centralisation (see Chapter 2.2).



(a) Single CAN network (G_1)



(b) Central-gateway in-vehicle network (G_2)



(c) Ethernet-based in-vehicle network (G_3)

🟦 CAN bus 🟥 Ethernet link

Figure 3.5: Simulated in-vehicle networks to evaluate network tomography as a monitoring approach. ***GW*** stands for gateway.

Let G_1, G_2, and G_3 be the three simulated networks shown in Figure 3.5, then Table 3.2 shows the parameters used for each simulated network.

Table 3.2: Network parameters used in each simulated scenario for applying network tomography in in-vehicle networks.

Parameter	G_1	G_2	G_3
Number of nodes (η)	30	121	245
Number of edge nodes ($\left\|\mathcal{E}\right\|$)	30	120	240
Number of CAN buses ($\left\|B\right\|$)	1	4	8
Number of links (γ)	1	4	12

Table 3.2: Network parameters used in each simulated scenario for applying network tomography in in-vehicle networks.

Parameter	G_1	G_2	G_3
Number of monitors ($\left\|\mathcal{E}_m\right\|$)	2	4	8
Number of measured paths (κ)	1	4	12
CAN bus bandwidth	1 Mbps	1 Mbps	1 Mbps
Ethernet link bandwidth	N/A	N/A	100 Mbps
Switch processing delay	N/A	N/A	8 μs

Note that for the single CAN network G_1, there is only one link which corresponds to the CAN bus $B(G_1) = \{b_1\}$. Moreover, because CAN network is identifiable using only two monitors, only one path between these monitors needs to be measured. As the maximum number of nodes CAN can accommodate is 30 (see Chapter 2.1.2.2), in the central-gateway and Ethernet-based in-vehicle network architectures, there were 30 ECUs connected to each CAN bus. Within each CAN, we simulated traffic similar to real car traffic from the available dataset provided by [97]. Note that [97] provides more than one dataset. In this experiment, we use the dataset for normal traffic (attack-free). In addition, cross-traffic existed between different domains. The payload size for all normal traffic was 8 bytes (see Appendix A for more details about traffic frequency). While monitoring traffic did not have a payload, its packet size was 47 bits (header size). To avoid asymmetric behaviour, the first monitoring messages were transmitted at different times. Then, each message was periodically sent every $10ms$. We ran the simulations for $130s$ and the total number of probing messages was 13000.

As used in OTIDS, we simulated remote frames (see Chapter 2.1.2.2) as monitoring messages in the central-gateway scenario where a CAN node requests such frames every $10ms$. The network parameters used for applying OTIDS are shown in Table 3.3.

Table 3.3: Network parameters used in each simulated scenario for applying OTIDS in in-vehicle networks.

Parameter	G_1	G_2	G_3
Number of nodes (η)	30	121	245
Number of edge nodes ($\left\|\mathcal{E}\right\|$)	30	120	240

Table 3.3: Network parameters used in each simulated scenario for applying OTIDS in in-vehicle networks.

Parameter	G_1	G_2	G_3
Number of CAN buses ($\left\|B\right\|$)	1	4	8
Number of links (γ)	1	4	12
Number of monitors ($\left\|\mathcal{E}_m\right\|$)	27	29	35
Number of measured paths (κ)	26	28	34
CAN bus bandwidth	1 Mbps	1 Mbps	1 Mbps
Ethernet link bandwidth	N/A	N/A	100 Mbps
Switch processing delay	N/A	N/A	8 μs

3.5.2 Results

We evaluate the network performance when monitoring traffic exists alongside normal traffic. Moreover, we compare the monitoring overhead against the OTIDS approach. Figure 3.6 and Figure 3.7 show the average bandwidth utilisation for each CAN bus and the average latency for normal traffic, respectively.

As shown in Figure 3.6, the consumed bandwidth has increased after adding network tomography monitoring traffic. This increase, however, is not significant (maximum increase is about 0.96% for b_1 in the central-gateway topology shown in Figure 3.5(b) and 1.9% for b_7 in Ethernet-based scenario shown in Figure 3.5(c)). However, when OTIDS is used, the consumed bandwidth has significantly increased. This is because OTIDS frequently requests for remote frames produced by all CAN nodes in the network; as the number of CAN IDs increases, the utilised bandwidth increases too.

Besides network bandwidth, it is crucial that the monitoring traffic does not affect the latency of normal traffic. In Figure 3.7, we show the average end-to-end latency for some of the normal traffic in each CAN bus (reporting the results for all existing traffic is not feasible due to the large volume of traffic). The increase in latency with network tomography monitoring traffic is negligible, around $0.4\mu s$ and $0.6\mu s$ at maximum for central-gateway (Figure 3.5(b)) and Ethernet-based (Figure 3.5(c)) scenarios, respectively. In contrast, adding OTIDS monitoring traffic results in increased latency for the existing traffic. This is due to the arbitration process

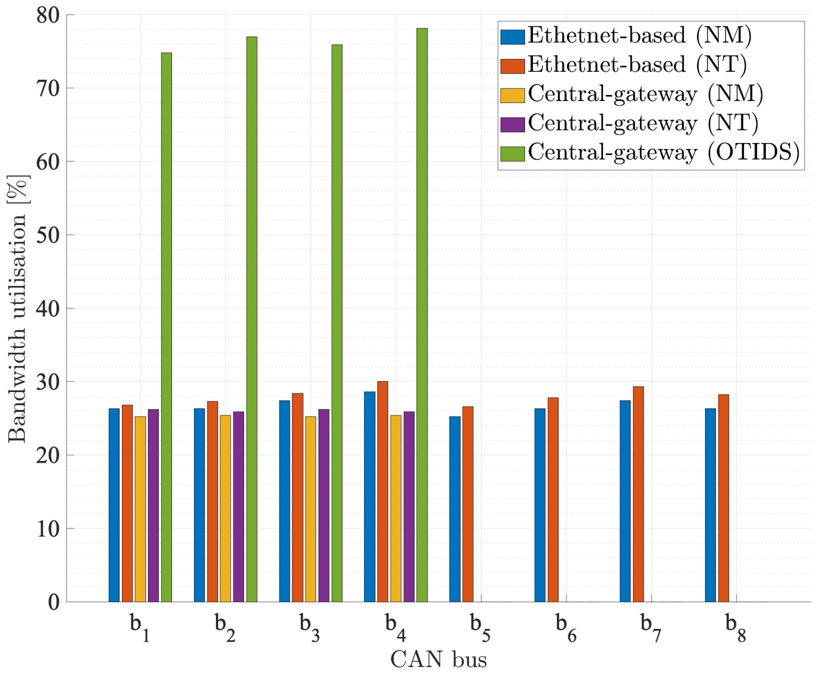


Figure 3.6: Average bandwidth of CAN buses where monitoring traffic exists. *NM*: No Monitoring, and *NT*: with Network Tomography monitoring traffic.

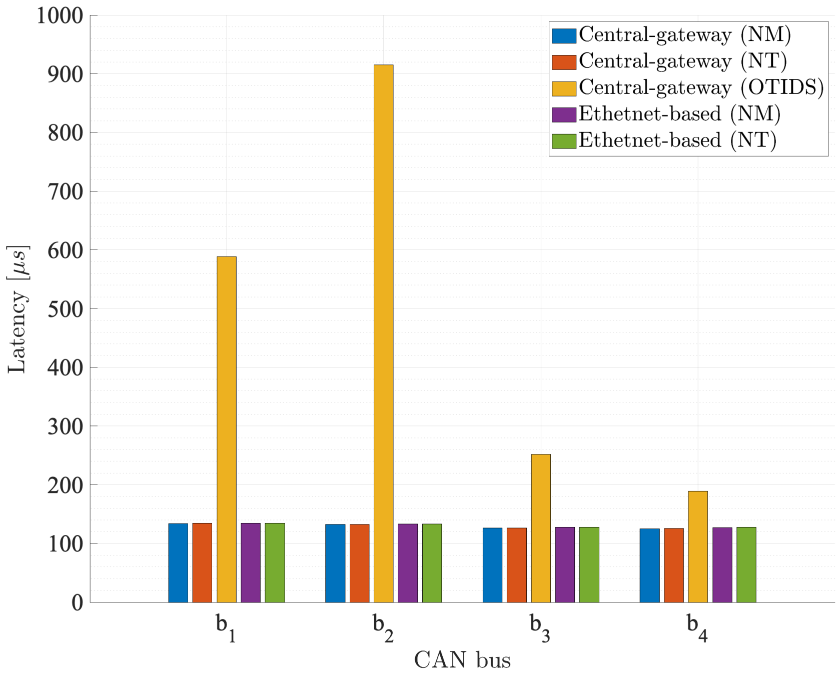


Figure 3.7: Average latency of network traffic where monitoring traffic exists. *NM*: No Monitoring, and *NT*: with Network Tomography monitoring traffic.

that CAN nodes engage with, as OTIDS periodically requests remote frames for all CAN IDs where each ID has different priority levels, giving the response frames a chance to compete with the normal traffic to access the bus, hence the increased latency in the existing traffic.

Overall, compared with OTIDS, it can be observed that the network tomography approach achieves better performance for both bandwidth and latency. Network tomography saves up to 52.2% bandwidth, while it reduces latency by $782.3\mu s$. This improvement is a result of monitoring a limited number of end-to-end paths as compared with OTIDS which monitors the whole network.

Next, we evaluate network identifiability based on the number of monitors. The result is shown in Figure 3.8. As shown, single CAN is identifiable when $\left|\mathcal{E}_m(G_1)\right| \geq 2$. This is aligned with Corollary 3.4.2. On the other hand, all links are identifiable if number of monitors $|\mathcal{E}_m(G_2)| \geq 4$ and $|\mathcal{E}_m(G_3)| \geq 8$ in central-gateway and Ethernet-based scenarios, respectively, which confirms that the minimum number of monitors required is $\left|B(G)\right|$ as stated in Corollary 3.4.3, where $B(G)$ is the set of CAN buses in the in-vehicle network G. For the central-gateway architecture, it is possible to identify part of the network using three monitors, as shown in Figure 3.8, 75% of the network can be identified with three monitors. On the other hand, five monitors can identify 25% of the Ethernet-based network, while six monitors can identify 50% and 75% can be identified with seven monitors. It is not possible for CAN (or any other network in this matter that does not allow cyclic routing) to be fully identified using only one monitor as this monitor alone cannot provide any end-to-end measurements unless a second monitor exists.

Additionally, we compare the link-level delays inferred by network tomography with the actual values using absolute error, $|x_i - \hat{x}_i|$, where x_i is the actual link-level delay and $\hat{x}_i$ is the inferred delay. The results are shown in Figure 3.9 and Figure 3.10 where the link-level delay for b_i is x_i, $i \in \{1, \ldots, 8\}$ and for e_i is x_i, $i \in \{9, \ldots, 12\}$.

For the central-gateway scenario (Figure 3.9), the maximum error is $41\mu s$. On the other hand, the maximum error observed for the Ethernet-based scenario (Figure 3.10) is $174\mu s$. The reported error is attributed to the fact that monitoring nodes transmit the monitoring traffic at different times due to the arbitration process. We can argue that these error values are small and that the inferred delay values would still be falling within the actual range.

Moreover, the reason that different links experience different error values is that each link is traversed by a different number of measurement paths. For instance, links $e_2, e_4 \in E(G_2)$

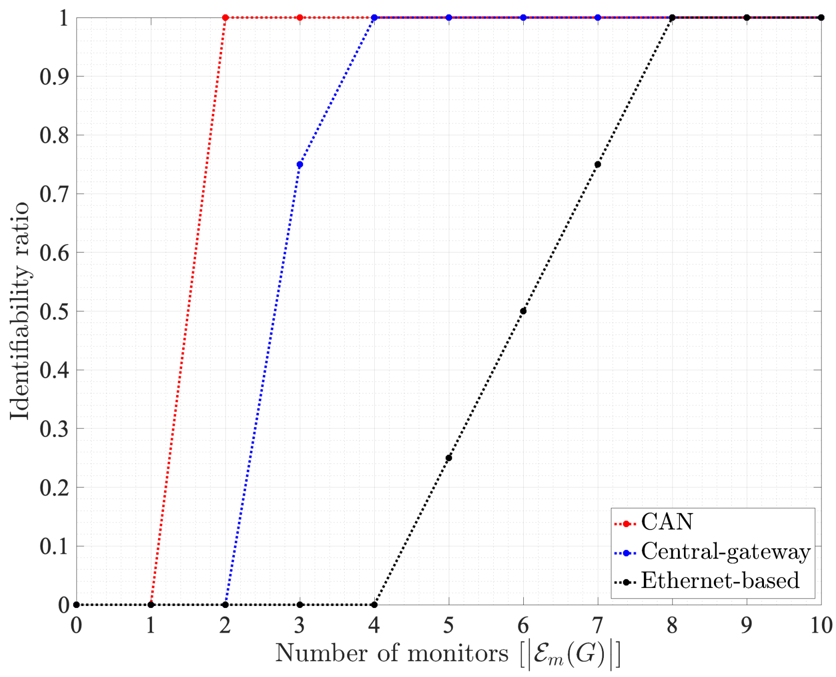


Figure 3.8: Identifiability ratio of CAN, central-gateway, and Ethernet-based in-vehicle networks.

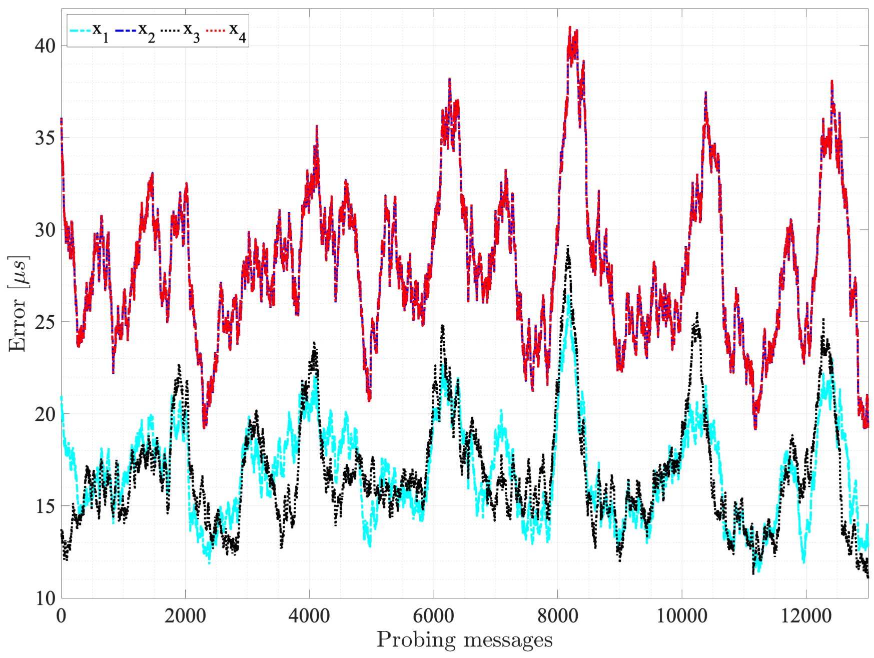


Figure 3.9: Absolute error of inferred link-level delay in central-gateway architecture. Maximum error is $41 \mu s$.

in Figure 3.5(b) are traversed by only one path in $\mathcal{P}_m(G_2)$ while $e_1, e_3 \in E(G_2)$ are traversed by two paths. As the link is measured by more paths the inference accuracy for such a link increases, hence, the error values for x_2 and x_4 are larger than those for x_1 and x_3.

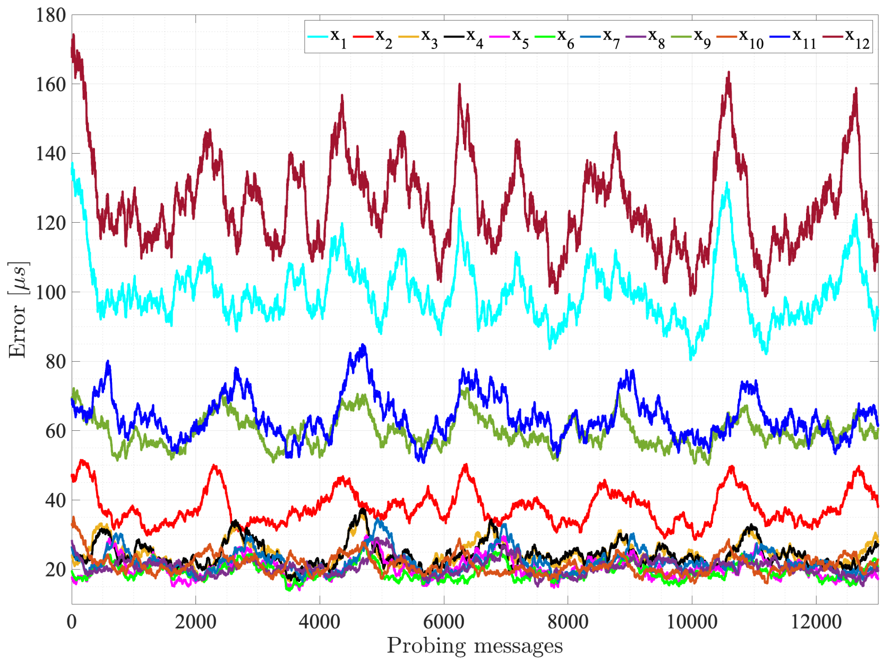


Figure 3.10: Absolute error of inferred link-level delay in Ethernet-based architecture. Maximum error is $174\mu s$.

3.6 Summary

In this chapter, we have introduced network tomography in in-vehicle networks as a performance monitoring approach. We theoretically investigated the applicability of network tomography on three different in-vehicle network architectures. Our analysis demonstrated that network tomography can be applied in in-vehicle networks to uniquely identify all links or a subset of links, given that the topological and monitor placement conditions are satisfied. We analysed network tomography using delay metrics, other metrics can also be used such as loss/success packet rate.

One challenge we found is that not all Ethernet-based architectures can be identifiable, as compared with central-gateway and single CAN in-vehicle networks. It is important for applying network tomography in an Ethernet-based network that the topology is fully identifiable, as next-generation in-vehicle networking architectures will be heavily relying on Ethernet.

In-vehicle network tomography requires further investigations and in this chapter, we have built a stepping stone into this field. The following chapter further investigates network identifiability and proposes new solutions to tackle the identifiability challenges imposed by lack of measurements or violation of topological conditions.

4 DNN-based Partial Tomography

"The most beautiful thing we can experience is the mysterious. It is the source of all true art and science."

Albert Einstein

4.1 Overview

The last chapter discussed the constraints that network tomography imposes in order to use it to infer the performance of *all* links in the network. An important constraint is that the measurement matrix has to be full rank. In this chapter, we devise a partial tomography algorithm in addition to a deep learning-based solution to tackle the problem of having a rank-deficient measurement matrix.

Particularly, this chapter proposes a novel approach for inferring link-level and path-segment as well as end-to-end path-level performance metrics in an accurate and timely fashion, by measuring performance along selected paths. To do so, we combine traditional network tomography with deep learning. The former is used to infer performance metrics in the network, partially depending on the availability of passive end-to-end measurements. The latter takes the results of this partial network tomography as input in order to train a deep neural network which is then used to estimate values of path-level metrics for the whole network. End-to-end path metrics can be trivially calculated by aggregating inferred (by partial network tomography) and estimated (by our trained model) metrics.

This chapter focuses on passive measurements (see Chapter 2.4.3.2). The desire to use passive measurements is due to multiple reasons: first, for a complicated in-vehicle network, inserting a large number of probes might consequently affect the mission-critical network performance. Second, with passive tomography, the existing traffic can provide more realistic and accurate measurements than the ones with inserted probes, which are different to the actual traffic. Third, the issues related to placing and minimising the number of monitors [130] are eliminated with passive tomography. Hence, in this chapter, we employ passive measurements (assuming that there are traffic with same parameters in the network). In spite of this, the approach can be extended to active measurements where the measurement matrix is not a full-rank matrix.

4.2 Introduction

As we know from the last chapter, in order to uniquely solve for $\boldsymbol{x}$ in (3.1), the routing matrix $\boldsymbol{A}$ used for the measurement should be invertible, and for $\boldsymbol{A}$ to be invertible, it has to be a full-rank square matrix. Specifically, for our network tomography problem, in order to identify all link-level metrics, two conditions should be satisfied: (i) the number of end-to-end measurements should be equal to the number of links in the network ($\kappa_G = \gamma_G$), and (ii) the available end-to-end measurements should be linearly independent (see Definition 3.4.3). The first condition ensures that $\boldsymbol{A}$ is a square matrix, while the second condition ensures that the square matrix $\boldsymbol{A}$ is a full-rank matrix with $r(\boldsymbol{A}) = \gamma_G$ (see Table 4.1).

Because in this chapter we focus on passive tomography, which is based on measuring the existing traffic, the available measurements can form a routing matrix that can be either one of the following cases

1. full-rank matrix with $r(\boldsymbol{A}) = \gamma_G$ and $\kappa_G = \gamma_G$;
2. rank-deficient matrix with $r(\boldsymbol{A}) < \gamma_G$ and $\kappa_G = \gamma_G$; and
3. rank-deficient matrix with $r(\boldsymbol{A}) < \gamma_G$ and $\kappa_G < \gamma_G$.

The first case satisfies both conditions (i) and (ii), and therefore $\boldsymbol{x}$ can be uniquely identified using (3.1). The second case only satisfies condition (i), while the third case does not satisfy any of the conditions. Thus, the last two cases cannot use (3.1) to infer all link-level metrics in the network. In this chapter, we assume that the in-vehicle network falls under either one of the last two cases, where the available measurements cannot form a full-rank matrix. The following examples further illustrate these cases.

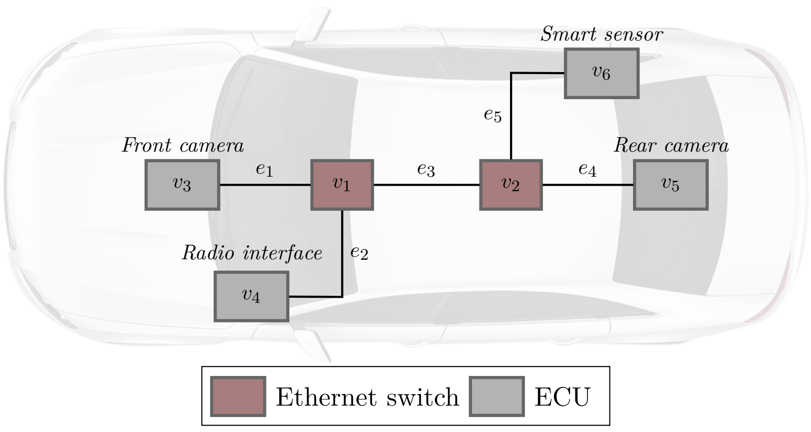


Figure 4.1: Example of simple in-vehicle network topology with six nodes - based on Ethernet-based architecture.

Example 4.2.1. *Consider the network shown in Figure 4.1. There are five links, $\gamma_G = 5$, corresponding to $E(G) = \{e_1, e_2, \ldots, e_5\}$. In this example, the set of all possible paths is $\mathcal{P}(G) = \{p_1, p_2, p_3, p_4, p_5, p_6\}$ with*

$$p_1 = \{e_1, e_2\},\ \ p_2 = \{e_1, e_3, e_4\},\ \ p_3 = \{e_1, e_3, e_5\}$$

$$p_4 = \{e_2, e_3, e_4\},\ \ p_5 = \{e_2, e_3, e_5\},\ \ p_6 = \{e_4, e_5\}.$$

Now suppose that the traffic going on in this network is passing only through p_1, p_2, and p_6. Hence, because the measurements are passive, we have $\mathcal{P}_m(G) = \{p_1, p_2, p_6\}$. And (3.1) can be written as

$$\begin{pmatrix} y_1 \\ y_2 \\ y_6 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}. \tag{4.1}$$

In (4.1), only a limited number of measurements are available (y_1, y_2, y_6), while others for paths in $\mathcal{P}(G) \backslash \mathcal{P}_m(G)$ are unknown (i.e., for paths y_3, y_4, and y_5). Thus, $\kappa_G < \gamma_G$, which makes the measurement matrix rank-deficient with $r(\boldsymbol{A}) = 3$.

In Example 4.2.1, note that the in-vehicle network topology G is identifiable according to Theorem 3.4.1. However, the available traffic cannot form a full-rank measurement matrix $\boldsymbol{A}$.

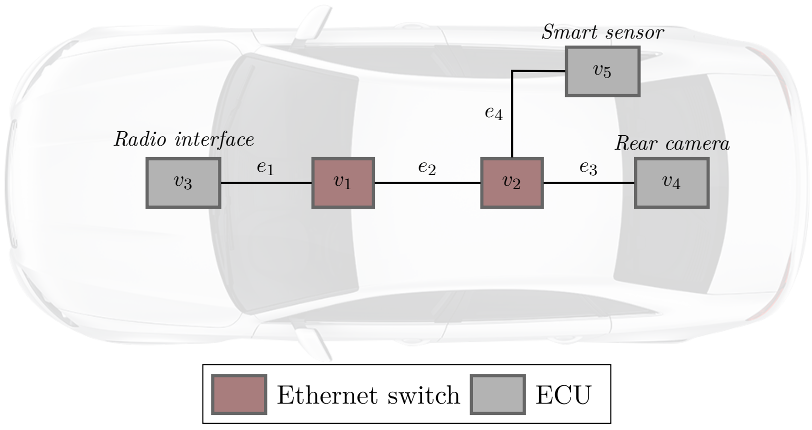


Figure 4.2: Example of simple in-vehicle network topology with five nodes - based on Ethernet-based architecture.

Example 4.2.2. *In this example, let us consider the topology shown in Figure 4.2. This topology has* $\gamma_G = 4$ *for* $E(G) = \{e_1, e_2, e_3, e_4\}$. *The set of all possible paths is* $\mathcal{P}(G) = \{p_1, p_2, p_3\}$, *with*

$$p_1 = \{e_1, e_2, e_3\}, \;\; p_2 = \{e_1, e_2, e_4\}, \;\; p_3 = \{e_3, e_4\}.$$

Assuming that the existing traffic uses all these paths, then $\kappa_G = 3$. *By writing* (3.1) *as*

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}. \tag{4.2}$$

we can see that $\kappa_G < \gamma_G$, *and the columns of* $\boldsymbol{A}$ *that correspond to* e_1 *and* e_2 *are identical. This is because the measurement paths that pass through* e_1 *always pass through* e_2 *too. The reason for this is that the topology is* ***not*** *identifiable; since* $d(v_1) = 2 < 3$ *(see Theorem 3.4.1). In fact, one can immediately know that the measurement matrix is not full-rank by computing the maximum number of paths* $\left|\mathcal{P}(G)\right|$ *using* (3.2), *which results in* $\left|\mathcal{P}(G)\right| = 3$ *for this example.*

Example 4.2.1 and Example 4.2.2 are two examples of the third case mentioned above, i.e., having rank-deficient measurement matrix $\boldsymbol{A}$ with $r(\boldsymbol{A}) < \gamma_G$ and $\kappa_G < \gamma_G$. The following shows an example of the second case where $r(\boldsymbol{A}) < \gamma_G$ and $\kappa_G = \gamma_G$.

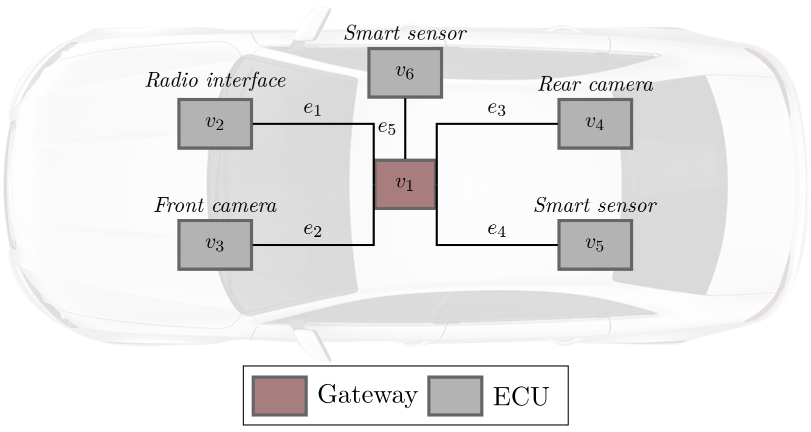


Figure 4.3: Example of simple in-vehicle network topology with six nodes - based on central-gateway architecture.

Example 4.2.3. *Consider the topology shown in Figure 4.3. Here,* $\gamma_G = 5$ *as* $E(G) = \{e_1, e_2, \ldots, e_5\}$ *and* $\mathcal{P}(G) = \{p_1, p_2, \ldots, p_{10}\}$ *where*

$$p_1 = \{e_1, e_2\},\ \ p_2 = \{e_1, e_3\},\ \ p_3 = \{e_1, e_4\},\ \ p_4 = \{e_1, e_5\},\ \ p_5 = \{e_2, e_3\}$$

$$p_6 = \{e_2, e_4\},\ \ p_7 = \{e_2, e_5\},\ \ p_8 = \{e_3, e_4\},\ \ p_9 = \{e_3, e_5\},\ \ p_{10} = \{e_4, e_5\}.$$

Assuming that the available traffic passes through p_3, p_4, p_5, p_8, *and* p_9, *hence, the set of measured paths is* $\mathcal{P}_m(G) = \{p_3, p_4, p_5, p_8, p_9\}$. *Then,* (3.1) *can be written as*

$$\begin{pmatrix} y_3 \\ y_4 \\ y_5 \\ y_8 \\ y_9 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}. \tag{4.3}$$

Here, the number of available measurements is $\kappa_G = 5 = \gamma_G$. *However, the rank of the measurement matrix is* $r(\boldsymbol{A}) = 4 < \gamma_G$. *Therefore,* (3.1) *cannot uniquely infer* $x_i, \forall e_i \in E(G)$. *The reason for this is that the measured paths in* $\mathcal{P}_m(G)$ *are* ***linearly dependent*** *(see*

Definition 3.4.3), where p_9 *is a linear combination of* p_3*,* p_4*, and* p_8*. That is,*

$$p_9 = p_8 + p_4 - p_3.$$

When the available end-to-end measurements are dependent, even if $\kappa_G = \gamma_G$, the resulting measurement matrix is rank-deficient with $r(\boldsymbol{A}) < \gamma_G$.

To uniquely solve $\boldsymbol{x}$ in (3.1), γ_G *independent* measurements should be available, so that $\boldsymbol{A}$ is a full-rank matrix with $r(\boldsymbol{A}) = \gamma_G$. However, as seen from the above examples, the measurement matrix for passive monitoring in an in-vehicle network is unlikely to form a full-rank matrix due to dependent (as in Example 4.2.3) or inadequate (see Example 4.2.1) end-to-end measurements, or even an unidentifiable topology (as shown in Example 4.2.2).

Table 4.1: Chapter 4 notations and their descriptions.

Notation	Description
Network & Graph Theory	
$G = (V, E)$	Network represented as graph G
$V(G)$	Set of vertices (nodes) in network G
$E(G)$	Set of edges (links) in network G
$\mathcal{E}(G) \subset V(G)$	Set of edge nodes in G (see Definition 3.3.1)
$\mathcal{R}(G) \subset V(G)$	Set of intermediate nodes in G (see Definition 3.3.1)
$\mathcal{P}(G)$	Set of all possible paths in network G
$\mathcal{P}_m(G) \subseteq \mathcal{P}(G)$	Set of measured paths in network G
$\mathcal{S} \subset G$	Partial network of G
$B(G)$	Set of CAN buses in G
$d(v_i)$	Degree of node $v_i \in V(G)$
$p_i \in \mathcal{P}(G)$	i^{th} path connecting two non-adjacent nodes $v_i, v_j \in \mathcal{E}(G)$
s_{v_i,v_j}	Segment connecting two non-adjacent nodes $v_i, v_j \in V(G)$ with $v_i \vee v_j \in \mathcal{R}(G)$
$e_i \in E(G)$	i^{th} link in network G
$b_i \in B(G)$	The i^{th} CAN bus in G

Table 4.1: Chapter 4 notations and their descriptions.

Notation	Description
Network Tomography	
$\boldsymbol{y}$	Vector of end-to-end measurements
$\boldsymbol{x}$	Vector of link-level measurements
$\boldsymbol{A}$	Measurement matrix
$y_i \in \boldsymbol{y}$	End-to-end measurement of path $p_i \in \mathcal{P}_m(G)$
$x_i \in \boldsymbol{x}$	Link-level metric of i^{the} link $e_i \in E(G)$
$a_{ij} \in \{0,1\}$	The element of the measurement matrix $\boldsymbol{A}$ at the i^{th} row (for $p_i \in \mathcal{P}_m(G)$) and j^{th} column (for $e_j \in E(G)$)
$r(\boldsymbol{A})$	Rank of the measurement matrix $\boldsymbol{A}$
Deep Neural Network	
$\boldsymbol{\beta}$	Bias vector
$\boldsymbol{W}$	Matrix of weights
$\boldsymbol{h}_i$	Vector of i^{th} hidden layer
$f(z)$	Hidden layer activation function
$f_o(z)$	Output function
$\hat{\boldsymbol{y}}$	Vector of estimated end-to-end measurements
$\hat{\boldsymbol{x}}$	Vector of estimated link-level measurements
L	Loss function
α	Learning rate
Numbers & Cardinalities	
$\eta_G := \Big\lvert V(G) \Big\rvert$	Total number of nodes in network G
$\gamma_G := \Big\lvert E(G) \Big\rvert$	Total number of links in network G
$\kappa_G := \Big\lvert \mathcal{P}_m(G) \Big\rvert$	Total number of measured paths in G
ϑ	Total number of partial networks

In the following, we show how partial tomography and deep learning can be utilised to tackle the rank-deficiency problem and infer the overall network performance.

4.3 Partial Network Tomography

In this section, we describe the proposed partial network tomography approach, where some link-level (or segment) metrics are inferred using the available end-to-end measurements.

For the network shown in Figure 4.1, consider the case where the measurement matrix $\boldsymbol{A}$ is rank-deficient, as in (4.1). In such a case, it is not possible to uniquely identify link-level metrics $\boldsymbol{x}$ for all γ_G links in G. However, a subset of link-level and/or segment metrics can be identified using partial network tomography of a partial network $\mathcal{S} \subset G$.

The following theorems state the requirements that need to be fulfilled when considering the use of partial tomography.

Theorem 4.3.1. *To perform partial network tomography on a given network G, the selected partial network $\mathcal{S} \subset G$ should include at least three edge nodes, $\Big|\mathcal{E}(\mathcal{S})\Big| \geq 3$.*

Proof. From Definition 3.3.1, we know that each edge node $v_i \in \mathcal{E}(G)$ has at most one connected link, hence $d(v_i) = 1$. And if $\Big|\mathcal{E}(\mathcal{S})\Big| < 3$, e.g., $\Big|\mathcal{E}(\mathcal{S})\Big| = 2$, there are at least two links in $\mathcal{S}$, i.e., $\gamma_{\mathcal{S}} \geq 2$. Because G is an undirected network and the routing between any pair is deterministic, then by (3.2), the maximum number of end-to-end measurements of partial network $\mathcal{S}$ with $\Big|\mathcal{E}(\mathcal{S})\Big| = 2$ is

$$\frac{\Big|\mathcal{E}(\mathcal{S})\Big|\Big(\Big|\mathcal{E}(\mathcal{S})\Big| - 1\Big)}{2} = 1. \tag{4.4}$$

Thus, $\kappa_{\mathcal{S}} = 1$, which means that $\kappa_{\mathcal{S}} < \gamma_{\mathcal{S}}$ and hence the measurement matrix $\boldsymbol{A}$ for $\mathcal{S}$ is not full-rank because $r(\boldsymbol{A}) = 1$, so $\boldsymbol{x}$ for links in $\mathcal{S}$ cannot have unique solution. □

Theorem 4.3.2. *To perform partial network tomography on a partial network $\mathcal{S} \subset G$ with $\Big|\mathcal{E}(\mathcal{S})\Big| \geq 3$, the number of available end-to-end measurements should be at least three, i.e., $\kappa_{\mathcal{S}} \geq 3$.*

Proof. We know from Theorem 4.3.1 that partial network tomography should be performed on a partial network $\mathcal{S}$ with at least $\Big|\mathcal{E}(\mathcal{S})\Big| = 3$, and by Definition 3.3.1, each $v_i \in \mathcal{E}(\mathcal{S})$ is incident to one link, hence there are at least $\gamma_{\mathcal{S}} \geq 3$. Now suppose that $\kappa_{\mathcal{S}} = 2$, then the measurement matrix $\boldsymbol{A}^{\mathcal{S}}$ for partial network $\mathcal{S}$ is $\kappa_{\mathcal{S}} \times \gamma_{\mathcal{S}}$ with $\kappa_{\mathcal{S}} < \gamma_{\mathcal{S}}$. Thus, $\boldsymbol{A}^{\mathcal{S}}$ is rank-deficient and cannot be reduced to a full-rank matrix, consequently $\boldsymbol{x}$ cannot be uniquely solved. □

For partial network tomography, the problem is formulated as

$$\boldsymbol{y}^{\mathcal{S}} = \boldsymbol{A}^{\mathcal{S}}\boldsymbol{x}^{\mathcal{S}}, \tag{4.5}$$

where $\boldsymbol{y}^{\mathcal{S}}$, $\boldsymbol{A}^{\mathcal{S}}$, and $\boldsymbol{x}^{\mathcal{S}}$ are end-to-end measurements vector, measurement matrix, and link-level (or segment) measurements vector, respectively, for partial network $\mathcal{S} \subset G$. If the topology is unidentifiable (see Theorem 3.4.1), the system then can be reduced to (4.6) by removing redundant columns. Hence, $\tilde{\boldsymbol{x}}^{\mathcal{S}}$ corresponds to segment metrics, and the reduced partial tomography can be formulated as

$$\boldsymbol{y}^{\mathcal{S}} = \tilde{\boldsymbol{A}}^{\mathcal{S}}\tilde{\boldsymbol{x}}^{\mathcal{S}}, \tag{4.6}$$

where $\left|\tilde{\boldsymbol{x}}^{\mathcal{S}}\right| = \left|\boldsymbol{y}^{\mathcal{S}}\right|$.

Algorithm 1: Partial Network Tomography

```
Inputs  : Network $G$, end-to-end measurement vector $\boldsymbol{y}$ with $y_i$ for each path
          $p_i \in \mathcal{P}_m(G)$
Output  : A set of inferred link-level metrics ($\boldsymbol{x}^{\mathcal{S}}$) and/or segment metrics ($\tilde{\boldsymbol{x}}^{\mathcal{S}}$) for
          partial network $\mathcal{S}$
Initialize:
$\mathcal{P}_m(G) \leftarrow$ all measured paths $p_i \in \mathcal{P}(G)$
$\kappa_G \leftarrow \left|\mathcal{P}_m(G)\right|$
$\boldsymbol{A} \leftarrow$ measurement matrix
1  while $r(\boldsymbol{A}) < \gamma_G$
2      find sub-network $\mathcal{S}$ s.t. $\left|\mathcal{E}(\mathcal{S})\right| \geq 3$ and $\kappa_{\mathcal{S}} \geq 3$
3      if $r(\boldsymbol{A}^{\mathcal{S}}) < \gamma_{\mathcal{S}}$ then
4          if $d(v_i) < 3, \exists v_i \in \mathcal{R}(\mathcal{S})$ then
5              reduce $\boldsymbol{A}^{\mathcal{S}}$ to $\tilde{\boldsymbol{A}}^{\mathcal{S}}$ by removing redundant column/s
6              solve for $\tilde{\boldsymbol{x}}^{\mathcal{S}}$ in (4.6)
7              return $\tilde{\boldsymbol{x}}^{\mathcal{S}}$
8          else
9              system is inconsistent
10             go to line 2
11     else
12         solve for $\boldsymbol{x}^{\mathcal{S}}$ in (4.5)
13         return $\boldsymbol{x}^{\mathcal{S}}$
```

The process of the proposed partial network tomography is illustrated in Algorithm 1. The algorithm starts by finding any partial network $\mathcal{S}$ with $\left|\mathcal{E}(\mathcal{S})\right| \geq 3$ and $\kappa_{\mathcal{S}} \geq 3$ (line 2). For such partial network $\mathcal{S}$, it checks for the rank of its measurement matrix $r(\boldsymbol{A}^{\mathcal{S}})$. If it is full-rank, it directly solves $\boldsymbol{x}^{\mathcal{S}}$ using (4.5) (line 10-12). Otherwise, if $\boldsymbol{A}^{\mathcal{S}} < \gamma_{\mathcal{S}}$ due to violation of topological condition in Theorem 3.4.1 (topology is unidentifiable), then $\boldsymbol{A}^{\mathcal{S}}$ will be reduced to $\tilde{\boldsymbol{A}^{\mathcal{S}}}$ by removing redundant columns (line 4-5). The algorithm then solves for $\tilde{\boldsymbol{x}^{\mathcal{S}}}$ using $\tilde{\boldsymbol{A}^{\mathcal{S}}}$ in (4.6). Otherwise, if the system (4.5) is inconsistent due to having dependent measurements, the algorithm finds another partial network (lines 8-10).

Complexity Analysis For a $\kappa_G \times \gamma_G$ measurement matrix with $\kappa_G = \gamma_G$, the time complexity of solving (3.1) using LU decomposition [210] is $\mathcal{O}(\kappa_G^2)$. Thus, in the worst-case scenario, Algorithm 1 runs in $\mathcal{O}(\vartheta\kappa_G^2)$, where ϑ is the total number of partial networks with $\left|\mathcal{E}(\mathcal{S})\right| \geq 3$ and $\kappa_{\mathcal{S}} \geq 3$.

Example 4.3.1. *To further illustrate Algorithm 1, consider a partial network $\mathcal{S}$ from the network shown in Figure 4.1 with $\mathcal{E}(\mathcal{S}) = \{v_3, v_4, v_6\}$. In such a partial network, we have $\gamma_{\mathcal{S}} = 4$, therefore, in order to identify all link-level metrics, we need $\kappa_{\mathcal{S}} = 4$ independent measurements. However, with $\left|\mathcal{E}(\mathcal{S})\right| = 3$, the maximum number of measurements is $\kappa_{\mathcal{S}} = 3$ (see proof of Theorem 4.3.1). As a result, not all link-level metrics can be uniquely identified, instead, segments can be identified. The matrix for such partial network with $\mathcal{P}_m(\mathcal{S}) = \{p_1, p_3, p_5\}$ is*

$$\boldsymbol{A}^{\mathcal{S}} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}. \tag{4.7}$$

It is clear that (4.7) *has one redundant column. The matrix then can be reduced to* (4.8) *by removing the redundant column and assigning the two links $e_3, e_5 \in E(G)$ to a segment s_{v_1,v_6}.*

$$\tilde{\boldsymbol{A}^{\mathcal{S}}} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}. \tag{4.8}$$

The complete system then will be

$$\begin{pmatrix} y_1 \\ y_3 \\ y_5 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_{s_{v_1,v_6}} \end{pmatrix}. \tag{4.9}$$

The system in (4.9) *now can be used to uniquely solve for x_1, x_2, and $x_{s_{v_1,v_6}}$.*

4.4 Delay Estimation with DNN

To further complement the partial network tomography and infer the full path-level performance of the in-vehicle network, we propose to use a DNN approach (see Chapter 2.5) to estimate the performance of unmeasured paths given the available measurements. Specifically, we leverage DNN to tackle the problem of rank-deficient matrix $\boldsymbol{A}$. In particular, we use DNN to estimate the performance of unmeasured paths $\mathcal{P}(G)\backslash\mathcal{P}_m(G)$ given only the performance of the set of measured paths $\mathcal{P}_m(G)$.

A neural network is utilised in [165] to estimate path-level performance based on actively measuring parts of the network. Our approach is different to the one presented in [165] in that it deals with varying performance characteristics in the network, and does not rely on active measurements. Moreover, in our proposal, end-to-end measurements are used to extract link-level performance metrics through partial network tomography, and as a result, training our neural network-based model is done with both end-to-end and link-level data.

In the following, we explain different approaches that can be used to infer the link-level performance of the network. These approaches are based on algebraic and pure DNN-based solutions.

4.4.1 DNN-based Algebraic Tomography

In order to infer the link-level performance $x_i \in \boldsymbol{x}, \forall e_i \in E(G)$, the DNN-based algebraic tomography seeks to solve the system in (3.1) by taking advantage of DNN to compensate for the deficiency of the measurement matrix. This approach can further be categorised into two approaches: Neural Network Delay Estimation (NNDE) and Neural Network Delay Tomography (NNDT).

4.4.1.1 Neural Network Delay Estimation (NNDE)

In this type, DNN is utilised to estimate the performance of unmeasured paths $\mathcal{P}(G)\backslash\mathcal{P}_m(G)$. In particular, when there are limited end-to-end available measurements ($\kappa_G < \gamma_G$), the DNN takes these measurements as input to estimate the performance of unmeasured paths $\mathcal{P}(G)\backslash\mathcal{P}_m(G)$. The DNN structure in this case is shown in Figure 4.4. Specifically, the neural network is fed with the available κ_G end-to-end measurements $\boldsymbol{y}$ for paths in $\mathcal{P}_m(G)$ to estimate the performance of the remaining $\gamma_G - \kappa_G$ unmeasured paths in $\mathcal{P}(G)\backslash\mathcal{P}_m(G)$. For example,

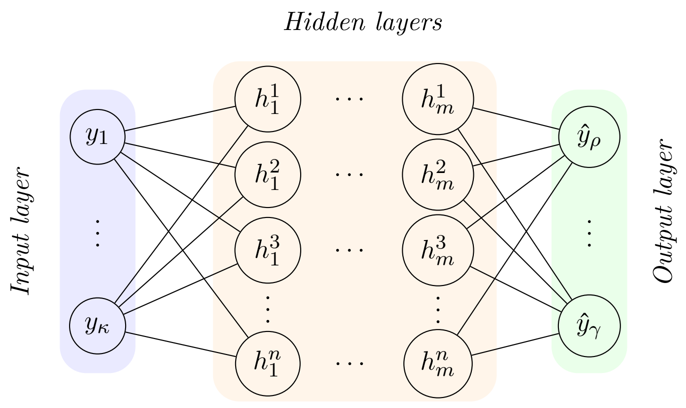


Figure 4.4: DNN structure in NNDE. The input to the DNN is a set of end-to-end measurements $\boldsymbol{y}$ available from the edge nodes $\mathcal{E}(G)$ in the in-vehicle network G. The output is the set of estimated measurements $\hat{\boldsymbol{y}}$ for unmonitored paths $\mathcal{P}(G)\backslash\mathcal{P}_m(G)$. Note that $\rho := \kappa_G + 1$.

for the network shown in Figure 4.1, suppose that $\mathcal{P}_m(G) = \{p_1, p_2, p_5\}$, the input to the neural network then will be a vector $\boldsymbol{y} = (y_1, y_2, y_5)^T$. The neural network then outputs estimations for unmeasured paths in $\mathcal{P}(G)\backslash\mathcal{P}_m(G)$, in this case, the output will be $\hat{y}_4$ and $\hat{y}_6$.

4.4.1.2 Neural Network Delay Tomography (NNDT)

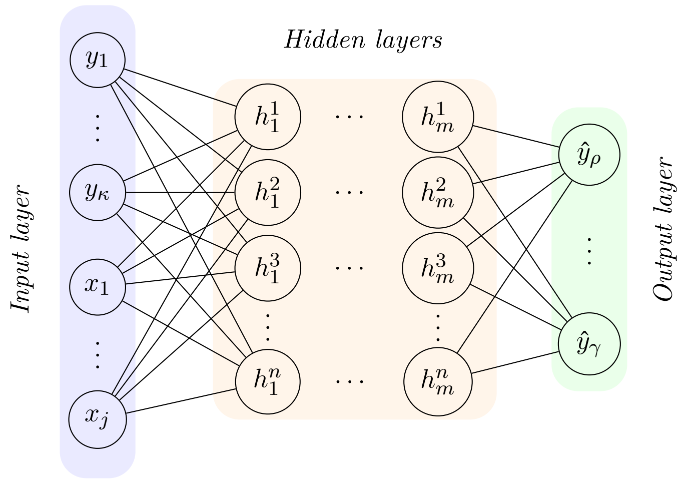


Figure 4.5: DNN structure in NNDT. The input to the DNN is a set of end-to-end measurements $\boldsymbol{y}$ available from the edge nodes $\mathcal{E}(G)$ in the in-vehicle network G, in addition to the set of inferred link-level metrics $\boldsymbol{x}$ obtained from partial network tomography. The output is the set of estimated measurements $\hat{\boldsymbol{y}}$ for unmonitored paths $\mathcal{P}(G)\backslash\mathcal{P}_m(G)$. Note that $\rho := \kappa_G + 1$.

Using partial network tomography explained in Chapter 4.3, one can infer proportion of link-level performance. Such inferred metrics can be further combined with NNDE to improve the estimation accuracy of unmeasured paths. Therefore, the input layer of DNN in this case includes $\boldsymbol{y}$ for paths in $\mathcal{P}_m(G)$ in addition to the inferred link-level or segment metrics $\boldsymbol{x}^{\mathcal{S}}$. Figure 4.5 shows the structure of DNN used in this type. As in NNDE, the network estimates the values for unmeasured paths in $\mathcal{P}(G)\backslash\mathcal{P}_m(G)$.

4.4.2 DNN-based Tomography

In contrast to DNN-based algebraic tomography, DNN here will be directly used to infer the set of link-level metrics $\boldsymbol{x}$ from the set of end-to-end measurements $\boldsymbol{y}$ without the need to solve (3.1). The input for the neural network in this case is $\boldsymbol{y}$ whilst the output is $\boldsymbol{x}$ as depicted in Figure 4.6.

Using this approach, there is no need to have a full-rank $\boldsymbol{A}$. Also, there is no condition on the number of end-to-end measurements. This approach can infer the link-level performance for all γ_G links, given that each link is traversed by *at least one* path in $\mathcal{P}_m(G)$.

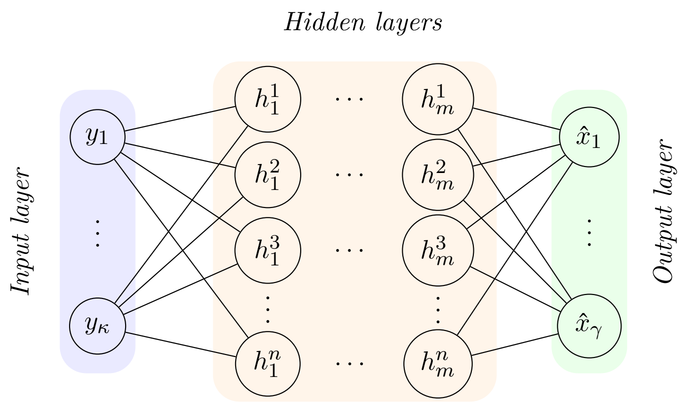


Figure 4.6: DNN structure in DNN-based tomography. The input to the DNN is a set of available end-to-end measurements $\boldsymbol{y}$ provided by edge nodes $\mathcal{E}(G)$ in an in-vehicle network G. The output is a set of estimated link-level performance $\hat{\boldsymbol{x}}$.

4.5 Performance Evaluation

This section first describes the experiment setup used to evaluate the proposed approaches. Second, simulation results are discussed, where the proposed approaches are evaluated to infer the path-level as well as the link-level performances.

4.5.1 Experiment Setup

The following describes the network simulation setup as well as the setup for the deep neural network and pre-processing of the dataset. Note that this study focuses on delay tomography. However, the approach can be applied to other metrics, e.g., packet loss rate.

4.5.1.1 Network Simulation

To evaluate the performance of the proposed approach, we conducted simulations using OMNeT++ [198] and CoRE4INET [211]. An in-vehicle network topology with an Ethernet backbone as shown in Figure 4.7 was simulated. Parameters for the simulated network are summarised in Table 4.2. Different traffic types using TSN standards as well as non-TSN traffic (BE traffic) (see Chapter 2.1.2.6) were used. The simulated network G consisted of

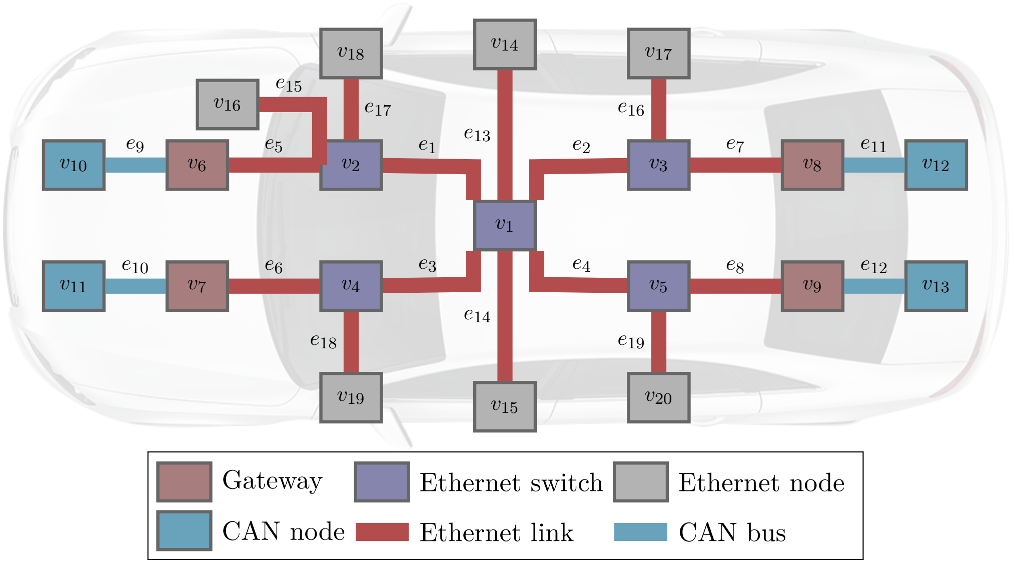


Figure 4.7: Ethernet-backbone in-vehicle network topology used in the simulation to evaluate DNN-based partial tomography.

$\eta_G = 20$ total number of nodes with $\left|\mathcal{E}(G)\right| = 11$ edge nodes and $\left|\mathcal{R}(G)\right| = 9$ intermediate nodes. The topology had in total $\gamma_G = 19$ links. This topology is based on a synthetic in-vehicle network topology derived from [212]. The simulation ran for 5000 times with random switches' processing delays uniformly distributed in the range $\left[10\mu s, 80\mu s\right]$ to ensure that we have different network measurements.

Table 4.2: Network parameters used in simulated scenarios for evaluating partial and DNN-based network tomography in in-vehicle networks.

Parameter	Value
Number of nodes (η_G)	20
Number of edge nodes ($\left\|\mathcal{E}(G)\right\|$)	11
Number of intermediate nodes ($\left\|\mathcal{R}(G)\right\|$)	9
Number of CAN buses ($\left\|B\right\|$)	4
Number of links (γ_G)	19
CAN bus bandwidth	1 Mbps
Ethernet link bandwidth	100 Mbps
Switch processing delay	Uniformly distributed between $10\mu s$ and $80\mu s$

Two types of traffic were used in the simulation based on Ethernet and CAN traffic. For the Ethernet traffic, different traffic types based on TSN standards, including AVB and IEEE802.1Q as well as best effort traffic, were used (see Chapter 2.1.2.6 for a detailed description of such standards). For non-TSN traffic, only BE traffic was used. Simulated traffic passes through different paths with $\mathcal{P}(G) = \{p_1, p_2, \ldots, p_{10}\}$ as illustrated in Table 4.3.

Table 4.3: Paths and traffic types used in simulating in-vehicle network with TSN traffic.

Path	Source	Destination	Traffic type	Payload
$p_1 = \{e_1, e_2, e_5, e_7, e_9, e_{11}\}$	v_{10}	v_{12}	CAN, BE	8 Bytes
$p_2 = \{e_3, e_4, e_6, e_8, e_{10}, e_{12}\}$	v_{11}	v_{13}	CAN, BE	8 Bytes
$p_3 = \{e_1, e_{14}, e_{17}\}$	v_{18}	v_{15}	AVB class A	393 Bytes
$p_4 = \{e_3, e_{14}, e_{18}\}$	v_{19}	v_{15}	AVB class A	393 Bytes
$p_5 = \{e_{13}, e_{14}\}$	v_{15}	v_{14}	AVB class B	786 Bytes
$p_6 = \{e_1, e_4, e_{15}, e_{19}\}$	v_{16}	v_{20}	IEEE 802.1Q	100 Bytes
$p_7 = \{e_2, e_4, e_{16}, e_{19}\}$	v_{17}	v_{20}	IEEE802.1Q	100 Bytes
			BE	46 Bytes

Table 4.3: Paths and traffic types used in simulating in-vehicle network with TSN traffic.

Path	Source	Destination	Traffic type	Payload
$p_8 = \{e_1, e_3, e_{17}, e_{18}\}$	v_{18}	v_{19}	BE	46 Bytes
$p_9 = \{e_1, e_2, e_{16}, e_{17}\}$	v_{18}	v_{17}	BE	46 Bytes
$p_{10} = \{e_2, e_3, e_{16}, e_{18}\}$	v_{19}	v_{17}	BE	46 Bytes

The payload value of CAN traffic is the maximum value which is 8 Bytes (see Chapter 2.1.2.2). As the AVB traffic is used for audio and video applications, its payload value is higher with 393 Bytes – 786 Bytes. On the other hand, since the IEEE802.1Q traffic is used for time-sensitive applications (see Chapter 2.1.2.6), it does not require a high payload value as compared with AVB, whereas the payload for the best effort traffic is 46 Bytes which is the minimum payload for Ethernet messages [30] as this kind of traffic is mainly used here for the measurements of the network performance.

Note that path p_7 has two different traffics: IEEE802.1Q and BE; to distinguish between the two types, we denote the path with IEEE802.1Q traffic and the one with BE traffic respectively by ${}^1p_7$ and ${}^2p_7$. For each traffic, we recorded its end-to-end delay to construct the dataset we used for training the DNN model.

4.5.1.2 Neural Network Model and Data Processing

The structures of the used DNNs for NNDE and NNDT are similar to the ones shown in Figure 4.4 and Figure 4.5. These DNNs consist of two hidden layers, where the number of hidden neurons in each layer is $2\kappa_G - 1$ in case of NNDE and $2(\kappa_G + \left|\boldsymbol{x}^{\mathcal{S}}\right|) - 1$ in NNDT. The model has trained over 1000 epochs. It is worth noting that early stopping [179], with a patience value of 30 epochs, was utilised to avoid model overfitting. We used the dataset of 5000 samples generated by our simulation, then split it into 60%, 25% and 15% as training, validation and testing sets, respectively. Moreover, *MinMaxScaler* [213] was applied to scale all the data values between 0 and 1. The training data was used to train the model, the validation data with early stopping was used for cross-validation, and the test set was used to test the model performance on the new data that the model had not seen before. In addition, we used

ReLU activation function (2.2) for all hidden layers and linear activation function (4.10) for the output layer.

$$f(z) = z. \tag{4.10}$$

For backpropagation, *Adam* optimisation function [214] was used to update the weights and minimise the loss function (2.6). The model was trained using mini-batches of size 40, with a learning rate of $\alpha = 0.001$. Summary of the DNNs settings are shown in Table 4.4

Table 4.4: Parameters used for the DNN model.

Parameter	Value
Number of hidden layers	2
Number of neurons in each hidden layer	NNDE: $2\kappa_G - 1$, NNDT: $2(\kappa_G + \left\|\boldsymbol{x}^{\mathcal{S}}\right\|) - 1$
Hidden layers activation function ($f(z)$)	ReLU (2.2)
Output layer activation function ($f_o(z)$)	Linear (4.10)
Backpropagation optimisation function	Adam [214]
Size of training set	60%
Size of validation set	25%
Size of test set	15%
Feature scalar	MinMaxScaler
Learning rate (α)	0.001
Patience value	30

4.5.1.3 Partial Network Tomography Settings

To perform partial network tomography for the network shown in Figure 4.7, Algorithm 1 used partial network $\mathcal{S}$ with $\mathcal{E}(\mathcal{S}) = \{e_{17}, e_{18}, e_{19}, e_{20}\}$ and $\mathcal{P}_m(\mathcal{S}) = \{^2p_7, p_8, p_9, p_{10}\}$ which satisfies both of the conditions in Theorem 4.3.1 and Theorem 4.3.2. Using (4.5), the segments' metrics $\boldsymbol{x}^{\mathcal{S}}$ for $s_{v_1,v_{17}}$, $s_{v_1,v_{18}}$, $s_{v_1,v_{19}}$, and $s_{v_1,v_{20}}$ were inferred. The inferred metrics were then added to the input of the neural network, in addition to the available end-to-end measurements, in the NNDT case as shown in Figure 4.5.

4.5.2 Results

This section illustrates the results of applying the proposed approaches to the simulated environment described above. First, the results for path-level delay estimation are discussed. In addition, we compare the proposed approach with the one presented in [165]. Next, estimating the link-level delay performance is evaluated using the proposed algebraic and DNN tomography approaches discussed in Chapter 4.4.1 and Chapter 4.4.2.

4.5.2.1 Path-Level Estimation Results

The following evaluates the proposed approaches to estimating the end-to-end, path-level, and delay performance when there is a limited number of available measurements.

Training Results Using cross-validation, we first evaluate the model's learning performance to ensure that it is not overfitting. The results are shown in Figure 4.8 and Figure 4.9 for TSN and non-TSN scenarios, respectively. These results show that in both NNDE and NNDT cases, training and validation are almost aligned, and the model stops training before it starts to overfit with maximum number of epochs equal to 300 epochs in case of NNDE for TSN when $\kappa_G = 4$ as shown in Figure 4.8 and 448 epochs in case of NNDE for non-TSN when the number of available measurements is $\kappa_G = 5$ as shown in Figure 4.9. We can see that when the number of measured paths, i.e., κ_G, increases, the model with TSN learns faster than with non-TSN traffic as shown in Figure 4.8. The reason behind this is that with TSN, the traffic characteristics (including the end-to-end delays) are deterministic, therefore, it is easier for the DNN to learn such characteristics than with non-TSN traffic which has non-deterministic behaviour.

In addition, in all different values of κ_G, the number of epochs in NNDT is less than in NNDE. This means that the more information added from the partial network tomography in the NNDT case, allowed the model to learn quicker than when the only available information is the path-level measurements (which is the case with NNDE).

Testing Results After training our model, we evaluate its performance on new data points[1] that the model has never seen before using the test set. In Table 4.5 and Table 4.6, we show the Mean Absolute Percentage Error (MAPE) values for model estimations on the test set (here,

[1] In machine and deep learning, a data point refers to an entry of the dataset.

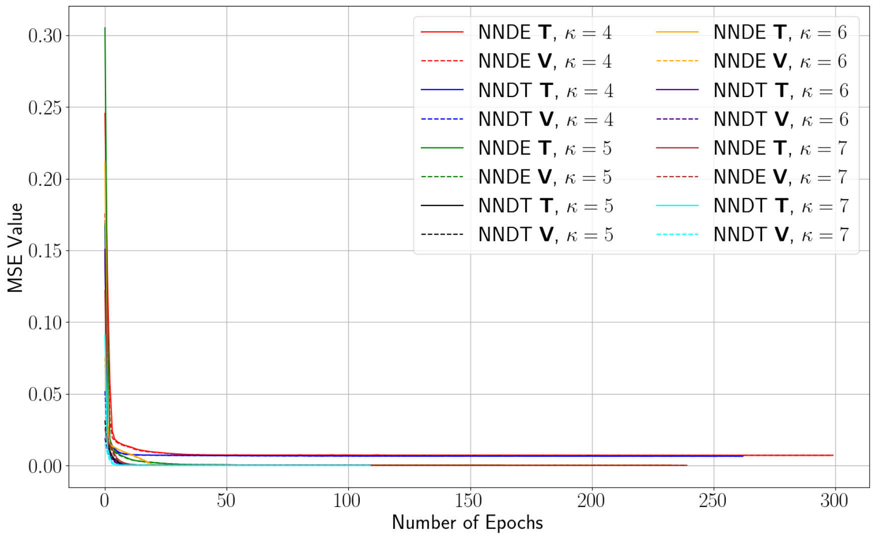


Figure 4.8: Training performance with cross-validation in TSN scenario. **T**: training, **V**: validation. The y-axis represents the MSE (used as a loss function for training).

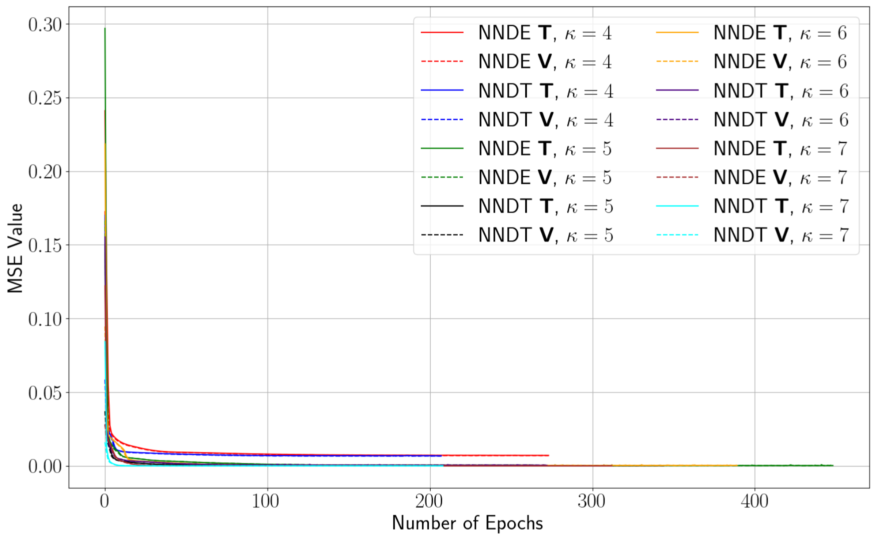


Figure 4.9: Training performance with cross-validation in non-TSN scenario. **T**: training, **V**: validation. The y-axis represents the MSE (used as a loss function for training).

we report the best performance results, see Appendix B for more results). MAPE is computed as

$$\text{MAPE} = 100\left(\frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right|\right). \tag{4.11}$$

In the presented scenario, partial network tomography is performed when 40% or more of the network is measured with $\mathcal{P}_m(G) = \{{}^2p_7, p_8, p_9, p_{10}\}$, hence $\kappa_G = 4 > 3$ which satisfies the condition that the number of end-to-end measurements should be at least 3 (see Theorem 4.3.2). As seen in Table 4.5 and Table 4.6, for both TSN and non-TSN scenarios, the error rates of both models, NNDE and NNDT, decrease when the number of available measurements increases, reaching up to 99% accuracy when 50% or more of the network is measured. Moreover, both NNDE and NNDT have similar performance in estimating the unmeasured paths, sometimes with slight improvement when using NNDT.

Table 4.5: MAPE values for test results in **TSN** scenario.

Measured ratio (%)	Measured paths ($\mathcal{P}_m(G)$)	NNDE (%)	NNDT (%)
10	$\mathcal{P}_m = \{{}^1p_7\}$	10.009	N/A
20	$\mathcal{P}_m = \{p_8, {}^2p_7\}$	8.225	N/A
30	$\mathcal{P}_m = \{{}^1p_7, p_8, p_{10}\}$	4.854	N/A
40	$\mathcal{P}_m = \{{}^2p_7, p_8, p_9, p_{10}\}$	2.522	2.395
50	$\mathcal{P}_m = \{p_5, {}^2p_7, p_8, p_9, p_{10}\}$	1.176	1.162
60	$\mathcal{P}_m = \{p_5, p_6, {}^2p_7, p_8, p_9, p_{10}\}$	0.780	0.819
70	$\mathcal{P}_m = \{p_3, p_5, p_6, {}^2p_7, p_8, p_9, p_{10}\}$	0.665	0.606

Table 4.6: MAPE values for test results in **non-TSN** scenario.

Measured ratio (%)	Measured paths ($\mathcal{P}_m$)	NNDE (%)	NNDT (%)
10	$\mathcal{P}_m = \{{}^1p_7\}$	10.244	N/A
20	$\mathcal{P}_m = \{p_8, {}^2p_7\}$	8.881	N/A
30	$\mathcal{P}_m = \{{}^1p_7, p_8, p_{10}\}$	5.292	N/A
40	$\mathcal{P}_m = \{{}^2p_7, p_8, p_9, p_{10}\}$	2.806	1.886
50	$\mathcal{P}_m = \{p_5, {}^2p_7, p_8, p_9, p_{10}\}$	0.934	0.492
60	$\mathcal{P}_m = \{p_5, p_6, {}^2p_7, p_8, p_9, p_{10}\}$	0.389	0.562

Table 4.6: MAPE values for test results in **non-TSN** scenario.

Measured ratio (%)	Measured paths ($\mathcal{P}_m$)	NNDE (%)	NNDT (%)
70	$\mathcal{P}_m = \{p_3, p_5, p_6, {}^2p_7, p_8, p_9, p_{10}\}$	0.143	0.155

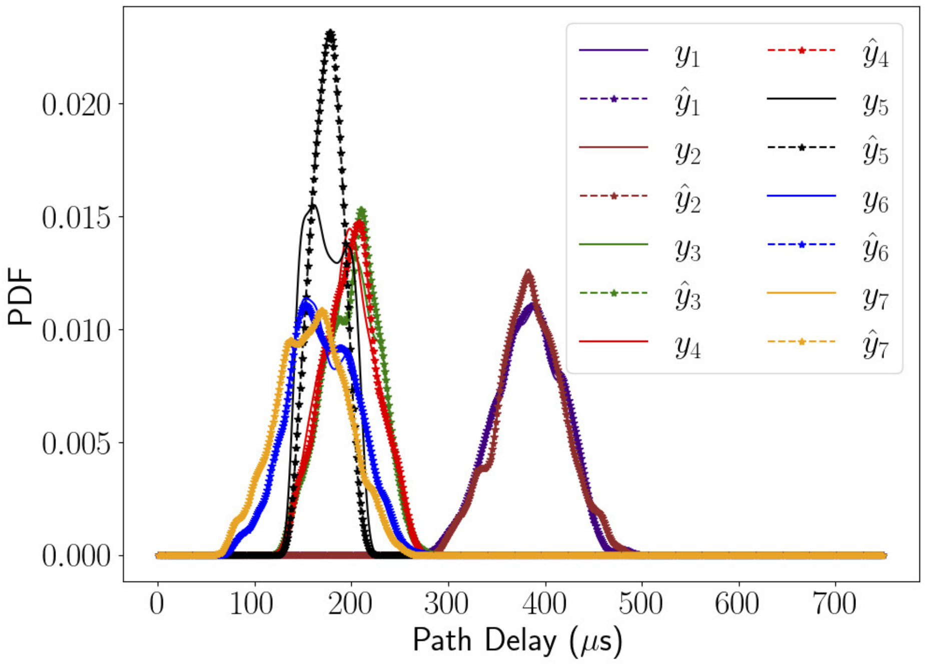


Figure 4.10: PDF of NNDT in case 40% of the network is measured.

Figure 4.10 and Figure 4.11 further show the Probability Density Function (PDF) of the estimated paths' delays in NNDT when 40% and 70% of the network is measured, respectively, for the TSN scenario. We can see in Figure 4.10 that the estimated and actual delays are very close to each other except for y_5 and $\hat{y}_5$ where there is a marginal difference. This is because the available measurements are for paths in $\mathcal{P}_m(G)$ that do not share any common link with path $p_5 \in \mathcal{P}(G) \backslash \mathcal{P}_m(G)$, as compared with other paths where they share one or more links with at least one path in $\mathcal{P}_m(G)$. On the other hand, when more paths are measured, as shown in Figure 4.11 (70% of the network is measured), the distribution of actual and estimated values are nearly overlapped. Note that in both cases, $\hat{y}_7$ is accurately estimated for ${}^1p_7$ due to the available measurement of the same path but different traffic type (i.e., ${}^2p_7$ for best effort traffic).

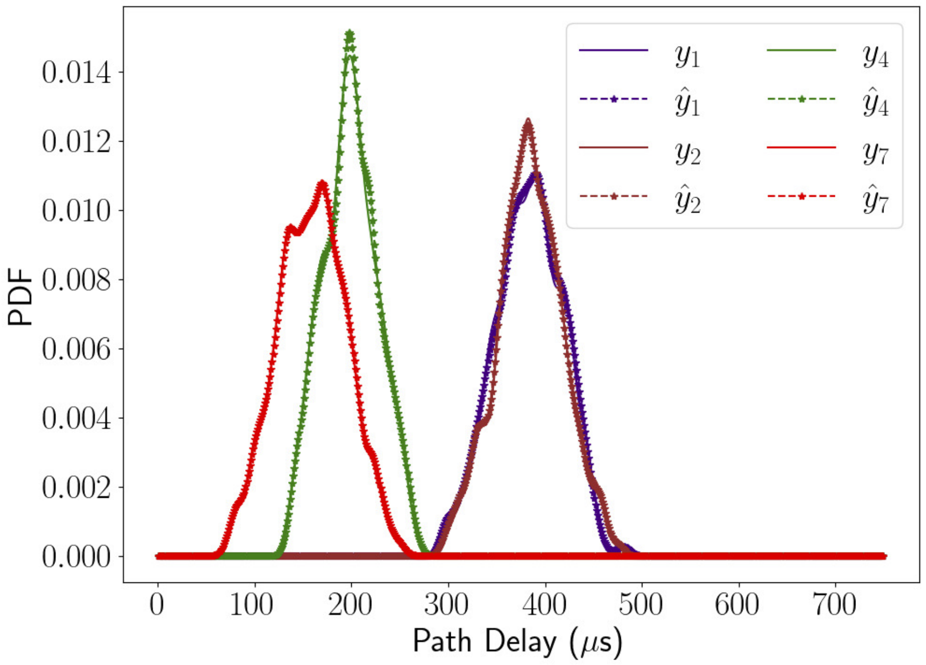


Figure 4.11: PDF of NNDT in case of 70% of the network is measured.

To further evaluate the contribution of each path's measurement in estimating the overall network delay performance, we show in Figure 4.12 the MAPE value for each measured path in $\mathcal{P}_m(G)$ when only 10% of the network is measured, i.e., $\kappa_G = 1$. As shown, when $\mathcal{P}_m(G) = \{p_7\}$, the MAPE is at its lowest value for both TSN and non-TSN. In contrast, the highest MAPE value for TSN and non-TSN was achieved when $\mathcal{P}_m(G) = \{p_5\}$ and $\mathcal{P}_m(G) = \{p_4\}$, respectively. This is because, e.g., p_5 has only one link that is shared with two paths only in the network (i.e., link e_{14} is shared with p_3 and p_4). While p_7 has two segments ($s_{v_1,v_{17}}$ and $s_{v_1,v_{20}}$) shared with p_6, p_9 and p_{10}, in addition to two more links (e_2 and e_4) that are shared with p_1 and p_2.

Additionally, we compare the proposed approach with *NeuTomography* [165]. The neural network used in [165] is depicted in Figure 4.13; it takes as input a vector of binary entries corresponding to the set of nodes in the network. The input is a one-hot vector indicating the source and destination of the measured path (with 1 in corresponding source and destination positions, and 0 elsewhere), while the output of this neural network is the delay estimation of this path.

Table 4.7 shows the MAPE results using the approach proposed in [165] on the simulated

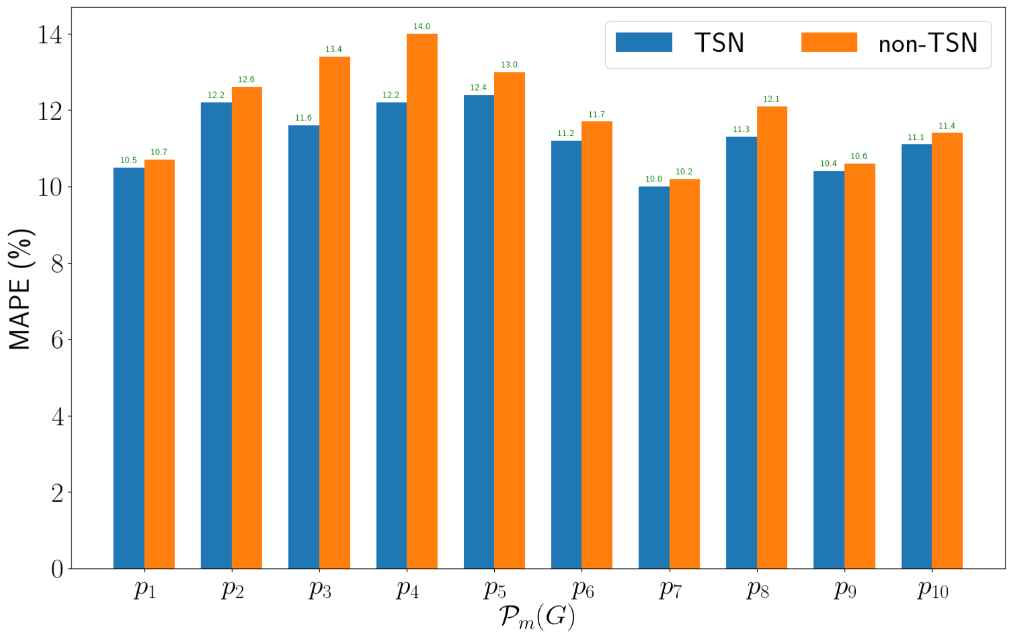


Figure 4.12: MAPE values on test set for NNDE when only 10% of the network is measured (i.e., $\kappa_G = 1$).

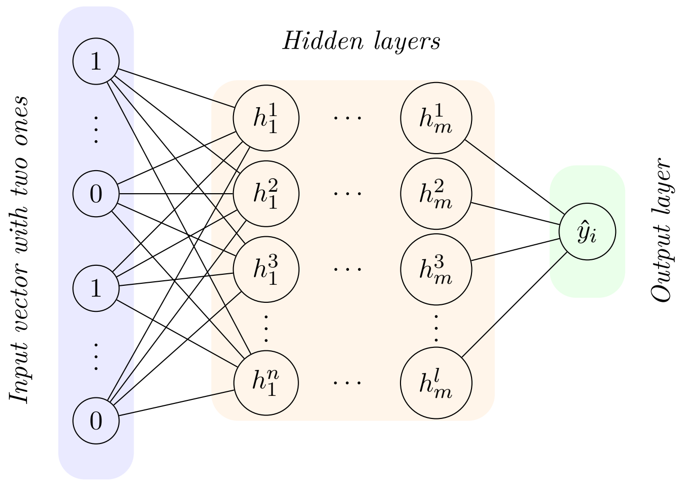


Figure 4.13: Neural network structure used in [165].

in-vehicle network shown in Figure 4.7. Here, like in [165], we evaluate the performance when 20%, 25%, and 30% of the network is measured. As shown in Table 4.7, NeuTomography yields large error values compared with the approach proposed in this thesis. The main reason for this is that NeuTomography only uses information about the source and destination. Such information cannot provide much useful information in the case of dynamic traffic features such as the ones used in automotive networks. In comparison, our approach uses the measured end-to-end performance across selected paths to help the neural network estimate the performances of unmeasured ones. In addition, the estimation performance can be further improved using the proposed partial tomography approach, which provides even more useful information about the delay performance in the network. Hence, better results are achieved using the approach proposed in this thesis.

Table 4.7: MAPE values using the neural network tomography approach proposed in [165].

Measured ratio (%)	MAPE (%)
20	102.6
25	56.4
30	34.4

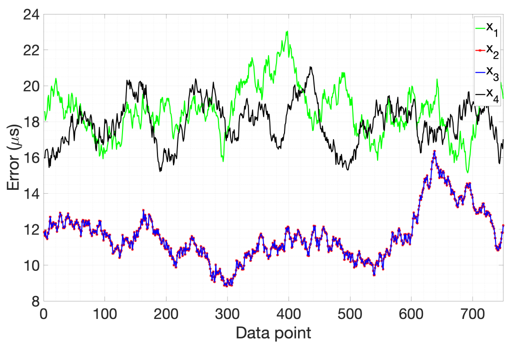
4.5.2.2 Link-Level Estimation Results

Here, the proposed approach is evaluated to infer the link-level delay metrics of the simulated in-vehicle network. In particular, we evaluate the two types of tomographic approaches discussed earlier, i.e., *DNN-based Algebraic Tomography* and *DNN-based Tomography.* The former uses the DNN structure as the one shown in Figure 4.4, while the latter uses the structure shown in Figure 4.6.

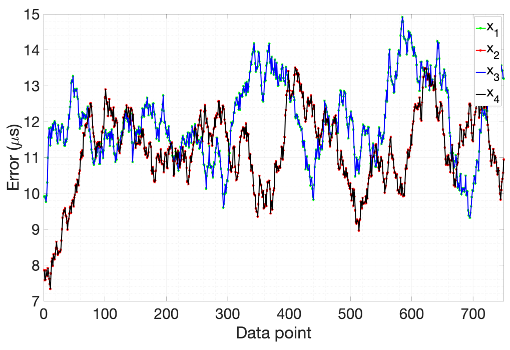
DNN-based Algebraic Tomography Results We compare the results obtained using the actual end-to-end delays with the ones obtained using the estimated end-to-end delays in the DNN-based algebraic tomography approach.

In Figure 4.14, we show the absolute error value of actual and inferred delay, $|x_i - \hat{x}_i|$, in case TSN traffic is used. We observe that as the number of measured paths (i.e., κ_G) increases, the performance appears to improve. For instance, when only one path is measured and the remaining were estimated using DNN, the maximum error is about $23.0\mu s$ as shown in Figure 4.14(a). The performance slightly improved when the number of measured paths increased to $\kappa_G = 2$ as shown in Figure 4.14(b) where the maximum error value is $14.9\mu s$.

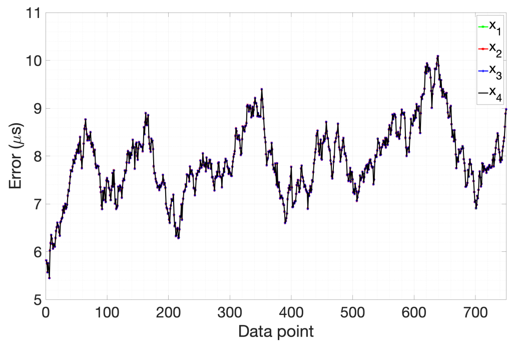
In Figure 4.14(c), the performance improved even further as the number of measured paths increased to $\kappa_G = 3$, in such case the maximum error is only $10.1\mu s$.



(a) $\kappa_G = 1$



(b) $\kappa_G = 2$

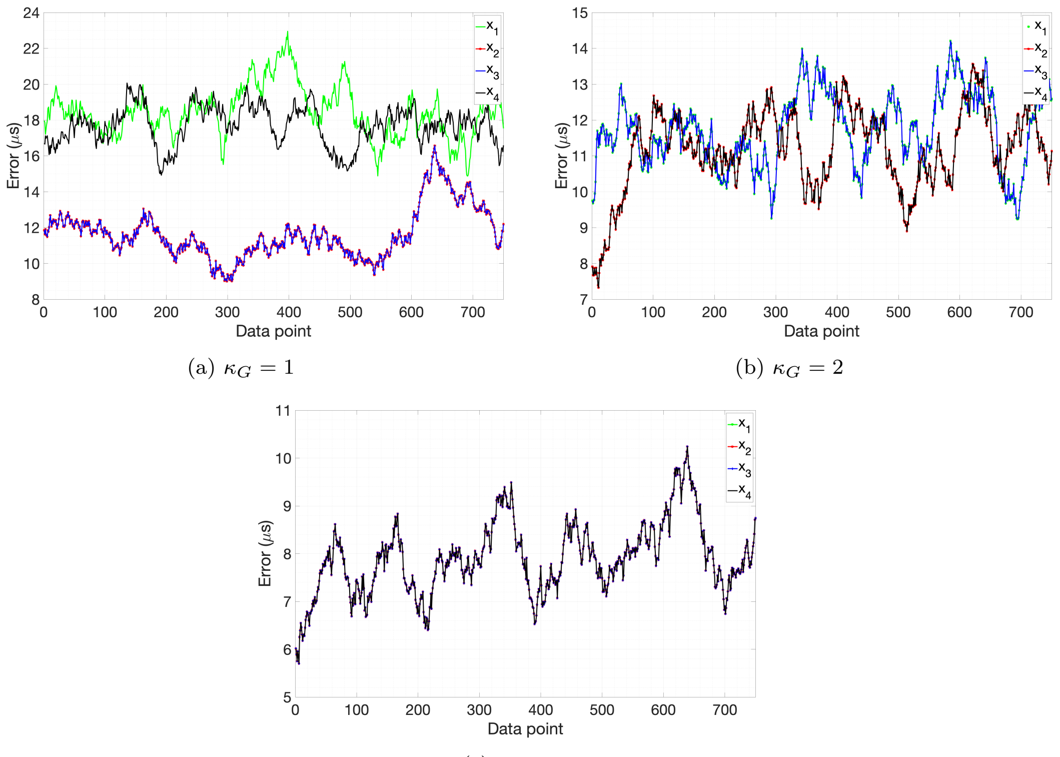


(c) $\kappa_G = 3$

Figure 4.14: Absolute error value averaged over 50 data points for DNN-based algebraic tomography approach. Scenario with **TSN** traffic. Here x_i represents $|x_i - \hat{x_i}|$ for link $e_i \in E(G)$.

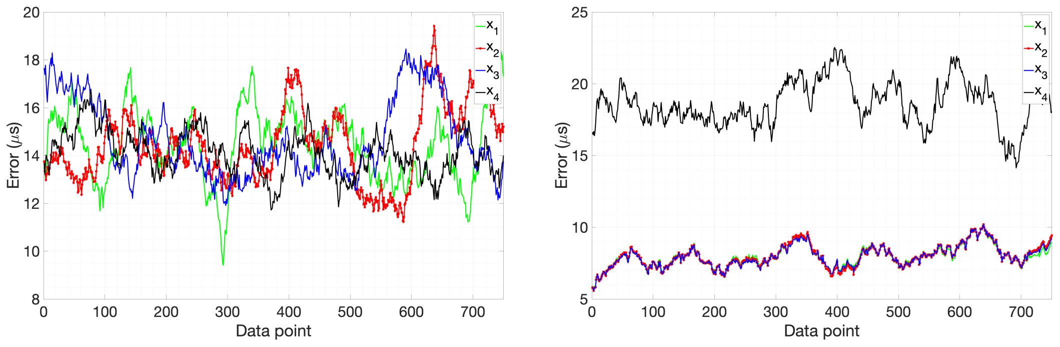
A similar observation was noted for the scenario with no TSN traffic, as shown in Figure 4.15.

DNN-based Tomography Results In this approach, only two paths were chosen to be measured to infer the link-level delays, i.e., $\mathcal{P}_m(G) = \{p_7, p_8\}$ (see Table 4.3). In Figure 4.16, the absolute error values of actual and inferred delays are depicted. The maximum error observed for the scenario with TSN traffic (see Figure 4.16(a)) is $19.4\mu s$ which corresponds to x_2, while the scenario with non-TSN traffic (see Figure 4.16(b)) has a maximum error of $22.5\mu s$ corresponding to x_4. As noted, the error value for the case with non-TSN traffic is larger than the case with TSN traffic. As mentioned earlier, this is because most TSN traffic has bounded delays, hence deterministic behaviour, which makes it easier for the neural network to predict the behaviour of such traffic.



(a) $\kappa_G = 1$ (b) $\kappa_G = 2$ (c) $\kappa_G = 3$

Figure 4.15: Absolute error value averaged over 50 data points for DNN-based algebraic tomography approach. Scenario with **non-TSN** traffic. Here x_i represents $|x_i - \hat{x_i}|$ for link $e_i \in E(G)$.



(a) With TSN traffic (b) With *no* TSN traffic

Figure 4.16: Absolute error value averaged over 50 data points for DNN-based tomography approach.

Discussion Comparing the results of both approaches (DNN-based algebraic tomography and DNN-based tomography), we found that the DNN-based algebraic tomography approach performed better than the DNN-based tomography approach in terms of estimation accuracy. As the DNN-based tomography approach in our scenario only measured two paths to predict the link-level delays, one can compare it with DNN-based algebraic tomography when $\kappa_G = 2$. Therefore, it can be observed that DNN-based algebraic tomography outperforms DNN-based tomography by about $4.5\mu s$ less estimation error. Because the DNN-based algebraic tomography approach needs at least γ_G number of end-to-end measurements to uniquely solve for $\boldsymbol{x}$ in (3.1), it uses more end-to-end measurements (either measured or estimated) than DNN-based tomography approach. As the number of end-to-end measurements increases, the uncertainty of the inferred link-level delay decreases. Hence, the DNN-based algebraic tomography approach yields more accurate results than the DNN-based tomography approach.

The comparison between the two proposed approaches to infer the link-level metrics presents a trade-off between inference accuracy and computational overhead. If devices in in-vehicle networks are powerful enough to perform intensive computation, then one should opt for a DNN-based algebraic tomography approach. Otherwise, the DNN-based tomography approach can be used, bearing in mind that it sacrifices accuracy with marginal error.

4.6 Summary

In this chapter, we have proposed a partial network tomography approach to infer link-level and/or path-segment metrics of partial networks using only the available end-to-end measurements. Moreover, we have proposed a deep neural network delay estimation approach to estimate the unmeasured end-to-end delay of an in-vehicle network. In the proposed approach, we have used a subset of measured paths as input to train the model and estimate the delay of remaining paths in the network. Additionally, we have used the inferred metrics from the partial tomography to estimate the performance of unmeasured paths. The results showed that by only measuring the end-to-end subset of the network, the overall performance can be accurately estimated with up to 99% accuracy. We believe that our approach is suitable for networks with variable traffic such as in-vehicle networks where direct monitoring of the full network is not desirable as such monitoring traffic may overwhelm the network.

The following chapter examines the proposed approaches in anomaly detection and localisation applications for in-vehicle networks.

5 Anomaly Detection and Localisation using Network Tomography

"I am one of those who think, like Nobel, that humanity will draw more good than evil from new discoveries."

Marie Curie

5.1 Overview

This chapter examines the proposed monitoring approaches, based on network tomography and deep neural networks, to an anomaly detection and localisation for in-vehicle networks that is based on central-gateway architecture. Any anomalous behaviour is defined as the behaviour that deviates from the normal behaviour of the network by exceeding some threshold values. In particular, this chapter investigates three types of proposed monitoring approaches: Delay Network Tomography (DNT), Binary Network Tomography (BNT), and Deep Neural Network (DNN).

5.2 Anomaly Detection and Localisation Problem

In this chapter, we focus on one of the new E/E architectures that are based on a central-gateway. Such architectures often divide the vehicle's components into multiple domains. Each domain is defined as follows (refer to Table 5.1 for the notations used in this chapter):

Table 5.1: Chapter 5 notations and their descriptions.

Notation	Description
Network & Graph Theory	
$G = (V, E)$	Network represented as graph G
$V(G)$	Set of vertices (nodes) in network G
$E(G)$	Set of edges (links) in network G
$\mathcal{E}(G) \subset V(G)$	Set of edge nodes in G (see Definition 3.3.1)
$\mathcal{R}(G) \subset V(G)$	Set of intermediate nodes in G (see Definition 3.3.1)
$\mathcal{P}(G)$	Set of all possible paths in network G
$\mathcal{P}_m(G) \subseteq \mathcal{P}(G)$	Set of measured paths in network G
$\mathcal{D}(G)$	Set of domains in in-vehicle network G
$p_i \in \mathcal{P}(G)$	i^{th} path connecting two non-adjacent nodes $v_i, v_j \in \mathcal{E}(G)$
$e_j \in E(G)$	j^{th} link in network G
$d_i \in \mathcal{D}(G)$	i^{th} domain in network G
Network Tomography	
$\boldsymbol{y}$	Vector of end-to-end measurements
$\boldsymbol{y}_m$	Vector of end-to-end measurements for paths in $\mathcal{P}_m(G)$
$\boldsymbol{x}$	Vector of link-level measurements
$\boldsymbol{A}$	Measurement matrix
$y_i \in \boldsymbol{y}$	End-to-end measurement of i^{the} path $p_i \in \mathcal{P}_m(G)$
$x_j \in \boldsymbol{x}$	Link-level metric of j^{the} link $e_j \in E(G)$
$r(\boldsymbol{A})$	Rank of the measurement matrix $\boldsymbol{A}$
$S(G) \in \{0, 1\}$	Status of network G, 0 is normal and 1 is anomalous
$S(p_i) \in \{0, 1\}$	Status of i^{th} path $p_i \in \mathcal{P}(G)$, 0 is normal and 1 is anomalous
$S(e_j) \in \{0, 1\}$	Status of j^{th} link $e_j \in E(G)$, 0 is normal and 1 is anomalous
Deep Neural Network	
$\boldsymbol{\beta}$	Bias vector
$\boldsymbol{W}$	Matrix of weights
$\boldsymbol{h}_i$	Vector of i^{th} hidden layer
$f(z)$	Hidden layer activation function

Table 5.1: Chapter 5 notations and their descriptions.

Notation	Description
Deep Neural Network	
$f_o(z)$	Output function
$\hat{\boldsymbol{y}}_u$	Vector of estimated end-to-end measurements for paths in $\mathcal{P}(G)\backslash\mathcal{P}_m(G)$
$\hat{\boldsymbol{x}}$	Vector of estimated link-level measurements
L	Loss function
α	Learning rate
Numbers & Cardinalities	
$\gamma_G := \Big\lvert E(G) \Big\rvert$	Total number of links in network G
$\kappa_G := \Big\lvert \mathcal{P}_m(G) \Big\rvert$	Total number of measured paths in G
a_{p_i}	Minimum path-level delay of i^{th} path $p_i \in \mathcal{P}(G)$
b_{p_i}	Maximum path-level delay of i^{th} path $p_i \in \mathcal{P}(G)$
a_{e_j}	Minimum link-level delay of j^{th} link $e_j \in E(G)$
b_{e_j}	Maximum link-level delay of j^{th} link $e_j \in E(G)$
$c_i \in \mathbb{Z}^+$	Priority level of CAN node $v_i \in \mathcal{E}(G)$

Definition 5.2.1. A domain $d_i \in \mathcal{D}(G)$, $i \in \{1, 2, \ldots, \Big\lvert \mathcal{D}(G) \Big\rvert\}$ is a subnetwork of in-vehicle network G that includes *at least one* intermediate node $v_i \in \mathcal{R}(G)$ and *two* edge nodes $v_j, v_k \in \mathcal{E}(G)$ where v_j, v_k are connected to v_i through links in $E(G)$. Multiple domains can share nodes in $\mathcal{R}(G)$.

The in-vehicle network we focus on in this study is the one with multiple domains interconnected with each other through a central-gateway, as shown in Figure 5.1. In this network, there are four domains, $\mathcal{D}(G) = \{d_1, d_2, d_3, d_4\}$, that correspond to *chassis*, *telematics*, *powertrain* and *body* domains.

Definition 5.2.2. We refer to a non-empty in-vehicle network $G \neq (\emptyset, \emptyset)$ with *at least two* domains (see Definition 5.2.1), i.e., $\Big\lvert \mathcal{D}(G) \Big\rvert \geq 2$, as a central-gateway in-vehicle network.

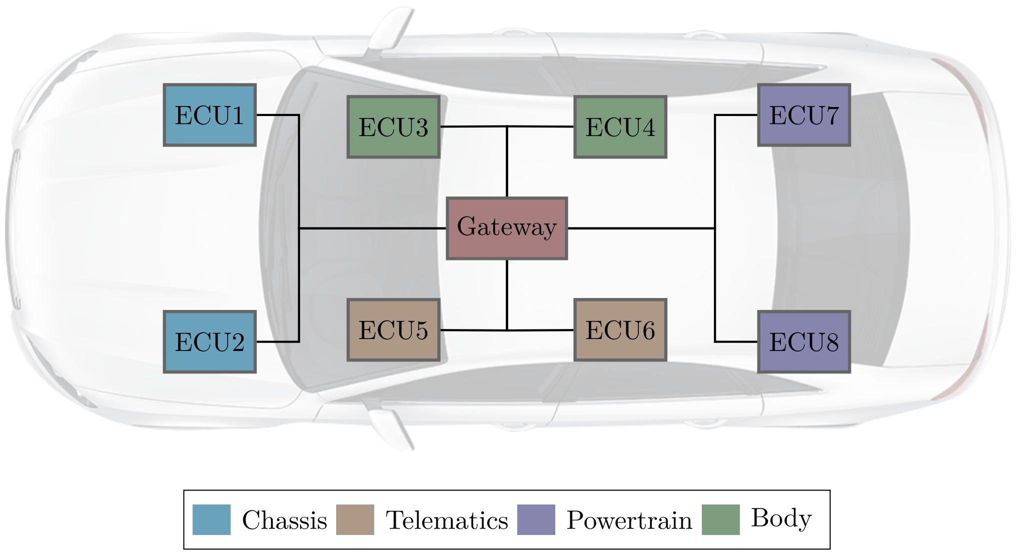


Figure 5.1: Example of in-vehicle network with four domains, based on central-gateway architecture.

Given an in-vehicle network G, let $S(G) \in \{0, 1\}$ be the status of such network, then our first objective is to detect if G is anomalous or normal, that is $S(G) = 1$ or $S(G) = 0$, respectively. To detect the status of G, its components have to be examined. Let $S(p_i) \in \{0, 1\}$ be the status of path $p_i \in \mathcal{P}(G)$, then we say that $S(G) = 1$ if there is *at least one* anomalous path p_i with $S(p_i) = 1$. Similarly, we say that $S(G) = 0$ if the status of all paths in $\mathcal{P}(G)$ is normal. Formally,

$$S(G) = \begin{cases} 0, & \text{if } S(p_i) = 0,\ \forall p_i \in \mathcal{P}(G) \\ 1, & \text{if } S(p_i) = 1,\ \exists p_i \in \mathcal{P}(G) \end{cases}, \tag{5.1}$$

where the status of each path can be determined by measuring the status of each link it passes through, i.e., $S(e_j) \in \{0, 1\}$. The status of p_i is given by

$$S(p_i) = \begin{cases} 0, & \text{if } S(e_j) = 0,\ \forall e_j \in p_i \\ 1, & \text{if } S(e_j) = 1,\ \exists e_j \in p_i \end{cases}, \tag{5.2}$$

$i \in \{1, \ldots, \left|\mathcal{P}(G)\right|\}$, $j \in \{1, \ldots, \gamma_G\}$,

and

$$S(e_j) = \begin{cases} 0, & \text{if } a_{e_j} \leq x_j \leq b_{e_j} \\ 1, & \text{if } x_j < a_{e_j} \text{ or } x_j > b_{e_j} \end{cases}, \tag{5.3}$$

where a_{e_j} and b_{e_j} are threshold values for the minimum and maximum link-level delay, respectively, of link $e_j \in E(G)$ in the normal network behaviour, and x_j is the link-level delay of link $e_j \in E(G)$.

Our main goal here is to determine the status of G, $S(G)$, as either normal ($S(G) = 0$) or anomalous ($S(G) = 1$). And in case $S(G) = 1$, the next goal is to locate the anomalous link/node within the network. To classify G based on (5.1), which in turn uses (5.2), all paths and links in $\mathcal{P}(G)$ and $E(G)$, respectively, have to be monitored. This monitoring overhead incurs an extra burden on the network. Therefore, inspired by the network tomography-based monitoring approach described in previous chapters, in this chapter, we propose to monitor a limited number of paths in $\mathcal{P}(G)$ in order to infer the overall network status $S(G)$. Hence, reducing the monitoring and measurement overheads.

However, in order to uniquely identify all link-level metrics, the measurement matrix $\boldsymbol{A}$ has to be of full rank (recall the conditions stated in Remark 3.4.1). To compensate for the rank deficiency, we leverage the DNN-based approach proposed in the last chapter.

5.3 Network Tomography and DNN-based Anomaly Detection and Localisation

The subsequent sections explain three types of tomographic approaches that can be used to monitor the in-vehicle network. The first (i.e., *Delay Network Tomography (DNT)*) uses actual delay measurements, while the second (i.e., *Binary Network Tomography (BNT)*) uses binary measurements, and the third (i.e., *Deep Neural Network (DNN)-based Tomography*) uses DNN to improve the rank deficiency of the measurement matrix. The first two approaches (i.e., DNT and BNT) assume that the measurement matrix is full rank, i.e., $r(\boldsymbol{A}) = \gamma_G$, while the third (i.e., DNN-based tomography) assumes that the measurement matrix is rank-deficient with $r(\boldsymbol{A}) < \gamma_G$. The main difference between DNT and BNT is that the former uses delay metrics (see Chapter 2.4.4.1) to measure the network performance, while the latter uses binary status (see Chapter 2.4.4.3). If the goal is to infer the exact link-level delay then DNT can be used, whereas BNT can be used if the goal is to identify the link-level status. Both DNT and BNT can be used to detect and locate the anomalous link, they only differ in the process in which each one achieves this goal as will be discussed next. DNN-based tomography can be used to compensate for the rank deficiency of the measurement matrix $\boldsymbol{A}$, hence it can be used for both DNT (*DNN for DNT*) and BNT (*DNN for BNT*).

5.3.1 Delay Network Tomography (DNT)

This type of tomography uses delay metric to monitor the network and infer the link-level delays using (3.1). In this type, the operation $\otimes$ is for matrix multiplication, since the problem involves actual values for the delay measurements.

The only measurements that need to take place in this kind of network tomography are the end-to-end delay measurements. The end-to-end delay can be measured using timestamps [215], [216] as

$$y_i^{delay} = t_{recv} - t_{trans}, \tag{5.4}$$

where t_{recv} is the time when a monitoring message is received and t_{trans} is the transmission time of the same message. Then, by solving (3.1) for vector $\boldsymbol{x}$ (assuming $r(A) = \gamma_G$), one can determine the status of the network. In particular, by comparing the inferred delay value x_j of link e_j with the value of normal behaviour using (5.3).

The exact steps of this approach are as follows:

1. Using (5.4), collect path-level delay measurements $\boldsymbol{y}$ for paths in $\mathcal{P}_m(G)$.
2. Solve (3.1) to infer the link-level delays $\boldsymbol{x}$.
3. Using (5.3), determine the status for the inferred value of x_j, if $S(e_j) = 1$, then $S(G) = 1$ and link e_j is anomalous link.

5.3.2 Binary Network Tomography (BNT)

A branch of network tomography that is used to infer links' states from path-level states is called *Binary*[1] network tomography [146], [147] (see Chapter 2.4.4.3). For binary network tomograph, $\otimes$ in (3.1) is for boolean matrix multiplication, i.e., $y_i = \vee_j (a_{ij} \wedge x_j)$.

[1]Sometimes also called *Boolean*.

This approach only requires knowing the normal delay behaviour of paths in $\mathcal{P}_m(G)$ which then will be used to compare against the obtained measurements using

$$y_i = \begin{cases} 0, & \text{if } a_{p_i} \leq y_i^{delay} \leq b_{p_i} \\ 1, & \text{if } y_i^{delay} < a_{p_i} \text{ or } y_i^{delay} > b_{p_i} \end{cases}, \tag{5.5}$$

where a_{p_i} and b_{p_i} are threshold values for the minimum and maximum path-level delays of path $p_i \in \mathcal{P}_m(G)$ in the normal network behaviour, and y_i^{delay} is the measured path-level delay of $p_i \in \mathcal{P}_m(G)$ obtained by computing (5.4). Then, the status of each link $e_j \in E(G)$, $x_j \in \boldsymbol{x}$, can be determined following below steps:

1. Using (5.4), collect path-level delay measurements $y_{p_i}^{delay}, \forall p_i \in \mathcal{P}_m(G)$.
2. Determine the path-level status $y_i, \forall p_i \in \mathcal{P}_m(G)$ using (5.5).
3. If $y_i = 1, \exists p_i \in \mathcal{P}_m(G)$, then $S(G) = 1$.
4. If $S(G) = 1$, then solve (3.1) for $\boldsymbol{x}$ to obtain the status $x_j, \forall e_j \in E(G)$.
5. The anomalous link e_j is the one with $x_j = 1$.

5.3.3 Deep Neural Network (DNN)-based Tomography

Recall that the first condition in Remark 3.4.1 requires that $\kappa_G = \gamma_G$, usually, in-vehicle networks do not guarantee the satisfiability of this condition [208], [217]. Also, it is uncertain that the end-to-end measurements would be linearly independent. To tackle this, we adopt the deep-learning-based approach using a deep neural network as proposed in the last chapter (see Chapter 4.4). In particular, this approach assumes that there are a limited number of end-to-end measurements so that the measurement matrix $\boldsymbol{A}$ is rank-deficient (i.e., $r(\boldsymbol{A}) < \gamma_G$). Then, the deep neural network can be used to improve such a matrix so that it becomes a full-rank matrix. The deep neural network shown in Figure 4.4 is used here, where the output of such network is a set of estimated end-to-end measurements for paths in $\mathcal{P}(G) \backslash \mathcal{P}_m(G)$. Let such measurements be $\hat{\boldsymbol{y}}_u$, then based on (2.5), the output is

$$\hat{\boldsymbol{y}}_u = f_o(\boldsymbol{W}_{m+1}^T \boldsymbol{h}_m + \beta_m). \tag{5.6}$$

And the input is the set of end-to-end measurements for measured paths in $\mathcal{P}_m(G)$. Let such measurements be $\boldsymbol{y}_m$, then one can achieve

$$\left|\boldsymbol{y}_m\right| + \left|\hat{\boldsymbol{y}}_u\right| = \gamma_G, \tag{5.7}$$

From (5.7), it can be seen that after using DNN to estimate $\hat{\boldsymbol{y}}_u$, the total number of measurements can reach γ_G. If such measurements' paths are linearly independent, then they can be used to infer the link-level delays of the network by solving (3.1).

Remark 5.3.1. In order to use the DNN-based tomography approach, the only condition that should be satisfied is that the measurements should cover all links in $E(G)$. That is, each link $e_j \in E(G)$ should appear *at least once* in $\mathcal{P}_m(G)$.

The condition stated in the above remark is necessary to ensure that all links' existence is recognised by the DNN. Otherwise, the DNN cannot estimate the performance for those paths that traverse uncovered links.

5.3.4 Discussion

It is worth mentioning that in DNT, both the detection phase and localisation phase are performed in the same step (i.e., step 3), while in BNT, the detection phase occurs before the localisation phase. In other words, in BNT, only if an anomaly is detected, $S(G) = 1$, then the localisation phase would take place by solving (3.1). Unlike BNT, with DNT, the system has first to solve for $\boldsymbol{x}$ in (3.1) to detect and locate the anomaly. For an in-vehicle network with its limited capacity and resources, periodically solving (3.1) in DNT adds unnecessary computational overhead. Thus, BNT is favoured over DNT for this matter as it only needs to solve (3.1) once an anomaly has been detected. Moreover, the findings, as will be shown later, of BNT yield more accurate results than DNT.

5.4 Performance Evaluation

This section evaluates the proposed monitoring approaches for detecting and locating anomalies in in-vehicle networks that are based on central-gateway architecture. Note that the approach can be applied to any architecture with $\left|\mathcal{E}(G)\right| \geq 3$ and $\left|\mathcal{R}(G)\right| \geq 1$. The experiment settings for the simulated in-vehicle network are first described. Then, the evaluation results of all three types of tomography approaches are discussed.

5.4.1 Experiment Setup

We used a framework provided by CoRE project [218] based on OMNeT++ simulator [198] to simulate the in-vehicle network shown in Figure 5.1. Nodes in each subsystem are connected using a CAN bus, while all the subsystems are connected through a central-gateway. Table 5.2 shows the traffic details used for the normal network behaviour. Note that $v_1, v_5 \in \mathcal{E}(G)$ are edge nodes in the chassis domain, $v_2, v_6 \in \mathcal{E}(G)$ in the telematics domain, $v_3, v_7 \in \mathcal{E}(G)$ in the body domain and $v_4, v_8 \in \mathcal{E}(G)$ in the powertrain domain. In particular, in Figure 5.1, ECU1= v_1, ECU2= v_5, ECU5= v_2, ECU6= v_6, ECU3= v_3, ECU4= v_7, ECU7= v_4, and ECU8= v_8.

Since nodes v_3 and v_4 are in different domains, the start transmission times for the traffic transmitted by such nodes can be the same as they cannot contend to access the CAN bus. Hence, the same t_0 values for these source nodes as shown in Table 5.2. In the new E/E architectures, communication between different domains becomes common. For instance, ADAS applications can involve intense interaction between different vehicle domains [43]. Therefore, some entries shown in Table 5.2 allow cross-traffic communications.

Table 5.2: Normal traffic details used for network shown in Figure 5.1. t_0 is the start time.

Source	Destination	Priority (c_i)	t_0 (s)	Frequency (s)	Payload
v_1	v_2, v_3, v_4, v_5	2	0.50	0.60	8 Bytes
v_2	v_4, v_6	3	0.9	0.05	5 Bytes
v_3	v_7	4	0.005	0.06	3 Bytes
v_4	v_8	5	0.005	0.08	8 Bytes

The measured paths in case of DNT and BNT are $\mathcal{P}_m(G) = \{p_1, p_2, p_3, p_4\}$, where $p_1 = \{e_1, e_2\}$, $p_2 = \{e_1, e_3\}$, $p_3 = \{e_1, e_4\}$ and $p_4 = \{e_3, e_4\}$ (e_1 corresponds to the CAN bus in the chassis domain, e_2 in the telematics domain, e_3 in the body domain and e_4 in the powertrain domain). Table 5.3 shows the characteristics of the monitoring traffic used to measure paths in $\mathcal{P}_m(G)$. The monitoring traffic has no payload, its header size is 47 bits, and it is sent periodically every 1 second. As the source node is the same for the first three paths shown in Table 5.3, the priority level and start of transmission time are the same.

Table 5.3: Monitoring traffic details. t_0 is the start time.

Measured path ($p_i \in \mathcal{P}_m(G)$)	Source	Destination	Priority (c_i)	t_0 (s)
$p_1 = \{e_1, e_2\}$	v_1	v_2	20	1
$p_2 = \{e_1, e_3\}$	v_1	v_3	20	1
$p_3 = \{e_1, e_4\}$	v_1	v_4	20	1
$p_4 = \{e_3, e_4\}$	v_3	v_4	21	1.05

5.4.1.1 Anomalous Behaviour (DoS Attack)

As discussed in Chapter 2.3.2 of this thesis, DoS attack is one of the critical attacks that needs to be properly detected and located in order to minimise any harm it may cause. Therefore, for the anomalous behaviour, we simulated four DoS attacks, each originating from one of the four domains. The frequency of the attack is every 0.5 millisecond with a payload size of 8 bytes. The priority of the DoS attack is 1, i.e., the highest priority among other normal messages (see Table 5.2). As we assume that only one anomalous link can exist at a time, the attack in this scenario targets a single CAN bus.

5.4.1.2 DNN Setup

We used a deep neural network structure as depicted in Figure 4.4 with two hidden layers in addition to the input and output layers. Each hidden layer consists of $2\kappa_G - 1$ neurons. We used the ReLU activation function (2.2) for the hidden layers and linear function (4.10) for the output layer. To avoid model overfitting, early stopping [179] was utilised. For optimisation during backpropagation, the Adam optimisation function was used [214] to minimise (2.6). The set of measured paths here is $\mathcal{P}_m(G) = \{p_1, p_4\}$, while the DNN estimates the performance for $\mathcal{P}(G) \backslash \mathcal{P}_m(G) = \{p_2, p_3\}$. Note that p_1 and p_4 satisfy the condition in Remark 5.3.1 that all links in $E(G)$ are covered by these two paths.

We simulated five different scenarios, one for normal behaviour, an attack behaviour targeting the chassis domain, an attack behaviour targeting the telematics domain, an attack behaviour targeting the powertrain domain, and an attack behaviour targeting the body domain. Each simulation lasted for 120 seconds.

5.4.2 Results

In the following, we show the results of the proposed monitoring approaches to detect and locate anomalies in the simulated in-vehicle network. We note that all approaches could accurately detect the anomalies. Hence, in the following, we show the results for the localisation phase.

Let TP be true positive, TN true negative, FP false positive and FN false negative, then the following metrics are used to evaluate the status $x_j \in \boldsymbol{x}$ of each link $e_j \in E(G)$.

- **Precision:** It measures the ability to correctly classify anomalous links out of all positive predictions. Precision is calculated as

$$\text{Precision} = \frac{TP}{TP + FP} \quad .$$

- **Recall:** It measures the ability to correctly classify anomalous links out of all actual anomalous links. It is calculated as

$$\text{Recall} = \frac{TP}{TP + FN} \quad .$$

- **F1-measure:** It measures the harmonic mean between precision and recall. It is calculated as

$$\text{F1-measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad .$$

- **Accuracy:** It measures the ability to correctly classify normal and anomalous links. Accuracy is calculated as

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad .$$

We compare the inference results using DNT, BNT, DNN-based tomography for DNT (DNN for DNT), and DNN-based tomography for BNT (DNN for BNT). Additionally, we compare with a baseline solution (BL) that uses threshold values to determine the status of each link using (5.3).

All the results are shown in Table 5.4 — Table 5.7. It is worth noting that BNT achieved the best performance for all metrics. It correctly located all the anomalous links, with no false positives or false negatives. This is due to the fact that BNT can detect if an anomaly occurred by measuring the deviations from the normal network behaviour. In addition, it uses binary values to determine the link status, hence accurately locating the anomalous link without the need to infer the exact link-level delay value. For DNT, it achieved 100% recall for correctly detecting all anomalous links. However, it sometimes misclassified normal links as anomalous. This means that it cannot uniquely locate the anomalous link. Compared with the baseline solution, BNT yielded better results while DNT performed worse (for all metrics except for recall).

On the other hand, when the DNN-based tomography approach is used, DNN for BNT performed worse than DNN for DNT. Especially, it could not detect when link e_2 is anomalous. This is clear from the precision, recall, and F1-measure results shown in Table 5.4, Table 5.5, and Table 5.6, respectively as 00.00%. However, we observed that DNN for BNT can sometimes correctly classify e_2 when it is normal. This is reflected in the accuracy score shown in Table 5.7 as 40.00%. Moreover, it correctly detected anomalies when links other than e_2 were anomalous, but it could not uniquely locate the anomalous link. On the contrary, DNN for DNT could detect all anomalous links but could not uniquely determine the abnormal one. As compared with the baseline solution, both DNN-based approaches performed worse than the baseline solution in all metrics except for recall where DNN for DNT performed better. With DNN, the measurement results for paths in $\mathcal{P}(G)\backslash\mathcal{P}_m(G)$ are estimation values and not exact values like the case with BNT and DNT. This means that the estimated values can deviate from the actual exact measurements. Therefore, this affects the performance of DNN-based tomography in that it results in a higher number of false positives and/or false negatives than with BNT and DNT.

Table 5.4: Results of detecting and locating anomalies in centralised in-vehicle networks - Precision (%).

Link	Baseline	DNT	BNT	DNN for DNT	DNN for BNT
e_1	100.0	50.00	100.0	20.00	25.00

Table 5.4: Results of detecting and locating anomalies in centralised in-vehicle networks - Precision (%).

Link	Baseline	DNT	BNT	DNN for DNT	DNN for BNT
e_2	50.00	20.00	100.0	20.00	00.00
e_3	100.0	50.00	100.0	20.00	50.00
e_4	50.00	33.30	100.0	20.00	50.00

Table 5.5: Results of detecting and locating anomalies in centralised in-vehicle networks - Recall (%).

Link	Baseline	DNT	BNT	DNN for DNT	DNN for BNT
e_1	100.0	100.0	100.0	100.0	100.0
e_2	100.0	100.0	100.0	100.0	00.00
e_3	100.0	100.0	100.0	100.0	100.0
e_4	100.0	100.0	100.0	100.0	100.0

Table 5.6: Results of detecting and locating anomalies in centralised in-vehicle networks - F1-measure (%).

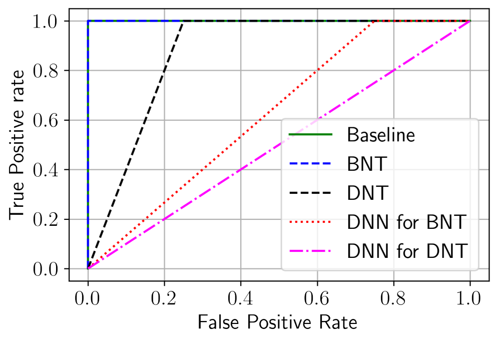
Link	Baseline	DNT	BNT	DNN for DNT	DNN for BNT
e_1	100.0	66.70	100.0	33.30	40.00
e_2	66.70	33.30	100.0	33.30	00.00
e_3	100.0	66.70	100.0	33.30	66.70
e_4	66.70	50.00	100.0	33.30	66.70

Table 5.7: Results of detecting and locating anomalies in centralised in-vehicle networks - Accuracy (%).

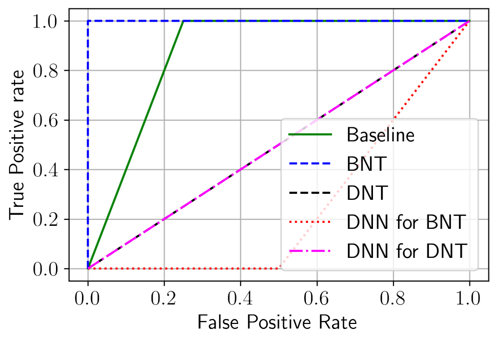
Link	Baseline	DNT	BNT	DNN for DNT	DNN for BNT
e_1	100.0	80.00	100.0	20.00	40.00
e_2	80.00	20.00	100.0	20.00	40.00
e_3	100.0	80.00	100.0	20.00	80.00
e_4	80.00	60.00	100.0	20.00	80.00

In addition, Figure 5.2 shows the Receiver Operating Characteristic (ROC) curve results. As seen, BNT outperforms the other solutions in uniquely and accurately locating the anomalous links. On the other hand, the performance of DNT is somehow comparable with the DNN-based solution. For e_1 and e_3, the baseline solution achieved a 100% true positive rate (see Figure 5.2(a) and Figure 5.2(c)) as for these two links, the baseline solution could accurately locate when these links are anomalous. This is also reflected in the previous tables where the baseline solution achieves 100% in all metrics for links $e_1, e_3 \in E(G)$). In contrast, for links e_2 and e_4 (see Figure 5.2(b) and Figure 5.2(d)), the baseline solution did not perform as well as for e_1 and e_3. The DNT approach, on the other hand, achieved the same true positive rate for both links e_1 and e_3. Again, this is because the inference results for these two links are the same as shown in the precision and recall metrics. For link e_2, the true positive rate is worse than for other links. This is also clear from the precision and F1-measure results. The DNN for BNT approach resulted in a high false positive rate for link e_2 as shown in Figure 5.2(b). This is because DNN for BNT approach could not locate when link e_2 is anomalous as discussed earlier. On the other hand, the DNN for DNT approach achieved similar true positive rates for all links in G. This has also been shown in the precision and recall results in Table 5.4 and Table 5.5.

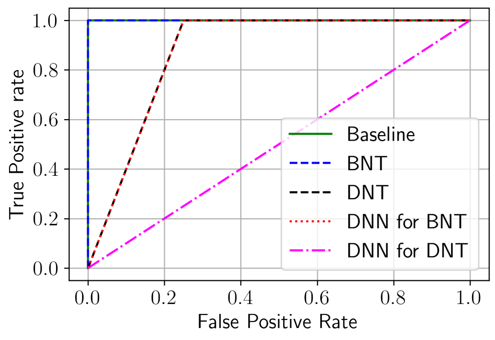
Compared with other monitoring approaches, our network tomography approach with BNT achieves the best results as shown in Table 5.8, where the performance of the proposed approaches is compared against the performance reported in [96] for CANTransfer [96] and OTIDS [79] (see Chapter 2.3.4).



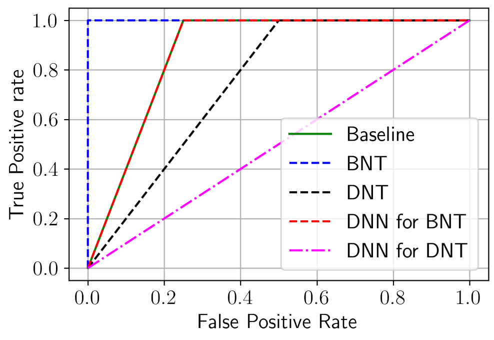
(a) Link metric x_1 for $e_1 \in E(G)$



(b) Link metric x_2 for $e_2 \in E(G)$



(c) Link metric x_3 for $e_3 \in E(G)$



(d) Link metric x_4 for $e_4 \in E(G)$

Figure 5.2: ROC curve results for each link-level status $x_j \in \boldsymbol{x}$.

Table 5.8: Comparison results of network tomography-based monitoring approach with CANTransfer [96] and OTIDS [79].

Monitoring approach	Precision (%)	Recall (%)	F1-Measure (%)
Baseline	75.00	100.0	83.35
DNT	38.33	100.0	54.18
BNT	100.0	100.0	100.0
DNN for DNT	20.00	100.0	33.30
DNN for BNT	31.25	75.00	43.35
CANTransfer	94.93	95.57	95.25
OTIDS	99.82	71.68	83.44

5.4.3 Discussion

The above results show that BNT can be used to accurately detect and locate anomalies within the in-vehicle network. DNT, however, cannot uniquely locate the anomalous links. This is due to the fact that message priority in CAN affects the performance of DNT. Particularly, if, for example, two nodes v_i and v_j with priorities $c_i = 53$ and $c_j = 52$, are transmitting simultaneously, then node v_j will win the arbitration and access the bus before v_i because v_j has higher priority than v_i, thus the delay of the message transmitted by v_i is larger than that transmitted by v_j. As mentioned in [208], this can result in asymmetric behaviour, which can perturb the actual delay measurements. Hence, DNT cannot accurately locate the anomaly.

On the other hand, BNT does not rely on the delay measurements to locate the anomalies. Instead, it uses the path-level binary performance which, due to the deterministic behaviour of in-vehicle networks, can accurately classify paths' states. Hence, uniquely locates the anomalous links.

The results of DNN-based tomography solutions show that they are not very reliable, especially in locating the anomalies. Therefore, it is advisable to use a more reliable solution such as BNT which, due to its algebraic nature, requires satisfying the topological conditions, but once the conditions are satisfied, BNT is the best choice to detect and locate anomalies in in-vehicle networks.

5.5 Summary

This chapter proposed leveraging a network tomography monitoring approach to detect and locate anomalies in an in-vehicle network with centralised architecture. Moreover, if the measurement matrix does not meet the rank requirements, a deep neural network-based approach is used to compensate for the rank deficiency. In addition to the baseline solution, two types of network tomography have been evaluated: BNT and DNT. The evaluation results show that, by monitoring only a subset of the network, BNT can achieve the best performance in correctly detecting the anomaly and locating the anomalous link in the network, while DNT and deep neural network-based tomography approaches did not perform well in locating all the anomalous links. These results indicate that to accurately detect and locate anomalies, BNT is the choice to be used. However, the rank requirement of the measurement matrix should be satisfied to obtain accurate results. Hence, it is important to ensure that the topology is identifiable (satisfies topological conditions stated in Theorem 3.4.1). The next chapter proposes a new in-vehicle network topology that supports this property.

One aspect of our approach that can be seen as a limitation is that it cannot detect attacks that result in behaviour that conforms to the normal performance of the network, as it highly depends on measuring how the network performance deviates from the normal behaviour. However, we argue that most anomalies should yield some deviations from the normal behaviour, otherwise, the network operation would not be affected. In addition, the proposed approach requires that the threshold values for the normal behaviour are accurately determined, otherwise, the classification results might not be precise. Therefore, it is important to accurately define the normal behaviour of the in-vehicle network in order to determine the threshold values that can be used in the anomaly detection and localisation process.

Another limitation of network tomography is that the collection of end-to-end measurements and computation of link-level metrics require a powerful and central unit to perform these tasks. Such a unit can be facilitated by the central architecture of the in-vehicle network. For instance, the central-gateway can be part of a SDN controller that is responsible for inferring the results and taking actions to mitigate the anomalies effect. This is also discussed in the new proposed topology presented in the following chapter.

6 A New SDN-enabled In-Vehicle Network Topology

"The wonders of yesterday are today common occurrences."

Nikola Tesla

6.1 Overview

The promising results of BNT proposed in the last chapter are strong drivers to propose a new topology. Specifically, algebraic approaches used to monitor the in-vehicle networks often result in more accurate performance than the DNN counterparts. However, to fully benefit from algebraic network tomography, the in-vehicle network topology should satisfy certain conditions, as mentioned in Chapter 3 where we briefly shed light on one important challenge we faced when analysing network identifiability for different E/E in-vehicle network architectures. In particular, the findings showed that the Ethernet-based architecture is not always identifiable. To this end, this chapter seeks to investigate possible solutions to achieve a fully-identifiable in-vehicle network topology. Additionally, this chapter aims to propose an overall monitoring approach that is capable of performing all three tasks that any robust monitoring approach should support. These tasks are *anomaly detection*, *localisation* and *mitigation* (see Chapter 2.3.3).

An overall view of the main contributions in this chapter is demonstrated in Figure 6.1.

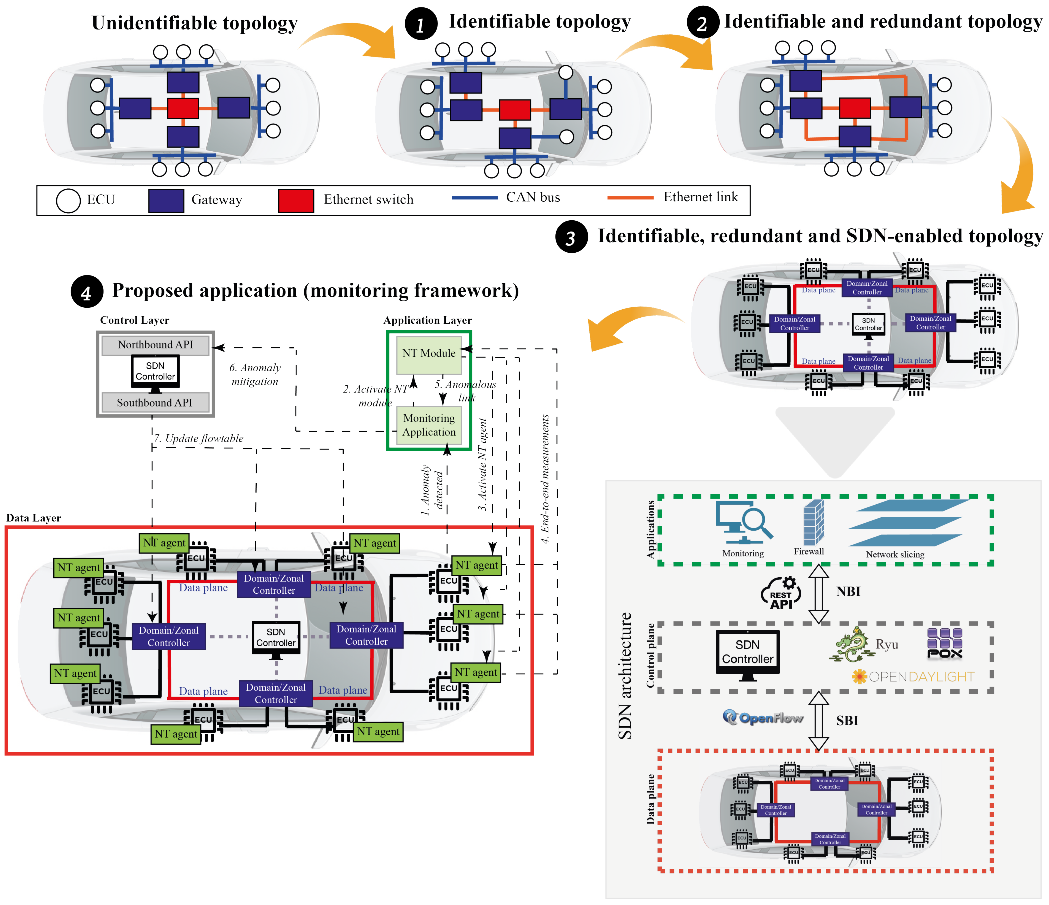


Figure 6.1: Overall view of the main contributions in this chapter.

6.2 Introduction and Motivation

As shown in the last chapter, algebraic network tomography can achieve the best results in detecting and locating anomalies in in-vehicle networks. Typically, to use algebraic network tomography, the network topology has to be *identifiable* [20], [117]. In Chapter 3, it has been shown that not all in-vehicle network architectures satisfy this property. Therefore, in this chapter, we formalise the topological requirements for in-vehicle networks so that they satisfy the identifiability conditions. Further, current in-vehicle network architectures were designed to be fail-safe and not fail-operational, which means they are not robust enough against failures, especially for autonomous vehicles of level 4 and 5 [21]. For instance, if a failure occurs in the network due to, for example, an attack, the impact can extend to the whole network. This can, not only affect the network performance, but it can also endanger lives, if no proper action is taken to stop the malicious behaviour or at least allow the operation of critical subsystems under such behaviour to ensure having a fail-operational mechanism. A solution to this problem is to have a redundant network that can be activated once the original one is at risk.

The main contribution of this chapter is to propose a novel in-vehicle network topology that has three important properties. First, it is *identifiable* based on the topological condition defined in Chapter 3. Second, it supports *redundancy*, in which the redundant paths can be used in case of any occurrences of failure in the network. Third, it is enabled with *SDN* capabilities. Furthermore, the topological conditions required to achieve identifiable and redundant in-vehicle networks are studied. These conditions can be very useful in the design stage of any in-vehicle network. Additionally, based on the theoretical analysis, a number of algorithms to transform any existing in-vehicle network topology into an identifiable and redundant topology are devised. Such algorithms can further assist in the modification of existing in-vehicle networks' topologies rather than building the network from scratch, thus, saving time and resources that otherwise will be costly [21].

The first property of the new topology (identifiable topology) is to ensure that every element of the network will be monitored using only end-devices, $\mathcal{E}(G)$, and without the need to access internal networking elements. The second (redundant topology) and third (SDN-enabled topology) properties play significant roles in facilitating the monitoring of the network that involves detecting, locating, and mitigating anomalies. For instance, if part of the network fails, e.g., due to an attack, the safety-critical traffic can still be transmitted using the redundant paths. On the other hand, the benefits of enabling the in-vehicle network with SDN capabilities are numerous. First, SDN can aid in the network tomography-based monitoring process, where the SDN controller can set the measurement paths and collect their performance metrics from end devices. In addition, SDN is well-suited for the new E/E architecture (see Chapter 2.2) that is based on utilising a cross-domain, centralised architecture rather than domain-specific architecture [48]. So, instead of a large number of ECUs, fewer number of *powerful* devices will be adopted by the new E/E architectures. Such powerful devices are called *High-Performance Computing Platforms (HPCPs)* [21], [59]. Therefore, by hosting the SDN controller in one of these platforms, it should be able to perform the necessary network tomography computations. Moreover, if a failure is detected within the network, the SDN controller can mitigate its effect by instructing the underlying devices to activate the redundant paths, reroute the normal traffic and drop the malicious one.

6.3 Centralised In-Vehicle Networking Architectures

The issue with the decentralised conventional E/E architectures is that they are not robust enough against malicious behaviours [43]. Centralised architectures, on the other hand, can help in achieving more secure in-vehicle systems [43], [48]. In addition, they have the benefit of reduced latency compared with decentralised architectures. This is due to two main reasons: first, the coordination between different ECUs is done in a single ECU in the centralised architecture, which results in zero control latency. The second reason is the reduced number of different CAN messages in the centralised architecture because messages that were sent by different ECUs in the decentralised architecture are now transmitted by the same ECU in the centralised one. This minimises bus contention, which in turn reduces latency [43]. Other benefits of centralised architectures include reduced wiring length, reduced data volume and minimised bus utilisation [43]. Hence, for these benefits, next-generation automotive networks tend to employ centralised architectures.

6.4 Redundancy in In-Vehicle Networks

To cope with failures, redundancy should be supported for all new E/E architectures [219]. Due to their low weight cost, conventional in-vehicle networks follow star-based topologies [220]. However, such topologies do not support redundancy. To have a redundant network, the network topology should follow a ring-based topology [22], [23]. Therefore, the topology proposed in this chapter is also ring-based, in addition to being fully identifiable to allow monitoring of all network components using only edge devices as monitors.

6.5 Software-Defined Network (SDN)

SDN is a new networking paradigm that supports the shift towards programmable networks [221] where the control plane is decoupled from the data plane, allowing for more flexible management. Generally, SDN architecture is divided into three main layers as shown in Figure 6.2. The first layer includes physical network devices such as routers, gateways, switches, etc. This layer is called *data plane* layer. The second layer, called *control layer*, includes an SDN controller which is a central entity in the network that can handle the decision-making of the forwarding devices at the data plane layer. There are multiple SDN controllers available nowadays, e.g., Ryu, OpenDaylight, and POX. The communication between the data plane and control plane is done through an interface called *Southbound Interface (SBI)*, e.g., OpenFlow protocol [222]. The third layer is the *applications* layer[1]. This layer hosts networking applications which can be used to manage and control the underlying network through the control plane layer. Examples of such applications include network monitoring [223], [224], network slicing [225], [226], firewall [227], etc. Communication between the control plane and the application plane can utilise a *Northbound Interface (NBI)* such as REST API [228].

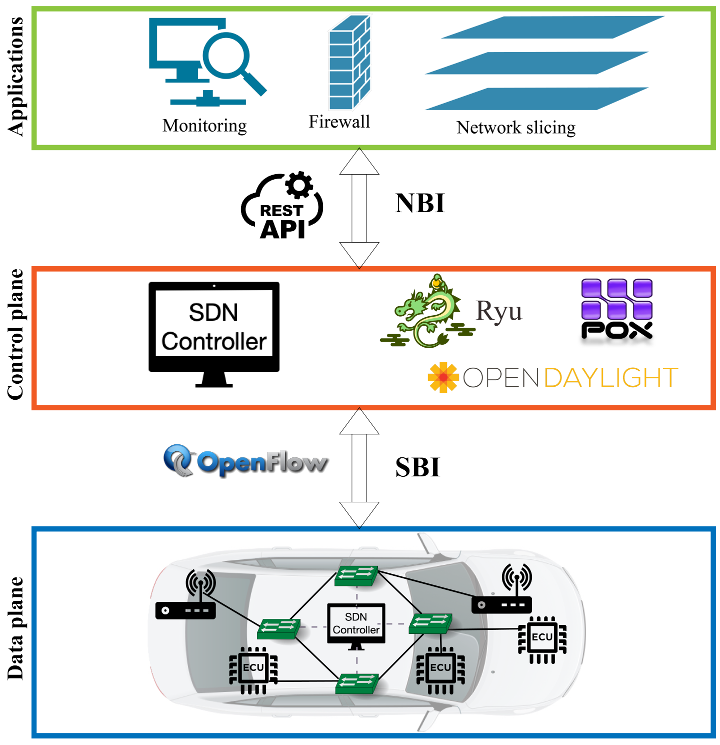


Figure 6.2: SDN layered architecture.

There are two types of flows in SDN: *proactive* and *reactive* [24]. In proactive flows, the SDN application presets these flows so that if a packet arrives, the receiving device has the destination address for such packet set in its flow table. This is also called *static flow*. On the contrary, reactive flows respond to the packet by sending it to the SDN controller. The SDN application then instructs the controller on how to handle such a packet.

Employing SDN in in-vehicle networks has several benefits. First, the SDN paradigm with the centralisation feature is the perfect fit for the new E/E architectures that are shifting more towards centralisation [43], [48]. Second, incorporating SDN into the in-vehicle network

[1]Sometimes it is also called *management* layer.

can drastically minimise the number of ECUs in which hardware-specific functions can be replaced by software components. This provides robustness to design changes where additional functionalities can easily be added as software applications to the central control unit, e.g., the SDN controller [43]. In addition, the SDN concept can be combined with network tomography [169]. For instance, an SDN controller can set up the measurement paths between monitors more efficiently so that the estimation accuracy is improved as compared with vanilla network tomography [229]. In addition, the collection of end-to-end measurements can be done by the controller, which can also perform network tomography-related computations. Moreover, with its overall view of the underlying network architecture, the SDN controller can assist in the decision-making process when part of the network experiences any failure. In particular, it can reroute the traffic to the redundant paths/links.

Recently, researchers started to propose using SDN in vehicular communications [230]–[233]. Our focus in this thesis is on the intra-vehicle networking. Employing SDN in this domain is still in its infancy with limited contributions. For instance, [50] proposed a flexible SDN-based architecture for automotive Ethernet networks with the goal of meeting high bandwidth requirements for multimedia applications while also maintaining the necessary bandwidth for time-critical applications. Furthermore, SDN enhances the security aspect of the network using network slicing, which requires the SDN concept to isolate different slices [231]. On the other hand, authors in [234] discussed various strategies for integrating control flows in automotive Ethernet networks. They transformed a traditional in-vehicle network into an SDN-controlled Ethernet network and evaluated these strategies. They found that, with SDN, the attack surfaces on cars can significantly be reduced.

In this chapter, we combine network tomography and SDN into an in-vehicle network, where the former is a monitoring mechanism and the latter can significantly aid in the monitoring process as well as management of the overall network. In addition, we propose a complete monitoring framework leveraging such a combination.

6.6 Problem Statement and Assumptions

6.6.1 Problem Statement

Table 6.1 describe the notations used throughout this chapter.

Table 6.1: Chapter 6 notations and their descriptions.

Notation	Description
Network & Graph Theory	
$G = (V, E)$	Network represented as graph G
$V(G)$, $E(G)$	Set of vertices (nodes) and edges (links) in network G
$\mathcal{E}(G) \subset V(G)$	Set of edge nodes in G (see Definition 3.3.1)
$\mathcal{R}(G) \subset V(G)$	Set of intermediate nodes in G (see Definition 3.3.1)
$\mathcal{R}_{3^+} \subseteq \mathcal{R}(G)$	Set of internal nodes having node degree larger than three (see Definition 6.7.2)
$\mathcal{R}_{3^-} \subseteq \mathcal{R}(G)$	Set of internal nodes having node degree less than three (see Definition 6.7.2)
$\mathcal{P}(G)$	Set of all possible paths in network G
$\mathcal{P}_m(G) \subseteq \mathcal{P}(G)$	Set of measured paths in network G
$\mathcal{B}(G)$	Set of bridges in G (see Definition 6.8.2)
$\mathcal{I}(G), \mathcal{T}(G) \subset E(G)$	Set of internal and externallinks in G (see Definition 6.7.1)
$\mathcal{W}(G)$	Set of internal nodes in G having node degree larger than three and are neighbours to more than one internal node
$\mathcal{N}(v_i)$	Set of neighbours directly connected to node $v_i \in V(G)$
$\mathcal{N}_{\mathcal{R}}(v_i) \subseteq \mathcal{N}(v_i)$	Set of internal neighbours in $\mathcal{R}(G)$ directly connected to node $v_i \in V(G)$
$G + \{v_iv_j\}$	Adding link v_iv_j to network G
$d(v_i, v_j)$	Distance between nodes $v_i, v_j \in V(G)$
$\mathcal{C}(G)$	Set of graph components in G
$v_h(e_i), v_t(e_i)$	Endpoints (head and tail) of i^{th} link $e_i \in E(G)$
$g_{\mathcal{B}} \subset g$	Subgraph of g that only contains bridges of g, $\mathcal{B}(g)$
$d(v_i)$	Degree of node $v_i \in V(G)$
$p_i \in \mathcal{P}(G)$	i^{th} path connecting two non-adjacent nodes $v_i, v_j \in \mathcal{E}(G)$
$e_i \in E(G)$	i^{th} link in network G
$v_iv_j \in E(G)$	A link connecting nodes v_i and v_j

Table 6.1: Chapter 6 notations and their descriptions.

Notation	Description
Network Tomography	
$\boldsymbol{y}$	Vector of end-to-end measurements
$\boldsymbol{x}$	Vector of link-level measurements
$\boldsymbol{A}$	Measurement matrix
$y_i \in \boldsymbol{y}$	End-to-end measurement of path $p_i \in \mathcal{P}_m(G)$
$x_i \in \boldsymbol{x}$	Link-level metric of i^{the} link $e_i \in E(G)$
$a_{ij} \in \{0, 1\}$	The element of the measurement matrix $\boldsymbol{A}$ at the i^{th} row (for $p_i \in \mathcal{P}_m(G)$) and j^{th} column (for $e_j \in E(G)$)
$r(\boldsymbol{A})$	Rank of the measurement matrix $\boldsymbol{A}$
$S_d(G) \in \{\text{false}, \text{true}\}$	Identifiability status of network G, false means *unidentifiable* and true means *identifiable*
$S_r(G) \in \{\text{false}, \text{true}\}$	Redundancy status of network G, false means *non-redundant* and true means *redundant*
Numbers & Cardinalities	
η_G, γ_G	Total number of nodes and links in network G
$\kappa_G := \Big\lvert \mathcal{P}_m(G) \Big\rvert$	Total number of measured paths in G
$\lambda_G := \Big\lvert \mathcal{R}(G) \Big\rvert$	Total number of intermediate nodes in network G
$\sigma := \Big\lvert \mathcal{R}_{3-} \Big\rvert$	Total number of internal nodes having node degree less than three
$\zeta_{v_i} := \Big\lvert \mathcal{N}_{\mathcal{R}}(v_i) \Big\rvert$	Total number of internal nodes that are neighbours to $v_i \in \mathcal{R}_{3+}$
$\psi := \Big\lvert R_{3+} \Big\rvert$	Number of internal nodes with node degree larger than three
$\varphi_{v_i} := d(v_i) - 3$	Number of remaining links incident to v_i when disconnecting it from three links
$\omega := \Big\lvert \mathcal{W} \Big\rvert$	Total number of internal nodes having node degree larger than three and are neighbours to more than one internal node

Table 6.1: Chapter 6 notations and their descriptions.

Notation	Description
Numbers & Cardinalities	
N_{v_i,v_j}	Total number of internally disjoint paths between $v_i, v_j \in \mathcal{E}(G)$
$\mathcal{H}_{p_i} := \left\| \bar{y_i} - y_i \right\|$	Difference between measured and actual performance of i^{th} path $p_i \in \mathcal{P}_m(G)$
$\mathcal{H}_{e_i} := \left\| \bar{x_i} - x_i \right\|$	Difference between measured and actual performance of i^{th} link $e_i \in E(G)$
δ_{p_i}	Predefined threshold value for i^{th} path $p_i \in \mathcal{P}_m(G)$
δ_{e_i}	Predefined threshold value for i^{th} link $e_i \in E(G)$
χ_η	Number of added nodes in the transformed topology
χ_γ	Number of added links in the transformed topology

Given an in-vehicle network G, the aim is to decide whether it is identifiable or not (i.e., $S_d(G) \in \{\text{false}, \text{true}\}$). If $S_d(G) = \text{false}$, the goal is to transform G into an identifiable topology G_i (i.e., $G \longrightarrow G_i$) where $S_d(G_i) = \text{true}$. Similarly, the second aim is to check for the redundancy status of G, $S_r(G) \in \{\text{false}, \text{true}\}$. If $S_r(G) = \text{false}$, then the goal is to transform G into redundant topology G_r (i.e., $G \longrightarrow G_r$) such that $S_r(G_r) = \text{true}$. The ultimate goal is to have a minimum number of links $\gamma_{G_i}, \gamma_{G_r}$ in the transformed topologies G_i and G_r, respectively.

The final topology G_{ir} should satisfy both states of identifiability and redundancy with $S_d(G_{ir}) = \text{true}$ *and* $S_r(G_{ir}) = \text{true}$.

6.6.2 Assumptions

Throughout this chapter, we adopt the following assumptions:

1. Only internal links can fail (a link e_i is internal if its both endpoints are intermediate nodes, see Definition 6.7.1 for a formal definition).

2. Communication between any node pair $v_i, v_j \in \mathcal{E}(G)$ should go through a simple path. Paths with cycles are not allowed.

3. All nodes in $\mathcal{E}(G)$ can communicate with each other.
4. Links in G are symmetric.
5. The in-vehicle network G is a connected network. The same principle, however, can be applied to individual disconnected components.
6. Only edge nodes in $\mathcal{E}(G)$ can be used as monitors.

6.7 Identifiable Topology

In this section, the focus is on networks forming only acyclic graphs. In a later section (Chapter 6.8), we discuss network identifiability for in-vehicle networks with cycles.

Recall the identifiability definition for network G in Definition 3.4.2, the aim here is to have a fully identifiable network topology[2].

6.7.1 Topological Conditions

The focus of this chapter is on achieving a topology that is fully identifiable, hence, we need to study the topological conditions needed to transform an unidentifiable topology into an identifiable one.

Definition 6.7.1. Given an in-vehicle network $G = (V, E)$, sets of internal and external links ($\mathcal{I}(G) \subset E(G)$ and $\mathcal{T}(G) \subseteq E(G)$) are defined as:

- $\mathcal{I}(G) = \{e_i \in E(G) : v_h(e_i) \in \mathcal{R}(G), v_t(e_i) \in \mathcal{R}(G)\}$;
- $\mathcal{T}(G) = \{e_i \in E(G) : v_h(e_i) \in \mathcal{E}(G), v_t(e_i) \in \mathcal{R}(G)\}$

where $\mathcal{I}(G) \cup \mathcal{T}(G) = E(G)$ and $\mathcal{I}(G) \cap \mathcal{T}(G) = \emptyset$. We denote a set of internal links that node v_i is incident to by $\mathcal{I}(v_i)$.

The following lemma states the condition required to have an acyclic-connected graph G.

Lemma 6.7.1. *Any connected graph G is acyclic if and only if it has $\eta - 1$ links, where $\eta := \Big|V(G)\Big|$.*

Proof. See proof of Corollary 1.5.3 in [196]. □

[2]In the remaining of this chapter, we simply use the term *identifiable topology* to refer to a fully identifiable topology. For unidentifiable or l-identifiable topology, we simply say *unidentifiable topology.*

From the assumption that an in-vehicle network is a connected graph, we derive the following theorem that is necessary for designing the transformation algorithm.

Theorem 6.7.1. *Any acyclic connected graph G with $\lambda_G \geq 2$ implies that $\mathcal{I}(v_i) \neq \emptyset, \forall v_i \in \mathcal{R}(G)$, where $\lambda_G := \left|\mathcal{R}(G)\right|$.*

Proof. For a connected graph G, let assume that $\mathcal{I}(v_i) = \emptyset, \exists v_i \in \mathcal{R}(G)$. By constructing a graph with $\lambda_G = 2$ corresponding to $v_1 \in \mathcal{R}(G)$ and $v_2 \in \mathcal{R}(G)$ with $\mathcal{I}(v_1) = \emptyset$, it implies that $\mathcal{I}(v_2) = \emptyset$, hence $\gamma < \eta - 1$. Then $d(v_1, v_2) = \infty$ and by Lemma 6.7.1, the graph is disconnected, which is a contradiction. $\square$

The above theorem indicates that all internal nodes, in any connected acyclic graph with $\lambda_G \geq 2$, should be connected to *at least* one internal link. In addition, the necessary and sufficient condition for G to be identifiable is defined in Theorem 3.4.1.

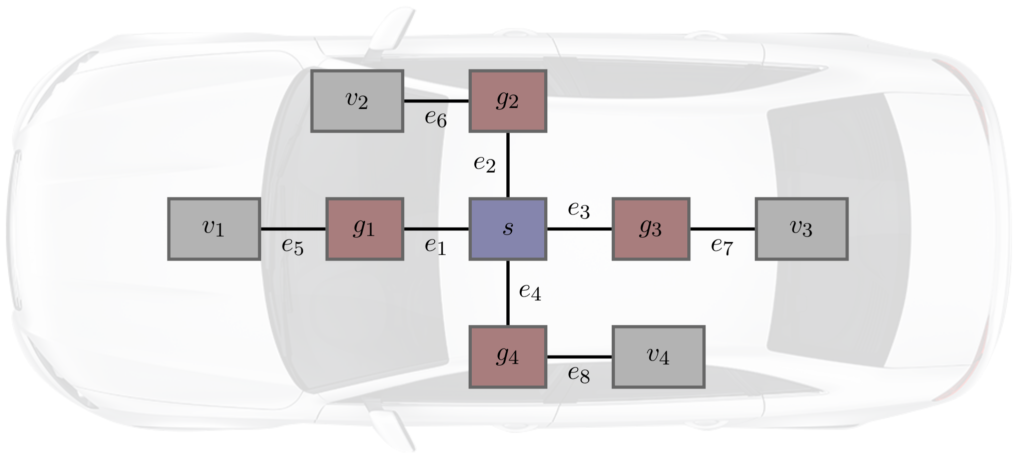


Figure 6.3: Unidentifiable Ethernet-based topology.

The Ethernet-based topology shown in Figure 6.3 is unidentifiable. This is due to the violation of the topological condition stated in Theorem 3.4.1. According to this theorem, in order to make the topology identifiable, the node degree for all $v_i \in \mathcal{R}(G)$ should be increased by at least 1 (the current degree of internal nodes, i.e., gateways $g_1, g_2, \dots, g_4$, is 2). However, in automotive networks, where the wiring adds up to the total weight and complexity of the vehicle, it is desirable to minimise the number of links. The following theorem characterizes the minimum number of links required to achieve identifiable topology.

Theorem 6.7.2. *The minimum number of links that can exist in an identifiable topology G with λ_G internal nodes is*

$$2\lambda_G + 1. \tag{6.1}$$

Proof. From Lemma 6.7.1, we know that we need to have at least $\eta - 1$ links in G, otherwise, the topology would be disconnected. However, Lemma 6.7.1 ensures connectivity, but it does not ensure identifiability. To ensure having a topology that is identifiable, the condition in Theorem 3.4.1 has to be satisfied.

Without loss of generality, we can assume a topology with internal nodes only. Because each $v_i \in \mathcal{R}(G)$ should have at least $d(v_i) = 3$ to satisfy the identifiability condition, $3\lambda_G$ links are needed, including the $\lambda_G - 1$ links required to have a connected graph. Therefore, the minimum possible number of links that can exist in identifiable topology is

$$3\lambda_G - (\lambda_G - 1) = 2\lambda_G + 1.$$

□

Note that (6.1) is just the minimum number of links in an identifiable topology. In practice, the total number of links can be larger. Thus, any network topology with fewer number of links implies that the topology is *unidentifiable*.

Based on the above theoretical analysis, we derive a procedure (i.e., Procedure 1) to check for the topological identifiability condition for any in-vehicle network G.

```
Procedure 1: $isIdentifiable(G)$
  Output : $S_d(G)$
  Initialize:
  $S_d(G) \leftarrow$ true
  $\mathcal{R}(G) \leftarrow \{v_i : d(v_i) \geq 2\}$
1 foreach $v_i \in \mathcal{R}(G)$ do
2     if $d(v_i) < 3$ then
3         $S_d(G) \leftarrow false$
4         break
5 return $S_d(G)$
```

Procedure 1 takes a network G and decides whether it is identifiable $S_d(G) =$ true or not $S_d(G) =$ false.

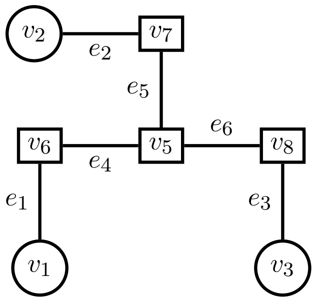
6.7.2 Transformation into Identifiable Topology

By transforming an *unidentifiable* topology G into an *identifiable* topology G_i, only the number of internal nodes can be preserved, i.e., $\lambda_G = \lambda_{G_i}$, while preserving the number of edge nodes is not guaranteed as in some cases we need to add more links, which in turn requires adding more edge nodes in case the topology is acyclic, hence $\eta_{G_i} \geq \eta_G$.

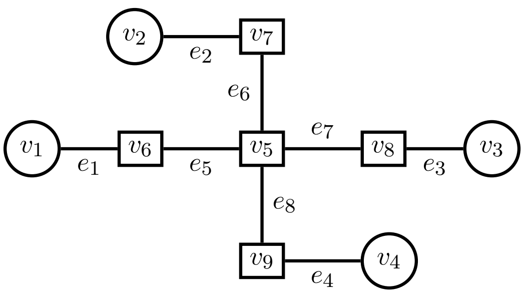
Any *unidentifiable* topology G can be classified as either one of the following two cases:

- **Case 1**: There is at least one internal node $v_i \in \mathcal{R}(G)$ with $d(v_i) < 3$ while $d(v_j) = 3$, $\forall v_j \in \mathcal{R}(G) \backslash v_i$, where $i \neq j$ and $i, j \in \{1, \ldots, \eta\}$.
- **Case 2**: There is at least one internal node $v_i \in \mathcal{R}(G)$ with $d(v_i) < 3$ while $d(v_j) \geq 3$, $\forall v_j \in \mathcal{R}(G) \backslash v_i$ with at least one node $v_k \in \mathcal{R}(G) \backslash v_i$ having $d(v_k) > 3$, where $i \neq j \neq k$ and $i, j, k \in \{1, \ldots, \eta\}$.

Figure 6.4 shows examples of such cases. Figure 6.4(a) shows an example of case 1 with $d(v_5) = 3$ while $d(v_6) = d(v_7) = d(v_8) = 2 < 3$. On the other hand, Figure 6.4(b) shows a case 2 example with $d(v_5) = 4 > 3$ while $d(v_6) = d(v_7) = d(v_8) = d(v_9) = 2 < 3$.



(a) Case 1



(b) Case 2

Figure 6.4: Topology examples of case 1 and case 2 topologies.

In the following, we will describe how to transform the unidentifiable topology G into an identifiable topology G_i considering each one of the above cases. But first, the following two sets are necessary to define.

Definition 6.7.2. Given an in-vehicle network G with a set of internal nodes $\mathcal{R}(G)$ defined as in Definition 3.3.1, let $\mathcal{R}_{3^-} \subseteq \mathcal{R}(G)$ and $\mathcal{R}_{3^+} \subseteq \mathcal{R}(G)$ be two sets of internal nodes with node degree less than and larger than three, respectively. These sets are formally defined as

- $\mathcal{R}_{3^-} = \{v_i \in \mathcal{R}(G) : d(v_i) < 3\}$;
- $\mathcal{R}_{3^+} = \{v_i \in \mathcal{R}(G) : d(v_i) > 3\}$.

6.7.2.1 Transforming Case 1 Topologies

Let $\sigma := \left|\mathcal{R}_{3^-}\right|$ be the number of internal nodes with node degree less than three, then, to transform the topology G to an identifiable topology G_i, for the first case mentioned above, σ number of links and edge nodes need to be added and connected to the internal nodes with degree < 3 so that the condition in Theorem 3.4.1 is met. In this case, the total number of links in the identifiable topology G_i will be

$$\gamma_{G_i} = \gamma_G + \sigma, \tag{6.2}$$

where γ_{G_i} and γ_G are the total number of links in G_i and G, respectively. Procedure 2 is used to transform the topology of this case into an identifiable one.

Procedure 2: $Case1(G, \mathcal{R}_{3^-})$

```
  **Output:** $G_i$
1 **foreach** $v_i \in \mathcal{R}_{3^-}$ **do**
2     $\mathcal{E}(G) \leftarrow \mathcal{E}(G) \cup \{v_j\}$
3     $G \leftarrow G + \{v_i v_j\}$
4 $G_i \leftarrow G$
5 **return** $G_i$
```

6.7.2.2 Transforming Case 2 Topologies

Unlike case 1, transforming a topology of case 2 is not straightforward. For topologies of this case, in order to keep the number of links to a minimum, instead of adding more links, existing links can be restructured. This is because in case 2 there is at least one internal node v_i with $d(v_i) > 3$. So we can disconnect some links incident to v_i and connect them to other nodes in $\mathcal{R}_{3^-}$ while maintaining node degree of at least three for all internal nodes, hence satisfying Theorem 3.4.1. And, if needed, extra links will be added, as described next.

There are, however, certain assertions that need to be taken care of when restructuring the topology to ensure that the resulting topology is connected and acyclic.

Remark 6.7.1. Let $G^{'}$ be a resulting topology after restructuring any link in unidentifiable topology G, then $G^{'}$ should satisfy the following conditions

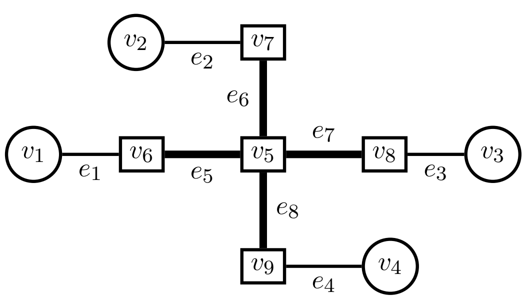
1. it is a connected graph;

2. it has no self-loops;

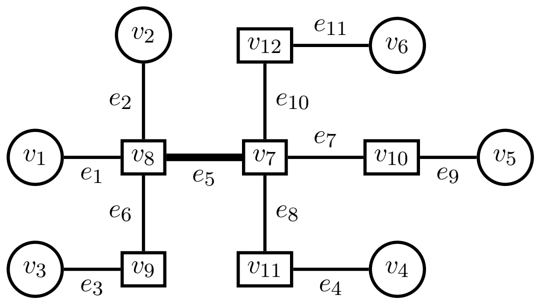
3. it is not multigraph; and

4. it has no cycles.

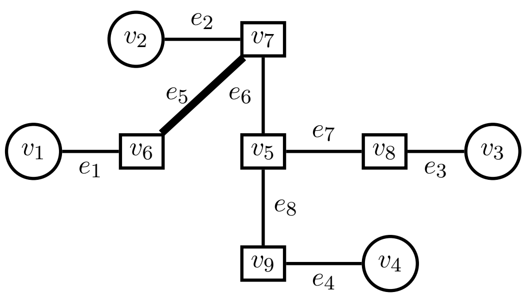
To restructure the unidentifiable topology, two types of links can be used: *Partially Restructurable Links (PRLs)* and *Fully Restructurable Links (FRLs)*. Figure 6.5 shows an example of the difference between the two. In the following, we will describe each one of these links and show how they can be used to transform the topology towards an identifiable one.



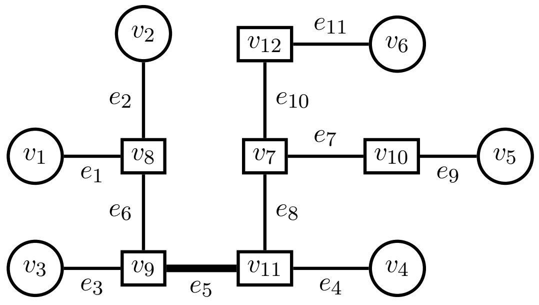
(a) $e_5, \dots, e_8$ are PRLs



(b) e_5 is FRL



(c) Topology after restructuring PRL e_5



(d) Topology after restructuring FRL e_5

Figure 6.5: Difference between PRLs and FRLs.

Transformation using PRLs Figure 6.5(a) and Figure 6.5(c) show examples of PRLs and how to restructure them. In addition, the following definition formally defines PRL.

Definition 6.7.3. For a node $v_i \in \mathcal{R}_{3^+}$, a *PRL* is an internal link, $e_i \in \mathcal{I}(G), i \in \{1, 2, \dots, \gamma\}$, that is incident to $v_i, i \in \{1, 2, \dots, \eta\}$ such that it can be disconnected from it while keeping $d(v_i) \geq 3$.

Let $\mathcal{N}(v_i)$ be a set of neighbours directly connected to $v_i \in V(G)$ and let $\mathcal{N}_{\mathcal{R}}(v_i) \subseteq \mathcal{N}(v_i)$ be a set of *internal* nodes in $\mathcal{R}(G)$ directly connected to $v_i \in V(G)$, then because PRL e_i is an internal link, and according to Definition 6.7.1, its both end-points are internal nodes.

Hence, the number of PRL incident to v_i can be computed by counting the number of internal nodes that are neighbours to v_i, let $\zeta_{v_i} := \left|\mathcal{N}_{\mathcal{R}}(v_i)\right|$ be this number and let $\varphi_{v_i} := d(v_i) - 3$ $:= d(v_i) - 3$ be the number of remaining links incident to node $v_i \in V(G)$ after disconnecting it from any three links. For identifiable topologies, $\varphi_{v_i} \geq 0, \forall v_i \in \mathcal{R}(G)$, otherwise if the topology is unidentifiable, then $\varphi_{v_i} < 0, \exists v_i \in \mathcal{R}(G)$. Further, let $\psi := \left|R_{3+}\right|$, then the following theorem quantifies the maximum number of links that can be restructured in G.

Theorem 6.7.3. *For unidentifiable topology G of case 2, if $\psi \geq 1$, then the maximum number of PRLs, γ_{PRL}, such that $d(v_i) \geq 3, \forall v_i \in \mathcal{R}(G)$, is*

$$\gamma_{PRL} = \sum_{i=1}^{\psi} \gamma_{PRL}(v_i), \tag{6.3}$$

where

$$\gamma_{PRL}(v_i) = \begin{cases} \varphi_{v_i}, & \text{if } \zeta_{v_i} > \varphi_{v_i} \\ \zeta_{v_i} - 1, & \text{otherwise} \end{cases}. \tag{6.4}$$

Proof. Based on Theorem 3.4.1, at least three links should be incident to v_i, $\forall v_i \in \mathcal{R}(G)$. Hence, for $\psi = 1$, no more than φ_{v_i} links can be restructured in G. This means that disconnecting φ_{v_i} links from v_i would leave $d(v_i) = 3$

$$d(v_i) - \varphi_{v_i} = d(v_i) - \big(d(v_i) - 3\big) = 3.$$

However, disconnecting φ_{v_i} links can only guarantee $d(v_i) = 3$ but cannot ensure that disconnecting any of φ_{v_i} links would keep the topology connected. For this, Theorem 6.7.1 requires that for each internal node, $v_i \in \mathcal{R}(G)$, it should be incident to at least one link in $\mathcal{I}(G)$. Therefore, such a link cannot be used as PRL. From Theorem 6.7.1, we know that it is necessary for each $v_i \in \mathcal{R}(G)$ to be connected to at least one link in $\mathcal{I}(G)$, therefore, only if $\zeta_{v_i} > \varphi_{v_i}$, we can restructure all φ_{v_i} links. Otherwise, if $\zeta_{v_i} \leq \varphi_{v_i}$, restructuring all φ_{v_i} links will result in $\left|\mathcal{I}(v_i)\right| < 1$ which violates the connectivity condition in Theorem 6.7.1. Therefore, to ensure satisfiability of Theorem 6.7.1 in this case, only up to $\zeta_{v_i} - 1$ can be restructured, which proves (6.4).

For $\psi > 1$, the same argument applies for individual nodes in $\mathcal{R}_{3+}$ for which (6.3) is proven. □

To use PRLs, it is important to consider the conditions in Remark 6.7.1. Let $\mathcal{W}(G) :=$

$\{v_i \in \mathcal{R}_{3+} : \zeta_{v_i} > 1\}$ be a set of internal nodes having node degree larger than three *and* are neighbours to more than one internal node, and let $\omega := \left|\mathcal{W}\right|$. The following proposition ensures that $G^{'}$ is not multigraph and does not have self-loops.

Proposition 6.7.1. *To ensure that $G^{'}$ is a simple graph, restructuring PRL link $e_i = v_iv_j$, in unidentifiable topology G, into $e_i^{'} = v_jv_k$, where $v_i \in \mathcal{W}$, $v_j \in \mathcal{N}(v_i)$ and $v_k \in \mathcal{R}_{3-}$, is the mapping between e_i and $e_i^{'}$, that is, $e_i \in E(G) \to e_i^{'} \in E(G^{'})$ such that the following conditions are met*

1. *$v_j \neq v_k$,*
2. *$v_k \notin \mathcal{N}(v_j)$ (or $v_j \notin \mathcal{N}(v_k)$).*

Proof. Let $G^{'}$ be a resulting topology after restructuring any link, then such topology has to be a simple graph.
Condition 1: assume that $v_j = v_k$ in $G^{'}$, then connecting v_j and v_k, which form link $e_i^{'}$, will result in self-loop with $v_h(e_i^{'}) = v_t(e_i^{'})$. This contradicts that $G^{'}$ is acyclic.
Condition 2: now assume that $v_k \in \mathcal{N}(v_j)$ (or $v_j \in \mathcal{N}(v_k)$), then connecting v_k and v_j will result in multigraph where v_k and v_j share two links. Again, this contradicts that $G^{'}$ is acyclic. □

The conditions defined in Proposition 6.7.1 are the necessary conditions to obtain a simple graph. These conditions, however, do not ensure that the resulting graph is connected. To ensure connectivity, Theorem 6.7.1 states that $\mathcal{I}(v_i) \neq \emptyset, \forall v_i \in \mathcal{R}(G)$. Therefore, $\zeta_{v_i} \geq 1, \forall v_i \in \mathcal{R}(G)$. To restructure any PRL, it is important to ensure that this is satisfied for all internal nodes in the resulting graph $G^{'}$.

Assuming that $\mathcal{W} \neq \emptyset$, and $\sigma \geq 1$, then the following is the sufficient condition to have acyclic and connected topology satisfying all conditions in Remark 6.7.1.

Proposition 6.7.2. *Any PRL $e_i = v_iv_j$ in G can be disconnected from its end-point $v_i \in \mathcal{W}$ and connected to $v_k \in \mathcal{R}_{3-}$ such that the resulting graph $G^{'} = G^* + v_jv_k$ is acyclic and connected iff $v_j \in c_1$ and $v_k \in c_2$ (or $v_j \in c_2$ and $v_k \in c_1$) in G^*, where $G^* = G - \{v_iv_j\}$ and $e_1, e_2 \in \mathcal{C}(G^*)$.*

Proof. $\to$ Provided that $v_jv_k \notin E(G)$, we assume $G^{'} = G^* + \{v_jv_k\}$, where $G^* = G - \{v_iv_j\}$, is acyclic and connected graph, and prove that $v_j \in c_1$ and $v_k \in c_2$ (or $v_j \in c_2$ and $v_k \in c_1$). We prove this by contradiction, assuming that $G^{'}$ is not acyclic and not connected.

We know that the original graph G is acyclic and connected with a number of links $\gamma_G = \eta_G - 1$, then for $G^* = G - \{v_i v_j\}$, the number of links decreases by one. Hence, $\gamma_{G^*} < \eta_{G^*} - 1$. Then if $v_j \in c_1$ and $v_k \in c_2$ (or if $v_j \in c_2$ and $v_k \in c_1$), the resulting graph $G^{'} = G^* + \{v_j v_k\}$ would have $\gamma_{G^{'}} = \eta_{G^{'}} - 1$ and according to Lemma 6.7.1, $G^{'}$ is acyclic and connected graph, which contradicts the assumption.

$\leftarrow$ We assume for G^* that $v_j \in c_1$ and $v_k \in c_2$ (or $v_j \in c_2$ and $v_k \in c_1$) and prove that the resulting graph $G^{'} = G^* + \{v_j v_k\}$ is an acyclic connected graph. For this, we prove the contrapositive that if both $v_j, v_k \in c_1$ (or $v_j, v_k \in c_2$), then $G^* + \{v_j v_k\}$ is *neither* acyclic *nor* connected graph.

Since G is maximally acyclic (see acyclic graph definition in Chapter 2.6), then $c_1, c_2 \in \mathcal{C}(G^*)$ must also be maximally acyclic. Thus, $c_1 + v_j v_k$ (or $c_2 + v_j v_k$) results in having a cycle in c_1 (or c_2) while the graph is still disconnected. □

Procedure 3 shows how PRLs are used to transform an unidentifiable topology towards an identifiable one.

While both σ and ω are larger than 0, the procedure starts by looping through elements of $\mathcal{W}$ and checking their neighbours. If a neighbour $v_j \in \mathcal{N}(v_i)$ is not an internal node, it will be skipped (lines 1-6). Otherwise, the link between $v_i \in \mathcal{W}$ and $v_j \in \mathcal{N}(v_i)$ will be disconnected ($G - \{v_i v_j\}$), and the resulting two components c_1, c_2 will be added to $\mathcal{C}$ (lines 7-9). If a node $v_k \in \mathcal{R}_{3-}$ is in a different component than v_j, a link between these nodes will be added, then ω and σ get updated (lines 10-14).

Transformation using FRLs In some cases, a link can be completely disconnected from both end-points to connect other two end-points. We call such link *Fully Restructurable Link (FRL).*

Definition 6.7.4. A *FRL* is an internal link, $e_i \in \mathcal{I}(G), i \in \{1, 2, \dots, \gamma\}$, with $e_i = v_i v_j$ such that both $v_i \in \mathcal{R}_{3+}, i \in \{1, 2, \dots, \eta\}$ and $v_j \in \mathcal{R}_{3+}, j \in \{1, 2, \dots, \eta\}$, and the number of internal links they are incident to is larger than 1, i.e., $\left|\mathcal{I}(v_i)\right| > 1$ and $\left|\mathcal{I}(v_j)\right| > 1$. Therefore, fully restructurable links can only exist in scenarios where $\omega > 1$.

An example of FRL and how it can be restructured is shown in Figure 6.5(b) and Figure 6.5(d).

Using FRLs instead of PRLs can speed up the transformation process by reducing σ by a factor of 2. This is because FRLs can be disconnected from their end-points and used to connect

Procedure 3: $PRL(G, \mathcal{R}_{3-}, \mathcal{W})$

```
Output : $G_{PRL}$
Initialize:
$G_{PRL} \leftarrow G$
$\sigma \leftarrow \left|\mathcal{R}_{3-}\right|$
$\omega \leftarrow \left|\mathcal{W}\right|$
$\mathcal{R}(G) \leftarrow \{v_i \in V(G) : d(v_i) \geq 2\}$
$\mathcal{C} \leftarrow \emptyset$
1  while $\sigma > 0$ and $\omega > 0$ do
2      for $i = 1 : \omega$ do
3          $v_i = \mathcal{W}[i]$
4          for each $v_j \in \mathcal{N}(v_i)$ do
5              if $v_j \notin \mathcal{R}(G)$ then
6                  continue
7              else
8                  $G \leftarrow G - \{v_i v_j\}$
9                  $\mathcal{C} \leftarrow \mathcal{C} \cup \{c_1, c_2\}$
10                 for $k = 1 : \sigma$ do
11                     $v_k = \mathcal{R}_{3-}[k]$
12                     if $(v_j \in c_1$ and $v_k \in c_2)$ or $(v_j \in c_2$ and $v_k \in c_1)$ then
13                         $G \leftarrow G + \{v_j v_k\}$
14                         update $\omega$ and $\sigma$
15                     else
16                         continue
17 $G_{PRL} \leftarrow G$
18 return $G_{PRL}$
```

other *two* nodes in $\mathcal{R}_{3-}$. However, as in the case of partially restructurable links, reconnecting fully restructurable links should satisfy the conditions in Remark 6.7.1. Thus, similar to PRLs, the following conditions are necessary for $G^{'}$ to be acyclic.

Proposition 6.7.3. *To ensure that $G^{'}$ is a simple graph, restructuring FRL link $e_i = v_i v_j$, in unidentifiable topology G, into $e_i^{'} = v_k v_m$, where $v_i, v_j \in \mathcal{W}$ with $v_j \in \mathcal{N}(v_i)$ and $v_k, v_m \in \mathcal{R}_{3-}$, is the mapping between e_i and $e_i^{'}$, i.e., $e_i \in E(G) \rightarrow e_i^{'} \in E(G^{'})$ such that the following conditions are met*

1. $v_k \neq v_m$,

2. $v_k \notin \mathcal{N}(v_m)$ *(or* $v_m \notin \mathcal{N}(v_k)$*).*

Proof. Replacing v_k by v_j and v_m by v_k, then the proof is similar to the one for Proposition 6.7.1 □

Assuming that $\omega \geq 2$ and there are at least two nodes in $\mathcal{W}$ that are neighbours, then the following is the sufficient condition to ensure having acyclic and connected topology $G^{'}$.

Proposition 6.7.4. *Any FRL in unidentifiable topology G can be disconnected from their end-points, $v_i \in \mathcal{R}_{3+}$ and $v_j \in \mathcal{R}_{3+}$, and reconnected to other end-points $v_k \in \mathcal{R}_{3-}$ and $v_m \in \mathcal{R}_{3-}$, such that the resulting graph $G^{'}$ is acyclic and connected iff for $G - \{v_i v_j\}$, $v_k \in c_1$ (or $v_k \in c_2$) and $v_m \in c_2$ (or $v_m \in c_1$), where $c_1, c_2 \in \mathcal{C}(G^*)$ and $G^* = G - \{v_i v_j\}$.*

Proof. Replacing v_k with v_j and v_m with v_k in Proposition 6.7.2 then the proof is similar to that of Proposition 6.7.2. □

Based on the above theoretical analysis, Procedure 4 is derived, illustrating how FRLs can be used to transform an unidentifiable topology towards an identifiable one. This procedure is similar to Procedure 3 except that it uses FRL when $\omega \geq 2$ and $\sigma \geq 2$. It finds two neighbouring nodes in $\mathcal{W}$ and disconnects them, then it checks if two nodes $v_k, v_m \in \mathcal{R}_{3-}$ are in different components, if so a link will be added between them.

Transforming an unidentifiable topology G into an identifiable one, G_i, by restructuring links using either PRLs or FRLs results in the total number of links γ_{G_i} in G_i being

$$\gamma_{G_i} = \begin{cases} \gamma_G, & \text{if } \varphi_{v_i} \geq \sigma \text{ and } \varphi_{v_i} < \zeta_{v_i} \\ \gamma_G, & \text{if } \zeta_{v_i} - 1 \geq \sigma \text{ and } \varphi_{v_i} \geq \zeta_{v_i} \\ \gamma_G + (\sigma - \varphi_{v_i}), & \text{if } \varphi_{v_i} < \sigma \text{ and } \varphi_{v_i} < \zeta_{v_i} \\ \gamma_G + (\sigma - \zeta_{v_i} + 1), & \text{if } \zeta_{v_i} - 1 < \sigma \text{ and } \varphi_{v_i} \geq \zeta_{v_i} \end{cases}. \quad (6.5)$$

6.7.2.3 Example

Figure 6.6 shows two possibilities of improving the unidentifiable topology shown in Figure 6.3 by transforming it into an identifiable one. Note that the original topology shown in Figure 6.3 falls under case 2[3]. Therefore, we can restructure and add links as discussed above.

Figure 6.3 has $\omega = 1$, thus there is no FRL, instead, PRL can be used according to Procedure 3. The resulting topology is shown in Figure 6.6(b). On the other hand, the identifiable topology shown in Figure 6.6(a) is achieved by simply adding more links to the gateways using Procedure 2.

[3]Notice that node $d(s) = 4$ while the degree of the other internal nodes is $2 < 3$.

Procedure 4: $FRL(G, \mathcal{R}_{3^-}, \mathcal{W})$

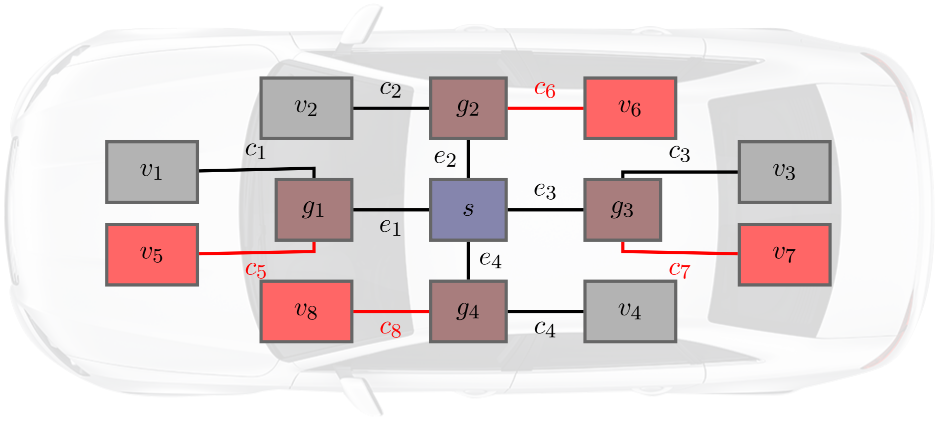
```
   Output : $G_{FRL}$
   Initialize:
   $G_{FRL} \leftarrow G$
   $\mathcal{C} \leftarrow \emptyset$
   $\sigma \leftarrow |\mathcal{R}_{3^-}|$
   $\omega \leftarrow |\mathcal{W}|$
1  while $\omega \geq 2$ and $\sigma \geq 2$ do
2      for $i = 1 : \omega$ do
3          $v_i = \mathcal{R}_{3^+}[i]$
4          for $j = i + 1 : \omega$ do
5              $v_j = \mathcal{R}_{3^+}[j]$
6              if $v_j \in \mathcal{N}(v_i)$ then
7                  $G \leftarrow G - \{v_i v_j\}$
8                  $\mathcal{C} \leftarrow \mathcal{C} \cup \{c_1, c_2\}$
9                  for $k = 1 : \sigma$ do
10                     $v_k = \mathcal{R}_{3^-}[k]$
11                     for $m = k + 1 : \sigma$ do
12                         $v_m = \mathcal{R}_{3^-}[m]$
13                         if $(v_k \in c_1$ and $v_m \in c_2)$ or $(v_m \in c_1$ and $v_k \in c_2)$ then
14                             $G \leftarrow G + \{v_k v_m\}$
15                             update $\omega$ and $\sigma$
16                         else
17                             continue
18 $G_{FRL} \leftarrow G$
19 return $G_{FRL}$
```

It is clear that following Procedure 3 is more efficient, in terms of the number of total links in the resulting topology, than adding more links to the internal nodes that have node degree of less than three (using Procedure 2). According to (6.5) and (6.2), for this example, $\gamma + (\sigma - \varphi_{v_i}) < \gamma + \sigma$. In fact, the total number of links in the topology shown in Figure 6.6(b) (using PRLs) is the minimum number of links. This can be confirmed by applying (6.1).

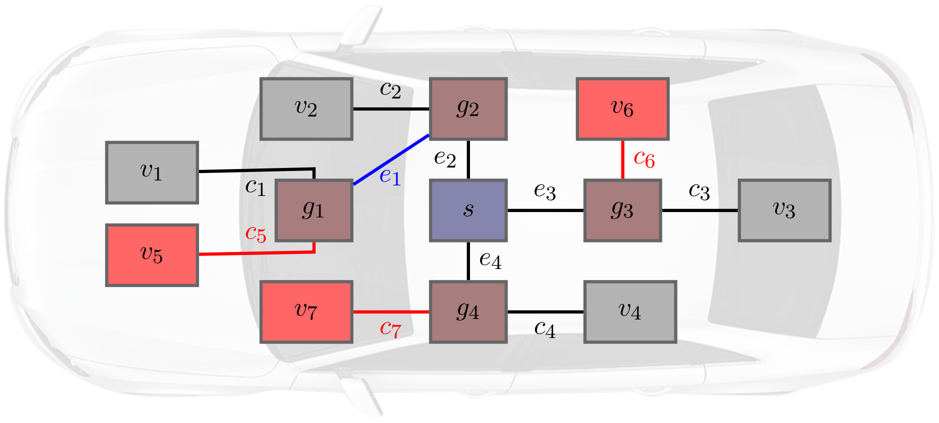
6.7.2.4 Transformation Algorithm

In this part, we propose a transformation algorithm that takes an unidentifiable in-vehicle network G and transforms it into an identifiable network G_i by restructuring the existing links, and/or, if necessary, adding more additional links.

Algorithm 2 starts by checking if $\omega > 0$, if so it checks if both $\omega \geq 2$ and $\sigma \geq 2$. In this case, it uses FRL for the transformation using Procedure 4. If the resulting topology is identifiable, it returns it and stops the algorithm (lines 4-6). Otherwise, it updates the values for ω and



(a) Achieving identifiable topology using Procedure 2, $\gamma = 12$



(b) Achieving identifiable topology using Procedure 3, $\gamma = 11$

Figure 6.6: Identifiable topology versions of Ethernet-based topology shown in Figure 6.3. Red link and nodes represent added nodes and links, blue link represents a restructured link.

σ, and uses the transformation with PRL using Procedure 3 (line 7-9). Again, it checks if the resulting topology is identifiable or not. If it is unidentifiable, it uses the transformation for case 1 using Procedure 2 after updating ω and σ (lines 13-16). If $\omega = \sigma = 1$, then the algorithm uses Procedure 3, checks for identifiability, and uses Procedure 2 if the resulting topology is unidentifiable (line 17-28).

In general, an unidentifiable topology is not guaranteed to have FRL for which Procedure 4 might be used, in this case, PRL, if existed, will be used (Procedure 3). Otherwise, the problem will be reduced to case 1 transformation (Procedure 2). A flowchart summary of how the overall transformation algorithm works is depicted in Figure 6.7. Note that Procedure 2, Procedure 3, and Procedure 4 will not be used in case $S_d(G) =$ true, in other words, if the network topology G is already identifiable. An example of this is the central-gateway architecture shown in Figure 5.1.

Algorithm 2: Transform to identifiable topology

```
Inputs  : $G$
Output  : $G_i$
Initialize:
$\mathcal{R}_{3^-} \leftarrow \{v_i \in \mathcal{R}(G) : d(v_i) < 3\}$
$\mathcal{R}_{3^+} \leftarrow \{v_i \in \mathcal{R}(G) : d(v_i) > 3\}$
$\mathcal{W} \leftarrow \{v_i \in \mathcal{R}_{3^+} : \zeta_{v_i} > 1\}$
$\omega \leftarrow |\mathcal{W}|$
$\sigma \leftarrow |\mathcal{R}_{3^-}|$
1  if $\omega > 0$ then
2      if $\omega \geq 2$ and $\sigma \geq 2$ then
3          $G_{FRL} \leftarrow FRL(G, \mathcal{R}_{3^-}, \mathcal{W})$
4          if $S_d(G_{FRL}) = true$ then
5              $G_i \leftarrow G_{FRL}$
6              go to line 28
7          else
8              update $\omega$ and $\sigma$
9              $G_{PRL} \leftarrow PRL(G_{FRL}, \mathcal{R}_{3^-}, \mathcal{W})$
10             if $S_d(G_{PRL}) = true$ then
11                 $G_i \leftarrow G_{PRL}$
12                 go to line 28
13             else
14                 update $\omega$ and $\sigma$
15                 $G_i \leftarrow Case1(G_{PRL}, \mathcal{R}_{3^-})$
16                 go to line 28
17     else
18         $G_{PRL} \leftarrow PRL(G, \mathcal{R}_{3^-}, \mathcal{W})$
19         if $S_d(G_{PRL}) = true$ then
20             $G_i \leftarrow G_{PRL}$
21             go to line 28
22         else
23             update $\mathcal{R}_{3^-}$
24             update $\mathcal{W}$
25             $G_i \leftarrow Case1(G_{PRL}, \mathcal{R}_{3^-})$
26             go to line 28
27 $G_i \leftarrow Case1(G, \mathcal{R}_{3^-})$
28 return $G_i$
```

Complexity Analysis The time complexity of checking for identifiability condition in Procedure 1 is $\mathcal{O}(\eta_G + \lambda_G)$ where η_G is the total number of nodes and λ_G is the total number of internal nodes in G. Procedure 2 takes $\mathcal{O}(\sigma)$, while restructuring links takes $\mathcal{O}(\omega \cdot d(v_i) \cdot \sigma)$ using PRL (Procedure 3) and $\mathcal{O}(\omega^2 \cdot \sigma^2)$ using FRL (Procedure 4), where $\omega := |\mathcal{W}|$ and $v_i \in \mathcal{W}$.

Thus, the complexity of the overall transformation algorithm is

$$\text{Complexity} = \begin{cases} \mathcal{O}(\eta_G + \lambda_G + \omega^2\sigma^2), & \text{if } \omega \geq 2 \text{ and } \sigma \geq 2 \\ \mathcal{O}(\omega \cdot d(v_i) + \frac{\eta + \lambda_G}{\sigma}), & \text{if } \omega = 1 \text{ or } \sigma = 1 \\ \mathcal{O}(\sigma), & \text{if } \omega = 0 \end{cases} \cdot \tag{6.6}$$

From (6.6), it is clear that the worst case scenario is when $\omega, \sigma \geq 2$, in which the algorithm takes $\mathcal{O}(\eta_G + \lambda_G + \omega^2\sigma^2)$, which is a quadratic time complexity, whereas in the other two cases, it takes linear time complexity.

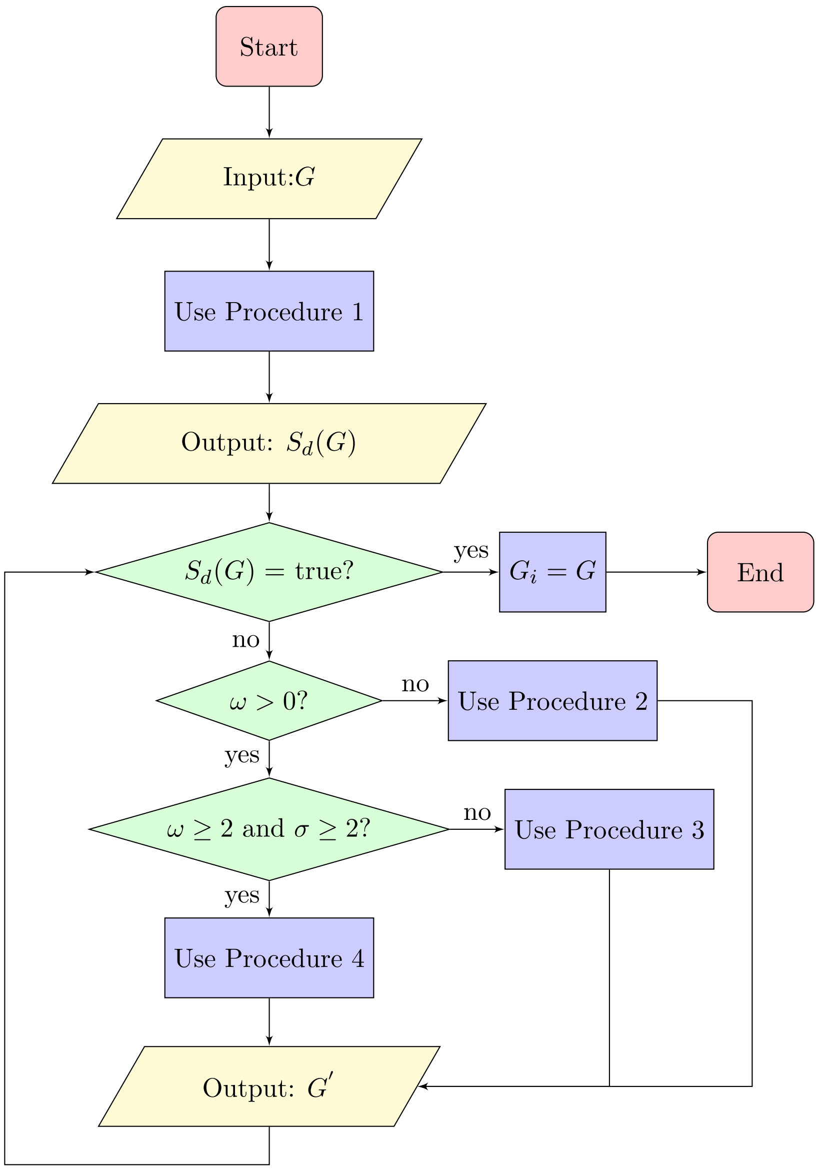


Figure 6.7: A flowchart illustrating the transformation algorithm of unidentifiable topology G into identifiable topology G_i.

Note that up to this point, this work is dealing with acyclic graphs. In the following, we need to have redundancy to ensure proper functionality of the vehicle under failure. As we will see, the topologies discussed next contain cycles.

6.8 Redundant Topology

The identifiable topologies discussed earlier do not have any redundant paths. In other words, there is only one single path between any communicating nodes in the network. This is particularly crucial for mission-critical traffic, which means that if there is any failure in the network, the network traffic might be disrupted. For example, suppose there is safety-related traffic between v_1 and v_2 in Figure 6.6(b), and the Ethernet link e_1 has failed. Because there are no other paths between v_1 and v_2, failure of link e_1 can result in disconnecting the communication between v_1 and v_2. This might cause serious issues in the network, which in turn could endanger lives. To tackle this, in-vehicle networks should have redundant paths that can be used in case there is any failure in the network. In this section, we further investigate topology identifiability and redundancy support at the same time. First, we address redundancy, then revisit identifiability under redundant topologies.

Adding a redundant link for each link in the topology is not efficient and can increase the weight cost and complexity of the vehicle. Therefore, like in the identifiability case, we focus on adding the minimum possible number of links while ensuring there are at least two *internally disjoint* paths (see definition of internally disjoint paths in Chapter 2.6) between any two edge nodes $v_i, v_j \in \mathcal{E}(G)$. In other words, the goal is to have $N_{v_i,v_j} \geq 2$ internally disjoint paths while keeping γ to a minimum, where N_{v_i,v_j} represents the total number of internally disjoint paths between v_i and v_j.

Let $p_i(v_i, v_j) \in \mathcal{P}(G)$ be the i^{th} path connecting edge nodes $v_i, v_j \in \mathcal{E}(G)$, then the following defines redundancy for in-vehicle network G.

Definition 6.8.1. An in-vehicle network G is *redundant*, if for any pair of edge nodes $v_i, v_j \in \mathcal{E}(G)$ such that $\mathcal{N}(v_i) \cap \mathcal{N}(v_j) = \emptyset$, there are *at least* two internally disjoint paths between them. Formally, we say that G is redundant if $\forall v_i, v_j \in \mathcal{E}(G) : \mathcal{N}(v_i) \cap \mathcal{N}(v_j) = \emptyset$ there are $N_{v_i,v_j} \geq 2$ paths between v_i and v_j: $p_1(v_i, v_j)$ and $p_2(v_i, v_j)$, such that $p_1(v_i, v_j) \cap p_2(v_i, v_j) = \{e_k, e_l\}$, where $e_k, e_l \in \mathcal{T}(G)$.

Based on our assumption that only internal links can fail, we claim the following.

Claim 6.8.1. *For communicating nodes $v_i, v_j \in \mathcal{E}(G)$ in an in-vehicle network G, if $\mathcal{N}(v_i) \cap \mathcal{N}(v_j) \neq \emptyset$, then even though there can be $N_{v_i,v_j} \geq 2$, it is sufficient and more efficient to use one path that goes through links only in $\mathcal{T}(G)$.*

This claim indicates that if there is common internal node(s) connected to both $v_i \in \mathcal{E}(G)$ and $v_j \in \mathcal{E}(G)$, then v_i and v_j are only incident to links in $\mathcal{T}(G)$ (external links). Therefore, the communication between v_i and v_j can go through these external links without passing through any internal links in $\mathcal{I}(G)$. In fact, paths that pass through links in $\mathcal{T}(G)$ only are shorter than those that also pass through $\mathcal{I}(G)$ links. Hence, because we assume that links in $\mathcal{T}(G)$ do not fail, the path between v_i and v_j does not need another redundant path. As a result, although there still can be two or more paths between v_i and v_j, having one path between such node-pair would suffice.

6.8.1 Topological Conditions

In order for an in-vehicle network G to be redundant, the following theorem states the necessary topological conditions.

Theorem 6.8.1 (necessary conditions)**.** *An in-vehicle network G is redundant if the following are satisfied*

1. *number of internal nodes in G is at least three, i.e., $\lambda_G \geq 3$;*
2. *each internal node in G is incident to at least two internal links, i.e., $|\mathcal{I}(v_i)| \geq 2, \forall v_i \in \mathcal{R}(G)$.*

Proof. We prove the necessity of both conditions by contradiction.

For the first condition, assume that G is redundant ($N_{v_i,v_j} \geq 2, \forall v_i, v_j \in \mathcal{E}(G)$) while having $\lambda_G < 3$ internal nodes. There are two cases for the values of λ_G: $\lambda_G = 2$ or $\lambda_G = 1$.

For $\lambda_G = 2$, let $\mathcal{R}(G) = \{v_i, v_j\}$ and $v_k, v_m \in \mathcal{E}(G)$ where $v_k \in \mathcal{N}(v_i)$ and $v_m \in \mathcal{N}(v_j)$. Then, according to Theorem 6.7.1, v_i and v_j must be connected. Hence, only when G allows multigraph (see definition of multigraph in Chapter 2.6) then $N_{v_k,v_m} \geq 2$. However, G has to be a simple graph, therefore, $N_{v_k,v_m} = 1, \forall v_k, v_m \in \mathcal{E}(G)$ which contradicts the assumption that it is redundant.

For the second case when $\lambda_G = 1$, let $\mathcal{R}(G) = \{v_i\}$ and $\mathcal{E}(G) = \{v_j, v_k\}$ where $v_j, v_k \in \mathcal{N}(v_i)$. In this case, $\mathcal{T}(G) = E(G)$ where all links are external links, then $N_{v_j,v_k} = 1, \forall v_j, v_k \in \mathcal{E}(G)$. This, again, contradicts that G is redundant.

For the second condition, assume that G is redundant and $\Big|\mathcal{I}(v_i)\Big| < 2, \exists v_i \in \mathcal{R}(G)$. According to the first condition, $\lambda_G \geq 3$, then for $\lambda_G = 3$, let $\mathcal{R}(G) = \{v_1, v_2, v_3\}$ where $\Big|\mathcal{I}(v_1)\Big| = 2$ and $\Big|\mathcal{I}(v_2)\Big| = \Big|\mathcal{I}(v_3)\Big| = 1$. In addition, let $\mathcal{E}(G) = \{v_4, v_5, v_6\}$ with $v_4 \in \mathcal{N}(v_1)$, $v_5 \in \mathcal{N}(v_2)$ and $v_6 \in \mathcal{N}(v_3)$. Then, $N_{v_j,v_k} = 1, \forall v_j, v_k \in \mathcal{E}(G)$ which contradicts that G is redundant. □

Let $g \subset G$ be an *internal subgraph* of an in-vehicle network G, that is $g = (V(g), E(g))$, such that $V(g) = \mathcal{R}(G)$ and $E(g) = \mathcal{I}(G)$, where $\mathcal{R}(G)$ and $\mathcal{I}(G)$ are the sets of internal nodes and internal links in G, respectively. Then the following theorem states the sufficient condition for G to be redundant.

Theorem 6.8.2 (sufficient condition)**.** *An in-vehicle network G is redundant iff $g \subset G$ is k-edge-connected with $k \geq 2$.*

Proof. We prove that a redundant topology G implies that $g \subset G$ is k-edge-connected with at least $k \geq 2$ and vice versa.

$\rightarrow$ Let us assume that G is redundant. Then according to Definition 6.8.1, for any node-pair $v_i, v_j \in \mathcal{E}(G)$ there are at least two internally disjoint paths between them. Let such paths be $p_1(v_i, v_j)$ and $p_2(v_i, v_j)$, then by definition, only internal links in such paths are disjoint while the external links are the same in both paths. Since all nodes in g are internal nodes in G, $V(g) = \mathcal{R}(G)$, then there are at least two disjoint paths ($p_1(v_i, v_j)$ and $p_2(v_i, v_j)$ where $p_1 \cap p_2 = \emptyset$) between any node-pair $v_i, v_j \in V(g)$. Therefore, removing any link $e_i \in E(g)$, keeps the internal graph g connected, hence by definition of k-edge-connectivity (see Chapter 2.6), g is at least 2-edge-connected.

$\leftarrow$ Assume $g \subset G$ is 2-edge-connected, then there are at least two disjoint paths ($p_1(v_i, v_j)$ and $p_2(v_i, v_j)$ where $p_1 \cap p_2 = \emptyset$) between any node-pair $v_i, v_j \in V(g)$. Because $V(g) = \mathcal{R}(G)$, there are at least two disjoint internal paths between any node-pair in $\mathcal{E}(G)$. By Definition 6.8.1, G in this case is redundant. □

6.8.2 Transformation into Redundant Topology

Based on the aforementioned conditions, we propose a transformation algorithm which converts an existing in-vehicle network topology G that is non-redundant into a redundant topology G_r. To achieve this, the following definition is needed.

Definition 6.8.2. A *bridge*[4] $e_i \in E(G), i \in \{1, 2, \dots, \gamma\}$ is a link in a connected network G such that its removal, $E(G) - \{e_i\}$, results in having more than one component, i.e., disconnected topology with $\left|\mathcal{C}(G)\right| = 2$ (see graph connectivity definition in Chapter 2.6).

Let $\mathcal{B}(G)$ be the set of bridges in G, then Theorem 6.8.2 can be translated into a condition of *not* having bridges in the internal graph $g \subset G$. In other words, $\mathcal{B}(g) = \emptyset$.

Theorem 6.8.3. *A k-edge-connected subgraph $g \subset G$ with $k \geq 2$ implies that $\mathcal{B}(g) = \emptyset$.*

Proof. We prove the contrapositive that k-edge-connected subgraph $g \subset G$ with $k < 2$ implies that $\mathcal{B}(g) \neq \emptyset$.

Assuming $g \subset G$ is 1-edge-connected, then by definition of k-edge-connectivity (see Chapter 2.6), removing some link $e_i \in E(g)$ results in g being disconnected. Hence, by Definition 6.8.2, $e_i \in \mathcal{B}(g)$. Therefore, $\mathcal{B}(g) \neq \emptyset$. $\square$

The above theorem provides the basis in which we can check for redundancy of a given topology G as shown in Procedure 6. Recall that the condition in Theorem 6.8.3 is to be satisfied for the internal network $g \subset G$, Procedure 5 is then needed to extract g out of G, which then will be used as input to Procedure 6.

Procedure 5: *extractInternalGraph*(G)

```
  Output : $g$
  Initialize:
  $g \leftarrow (\emptyset, \emptyset)$
  $V(g) \leftarrow \emptyset$
  $E(g) \leftarrow \emptyset$
  $\mathcal{R}(G) \leftarrow \{v_i \in V(G) : d(v_i) \geq 2\}$
  $\mathcal{I}(G) \leftarrow \{e_i \in E(G) : v_h(e_i), v_t(e_i) \in \mathcal{R}(G)\}$
1 foreach $v_i \in \mathcal{R}(G)$ do
2     $V(g) \leftarrow V(g) \cup \{v_i\}$
3 foreach $e_i \in \mathcal{I}(G)$ do
4     $E(g) \leftarrow E(g) \cup \{e_i\}$
5 $g \leftarrow (V(g), E(g))$
6 return $g$
```

Finding bridges can be done using the algorithm in [235], where it takes $\mathcal{O}(\lambda_G + \left|\mathcal{I}(G)\right|)$, and it can be used to check for redundancy condition in Procedure 6. However, we use a more efficient one that is based on Definition 6.8.2 and takes only $\mathcal{O}(\left|\mathcal{I}(G)\right|)$ (see Appendix C.2).

[4]Sometimes also called *cut-edge*.

Procedure 6: *isRedundant*(G)

```
   Output : $S_r(G)$
   Initialize:
   $g \leftarrow extractInternalGraph(G)$
   $\mathcal{B}(g) \leftarrow findBridges(g)$
1  if $\mathcal{B}(g) \neq \emptyset$ then
2      $S_r(G) \leftarrow false$
3  else
4      $S_r(G) \leftarrow true$
5  return $S_r(G)$
```

6.8.2.1 Transformation Algorithm

Let $\mathcal{E}(g) := \{v_i \in V(g) : d(v_i)_g = 1\}$ (where $d(v_i)_g$ is degree of v_i in the extracted subgraph $g \subset G$) be the set of leaves in g, and let $g_{\mathcal{B}} \subset g$ be a subgraph of g that only contains bridges of g. Then the transformation algorithm connects such leaves if $\left|\mathcal{E}(g)\right| > 1$. In particular, if $\left|\mathcal{E}(g)\right|$ is *even*, it will connect nodes in $\mathcal{E}(g)$ together until $S_r(G) = true$. Otherwise, if $\left|\mathcal{E}(g)\right|$ is *odd*, it will connect $\left|\mathcal{E}(g)\right| - 1$ nodes from $\mathcal{E}(g)$ together while the last node in $\mathcal{E}(g)$ will be connected to any internal node in $V(g)$. This transformation is illustrated in Algorithm 3. If there is only one bridge, the algorithm connects any of its end-points to any internal node that satisfies the conditions in Proposition 6.7.1, then G_r will be returned as redundant topology (line 1-7). Otherwise, all leaves of $g_{\mathcal{B}}$ will be stored in $\mathcal{E}(g)$ (line 8-10). Next, for each two leaves, the algorithm connects them together, provided that they satisfy the required conditions, then if the topology is redundant, it will be returned and the algorithm will stop, or else it will reinvoke itself (lines 11-21).

Complexity Analysis Extracting internal subgraph $g \subset G$ using Procedure 5 takes $\mathcal{O}(\eta_G + \gamma_G + \lambda_G + |\mathcal{I}(G)|)$. The complexity of finding bridges in g is $\mathcal{O}(\left|\mathcal{I}(G)\right|)$. Checking for redundancy condition using Procedure 6 takes $\mathcal{O}(\eta_G + \lambda_G + \left|\mathcal{I}(G)\right|)$. Therefore, the complexity of the overall transformation algorithm is

$$\text{Complexity} = \begin{cases} \mathcal{O}(\eta_G + \gamma_G + \lambda_G + \left|\mathcal{I}(G)\right| + \lambda_G\left|V(g_{\mathcal{B}})\right|), & \text{if } \left|\mathcal{B}(G)\right| = 1 \\ \mathcal{O}(\left|\mathcal{B}(G)\right|(\eta_G + \gamma_G + \lambda_G + \left|\mathcal{E}(g)\right|^2 + \left|V(g_{\mathcal{B}})\right|)), & \text{if } \left|\mathcal{B}(G)\right| > 1 \end{cases}. \tag{6.7}$$

The worst case scenario according to (6.7) is, thus, when $\left|\mathcal{B}(G)\right| > 1$ for which Algorithm 3 takes $\mathcal{O}(\left|\mathcal{B}(G)\right|(\eta_G + \lambda_G + \left|\mathcal{E}(g)\right|^2 + \left|V(g_{\mathcal{B}})\right|))$.

Algorithm 3: Transform to redundant topology

```
   Inputs   : $G$
   Output   : $G_r$
   Initialize:
   $g \leftarrow extractInternalGraph(G)$
   $\mathcal{B}(G) \leftarrow findBridges(g)$
   $g_{\mathcal{B}} \leftarrow$ bridges graph of $g$
   $\mathcal{E}(g) \leftarrow \emptyset$
1  if $\left|\mathcal{B}(G)\right| = 1$ then
2      foreach $v_i \in V(g)$ do
3          for $v_j \in V(g_{\mathcal{B}})$ do
4              if $v_i \neq v_j$ and $v_i \notin \mathcal{N}(v_j)$ then
5                  $G \leftarrow G + \{v_i v_j\}$
6                  $G_r \leftarrow G$
7                  go to line 22
8  for $v_j \in V(g_{\mathcal{B}})$ do
9      if $d(v_j)_{g_{\mathcal{B}}} = 1$ then
10         $\mathcal{E}(g) \leftarrow \mathcal{E}(g) \cup \{v_j\}$
11 for $i = 1 : \left|\mathcal{E}(g)\right|$ do
12     $v_i \leftarrow \mathcal{E}(g)[i]$
13     for $j = i + 1 : \left|\mathcal{E}(g)\right|$ do
14         $v_j \leftarrow \mathcal{E}(g)[j]$
15         if $v_i \notin \mathcal{N}(v_j)$ then
16             $G \leftarrow G + \{v_i v_j\}$
17         if $S_r(G) = true$ then
18             $G_r \leftarrow G$
19             go to line 22
20         else
21             go to line 1
22 return $G_r$
```

6.8.3 Revisiting Identifiability for Redundant Topology

Based on our assumption that only edge nodes can monitor the in-vehicle network, satisfying the above conditions only ensures redundancy but does not guarantee an identifiable topology. Given that only nodes in $\mathcal{E}(G)$ can act as monitors, like in the identifiability case, the topological condition in Theorem 3.4.1 has to be satisfied.

It is worth noting that [19] and [20] have studied the topological conditions for graphs with cycles to achieve identifiability under the assumption that monitors can be placed in internal nodes (nodes in $\mathcal{R}(G)$). However, the requirements for in-vehicle networks are different in which only nodes in $\mathcal{E}(G)$ can be used as monitors, hence the conditions defined in [19] and [20] can not be applied in in-vehicle network scenarios.

Generally, if the topology is unidentifiable, there are two methods to do the transformation in terms of which order to follow: *Redundancy then Identifiability Transformation (RIT)* or *Identifiability then Redundancy Transformation (IRT)*. The following describes how each one of these methods will be used to transform the unidentifiable *and* non-redundant topology into an identifiable *and* redundant one.

6.8.3.1 Redundancy then Identifiability Transformation (RIT)

In this method, the topology will be checked for redundancy first using Procedure 6. If $S_r(G)$ = false, then Algorithm 3 is used to transform G into redundant topology G_r. Next, Procedure 1 will be used to check if G_r is identifiable or not ($S_d(G_r)$ = true or $S_d(G_r)$ = false), if $S_d(G_r)$ = false, then Algorithm 2 will be used to transform G_r to identifiable topology G_{ir} as discussed in Chapter 6.7.2.

6.8.3.2 Identifiability then Redundancy Transformation (IRT)

Here, Procedure 1 will be used first to check if G is identifiable or not. If $S_d(G)$ = false, then G will be transformed into identifiable topology G_i using Algorithm 2. Then, it will be converted into redundant topology G_{ir} using Algorithm 3.

Having these two methods, the natural question then is which one is more efficient. By efficient here we mean that results in a minimum number of added links. The next proposition clears this doubt.

Proposition 6.8.1. *Given an in-vehicle network topology G that is unidentifiable and non-redundant, if $\exists v_i \in \mathcal{E}(g)$, with $d(v_i)_G < 3$, then using RIT yields fewer number of links than IRT.*

Proof. Transforming a non-redundant topology G into redundant topology G_r results in total number of links γ_{G_r} as

$$\gamma_{G_r} = \begin{cases} \gamma_G + 1, & \text{if } \left|\mathcal{B}(G)\right| = 1 \\ \gamma_G + \left\lfloor \frac{\left|\mathcal{E}(g)\right|}{2} \right\rfloor, & \text{if } \left|\mathcal{B}(G)\right| > 1 \end{cases}. \tag{6.8}$$

So only one link will be added if $\left|\mathcal{B}(G)\right| = 1$ and $\left\lfloor \frac{\left|\mathcal{E}(g)\right|}{2} \right\rfloor$ otherwise. Assuming $\omega = 0$, then Procedure 2 will be used to transform the topology into an identifiable one, which results in adding σ links.

If RIT is used, then G will be first transformed into redundant topology G_r which results in adding $\left\lfloor \frac{\left|\mathcal{E}(g)\right|}{2} \right\rfloor$ links (or one link if $\left|\mathcal{B}(G)\right| = 1$). Since there is at least one node $v_i \in V(g)$ with $d(v_i) < 3$ and by definition all internal nodes in $\mathcal{R}(G)$ have a degree of at least two, then $d(v_i) = 2$. If $v_i \in \mathcal{E}(g)$, transforming into redundant topology then will increase the degree of this node, which in turn reduces σ by at least one. Then transforming into identifiable topology will add $\sigma - 1$ links at maximum. By (6.8), in the worst-case scenario, the total number of links in the transformed topology using RIT is

$$\gamma_{G_{ir}}^{RIT} = \begin{cases} \gamma_G + \sigma, & \text{if } |\mathcal{B}(G)| = 1 \\ \gamma_G + \left\lfloor \frac{\left|\mathcal{E}(g)\right|}{2} \right\rfloor + \sigma - 1, & \text{if } \left|\mathcal{B}(G)\right| > 1 \end{cases}. \tag{6.9}$$

On the contrary, if IRT is used, then G will be first transformed into identifiable topology G_i which results in adding σ links (assuming worst case scenario where $\omega = 0$). Next, it will be transformed into redundant topology which, again, adds one link if $\left|\mathcal{B}(G)\right| = 1$ and $\left\lfloor \frac{\left|\mathcal{E}(g)\right|}{2} \right\rfloor$ otherwise. Thus, the total number of links in the transformed topology using IRT is

$$\gamma_{G_{ir}}^{IRT} = \begin{cases} \gamma_G + \sigma + 1, & \text{if } |\mathcal{B}(G)| = 1 \\ \gamma_G + \left\lfloor \frac{\left|\mathcal{E}(g)\right|}{2} \right\rfloor + \sigma, & \text{if } \left|\mathcal{B}(G)\right| > 1 \end{cases}. \tag{6.10}$$

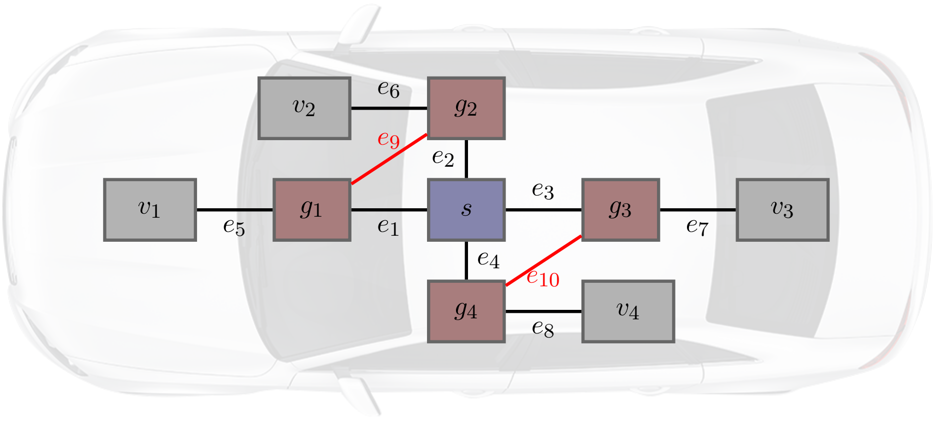
From (6.9) and (6.10) it is clear that

$$\gamma_G + \sigma < \gamma_G + \sigma + 1, \tag{6.11}$$

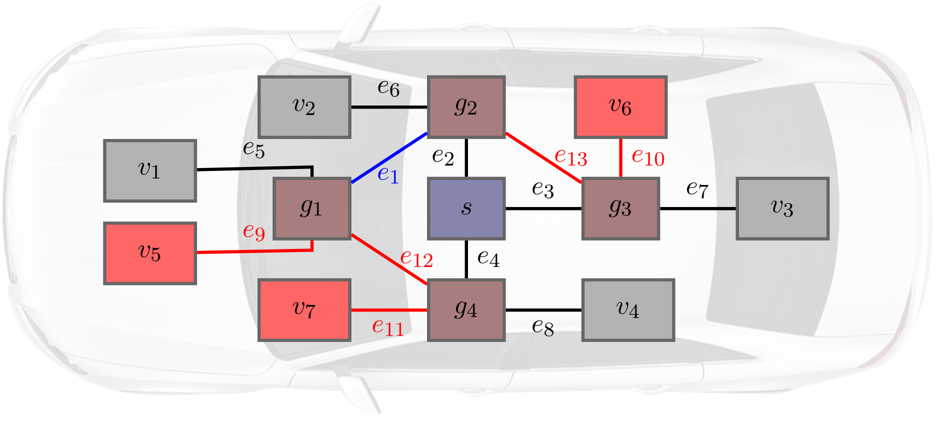
and

$$\gamma_G + \left\lfloor \frac{\left|\mathcal{E}(g)\right|}{2} \right\rfloor + \sigma - 1 < \gamma_G + \left\lfloor \frac{\left|\mathcal{E}(g)\right|}{2} \right\rfloor + \sigma. \tag{6.12}$$

Hence, using RIT results in fewer number of links than IRT. □



(a) Topology achieved using **RIT**, $\gamma_{G_{ir}}^{RIT} = 10$



(b) Topology achieved using **IRT**, $\gamma_{G_{ir}}^{IRT} = 13$

Figure 6.8: Redundant and identifiable topologies. Blue represents restructured link while red represent added links and nodes.

6.8.3.3 Example

Consider the original topology shown in Figure 6.3. This topology is unidentifiable and non-redundant with $\mathcal{E}(g) = \{g_1, g_2, g_3, g_4\}$ and $d(g_1)_G = d(g_2)_G = d(g_3)_G = d(g_4)_G = 2 < 3$. Figure 6.8(a) and Figure 6.8(b) show two transformed versions of this topology. Both versions are identifiable and redundant. Figure 6.8(a) is achieved using RIT, while Figure 6.8(b) is achieved using IRT. As shown, RIT results in identifiable and redundant topology G_{ir} with $\gamma_{G_{ir}}^{RIT} = 10$, while IRT results in more links with $\gamma_{G_{ir}}^{IRT} = 13$. Thus, Proposition 6.8.1 is true for this example.

6.9 SDN-enabled Topology

Incorporating SDN into the in-vehicle network can significantly facilitate the network tomography-based monitoring approach. For instance, in the new topology with redundancy feature, there can be multiple paths between edge nodes in $\mathcal{E}(G)$. SDN controller can coordinate these paths, in which it can distinguish between the path used in normal operation from the path that has to be activated only in case there is a failure in the original one. In addition, the SDN controller (which is a powerful unit, e.g., HPCP) can carry all the heavy work incurred by network tomography computations. For example, it can collect the end-to-end measurements from edge nodes, perform network tomography to infer the link-level performance, and then feedback to the underlying network to, e.g., deactivate the abnormal link and use the redundant one instead. Not only that network tomography can benefit from the advantages of SDN, but the opposite is also true. In fact, with network tomography being adopted as a monitoring approach for SDN-enabled networks, such networks can take advantage of the reduction in computational and monitoring traffic overhead that network tomography offers [169].

Given that the in-vehicle network topology is identifiable and redundant, we can discuss how to leverage network tomography and SDN to monitor the network. In the following, we show the final topology incorporating SDN. In addition, we propose a monitoring framework that leverages network tomography to detect, locate and mitigate anomalies in the network. Hence, achieving a complete monitoring system.

6.9.1 Proposed In-Vehicle Network Topology

Figure 6.9 shows an example of the final proposed topology that is SDN-enabled and at the same time, it supports both identifiability and redundancy properties as discussed in Chapter 6.7 and Chapter 6.8, respectively. In particular, $d(v_i) \geq 3, \forall v_i \in \mathcal{R}(G)$ (identifiability condition is satisfied) and $N_{v_i,v_j} \geq 2, \forall v_i, v_j \in \mathcal{E}(G)$ (topology is redundant). In addition, the proposed topology can support the new E/E architectures: either domain- or zonal-based architectures with domain/zonal controllers (see Chapter 2.2). Note that the domain/zonal controllers should support gateway as well as switching functionalities.

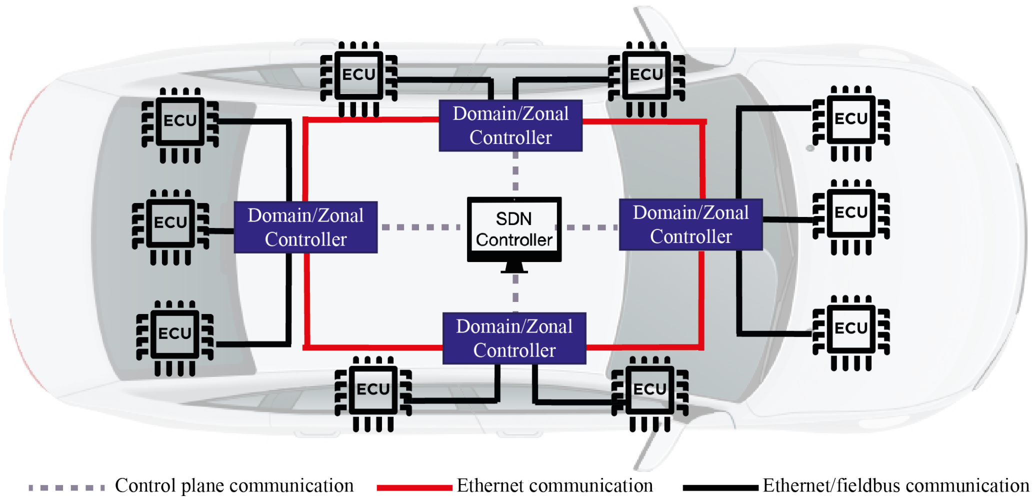


Figure 6.9: An example of the proposed SDN-enabled topology that is also identifiable and redundant.

6.9.2 Proposed Monitoring Framework

The proposed monitoring framework that takes advantage of the SDN paradigm and network tomography is shown in Figure 6.10. The main components reside at the application layer, which consists of *Monitoring Application* and *NT (Network Tomography) Module*. Note that other network management and monitoring applications can also be deployed in this layer.

The monitoring framework shown in Figure 6.10 follows three main steps: *anomaly detection*, *anomaly localisation* and *anomaly mitigation*. Each is explained in the following.

6.9.2.1 Anomaly Detection

Any anomaly in the in-vehicle network can be detected as follows:

- Edge nodes in $\mathcal{E}(G)$ measure the monitoring traffic along the measurement paths in $\mathcal{P}_m(G)$. For example, they can measure the end-to-end delay, end-to-end packet loss, etc.
- Each edge node $v_i \in \mathcal{E}(G), i \in \{1, 2, \ldots, \left|\mathcal{E}(G)\right|\}$ compares the end-to-end measurement $y_i, i \in \{1, 2, \ldots, \kappa_G\}$ for the monitored path $p_i \in \mathcal{P}_m(G), i \in \{1, 2, \ldots, \left|\mathcal{P}(G)\right|\}$ with the normal behaviour (predetermined) value $\bar{y}_i$.
- If $\mathcal{H}_{p_i} > \delta_{p_i}$, where $\mathcal{H}_{p_i} := \left|\bar{y}_i - y_i\right|$ and δ_{p_i} is a predefined threshold value, then v_i informs the monitoring application about an anomaly it detected in p_i (*step 1* as shown in Figure 6.10).

6.9.2.2 Anomaly Localisation

While $\mathcal{H}_{p_i} > \delta_{p_i}, \exists p_i \in \mathcal{P}_m(G)$, locating the anomalous link can be achieved as follows:

- The monitoring application activates the NT module (*step 2* in Figure 6.10).
- The NT module activates the NT agents at all nodes in $\mathcal{E}(G)$ (*step 3* in Figure 6.10).
- Nodes in $\mathcal{E}(G)$ send their latest end-to-end measurements $\boldsymbol{y}$ to the NT module (*step 4* in Figure 6.10).
- After solving (3.1), the NT module informs the monitoring application that e_i is the anomalous link (*step 5* in Figure 6.10) such that $\mathcal{H}_{e_i} > \delta_{e_i}$ where $\mathcal{H}_{e_i} := \Big|\bar{x}_i - x_i\Big|$, $\bar{x}_i$ is the normal behaviour (predetermined) value for link $e_i \in E(G)$, and δ_{e_i} is a predefined threshold value. Note that if binary network tomography is used, the anomalous link e_i is the one with $x_i = 1$.

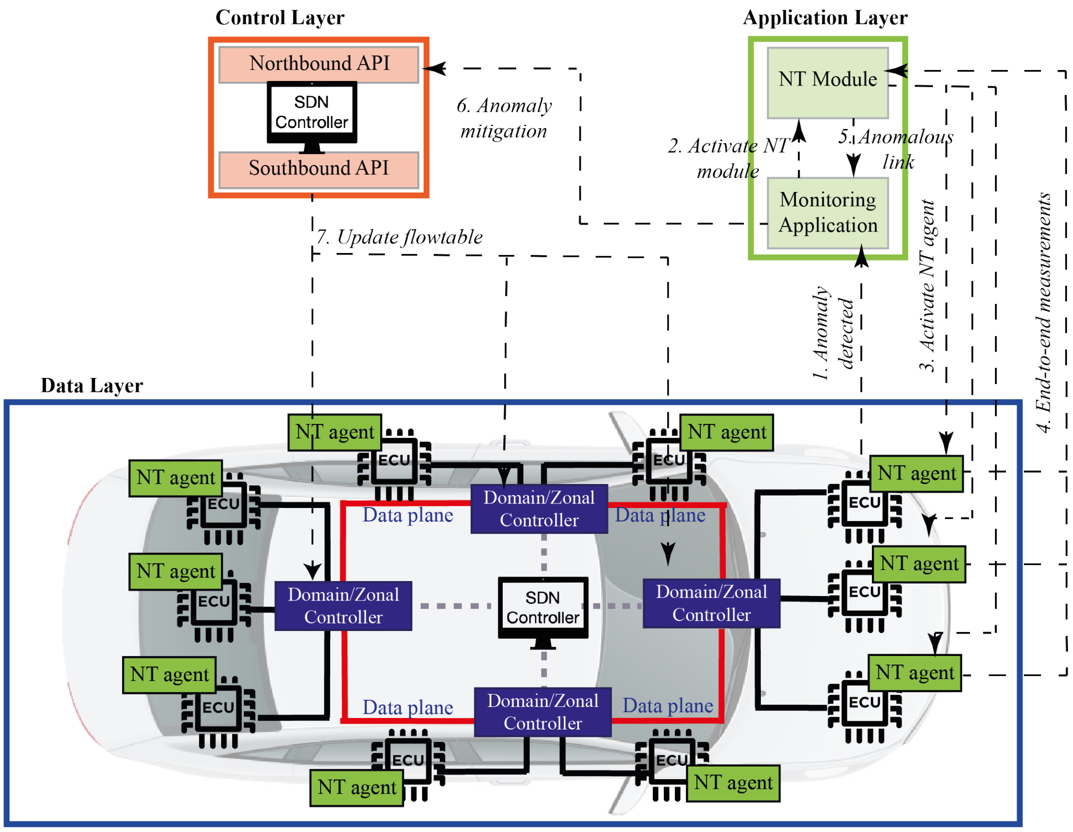


Figure 6.10: Proposed monitoring framework with SDN and network tomography. **NT**: Network Tomography.

6.9.2.3 Anomaly Mitigation

Once the anomalous link has been located using the anomaly localisation step, the mitigation step takes place as follows:

- The monitoring application instructs the SDN controller to mitigate the anomaly effect by specifying the anomalous link (*step 6* in Figure 6.10).
- In turn, the SDN controller performs the anomaly mitigation by instructing the underlying network nodes to update their flow tables to use the redundant path(s) $p_j \in \mathcal{P}(G)$ such that $e_i \notin p_j$ where e_i is the anomalous link, (*step 7* in Figure 6.10).

6.10 Performance Evaluation

In this section, we evaluate our transformation algorithms for both identifiability and redundancy. We first evaluate Procedure 3 that transforms a given unidentifiable topology into an identifiable one using PRLs. Then compare it with Procedure 4, where FRLs (if existed) can be used. Next, we evaluate Algorithm 3 to transform non-redundant topologies into redundant ones and compare the results of both methods, RIT and IRT, discussed in Chapter 6.8.3.1 and Chapter 6.8.3.2, respectively. Additionally, we perform an experiment that integrates network tomography and SDN, where SDN is used to set up the monitoring paths. The performance of such integration is evaluated on inferring delay and loss metrics.

6.10.1 Transformation Algorithms

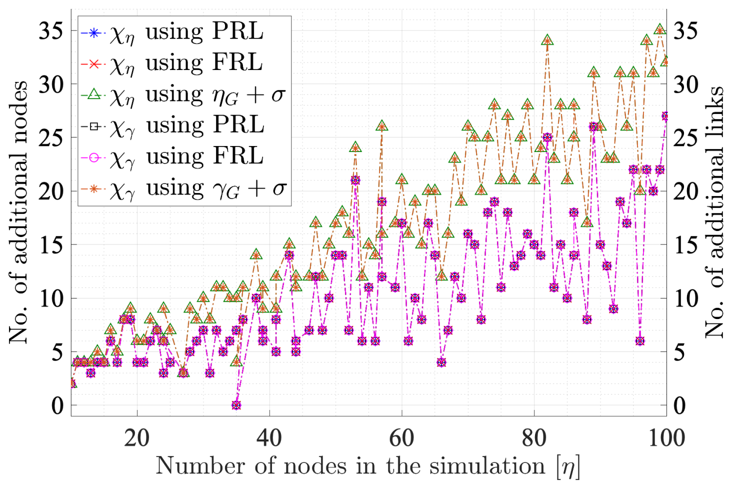
We evaluated the proposed transformation algorithms by simulating random topologies with different numbers of nodes in the range $[10, 100]$. We used MATLAB R2022b to simulate these topologies. Because topologies are generated randomly, they are neither guaranteed to be connected nor acyclic. Therefore, we checked each topology for these conditions to be met. We removed cycles if existed (see Appendix C.3), and connected the topology (see Appendix C.4) if it had more than one component (i.e., disconnected).

Let χ_η and χ_γ be the number of added nodes and links, respectively, in the transformed topology, then in the following, we discuss the results for identifiability and redundancy transformation algorithms. In addition, the results comparing the use of RIT and IRT for topologies that are originally unidentifiable and non-redundant are further discussed.

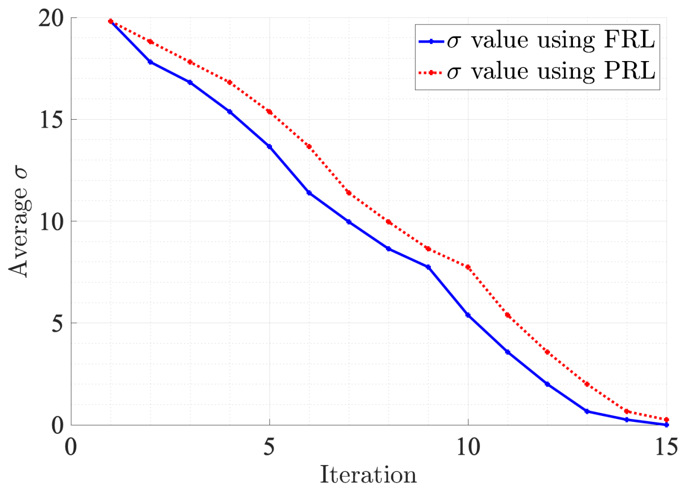
6.10.1.1 Transformation to Identifiable Topology ($G \longrightarrow G_i$)

Figure 6.11 shows identifiability results. The number of additional nodes and links are shown in Figure 6.11(a). We evaluated the transformation using PRLs, FRLs, and the basic method of adding more σ links and nodes as illustrated in Procedure 2.

As shown, the additional number of nodes and the additional number of links are the same in each scenario, this is because adding any node requires adding a link to connect it to the network. In addition, using either PRL or FRL results in the same number of added links and nodes. On the other hand, using Procedure 2 ($\eta_G + \sigma$ and $\gamma_G + \sigma$), results in more nodes and links than FRL and PRL.



(a) Number of added nodes and links after transformation



(b) Values of σ when using FRL and PRL

Figure 6.11: Identifiability results ($G \longrightarrow G_i$).

Although, the number of added nodes and links for PRL and FRL are exactly the same, we highlight the benefit of using FRL over PRL in terms of speed. This can be seen in Figure 6.11(b) where we show the average (over the number of topologies that had FRL links) value of σ during each iteration of the transformation. As seen, the σ value in the case of PRL is larger than that of FRL. This is because FRL reduces σ by 2 in a single iteration, while PRL reduces it by 1 in each iteration. Therefore, FRL can transform the topology much faster than PRL.

6.10.1.2 Transformation to Redundant Topology ($G \longrightarrow G_r$)

Figure 6.12 shows the redundancy results of transforming topologies that are non-redundant to redundant topologies using Algorithm 3. The results show that after the transformation, the number of added nodes η_{G_r}, stays the same as in the original topology. However, the number of links in the transformed topologies, γ_{G_r}, increases. This is expected as the algorithm tries to satisfy the redundancy conditions in Theorem 6.8.1. Specifically, the second condition in Theorem 6.8.1, implies that it is necessary to add more links for internal nodes with $\left|\mathcal{I}(v_i)\right| < 2$, hence the increased number of links.

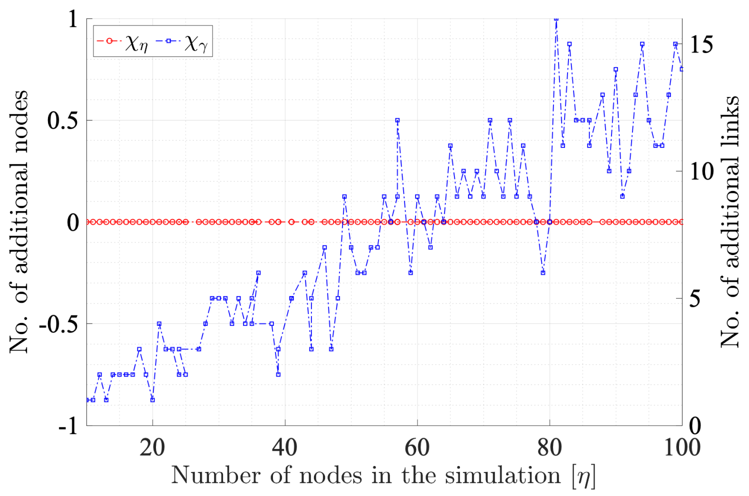
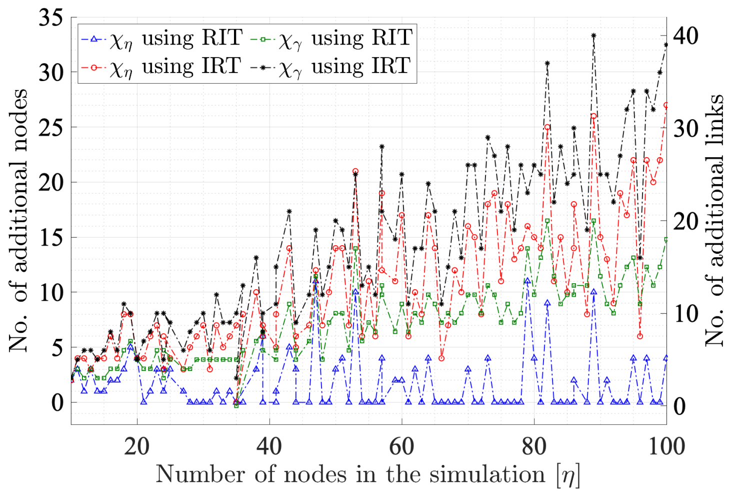


Figure 6.12: Redundancy results ($G \longrightarrow G_r$).

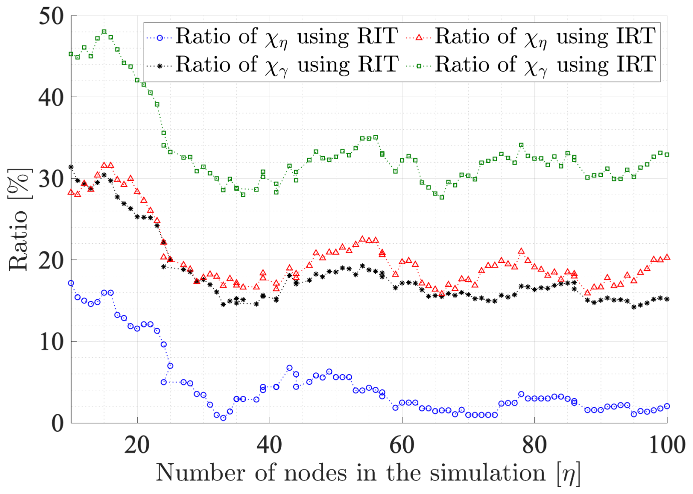
6.10.1.3 Transformation to Identifiable and Redundant Topology ($G \longrightarrow G_{ir}$)

Figure 6.13 shows the results of transforming unidentifiable and non-redundant topologies into identifiable and redundant ones using the methods discussed in Chapter 6.8.3: RIT and IRT. As observed in Figure 6.13(a), transforming unidentifiable and non-redundant topology

G into identifiable and redundant topology G_{ir} using RIT method results in fewer number of nodes and links as compared with IRT method. This is consistent with the theoretical result mentioned in Proposition 6.8.1.



(a) Number of added nodes and links after transformation



(b) Ratio of added number of nodes and links in the transformed topology

Figure 6.13: Results of transforming topology into identifiable and redundant ($G \longrightarrow G_{ir}$).

In addition, to quantify the added weight of the newly transformed topology, we show in Figure 6.13(b) the ratio of added nodes and links in the transformed topology G_{ir} as a percentage of the original topology before transformation i.e., G. This result is consistent with the theoretical analysis as stated in Proposition 6.8.1, where transforming unidentifiable *and* non-redundant topology into a topology that is both identifiable and redundant using RIT incur adding fewer number of nodes and links than using IRT. This is because RIT first (during the transformation into redundant topology) adds more links between internal nodes, then when it comes to the next step (transforming into identifiable topology), some (or all) of the internal nodes' degrees have already been increased including those with $d(v_i) < 3$, hence no additional nodes/links would be required in the second step, or at least only fewer than σ would be required.

Moreover, it is important to choose a transformation approach that results in minimal weight, and as expected, the maximum weight added is when IRT is used where it reaches 31.56% of added nodes and 48.03% of added links, whereas using RIT results in only 17.17% and 31.40% nodes and links, respectively.

6.10.2 SDN and Network Tomography Integration

In this part, we evaluate network tomography while integrated with SDN. We used Mininet 2.3.0[5] to simulate the network topology shown in Figure 6.14. Note that this topology satisfies all three properties defined for the new topology proposed in this chapter, i.e., identifiable, redundant, and SDN-enabled. For the SDN controller, we opted for Ryu controller[6]. Table 6.2 illustrates the network parameters used for the simulated in-vehicle network.

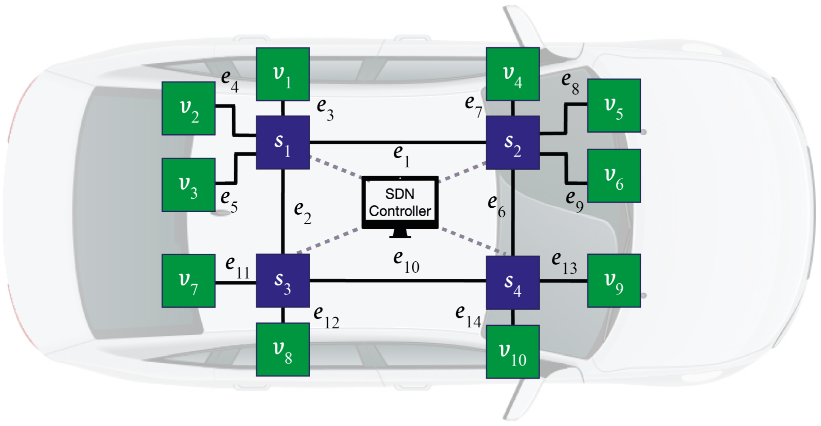


Figure 6.14: Simulated identifiable, redundant and SDN-enabled in-vehicle network topology.

[5]http://mininet.org
[6]https://ryu-sdn.org

Table 6.2: Network parameters for the simulated in-vehicle network shown in Figure 6.14.

Parameter	Value
Number of nodes (η_G)	14
Number of edge nodes ($\left\|\mathcal{E}(G)\right\|$)	10
Number of monitors ($\left\|\mathcal{E}_m(G)\right\|$)	10
Number of links (γ_G)	14
Number of all paths ($\left\|\mathcal{P}(G)\right\|$)	45
Number of measured paths (κ_G)	14

In addition, Table 6.3 shows the measurement paths for the simulated topology. These paths are independent and form a full-rank measurement matrix $\boldsymbol{A}$, see Appendix C.1 for code snippet used to construct such independent paths. We developed a monitoring application that proactively installs the flows for these paths. It also measures the end-to-end delay and loss rate between edge nodes in $\mathcal{E}(G)$ across the measurement paths $\mathcal{P}_m(G)$ defined in Table 6.3. A total of 100 probes were sent across these measurement paths. Note that the developed codes are available in our GitHub repository[7].

Table 6.3: Measurement paths used for the simulated topology shown in Figure 6.14.

Source	Destination	Path $p_i \in \mathcal{P}(G)_m$
v_1	v_4	$p_1 = \{e_2, e_3, e_6, e_7, e_{10}\}$
v_1	v_6	$p_2 = \{e_1, e_3, e_9\}$
v_2	v_7	$p_3 = \{e_1, e_4, e_6, e_{10}, e_{11}\}$
v_2	v_8	$p_4 = \{e_2, e_4, e_{12}\}$
v_2	v_9	$p_5 = \{e_1, e_4, e_6, e_{13}\}$
v_2	v_{10}	$p_6 = \{e_2, e_4, e_{10}, e_{14}\}$
v_3	v_4	$p_7 = \{e_1, e_5, e_7\}$
v_3	v_6	$p_8 = \{e_2, e_5, e_6, e_9, e_{10}\}$
v_3	v_7	$p_9 = \{e_2, e_5, e_{11}\}$

[7] https://gitfront.io/r/AmaniIbraheem/YPVrPAEk8KHm/New-IVN-Topology/

Table 6.3: Measurement paths used for the simulated topology shown in Figure 6.14.

Source	Destination	Path $p_i \in \mathcal{P}(G)_m$
v_4	v_6	$p_{10} = \{e_7, e_9\}$
v_5	v_9	$p_{11} = \{e_1, e_2, e_8, e_{10}, e_{13}\}$
v_5	v_{10}	$p_{12} = \{e_6, e_8, e_{14}\}$
v_8	v_9	$p_{13} = \{e_{10}, e_{12}, e_{13}\}$
v_8	v_{10}	$p_{14} = \{e_1, e_2, e_6, e_{12}, e_{14}\}$

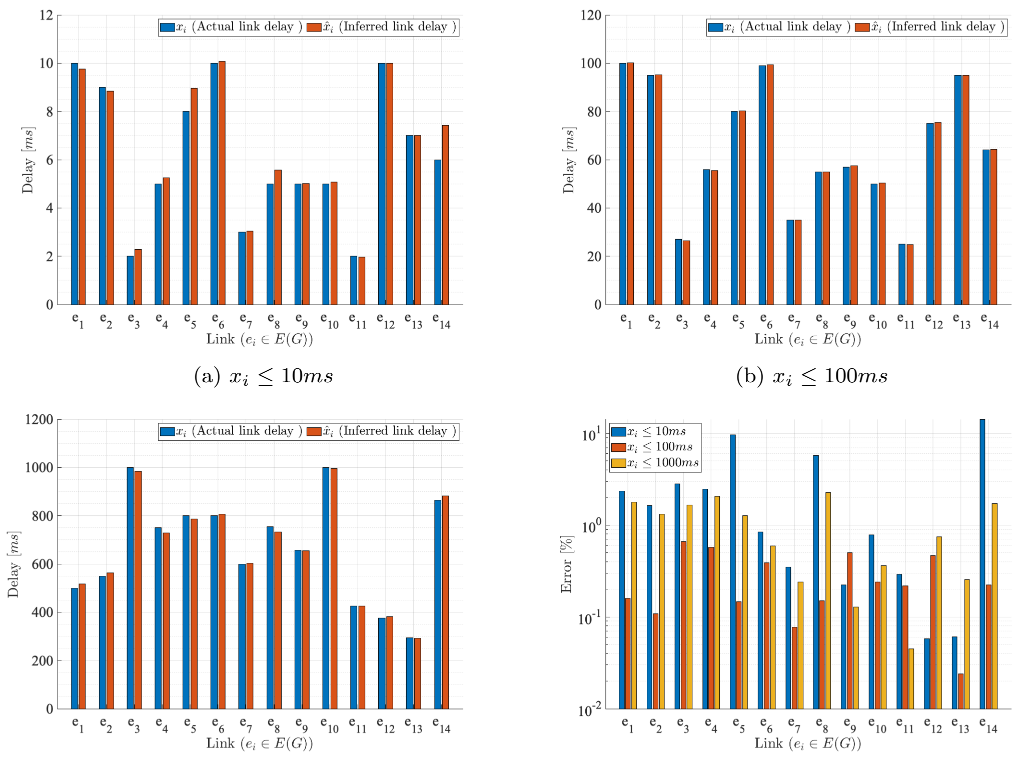
For delay tomography, we simulated three scenarios with different worst-case link-level delays (see Appendix D.1). Similarly, for loss tomography, we simulated three scenarios with different loss rates (see Appendix D.2).

In the following, we report the results of the developed SDN-based network tomography monitoring application to infer delay and loss metrics of links in the simulated network.

6.10.2.1 Delay Tomography

Figure 6.15 shows the delay tomography results for when the worst-case link-level delay is $10ms$ (Figure 6.15(a)), $100ms$ (Figure 6.15(b)), and $1000ms$ (Figure 6.15(c)). As shown, the inferred link-level delay $\hat{x}_i$ for all scenarios is close to the actual link-level delay x_i. The maximum error is for link e_{14} when the worst-case delay is $10ms$ for which the error is $\approx 14.27\%$, as shown in Figure 6.15(d). For link e_{14}, the inferred delay is $\hat{x}_{14} = 7.4ms$ while the actual delay is $x_{14} = 6ms$, hence the error value is relatively small, i.e., only $1.4ms$. On the other hand, the maximum error for when the worst-case delay is either $100ms$ or $1000ms$ does not exceed 3%; the maximum error for when the worst-case delay is $100ms$ is $\approx 0.66\%$ for link e_3, while when it is $1000ms$, the error is $\approx 2.26\%$ for link e_8.

These are promising results that show the potential of the proposed approach in inferring the delay metrics of the overall network by only monitoring the end-to-end delay performance.



(a) $x_i \leq 10ms$

(b) $x_i \leq 100ms$

(c) $x_i \leq 1000ms$

(d) Error percentage

Figure 6.15: **Delay** tomography results for the proposed in-vehicle network topology when the worst-case delay is: (a) $10ms$, (b) $100ms$, and (c) $1000ms$.

.

6.10.2.2 Loss Tomography

The end-to-end loss of each path $p_i \in \mathcal{P}_m(G)$ is given by

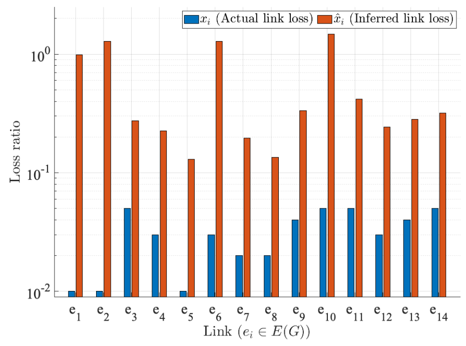
$$y_i = \prod_{e_j \in E(p_i)} x_{e_j}. \tag{6.13}$$

Thus, because the end-to-end loss is not additive, we take the natural log of both sides of (6.13) as

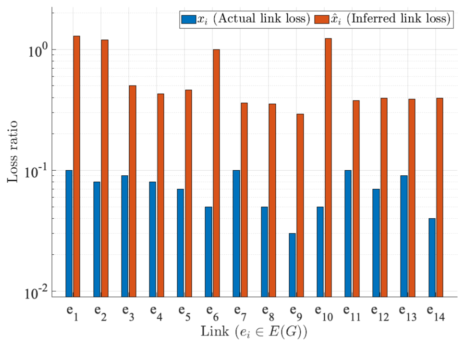
$$\ln(y_i) = \sum_{e_j \in E(p_i)} \ln(x_{e_i}). \tag{6.14}$$

Now (6.14) is in an additive form and (3.1) can be used to infer the link-level loss rate.

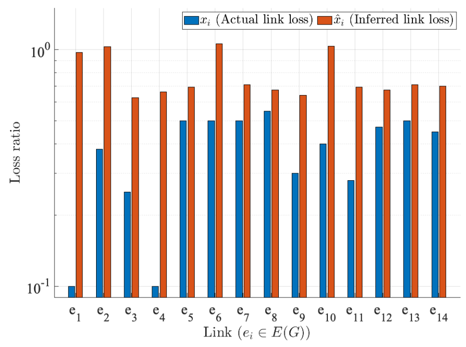
Loss tomography results are shown in Figure 6.16. We report the results for when the worst-case link-level loss rate is $\leq 5\%$ (Figure 6.16(a)), $\leq 10\%$ (Figure 6.16(b)), and $\leq 55\%$ (Figure 6.16(c)). As seen, there is an apparent contrast between actual, x_i, and inferred loss rate, $\hat{x}_i$, especially when the worst-case loss is $\leq 5\%$ as shown in Figure 6.16(a). In addition,



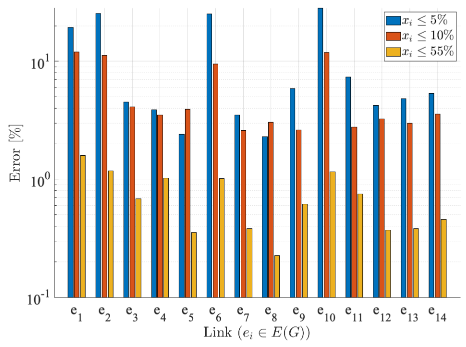
(a) $x_i \leq 5\%$



(b) $x_i \leq 10\%$



(c) $x_i \leq 55\%$



(d) Error percentage

Figure 6.16: **Loss** tomography results for the proposed SDN-enabled in-vehicle network topology when the worst-case loss rate is: (a) 5%, (b) 10%, and (c) 55%.

Figure 6.16(d) shows the percentage of the absolute error value of inferred link loss over the loss value in the worst-case scenario. It can be observed that as the loss rate increases, the inference error decreases as shown in Figure 6.16(d). Moreover, the maximum error is $\approx 28.36\%$ for e_{10} in the scenario where the maximum loss is 5%. This can be attributed to the measurement noise that is caused by the stochastic nature of end-to-end loss rate when assigning different loss rates to the links with high variance. Moreover, the maximum errors when the worst-case loss rates are $\leq 10\%$ and $\leq 55\%$ are around 11.96% and 1.59%, respectively for link e_1. The error values in these two cases are considered small, and arguably, in a real-world scenario, not all links are lossy. In addition, failures caused by attacks, e.g., DoS (Denial of Service) attacks, usually increase the loss rates per link, and from these results, it is clear that network tomography can accurately locate the anomalous link(s) affected by such attack.

The above results indicate that the integration of SDN with a network tomography-based monitoring approach is effective in inferring the internal network performance. For instance, in case of an anomaly and by monitoring the delay or loss metrics, one can locate such anomaly using the proposed approach.

6.11 Summary

In this chapter, we have proposed a new in-vehicle network topology that satisfies three properties: identifiable, redundant and SDN-enabled. The topology is designed towards efficient monitoring of the in-vehicle network using network tomography and SDN. The first property of the topology ensures that network tomography can infer the performance of all the link-level metrics, the second property helps in achieving a fail-operational behaviour, while the third property facilitates the monitoring process in terms of deploying the monitoring applications at the central entity that has an overall view and control of the underlying network topology. Furthermore, the proposed topology can act as a driver for the next-generation centralised E/E architectures.

Topological conditions to satisfy identifiability and redundancy properties have been extensively studied. Additionally, based on the derived theoretical results, we have proposed transformation algorithms to transform an existing topology that is unidentifiable or non-redundant into an identifiable and redundant topology while keeping the number of links to a minimum to avoid adding unnecessary weight to the vehicle. These algorithms allow the rewiring of existing in-vehicle network topologies. In addition, they support the gradual transformation of the existing in-vehicle network topologies, where designing new topologies from scratch can be extremely costly. Evaluation of the proposed transformation algorithms showed that the results are consistent with the theoretical analysis.

Moreover, leveraging the SDN paradigm, a complete monitoring framework has been proposed. Such a framework supports the three main tasks that should be provided in any robust monitoring system. These tasks are anomaly detection, localisation and mitigation. The integration of SDN with network tomography has further been examined and found that such integration is suitable to infer the delay and loss metrics of the network.

7 Conclusion

"I have good news and bad news: the bad news is you are fired, the good news is you get to spend more time with us."

Mohannad Alalmaei

This chapter concludes our thesis, and it is divided into two parts. The first part summarises all the research work and results presented in this thesis, whereas the second part focuses on future research directions and open questions that can be further investigated in future works.

7.1 Research Summary

Due to the nature of the environment, today's vehicles are involved with, in which nowadays they are exposed to the outside world, it is more crucial than ever that the network inside the vehicle is robust against any malicious behaviours that may put users' lives in danger. Therefore, to ensure proper functionality of the vehicle, which is then reflected on users' safety, one of the critical tasks is monitoring the in-vehicle network.

Only recently, researchers have started looking into monitoring solutions for in-vehicle networks. The main reason for the lack of monitoring approaches for in-vehicle networks is that, in the past, the vehicle was only connected internally, with no outside communications, and hackers would need to physically access the vehicle to launch their attacks. Today, this has dramatically changed, with adversaries being able to remotely launch different types of attacks without the need to physically access the vehicle. And for the same reason, traditional automotive networks, that many of which are still used in today's vehicles, do not support any means of securing the vehicle, or providing fail-operational behaviour.

Although many research efforts are focusing on this issue, the existing proposals focus on traditional networking architectures where the proposed monitoring approach can only handle a single subsystem, e.g., one connected by CAN. In particular, current solutions neglect the fact that the in-vehicle network is a large system with different subsystems that need to be monitored as a whole. Additionally, most proposals entirely use machine and deep learning-based solutions, ignoring the fact that components in in-vehicle networks are limited in both computation and storage capacities. Another limitation in the monitoring approaches that exist in the current literature is that they entail some sort of accessing the internal network components in order to modify them to perform the monitoring process, disregarding the fact that it is not always feasible to access the internal elements of the in-vehicle network. In addition, most existing approaches are only concerned with detecting anomalies, resulting from attacks on the vehicle, with no consideration of countermeasures or mitigation solutions.

This thesis presented a novel and complete monitoring approach for in-vehicle networks, including next-generation networks, where there is no need to access any internal element of the network. This monitoring feature leveraged a network tomography-based solution that has just been studied, in this thesis, for the first time as a vehicular monitoring application. The proposed approach is lightweight and does not incur heavy monitoring overhead. In addition, it supports three main tasks of robust monitoring systems, in which it can detect, locate and mitigate anomalies in in-vehicle networks.

Network tomography has been considered for other applications, such as computer networks and the Internet. By studying network tomography applications for in-vehicle networks, as reported in Chapter 3, we found that not all in-vehicle networking architectures, including the new ones that are based on modern E/E architectures, are identifiable, hence network tomography cannot infer the performance of *all* networking elements. The main issue with this is that the measurement matrix used in network tomography cannot form a full-rank matrix.

To tackle the rank-deficiency problem of the measurement matrix, we have proposed to use partial network tomography and leveraged the advances in deep learning, hence, achieving a full-rank measurement matrix where network tomography can be used to infer the overall network performance. Presented in Chapter 4, the results confirmed the suitability of the proposed tomography algorithm as well as the deep learning-based solution. Additionally, using the proposed approach to infer the link-level performance of all links in in-vehicle networks, we found that network tomography, specifically algebraic tomography, was able to accurately determine the link-level metric.

Furthermore, as shown in Chapter 5, the presented monitoring approach could detect and locate anomalies in in-vehicle networks. Three different tomographic approaches have been evaluated to detect and locate DoS attacks: DNT, BNT, and DNN-based network tomography. Evaluation results have shown that BNT achieved promising results with no false positive or false negative.

Inspired by the promising results achieved from applying algebraic BNT to detect and locate anomalies, in Chapter 6, we have proposed a novel in-vehicle network topology. The main reason is to allow for algebraic network tomography approaches to be applied in in-vehicle networks. Such application requires, however, a full-rank measurement matrix. This feature thus can be satisfied by having a fully-identifiable topology. Therefore, one of the properties of the proposed topology is that it is identifiable. In addition, it is redundant and SDN-enabled topology. The redundancy and SDN properties can greatly facilitate the monitoring process. Specifically, the redundant topology can be used in case there is an anomaly in the network, hence, ensuring fail-operational behaviour. On the other hand, SDN can make this happen dynamically and intelligently without the need for human intervention. Moreover, to support the gradual transition into new in-vehicle architectures, instead of creating identifiable and redundant topologies from scratch, this thesis proposed different algorithms to transform any existing topology into identifiable and redundant ones. Finally, a complete monitoring framework based on the proposed approach has been devised. The proposed framework supports all three monitoring tasks: detection, localisation, and mitigation. With this framework, the in-vehicle network becomes self-reactive and more intelligent when faced with malicious behaviours.

7.2 Future Work

This thesis has set the stepping stone for a new monitoring approach for in-vehicle networks that is based on network tomography. This approach can further be built upon to improve the overall monitoring system. In addition, the advances of SDNs have been exploited in this thesis to facilitate the monitoring process. In the following, we discuss some of the intriguing research directions for network tomography and SDN applications in vehicular communications, that can further be investigated.

7.2.1 Network Tomography for Vehicular Communications

The following lists some of the future directions for applying network tomography to vehicular communications:

- This thesis mostly focussed on using delay metrics when applying network tomography to detect anomalies on the network. Other metrics can be used, such as loss/success rate, bandwidth consumption, and throughput.

- As has been found that using CAN traffic in monitoring the network can result in having asymmetric behaviour (an attribute that hinders the accurate performance of network tomography), especially when relying on delay metric, such asymmetric behaviour can be avoided if CAN messages do not compete to access the CAN bus. This is achieved by setting proper parameters for transmission times of CAN messages transmitted by different ECUs. Benefiting from the advances in artificial intelligence, one can design a reinforcement learning model that allows the agent to set these parameters with the aim of preventing collisions in the CAN bus.

- The partial network tomography algorithm presented in Algorithm 1 can be improved by selectively choosing the optimal partial network such that the result leads to the maximum possible identifiability.

- Using network tomography to detect anomalies in the in-vehicle network shows that all approaches (BNT, DNT, DNN for BNT, and DNN for DNT) could accurately detect the anomaly, however, the detection time was not analysed. This aspect can further be investigated to help understand and validate the appropriate network tomography monitoring approach that can detect anomalies in real-time (or near real-time).

- The proposed monitoring approach in this thesis can detect and locate anomalies that lead to deviation from the normal behaviour of the network. This thesis proved that the proposed approach could detect and locate anomalies resulting from attacks, such as DoS attack on CAN. The proposal can further be examined to detect other types of attacks, such as fuzzy and replay attacks. Different types of attacks might need measuring different metrics, including delay and loss rate. And some other attacks can be accurately detected and located by measuring more than one metric.

- The proposed network tomography approach presented in this thesis has been investigated to detect and locate anomalies where only one link can fail at a time. It would be another interesting direction to further examine the approach to detect and locate multiple anomalous links that occur simultaneously.

- This thesis focussed on anomaly detection and localisation with network tomography. In addition to this, network tomography can be used for other different applications such as load-balancing and network optimisation applications.
- On a larger scale, beyond the in-vehicle network, an interesting research direction is to apply network tomography as a monitoring approach for VANETs. Vehicles in VANETs are dynamic where they constantly join and leave the network, hence, it is important to inspect the performance of network tomography for such a dynamic environment. One crucial question is how to discover such dynamic topology to be used for the measurement matrix. Moreover, finding a set of independent measurement paths is another interesting direction which is considered challenging to apply in a highly dynamic environment such as VANET.

7.2.2 Software-Defined Networks (SDNs) for Vehicular Communications

Leveraging SDN for the in-vehicle network is the starting point to fully incorporate the concept of Software-Defined Vehicle (SDV). Being an SDN-enabled vehicle allows the vehicle to be part of this system. In the following, we list a few of the interesting research directions considering SDN:

- The algorithm proposed in this thesis to transform any existing topology (with a constraint that only edge nodes are accessible) into an identifiable topology often results in adding more weight to the vehicle network. Additional effort for optimising the topology to reduce this weight is another desirable direction that is worth investigating especially for in-vehicle networks. For this, the SDN paradigm can be highly beneficial in that it can help to replace some of the hardware-based functionalities with software ones and consolidate multiple similar ECUs into a single powerful unit.
- One of the essential tasks in implementing the proposed monitoring framework in this thesis is that, if there is any anomaly that has been detected and located, the SDN controller is responsible for mitigating its effect by instructing the network to use redundant path(s) that do not traverse the anomalous link. This process requires that the SDN controller quickly and intelligently assign the new paths, and drop the anomalous traffic. This task can be achieved by applying machine and deep learning techniques.

- With the SDN-enabled in-vehicle network, the inference result of network tomography can further be enhanced by using *FlowStats* messages provided by OpenFlow protocol. These are per-flow statics that can be leveraged to minimise the uncertainty of the inference results provided by network tomography, hence, reducing the link-level inference error.
- Equipping the vehicle with SDN functionality is of great benefit, not only internally for the in-vehicle system, but externally as well. For instance, the SDN functionality can be applied on a larger scale of a fleet of vehicles, e.g., in IoVs. With network tomography and SDN, attacks on vehicular communications can be detected and located so that the compromised vehicle can be excluded. Network tomography can help detect attacks and locate anomalous vehicles, while SDN can exclude such anomalous vehicles from the network so that they will not compromise other vehicles in the fleet.
- As one of the aims of incorporating the in-vehicle network with SDN capabilities is to improve the overall vehicle weight by replacing the hardware-based components with software-based alternatives, it is important to note that not all vehicle components can be replaced with software-based components; the physical hardware-based components would still need to co-exist in conjunction with software components. Therefore, it is mandatory to properly define the delimiters for which hardware-based components versus software-based ones should be adopted.
- Although the SDN capabilities cannot be overlooked, the centralisation aspect of the SDN controller can be seen as a limitation in that it presents a single point of failure. For this reason, it is essential to have appropriate countermeasures in place for the SDN controller. Such measures may include:
  - Enhancing the SDN controller with the highest level of security protection.
  - The SDN controller may experience performance degradation due to system failure. Thus, it is important to have a redundant controller entity that can take over if the original controller has failed.
  - The SDN controller tasks can be distributed among several controllers as it is possible to have more than one SDN controller. For instance, in an in-vehicle network with multiple domains, the domain controller can also act as a local SDN controller.

For a larger scale application such as in IoVs, a vehicle can act as an SDN controller for a group of vehicles that are in close proximity. If such a vehicle has been compromised, one can investigate the solutions to overcome the effect of this failure and assign an alternative vehicle to act as the SDN controller.

A Appendix: Traffic Characteristics for Applying Network Tomography in In-Vehicle Networks

A.1 Traffic Characteristics in Single CAN Architecture

The simulated topology for a single CAN network is shown below.

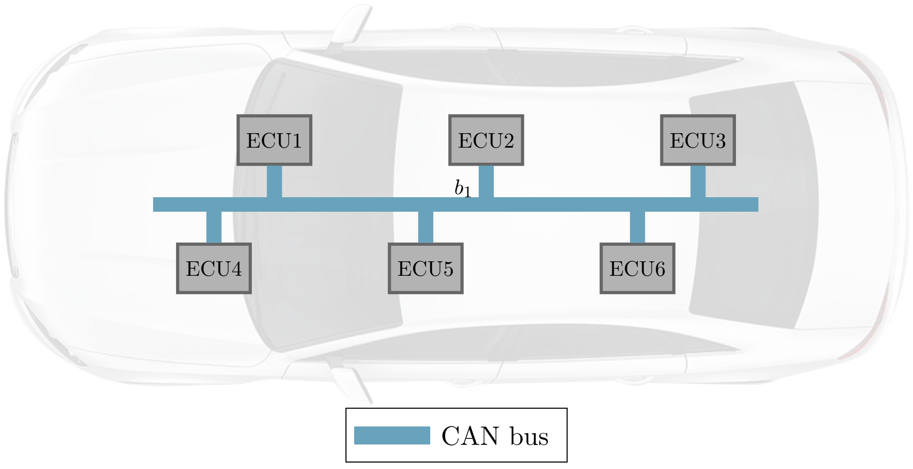


Figure A.1: Simulated single CAN network.

Traffic characteristics for this scenario are shown in the following table. All traffic has a payload of 64 bits = 8 bytes. Each CAN message is of length 111 bits.

Table A.1: Traffic characteristics for single CAN.

CAN ID	Frequency (s)	CAN ID	Frequency (s)
1	0.009887	15	0.008334
2	0.009776	16	0.018222
3	0.009665	17	0.008104
4	0.009554	18	0.007993
5	0.009443	19	0.007898
6	0.099249	20	0.097778
7	0.099138	21	0.007659
8	0.009103	22	0.007557
9	0.008992	23	0.017445
10	0.008881	24	0.017335
11	0.008775	25	0.996339
12	0.008667	26	0.996227
13	0.008548	27	0.047113
14	0.018444	-	-

A.2 Traffic Characteristics in Central-Gateway Architecture

The simulated topology for this scenario is shown in the following figure. Each CAN network consists of 30 ECUs.

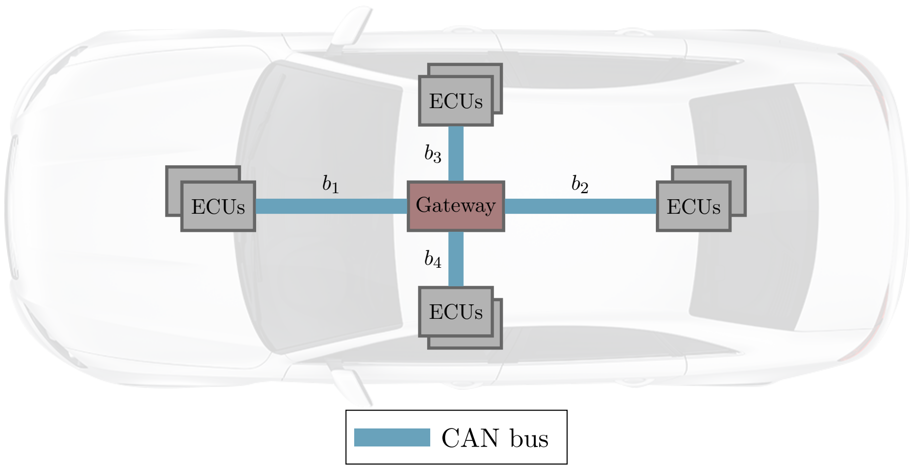


Figure A.2: Simulated central-gateway network.

Traffic within a single CAN is the same as in the above scenario. Cross-traffic between different CANs is shown in the following table.

Table A.2: Cross-traffic between different CANs in central-gateway network.

CAN ID	Frequency (s)
28	0.047113
29	0.047113

A.3 Traffic Characteristics in Ethernet-based Architecture

The simulated scenario for this case is shown in the below figure. As in other cases, the number of ECUs in each CAN is 30.

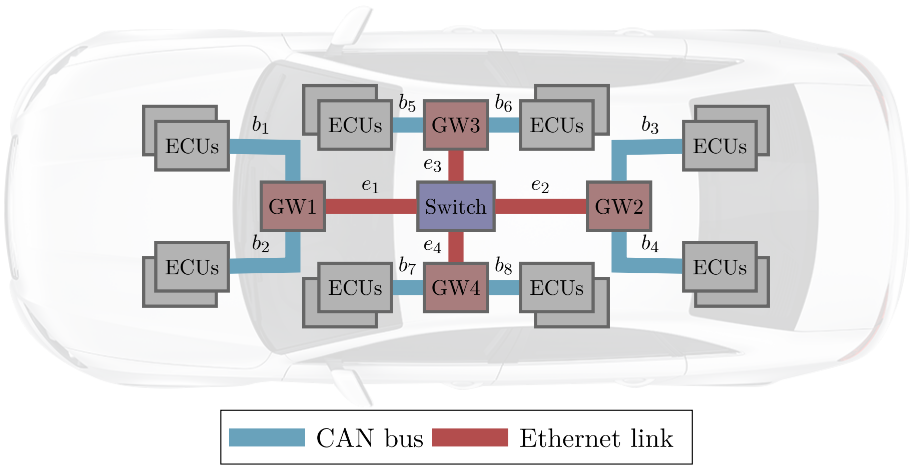


Figure A.3: Ethernet-based in-vehicle network

For this scenario, the traffic characteristics are shown in the following.

A.3.1 Cross-traffic between different CANs

Table A.3: Cross-traffic between different CANs in Ethernet-based network.

CAN ID	Frequency (s)
28	0.047113
29	0.047113

A.3.2 Cross-traffic between different gateways

Table A.4: Cross-traffic between different gateways in Ethernet-based network.

CAN ID	Frequency (s)
28	0.047113
29	0.047113
30	0.047113
31	0.047113
32	0.01
33	0.01
34	0.01

Table A.4: Cross-traffic between different gateways in Ethernet-based network.

CAN ID	Frequency (s)
35	0.01

B Appendix: Evaluation of DNN-based Partial Tomography

In the following, more evaluation results are shown when measuring different ratios of the network simulated in Chapter 4.

B.1 When 50% of the network is measured

Table B.1: DNN results when 50% of the network is measured.

Measured paths ($\mathcal{P}_m$)	NNDE (MAPE (%))	NNDT (MAPE (%))
$\mathcal{P}_m = \{p_3, {}^2p_7, p_8, p_9, p_{10}\}$	1.711	1.575
$\mathcal{P}_m = \{p_4, {}^2p_7, p_8, p_9, p_{10}\}$	2.481	1.515
$\mathcal{P}_m = \{p_6, {}^2p_7, p_8, p_9, p_{10}\}$	2.540	2.468
$\mathcal{P}_m = \{p_7, {}^2p_7, p_8, p_9, p_{10}\}$	2.955	2.771
$\mathcal{P}_m = \{p_5, {}^2p_7, p_8, p_9, p_{10}\}$	1.176	1.162

B.2 When 60% of the network is measured

Table B.2: DNN results when 60% of the network is measured.

Measured paths ($\mathcal{P}_m$)	NNDE (MAPE (%))	NNDT (MAPE (%))
$\mathcal{P}_m = \{p_3, p_4, {}^2p_7, p_8, p_9, p_{10}\}$	1.119	1.094
$\mathcal{P}_m = \{p_3, p_6, {}^2p_7, p_8, p_9, p_{10}\}$	1.170	1.255
$\mathcal{P}_m = \{p_3, {}^1p_7, {}^2p_7, p_8, p_9, p_{10}\}$	1.786	1.726
$\mathcal{P}_m = \{p_3, p_5, {}^2p_7, p_8, p_9, p_{10}\}$	0.896	0.900

Table B.2: DNN results when 60% of the network is measured.

Measured paths ($\mathcal{P}_m$)	NNDE (MAPE (%))	NNDT (MAPE (%))
$\mathcal{P}_m = \{p_4, p_6, {}^2p_7, p_8, p_9, p_{10}\}$	1.249	1.210
$\mathcal{P}_m = \{p_4, {}^1p_7, {}^2p_7, p_8, p_9, p_{10}\}$	1.754	1.733
$\mathcal{P}_m = \{p_4, p_5, {}^2p_7, p_8, p_9, p_{10}\}$	1.110	1.157
$\mathcal{P}_m = \{p_6, {}^1p_7, {}^2p_7, p_8, p_9, p_{10}\}$	2.975	2.804
$\mathcal{P}_m = \{p_5, p_6, {}^2p_7, p_8, p_9, p_{10}\}$	0.780	0.819
$\mathcal{P}_m = \{p_5, {}^1p_7, {}^2p_7, p_8, p_9, p_{10}\}$	1.343	1.463

B.3 When 70% of the network is measured

Table B.3: DNN results when 70% of the network is measured.

Measured paths ($\mathcal{P}_m$)	NNDE (MAPE (%))	NNDT (MAPE (%))
$\mathcal{P}_m = \{p_3, p_4, p_6, {}^1p_7, p_8, p_9, p_{10}\}$	0.940	1.058
$\mathcal{P}_m = \{p_3, p_4, {}^1p_7, {}^2p_7p_8, p_9, p_{10}\}$	1.263	1.262
$\mathcal{P}_m = \{p_3, p_4, p_5, {}^1p_7, p_8, p_9, p_{10}\}$	0.758	0.707
$\mathcal{P}_m = \{p_3, p_6, {}^1p_7, {}^2p_7, p_8, p_9, p_{10}\}$	1.428	1.635
$\mathcal{P}_m = \{p_3, p_5, p_6, {}^2p_7, p_8, p_9, p_{10}\}$	0.665	0.606
$\mathcal{P}_m = \{p_3, p_5, {}^1p_7, {}^2p_7, p_8, p_9, p_{10}\}$	1.024	1.049
$\mathcal{P}_m = \{p_4, p_6, {}^1p_7, {}^2p_7, p_8, p_9, p_{10}\}$	1.405	1.429
$\mathcal{P}_m = \{p_4, p_5, p_6, {}^2p_7, p_8, p_9, p_{10}\}$	0.672	0.655
$\mathcal{P}_m = \{p_6, p_7, {}^1p_7, {}^2p_7, p_8, p_9, p_{10}\}$	0.970	1.029

C Appendix: Procedures and Code Snippets

C.1 Constructing independent paths in network G

```
clc;
%Draw eaxmple topology with source s and target t
G = graph(s,t, [], names);
edges = G.Edges;
vertices = G.Nodes;
s = size(vertices);
end_nodes = zeros(s(1),1);
internal_nodes = zeros(s(1),1);
vertices_t =  transpose(table2array(vertices));
for i = vertices_t(:, :)
    if degree(G, i) == 1
        end_nodes = [end_nodes; i];
    else
        internal_nodes = [internal_nodes, i];
    end
end
paths_list = {};
for vi = transpose(end_nodes(2:end, 1))
    for vj = transpose(end_nodes(2:end, 1))
        if strcmp(vi,vj)==0
            fprintf('path between %s and %s\n', vi{1}, vj{1});
            [paths,edgepaths] = allpaths(G,vi{1},vj{1}, 'MaxNumPaths',3);
            for p=size(edgepaths)
                paths_list = [paths_list; edgepaths{p}];
            end
        end
    end
end
s_paths = size(paths_list,1);
s_links = size(edges,1);
links = [];
for a= transpose(paths_list)
    disp(a{1});
    for b=a{1}
    if ismember(b, links)
    else
        links = [links; b];
    end
    end
end
links = sort(links);
measurement_matrix = [];
for path=transpose(paths_list)
    for link = links
        measurement_matrix = [measurement_matrix; ismember(link, path{1})'];
    end
end
[Xsub,idx]=licols(measurement_matrix',1e-10);
A = Xsub';
```

C.2 Finding bridges in network G

```
Procedure 7: $findBridges(G)$
   Output : $\mathcal{B}(G)$
   Initialize:
   $\mathcal{B}(G) \leftarrow \emptyset$
1  foreach $e_i \in E(G)$ do
2      $G \leftarrow G - \{e_i\}$
3      if $isConnected(G)=false$ then
4          $\mathcal{B} \leftarrow \mathcal{B} \cup \{e_i\}$
5      $G \leftarrow G + \{e_i\}$
6  return $\mathcal{B}(G)$
```

C.3 Removing cycles in generated random graph G

```
function G = removeCycles(G)
cycles = allcycles(G);
        for c = 1:length(cycles)
            for r = cycles{c}
                for m = cycles{c}
                    if(strcmp(r,m)==0)
                        G = rmedge(G, r, m);
                        flag = 1;
                        break
                    end
                end
                if flag==1
                    break
                end
            end
        end
end
```

C.4 Connecting graph G of multiple components

```
function G = connect_graph(G)
[bins, binsSizes] = conncomp(G);
idx = binsSizes(bins) == max(binsSizes);
SG = subgraph(G, idx);
anynode = SG.Nodes;
anynode = table2array(anynode);
anynode = char(anynode(1));
idx_other = binsSizes(bins) ~= max(binsSizes);
for i = 1:length(idx_other)
    if (idx_other(i)==1)
        SG_other = subgraph(G, i);
        for node = SG_other.Nodes
            node = table2array(node);
            node = char(node);
            G = addedge(G, anynode, node);
            G = connect_graph(G);
            break
        end
        break
    end
end
end
```

D Appendix: Simulation Parameters for SDN-enabled In-Vehicle Network

D.1 Delay Tomography Parameters

D.1.1 When the worst-case delay is $10ms$

Table D.1: Link parameters when the worst-case delay is $10ms$.

Link ($e_i \in E(G)$)	Bandwidth (bps)	Delay (ms)
e_1	10	10
e_2	10	9
e_3	10	2
e_4	10	5
e_5	10	8
e_6	10	10
e_7	10	3
e_8	10	5
e_9	10	5
e_{10}	10	5
e_{11}	10	2
e_{12}	10	10
e_{13}	10	7

Table D.1: Link parameters when the worst-case delay is $10ms$.

Link ($e_i \in E(G)$)	Bandwidth (bps)	Delay (ms)
e_{14}	10	6

D.1.2 When the worst-case delay is $100ms$

Table D.2: Link parameters when the worst-case delay is $100ms$.

Link ($e_i \in E(G)$)	Bandwidth (bps)	Delay (ms)
e_1	10	100
e_2	10	95
e_3	10	27
e_4	10	56
e_5	10	80
e_6	10	99
e_7	10	35
e_8	10	55
e_9	10	57
e_{10}	10	50
e_{11}	10	25
e_{12}	10	75
e_{13}	10	95
e_{14}	10	64

D.1.3 When the worst-case delay is $1000ms$

Table D.3: Link parameters when the worst-case delay is 1000ms.

Link ($e_i \in E(G)$)	Bandwidth (bps)	Delay (ms)
e_1	10	500
e_2	10	550
e_3	10	1000
e_4	10	750
e_5	10	800
e_6	10	800
e_7	10	600
e_8	10	755
e_9	10	657
e_{10}	10	1000
e_{11}	10	425
e_{12}	10	375
e_{13}	10	295
e_{14}	10	864

D.2 Loss Tomography Parameters

D.2.1 When the worst-case loss rate is 5%

Table D.4: Link parameters when the worst-case loss rate is 5%.

Link ($e_i \in E(G)$)	Bandwidth (bps)	Loss (%)
e_1	10	1
e_2	10	1
e_3	10	5
e_4	10	3
e_5	10	1
e_6	10	3

Table D.4: Link parameters when the worst-case loss rate is 5%.

Link ($e_i \in E(G)$)	Bandwidth (bps)	Loss (%)
e_7	10	2
e_8	10	2
e_9	10	4
e_{10}	10	5
e_{11}	10	5
e_{12}	10	3
e_{13}	10	4
e_{14}	10	5

D.2.2 When the worst-case loss rate is 10%

Table D.5: Link parameters when the worst-case loss rate is 10%.

Link ($e_i \in E(G)$)	Bandwidth (bps)	Loss (%)
e_1	10	10
e_2	10	8
e_3	10	9
e_4	10	8
e_5	10	7
e_6	10	5
e_7	10	10
e_8	10	5
e_9	10	3
e_{10}	10	5
e_{11}	10	10
e_{12}	10	7
e_{13}	10	9

Table D.5: Link parameters when the worst-case loss rate is 10%.

Link ($e_i \in E(G)$)	Bandwidth (bps)	Loss (%)
e_{14}	10	4

D.2.3 When the worst-case loss rate is 55%

Table D.6: Link parameters when the worst-case loss rate is 55%.

Link ($e_i \in E(G)$)	Bandwidth (bps)	Loss (%)
e_1	10	10
e_2	10	38
e_3	10	25
e_4	10	10
e_5	10	50
e_6	10	50
e_7	10	50
e_8	10	55
e_9	10	30
e_{10}	10	40
e_{11}	10	28
e_{12}	10	47
e_{13}	10	50
e_{14}	10	45

Bibliography

[1] J. Contreras-Castillo, S. Zeadally, and J. A. Guerrero-Ibañez, “Internet of vehicles: architecture, protocols, and security,” *IEEE internet of things Journal*, vol. 5, no. 5, pp. 3701–3709, 2017.

[2] Sichitiu, Mihail L and Kihl, Maria, “Inter-vehicle communication systems: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 10, no. 2, pp. 88–105, 2008.

[3] Tuohy, Shane and Glavin, Martin and Hughes, Ciarán and Jones, Edward and Trivedi, Mohan and Kilmartin, Liam, “Intra-vehicle networks: A review,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 534–545, 2014.

[4] C. Miller and C. Valasek, “A survey of remote automotive attack surfaces,” *black hat USA*, vol. 2014, p. 94, 2014.

[5] S. Checkoway, D. McCoy, B. Kantor, *et al.*, “Comprehensive experimental analyses of automotive attack surfaces.,” in *USENIX security symposium*, San Francisco, vol. 4, 2011, p. 2021.

[6] M. Ring, D. Frkat, and M. Schmiedecker, “Cybersecurity evaluation of automotive e/e architectures,” in *ACM Computer Science In Cars Symposium (CSCS 2018)*, 2018.

[7] Longari, Stefano and Penco, Matteo and Carminati, Michele and Zanero, Stefano, “Copycan: An error-handling protocol based intrusion detection system for controller area network,” in *Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy*, 2019, pp. 39–50.

[8] S. C. HPL, “Introduction to the controller area network (CAN),” *Application Report SLOA101*, pp. 1–17, 2002.

[9] W. Voss, *A comprehensible guide to controller area network.* Copperhill Media, 2008.

[10] Y. Peng, B. Shi, T. Jiang, X. Tu, D. Xu, and K. Hua, “A Survey on In-vehicle Time Sensitive Networking,” *IEEE Internet of Things Journal*, 2023.

[11] Bozdal, Mehmet and Samie, Mohammad and Jennions, Ian, “A survey on can bus protocol: Attacks, challenges, and potential solutions,” in *2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE)*, IEEE, 2018, pp. 201–205.

[12] P. Weiss, A. Weichslgartner, F. Reimann, and S. Steinhorst, “Fail-operational automotive software design using agent-based graceful degradation,” in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2020, pp. 1169–1174.

[13] Castro, Rui and Coates, Mark and Liang, Gang and Nowak, Robert and Yu, Bin, “Network tomography: Recent developments,” *Statistical science*, vol. 19, no. 3, pp. 499–517, 2004.

[14] Lawrence, Earl and Michailidis, George and Nair, Vijayan N and Xi, Bowei, “Network tomography: A review and recent developments,” *Frontiers in statistics*, pp. 345–366, 2006.

[15] M. Rumez, D. Grimm, R. Kriesten, and E. Sax, “An overview of automotive service-oriented architectures and implications for security countermeasures,” *IEEE access*, vol. 8, pp. 221 852–221 870, 2020.

[16] W. Wu, R. Li, G. Xie, *et al.*, “A survey of intrusion detection for in-vehicle networks,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 919–933, 2019.

[17] P. Pelliccione, E. Knauss, R. Heldal, *et al.*, “Automotive architecture framework: The experience of volvo cars,” *Journal of systems architecture*, vol. 77, pp. 83–100, 2017.

[18] L. Zhang and D. Ma, “A hybrid approach toward efficient and accurate intrusion detection for in-vehicle networks,” *IEEE Access*, vol. 10, pp. 10 852–10 866, 2022.

[19] L. Ma, T. He, K. K. Leung, A. Swami, and D. Towsley, “Topological constraints on identifying additive link metrics via end-to-end paths measurements,” IBM THOMAS J WATSON RESEARCH CENTER HAWTHORNE NY, Tech. Rep., 2012.

[20] L. Ma, T. He, K. K. Leung, A. Swami, and D. Towsley, “Identifiability of link metrics based on end-to-end path measurements,” in *Proceedings of the 2013 conference on Internet measurement conference*, 2013, pp. 391–404.

[21] S. Saidi, S. Steinhorst, A. Hamann, D. Ziegenbein, and M. Wolf, "Special session: Future automotive systems design: Research challenges and opportunities," in *2018 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*, IEEE, 2018.

[22] S. Sommer, A. Camek, K. Becker, *et al.*, "Race: A centralized platform computer based architecture for automotive applications," in *2013 IEEE International Electric Vehicle Conference (IEVC)*, IEEE, 2013.

[23] W. Zeng, M. A. Khalid, and S. Chowdhury, "In-vehicle networks outlook: Achievements and challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1552–1571, 2016.

[24] P. Goransson, C. Black, and T. Culver, *Software defined networks: a comprehensive approach*. Morgan Kaufmann, 2016.

[25] L. Silva, N. Magaia, B. Sousa, *et al.*, "Computing paradigms in emerging vehicular environments: A review," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 3, pp. 491–511, 2021.

[26] A. Haddaji, S. Ayed, and L. C. Fourati, "Artificial Intelligence techniques to mitigate cyber-attacks within vehicular networks: Survey," *Computers and Electrical Engineering*, vol. 104, p. 108 460, 2022.

[27] W. Stolz, R. Kornhaas, R. Krause, and T. Sommer, "Domain control units-the solution for future E/E architectures?" *SAE International*, 2010.

[28] N. Navet and F. Simonot-Lion, "In-vehicle communication networks-a historical perspective and review," *Industrial Communication Technology Handbook, Second Edition*, 2013.

[29] J. Huang, M. Zhao, Y. Zhou, and C.-C. Xing, "In-vehicle networking: Protocols, challenges, and solutions," *IEEE Network*, vol. 33, no. 1, pp. 92–98, 2018.

[30] C. M. Kozierok, C. Correa, R. B. Boatright, and J. Quesnelle, *Automotive Ethernet: The Definitive Guide*. Intrepid Control Systems, 2014.

[31] M. Levi, Y. Allouche, and A. Kontorovich, "Advanced analytics for connected car cybersecurity," in *2018 IEEE 87th vehicular technology conference (VTC spring)*, IEEE, 2018, pp. 1–7.

[32] J. Takahashi, Y. Aragane, T. Miyazawa, *et al.*, "Automotive attacks and countermeasures on lin-bus," *Journal of Information Processing*, vol. 25, pp. 220–228, 2017.

[33] A. Gzemba, *MOST—The automotive multimedia network.—From MOST25 to MOST150*, 2011.

[34] T. Steinbach, "Ethernet-based network architectures for future real-time systems in the car," *ATZ worldwide*, vol. 121, no. 7, pp. 72–77, 2019.

[35] L. Zhao, F. He, E. Li, and J. Lu, "Comparison of time sensitive networking (TSN) and TTEthernet," in *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, IEEE, 2018, pp. 1–7.

[36] T. Steinbach, H.-T. Lim, F. Korf, T. C. Schmidt, D. Herrscher, and A. Wolisz, "Tomorrow's in-car interconnect? A competitive evaluation of IEEE 802.1 AVB and Time-Triggered Ethernet (AS6802)," in *2012 IEEE Vehicular Technology Conference (VTC Fall)*, IEEE, 2012, pp. 1–5.

[37] L. L. Bello, "The case for ethernet in automotive communications," *ACM SIGBED Review*, vol. 8, no. 4, pp. 7–15, 2011.

[38] J. Farkas, L. L. Bello, and C. Gunther, "Time-sensitive networking standards," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 20–21, 2018.

[39] J. L. Messenger, "Time-sensitive networking: An introduction," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 29–33, 2018.

[40] L. L. Bello, "Novel trends in automotive networks: A perspective on Ethernet and the IEEE Audio Video Bridging," *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pp. 1–8, 2014.

[41] X. Yang, D. Scholz, and M. Helm, "Deterministic networking (DetNet) vs time sensitive networking (TSN)," *Network*, vol. 79, 2019.

[42] G. Alderisi, G. Patti, and L. L. Bello, "Introducing support for scheduled traffic over IEEE audio video bridging networks," in *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*, IEEE, 2013, pp. 1–9.

[43] V. Bandur, G. Selim, V. Pantelic, and M. Lawford, "Making the case for centralized automotive E/E architectures," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1230–1245, 2021.

[44] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, *et al.*, "Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 88–145, 2018.

[45] L. Deng, G. Xie, H. Liu, Y. Han, R. Li, and K. Li, "A survey of real-time ethernet modeling and design methodologies: From AVB to TSN," *ACM Computing Surveys (CSUR)*, vol. 55, no. 2, pp. 1–36, 2022.

[46] N. Finn, "Introduction to time-sensitive networking," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 22–28, 2018.

[47] "Time-Sensitive Networking: A Technical Introduction White Paper," en, p. 8, 2017.

[48] H. Zhu, W. Zhou, Z. Li, L. Li, and T. Huang, "Requirements-driven automotive electrical/electronic architecture: A survey and prospective trends," *IEEE Access*, vol. 9, pp. 100 096–100 112, 2021.

[49] H. Askaripoor, M. Hashemi Farzaneh, and A. Knoll, "E/E architecture synthesis: Challenges and technologies," *Electronics*, vol. 11, no. 4, p. 518, 2022.

[50] M. Haeberle, F. Heimgaertner, H. Loehr, *et al.*, "Softwarization of automotive E/E architectures: A software-defined networking approach," in *2020 IEEE Vehicular Networking Conference (VNC)*, IEEE, 2020, pp. 1–8.

[51] N. Navet and F. Simonot-Lion, *Automotive embedded systems handbook.* CRC press, 2017.

[52] U. Keskin, "In-vehicle communication networks: a literature survey," 2009.

[53] S. Brunner, J. Roder, M. Kucera, and T. Waas, "Automotive E/E-architecture enhancements by usage of ethernet TSN," in *2017 13th Workshop on Intelligent Solutions in Embedded Systems (WISES)*, IEEE, 2017, pp. 9–13.

[54] B. Carlson, "Vehicle Network Processors Play Decisive Role in New Vehicle Architectures," *ATZelectronics worldwide*, vol. 16, no. 9, pp. 10–15, 2021.

[55] V. Bandur, V. Pantelic, T. Tomashevskiy, and M. Lawford, "A Safety Architecture for Centralized E/E Architectures," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, IEEE, 2021, pp. 67–70.

[56] V. M. Navale, K. Williams, A. Lagospiris, M. Schaffert, and M.-A. Schweiker, "(R) evolution of E/E architectures," *SAE International Journal of Passenger Cars-Electronic and Electrical Systems*, vol. 8, no. 2015-01-0196, pp. 282–288, 2015.

[57] J. Klaus-Wagenbrenner, "Zonal EE architecture: Towards a fully automotive Ethernet-based vehicle infrastructure," in *Proc. Automotive E/E Architecture Technology Innovation Conference*, 2019.

[58] U. Schifferdecker and C. Rätz, "High-Performance Computing Platforms in the Automobile," en, Feb. 2020.

[59] F. Rehm, J. Seitter, J.-P. Larsson, *et al.*, "The road towards predictable automotive high-performance platforms," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2021.

[60] V. L. Thing and J. Wu, "Autonomous vehicle security: A taxonomy of attacks and defences," in *2016 ieee international conference on internet of things (ithings) and ieee green computing and communications (greencom) and ieee cyber, physical and social computing (cpscom) and ieee smart data (smartdata)*, IEEE, 2016, pp. 164–170.

[61] C. Bernardini, M. R. Asghar, and B. Crispo, "Security and privacy in vehicular communications: Challenges and opportunities," *Vehicular Communications*, vol. 10, pp. 13–28, 2017.

[62] N. T. Courtois, G. V. Bard, and D. Wagner, "Algebraic and slide attacks on KeeLoq," *Lecture Notes in Computer Science*, vol. 5086, pp. 97–115, 2008.

[63] T. Eisenbarth, T. Kasper, A. Moradi, C. Paar, M. Salmasizadeh, and M. T. M. Shalmani, "On the power of power analysis in the real world: A complete break of the KeeLoq code hopping scheme," in *Advances in Cryptology–CRYPTO 2008: 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings 28*, Springer, 2008, pp. 203–220.

[64] I. Rouf, R. D. Miller, H. A. Mustafa, *et al.*, "Security and Privacy Vulnerabilities of In-Car Wireless Networks: A Tire Pressure Monitoring System Case Study.," in *USENIX Security Symposium*, vol. 10, 2010.

[65] C. Miller and C. Valasek, "Adventures in automotive networks and control units," *Def Con*, vol. 21, no. 260-264, pp. 15–31, 2013.

[66] A. Yadav, G. Bose, R. Bhange, K. Kapoor, N. Iyengar, and R. D. Caytiles, "Security, vulnerability and protection of vehicular on-board diagnostics," *International Journal of Security and Its Applications*, vol. 10, no. 4, pp. 405–422, 2016.

[67] D. Klinedinst and C. King, "On board diagnostics: Risks and vulnerabilities of the connected vehicle," *CERT Coordination Center, Tech. Rep*, 2016.

[68] "A practical wireless attack on the connected car and security protocol for in-vehicle CAN, author=Woo, Samuel and Jo, Hyo Jin and Lee, Dong Hoon," *IEEE Transactions on intelligent transportation systems*, vol. 16, no. 2, pp. 993–1006, 2014.

[69] B. Groza, H. Gurban, L. Popa, A. Berdich, and S. Murvay, "Car-to-Smartphone Interactions: Experimental Setup, Risk Analysis and Security Technologies," in *5th International Workshop on Critical Automotive Applications: Robustness & Safety*, 2019.

[70] Z. El-Rewini, K. Sadatsharan, D. F. Selvaraj, S. J. Plathottam, and P. Ranganathan, "Cybersecurity challenges in vehicular communications," *Vehicular Communications*, vol. 23, p. 100 214, 2020.

[71] S. Parkinson, P. Ward, K. Wilson, and J. Miller, "Cyber threats facing autonomous and connected vehicles: Future challenges," *IEEE transactions on intelligent transportation systems*, vol. 18, no. 11, pp. 2898–2915, 2017.

[72] N. Khatri, R. Shrestha, and S. Y. Nam, "Security issues with in-vehicle networks, and enhanced countermeasures based on blockchain," *Electronics*, vol. 10, no. 8, p. 893, 2021.

[73] K. Kim, J. S. Kim, S. Jeong, J.-H. Park, and H. K. Kim, "Cybersecurity for autonomous vehicles: Review of attacks and defense," *Computers & Security*, vol. 103, p. 102 150, 2021.

[74] Z. Petho, I. Khan, and Á. Torok, "Analysis of security vulnerability levels of in-vehicle network topologies applying graph representations," *Journal of Electronic Testing*, pp. 1–9, 2021.

[75] A. Chattopadhyay, K.-Y. Lam, and Y. Tavva, "Autonomous vehicle: Security by design," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, pp. 7015–7029, 2020.

[76] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network," in *2016 international conference on information networking (ICOIN)*, IEEE, 2016, pp. 63–68.

[77] M. R. Moore, R. A. Bridges, F. L. Combs, M. S. Starr, and S. J. Prowell, "Modeling inter-signal arrival times for accurate detection of can bus signal injection attacks: a data-driven approach to in-vehicle intrusion detection," in *Proceedings of the 12th Annual Conference on Cyber and Information Security Research*, 2017, pp. 1–4.

[78] C. Young, H. Olufowobi, G. Bloom, and J. Zambreno, "Automotive intrusion detection based on constant can message frequencies across vehicle driving modes," in *Proceedings of the ACM Workshop on Automotive Cybersecurity*, 2019, pp. 9–14.

[79] H. Lee, S. H. Jeong, and H. K. Kim, "OTIDS: A novel intrusion detection system for in-vehicle network by using remote frame," in *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, IEEE, 2017, pp. 57–5709.

[80] Q. Wang, Y. Qian, Z. Lu, Y. Shoukry, and G. Qu, "A delay based plug-in-monitor for intrusion detection in controller area network," in *2018 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, IEEE, 2018, pp. 86–91.

[81] P. Waszecki, P. Mundhenk, S. Steinhorst, M. Lukasiewycz, R. Karri, and S. Chakraborty, "Automotive electrical and electronic architecture security via distributed in-vehicle traffic monitoring," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 11, pp. 1790–1803, 2017.

[82] S. Halder, M. Conti, and S. K. Das, "COIDS: A clock offset based intrusion detection system for controller area networks," in *Proceedings of the 21st International Conference on Distributed Computing and Networking*, 2020, pp. 1–10.

[83] K. Huang, Q. Zhang, C. Zhou, N. Xiong, and Y. Qin, "An efficient intrusion detection approach for visual sensor networks based on traffic pattern learning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 10, pp. 2704–2713, 2017.

[84] M. Basseville, I. V. Nikiforov, *et al.*, *Detection of abrupt changes: theory and application.* prentice Hall Englewood Cliffs, 1993, vol. 104.

[85] M. Müter, A. Groll, and F. C. Freiling, "A structured approach to anomaly detection for in-vehicle networks," in *2010 Sixth International Conference on Information Assurance and Security*, IEEE, 2010, pp. 92–98.

[86] M. Müter, A. Groll, and F. Freiling, "Anomaly detection for in-vehicle networks using a sensor-based approach," *Journal of Information Assurance and Security*, vol. 6, no. 2, pp. 132–140, 2011.

[87] J. Ning, J. Wang, J. Liu, and N. Kato, “Attacker identification and intrusion detection for in-vehicle networks,” *IEEE communications letters*, vol. 23, no. 11, pp. 1927–1930, 2019.

[88] K.-T. Cho and K. G. Shin, “Error handling of in-vehicle networks makes them vulnerable,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1044–1055.

[89] G. Baldini, “Detection of cybersecurity spoofing attacks in vehicular networks with recurrence quantification analysis,” *Computer Communications*, vol. 191, pp. 486–499, 2022.

[90] M. Müter and N. Asaj, “Entropy-based anomaly detection for in-vehicle networks,” in *2011 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2011, pp. 1110–1115.

[91] W. Wu, Y. Huang, R. Kurachi, *et al.*, “Sliding window optimized information entropy analysis method for intrusion detection on in-vehicle networks,” *IEEE Access*, vol. 6, pp. 45 233–45 245, 2018.

[92] G. Baldini, “On the application of entropy measures with sliding window for intrusion detection in automotive in-vehicle networks,” *Entropy*, vol. 22, no. 9, p. 1044, 2020.

[93] D. Stabili, M. Marchetti, and M. Colajanni, “Detecting attacks to internal vehicle networks through Hamming distance,” in *2017 AEIT International Annual Conference*, IEEE, 2017, pp. 1–6.

[94] E. Seo, H. M. Song, and H. K. Kim, “GIDS: GAN based intrusion detection system for in-vehicle network,” in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, IEEE, 2018, pp. 1–6.

[95] J. Zhang, F. Li, H. Zhang, R. Li, and Y. Li, “Intrusion detection system using deep learning for in-vehicle security,” *Ad Hoc Networks*, vol. 95, p. 101 974, 2019.

[96] S. Tariq, S. Lee, and S. S. Woo, “CANTransfer: Transfer learning based intrusion detection on a controller area network using convolutional LSTM network,” in *Proceedings of the 35th annual ACM symposium on applied computing*, 2020, pp. 1048–1055.

[97] H. M. Song, J. Woo, and H. K. Kim, “In-vehicle network intrusion detection using deep convolutional neural network,” *Vehicular Communications*, vol. 21, p. 100 198, 2020.

[98] H. M. Song and H. K. Kim, “Self-supervised anomaly detection for in-vehicle network using noised pseudo normal data,” *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1098–1108, 2021.

[99] S. Jeong, B. Jeon, B. Chung, and H. K. Kim, “Convolutional neural network-based intrusion detection system for AVTP streams in automotive Ethernet-based networks,” *Vehicular Communications*, vol. 29, p. 100 338, 2021.

[100] J. Wei, Y. Chen, Y. Lai, Y. Wang, and Z. Zhang, “Domain adversarial neural network-based intrusion detection system for in-vehicle network variant attacks,” *IEEE Communications Letters*, vol. 26, no. 11, pp. 2547–2551, 2022.

[101] L. Fei-Fei, R. Fergus, and P. Perona, “One-shot learning of object categories,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[102] L. Y. Pratt, “Discriminability-based transfer between neural networks,” in *Advances in neural information processing systems*, S. Hanson, J. Cowan, and C. Giles, Eds., vol. 5, Morgan-Kaufmann, 1992. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1992/file/67e103b0761e60683e83c559be18d40c-Paper.pdf.

[103] Z. Deng, Y. Xun, J. Liu, S. Li, and Y. Zhao, “A Novel Intrusion Detection System for Next Generation In-Vehicle Networks,” in *GLOBECOM 2022-2022 IEEE Global Communications Conference*, IEEE, 2022, pp. 2098–2103.

[104] A. K. Dwivedi, “Anomaly detection in intra-vehicle networks,” *arXiv preprint arXiv:2205.03537*, 2022.

[105] D. Stabili, L. Ferretti, M. Andreolini, and M. Marchetti, “DAGA: Detecting attacks to in-vehicle networks via n-Gram analysis,” *IEEE Transactions on Vehicular Technology*, vol. 71, no. 11, pp. 11 540–11 554, 2022.

[106] H. Kwon, S. Lee, J. Choi, and B.-h. Chung, “Mitigation mechanism against in-vehicle network intrusion by reconfiguring ECU and disabling attack packet,” in *2018 International Conference on Information Technology (InCIT)*, IEEE, 2018, pp. 1–5.

[107] S.-F. Lokman, A. T. Othman, and M.-H. Abu-Bakar, “Intrusion detection system for automotive Controller Area Network (CAN) bus system: a review,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, pp. 1–17, 2019.

[108] S. Rajapaksha, H. Kalutarage, M. O. Al-Kadri, A. Petrovski, G. Madzudzo, and M. Cheah, “AI-based intrusion detection systems for in-vehicle networks: A survey,” *ACM Computing Surveys*, vol. 55, no. 11, pp. 1–40, 2023.

[109] Y. Xie, Y. Zhou, J. Xu, J. Zhou, X. Chen, and F. Xiao, “Cybersecurity protection on in-vehicle networks for distributed automotive cyber-physical systems: state-of-the-art and future challenges,” *Software: Practice and Experience*, vol. 51, no. 11, pp. 2108–2127, 2021.

[110] I. Zenden, H. Wang, A. Iacovazzi, A. Vahidi, R. Blom, and S. Raza, “On the Resilience of Machine Learning-Based IDS for Automotive Networks,” in *2023 IEEE Vehicular Networking Conference (VNC)*, IEEE, 2023, pp. 239–246.

[111] K. He, D. D. Kim, and M. R. Asghar, “Adversarial Machine Learning for Network Intrusion Detection Systems: A Comprehensive Survey,” *IEEE Communications Surveys & Tutorials*, 2023.

[112] Y. Vardi, “Network tomography: Estimating source-destination traffic intensities from link data,” *Journal of the American statistical association*, vol. 91, no. 433, pp. 365–377, 1996.

[113] K Claffy, T. E. Monk, and D. McRobb, “Internet tomography,” *Nature*, pp. 1–6, 1999.

[114] A. Coates, A. O. Hero III, R. Nowak, and B. Yu, “Internet tomography,” *IEEE Signal processing magazine*, vol. 19, no. 3, pp. 47–65, 2002.

[115] T. Bu, N. Duffield, F. L. Presti, and D. Towsley, “Network tomography on general topologies,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 30, no. 1, pp. 21–30, 2002.

[116] A. Gopalan and S. Ramasubramanian, “On identifying additive link metrics using linearly independent cycles and paths,” *IEEE/ACM Transactions on Networking*, vol. 20, no. 3, pp. 906–916, 2011.

[117] L. Ma, T. He, K. K. Leung, D. Towsley, and A. Swami, “Efficient identification of additive link metrics via network tomography,” in *2013 IEEE 33rd International Conference on Distributed Computing Systems*, IEEE, 2013, pp. 581–590.

[118] H. Li, Y. Gao, W. Dong, and C. Chen, “Preferential link tomography in dynamic networks,” *IEEE/ACM transactions on networking*, vol. 27, no. 5, pp. 1801–1814, 2019.

[119] P. Qin, B. Dai, G. Xu, K. Wu, and B. Huang, “Taking a free ride for routing topology inference in peer-to-peer networks,” *Peer-to-Peer Networking and Applications*, vol. 9, pp. 1047–1059, 2016.

[120] X. Zhang and C. Phillips, “A survey on selective routing topology inference through active probing,” *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 1129–1141, 2011.

[121] B. Eriksson, G. Dasarathy, P. Barford, and R. Nowak, “Efficient network tomography for internet topology discovery,” *IEEE/ACM Transactions on Networking*, vol. 20, no. 3, pp. 931–943, 2011.

[122] R. Zhang, Y. Li, and X. Li, “Topology inference with network tomography based on t-test,” *IEEE communications letters*, vol. 18, no. 6, pp. 921–924, 2014.

[123] B. Xi, G. Michailidis, and V. N. Nair, “Estimating network loss rates using active tomography,” *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1430–1448, 2006.

[124] E. Lawrence, G. Michailidis, and V. N. Nair, “Statistical inverse problems in active network tomography,” *Lecture notes-monograph series*, pp. 24–44, 2007.

[125] X. Fan, X. Li, and J. Zhang, “Compressed sensing based loss tomography using weighted $l1$ minimization,” *Computer Communications*, vol. 127, pp. 122–130, 2018.

[126] Y. Tsang, M. Coates, and R. Nowak, “Passive unicast network tomography based on TCP monitoring,” *Rice University, ECE Department Technical Report TR-0005*, 2000.

[127] Y. Tsang, M. Coates, and R. Nowak, “Passive network tomography using EM algorithms,” in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)*, IEEE, vol. 3, 2001, pp. 1469–1472.

[128] V. N. Padmanabhan and L. Qiu, “Network tomography using passive end-to-end measurements,” in *DIMACS Workshop on Internet and WWW Measurement, Mapping and Modeling*, Citeseer, 2002.

[129] H. Yao, S. Jaggi, and M. Chen, “Passive network tomography for erroneous networks: A network coding approach,” *IEEE Transactions on Information Theory*, vol. 58, no. 9, pp. 5922–5940, 2012.

[130] L. Ma, T. He, K. K. Leung, A. Swami, and D. Towsley, “Inferring link metrics from end-to-end path measurements: Identifiability and monitor placement,” *IEEE/ACM transactions on networking*, vol. 22, no. 4, pp. 1351–1368, 2014.

[131] N. G. Duffield, J. Horowitz, F. L. Presti, and D Towsley, “Network delay tomography from end-to-end unicast measurements,” in *IWDC*, Springer, vol. 1, 2001, pp. 576–595.

[132] M. J. Coates and R. D. Nowak, “Network tomography for internal delay estimation,” in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)*, IEEE, vol. 6, 2001, pp. 3409–3412.

[133] Y. Tsang, M. Coates, and R. D. Nowak, “Network delay tomography,” *IEEE Transactions on Signal Processing*, vol. 51, no. 8, pp. 2125–2136, 2003.

[134] N. G. Duffield and F. L. Presti, “Network tomography from measured end-to-end delay covariance,” *IEEE/ACM Transactions On Networking*, vol. 12, no. 6, pp. 978–992, 2004.

[135] E. Lawrence, G. Michailidis, and V. N. Nair, “Network delay tomography using flexicast experiments,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 5, pp. 785–813, 2006.

[136] V. Arya, N. G. Duffield, and D. Veitch, “Temporal delay tomography,” in *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*, IEEE, 2008, pp. 276–280.

[137] K. Deng, Y. Li, W. Zhu, Z. Geng, and J. S. Liu, “On delay tomography: Fast algorithms and spatially dependent models,” *IEEE Transactions on Signal Processing*, vol. 60, no. 11, pp. 5685–5697, 2012.

[138] P. Qin, B. Dai, K. Wu, B. Huang, and G. Xu, “DCE: A novel delay correlation measurement for tomography with passive realization,” *arXiv preprint arXiv:1307.5085*, 2013.

[139] Y. Gao, W. Dong, C. Chen, *et al.*, “Domo: passive per-packet delay tomography in wireless ad-hoc networks,” in *2014 IEEE 34th International Conference on Distributed Computing Systems*, IEEE, 2014, pp. 419–428.

[140] N. E. Rad, Y. Ephraim, and B. L. Mark, “Delay network tomography using a partially observable bivariate Markov chain,” *IEEE/ACM Transactions on Networking*, vol. 25, no. 1, pp. 126–138, 2016.

[141] R. N. Pradhan, M. S. Khan, M. Nijim, and R. Challoo, “Network delay modeling in Mobile Wireless Mesh Networks using Network Tomography,” in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, 2017, pp. 1–8.

[142] H.-T. Wei, S.-H. Hsieh, W.-L. Hwang, C.-S. Liao, and C.-S. Lu, “Link Delay Estimation Using Sparse Recovery for Dynamic Network Tomography,” *arXiv preprint arXiv:1812.00369*, 2018.

[143] D. Ghita, H. Nguyen, M. Kurant, K. Argyraki, and P. Thiran, “Netscope: Practical network loss tomography,” in *2010 Proceedings IEEE INFOCOM*, IEEE, 2010, pp. 1–9.

[144] Y. Qiao, G. Wang, X.-s. Qiu, and R. Gu, “Network loss tomography using link independence,” in *2012 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, 2012, pp. 000 569–000 574.

[145] X. Cao, Y. Wang, X. Qiu, and L. Meng, “End-to-end path loss inference algorithm with network tomography,” in *2013 15th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, IEEE, 2013, pp. 1–3.

[146] N. Duffield, “Simple network performance tomography,” in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, 2003, pp. 210–215.

[147] N. Duffield, “Network tomography of binary network performance characteristics,” *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5373–5388, 2006.

[148] M. Mukamoto, T. Matsuda, S. Hara, K. Takizawa, F. Ono, and R. Miura, “Adaptive boolean network tomography for link failure detection,” in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, IEEE, 2015, pp. 646–651.

[149] N. Ogino, T. Kitahara, S. Arakawa, G. Hasegawa, and M. Murata, “Decentralized boolean network tomography based on network partitioning,” in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, IEEE, 2016, pp. 162–170.

[150] N. Bartolini, T. He, V. Arrigoni, A. Massini, F. Trombetti, and H. Khamfroush, “On fundamental bounds on failure identifiability by boolean network tomography,” *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 588–601, 2020.

[151] N. Galesi and F. Ranjbar, “Counting and localizing defective nodes by Boolean network tomography,” *arXiv preprint arXiv:2101.04403*, 2021.

[152] S. Zarifzadeh, M. Gowdagere, and C. Dovrolis, "Range tomography: combining the practicality of boolean tomography with the resolution of analog tomography," in *Proceedings of the 2012 Internet Measurement Conference*, 2012, pp. 385–398.

[153] Y. Chen, D. Bindel, and R. H. Katz, "Tomography-based overlay network monitoring," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, 2003, pp. 216–231.

[154] M. H. Firooz and S. Roy, "Network tomography via compressed sensing," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, IEEE, 2010, pp. 1–5.

[155] M. Firooz and S. Roy, "Network Tomography via Compressed Sensing - with path selection," Dec. 2010, pp. 1–5. DOI: 10.1109/GLOCOM.2010.5684036.

[156] W. Wang, X. Gan, W. Bai, X. Wang, and X. Tian, "Compressed sensing based network tomography using end-to-end path measurements," in *2017 IEEE International Conference on Communications (ICC)*, IEEE, 2017, pp. 1–6.

[157] G Sharma, S Jaggi, and B. Dey, "Network tomography via network coding," in *2008 Information Theory and Applications Workshop*, IEEE, 2008, pp. 151–157.

[158] H. Yao, S. Jaggi, and M. Chen, "Network coding tomography for network failures," in *2010 Proceedings IEEE INFOCOM*, IEEE, 2010, pp. 1–5.

[159] P. Qin, B. Dai, B. Huang, G. Xu, and K. Wu, "A survey on network tomography with network coding," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1981–1995, 2014.

[160] V. N. Padmanabhan, L. Qiu, and H. J. Wang, "Passive network tomography using bayesian inference," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, 2002, pp. 93–94.

[161] X. Fan and X. Li, "Network tomography via sparse Bayesian learning," *IEEE Communications Letters*, vol. 21, no. 4, pp. 781–784, 2017.

[162] G. Liang and B. Yu, "Maximum pseudo likelihood estimation in network tomography," *IEEE Transactions on Signal Processing*, vol. 51, no. 8, pp. 2043–2053, 2003.

[163] G. Liang and B. Yu, "Pseudo likelihood estimation in network tomography," in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, IEEE, vol. 3, 2003, pp. 2101–2111.

[164] Z. Yang, R. Gu, T. Dong, *et al.*, "An artificial neural network based attenuation tomography in free space optical network," in *2018 International Conference on Networking and Network Applications (NaNA)*, IEEE, 2018, pp. 52–57.

[165] L. Ma, Z. Zhang, and M. Srivatsa, "Neural network tomography," *arXiv preprint arXiv:2001.02942*, 2020.

[166] M. Rahali, J.-M. Sanner, and G. Rubino, "TOM: a self-trained Tomography solution for Overlay networks Monitoring," in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, 2020, pp. 1–6.

[167] I. Sartzetakis and E. Varvarigos, "Machine Learning Network Tomography with partial topology knowledge and dynamic routing," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*, IEEE, 2022, pp. 4922–4927.

[168] G. I. Mary, Z. C. Alex, and L. Jenkins, "Response time analysis of messages in controller area network: a review," *Journal of Computer Networks and Communications*, vol. 2013, 2013.

[169] G. Kakkavas, A. Stamou, V. Karyotis, and S. Papavassiliou, "Network tomography for efficient monitoring in SDN-enabled 5G networks and beyond: Challenges and opportunities," *IEEE Communications Magazine*, vol. 59, no. 3, pp. 70–76, 2021.

[170] M. H. Raza, A. Nafarieh, and W. Robertson, "Application of network tomography in load balancing," *Procedia Computer Science*, vol. 52, pp. 1120–1125, 2015.

[171] W. Wang, H. Wang, B. Wang, Y. Wang, and J. Wang, "Energy-aware and self-adaptive anomaly detection scheme based on network tomography in mobile ad hoc networks," *Information Sciences*, vol. 220, pp. 580–602, 2013.

[172] T. He, "Distributed link anomaly detection via partial network tomography," *ACM SIGMETRICS Performance Evaluation Review*, vol. 45, no. 3, pp. 29–42, 2018.

[173] L. Ma, T. He, A. Swami, D. Towsley, K. K. Leung, and J. Lowe, "Node failure localization via network tomography," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, 2014, pp. 195–208.

[174] V. W. Bandara, A. P. Jayasumana, and R. Whitner, “An adaptive compressive sensing scheme for network tomography based fault localization,” in *2014 IEEE International Conference on Communications (ICC)*, IEEE, 2014, pp. 1290–1295.

[175] S. Pan, P. Li, C. Yi, D. Zeng, Y.-C. Liang, and G. Hu, “Edge intelligence empowered urban traffic monitoring: A network tomography perspective,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 2198–2211, 2020.

[176] R. Zhang, S. Newman, M. Ortolani, and S. Silvestri, “A network tomography approach for traffic monitoring in smart cities,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2268–2278, 2018.

[177] A. Paranjothi, M. S. Khan, R. Patan, R. M. Parizi, and M. Atiquzzaman, “VANETomo: A congestion identification and control scheme in connected vehicles using network tomography,” *Computer Communications*, vol. 151, pp. 275–289, 2020.

[178] G. Michailidis, “Power allocation to a network of charging stations based on network tomography monitoring,” in *2013 18th International Conference on Digital Signal Processing (DSP)*, IEEE, 2013, pp. 1–6.

[179] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT press, 2016.

[180] S. J. Russell, *Artificial intelligence a modern approach.* Pearson Education, Inc., 2010.

[181] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[182] C. C. Aggarwal *et al.*, “Neural networks and deep learning,” *Springer*, vol. 10, no. 978, p. 3, 2018.

[183] R. Rojas and R. Rojas, “The backpropagation algorithm,” *Neural networks: a systematic introduction*, pp. 149–182, 1996.

[184] A. F. Agarap, “Deep learning using rectified linear units (ReLU),” *arXiv preprint arXiv:1803.08375*, 2018.

[185] S. Sharma, S. Sharma, and A. Athaiya, “Activation functions in neural networks,” *Towards Data Sci*, vol. 6, no. 12, pp. 310–316, 2017.

[186] S. Narayan, “The generalized sigmoid activation function: Competitive supervised learning,” *Information sciences*, vol. 99, no. 1-2, pp. 69–82, 1997.

[187] P. Jeatrakul and K. W. Wong, “Comparing the performance of different neural networks for binary classification problems,” in *2009 Eighth International Symposium on Natural Language Processing*, IEEE, 2009, pp. 111–115.

[188] B. Gao and L. Pavel, “On the properties of the softmax function with application in game theory and reinforcement learning,” *arXiv preprint arXiv:1704.00805*, 2017.

[189] G. Ou and Y. L. Murphey, “Multi-class pattern classification using neural networks,” *Pattern recognition*, vol. 40, no. 1, pp. 4–18, 2007.

[190] D. F. Specht *et al.*, “A general regression neural network,” *IEEE transactions on neural networks*, vol. 2, no. 6, pp. 568–576, 1991.

[191] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, “Loss functions for image restoration with neural networks,” *IEEE Transactions on computational imaging*, vol. 3, no. 1, pp. 47–57, 2016.

[192] S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv preprint arXiv:1609.04747*, 2016.

[193] L. Bottou *et al.*, “Stochastic gradient learning in neural networks,” *Proceedings of Neuro-Nımes*, vol. 91, no. 8, p. 12, 1991.

[194] D. R. Wilson and T. R. Martinez, “The general inefficiency of batch training for gradient descent learning,” *Neural networks*, vol. 16, no. 10, pp. 1429–1451, 2003.

[195] S. Khirirat, H. R. Feyzmahdavian, and M. Johansson, “Mini-batch gradient descent: Faster convergence under data sparsity,” in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, IEEE, 2017, pp. 2880–2887.

[196] R. Diestel, “Graph theory,” *Grad. Texts in Math*, vol. 101, p. 867, 2005.

[197] A. Benjamin, G. Chartrand, and P. Zhang, *The fascinating world of graph theory.* Princeton University Press, 2017.

[198] A. Varga, “OMNeT++,” in *Modeling and tools for network simulation*, Springer, 2010, pp. 35–59.

[199] R. L. S. De Oliveira, C. M. Schweitzer, A. A. Shinoda, and L. R. Prete, “Using Mininet for emulation and prototyping software-defined networks,” in *2014 IEEE Colombian conference on communications and computing (COLCOM)*, Ieee, 2014, pp. 1–6.

[200] C. Fernandez and J. L. Munoz, "Software Defined Networking (SDN) with OpenFlow 1.3, Open vSwitch and Ryu," *UPC Telematics Department*, 2015.

[201] R. Khondoker, A. Zaalouk, R. Marx, and K. Bayarou, "Feature-based comparison and selection of Software Defined Networking (SDN) controllers," in *2014 world congress on computer applications and information systems (WCCAIS)*, IEEE, 2014, pp. 1–7.

[202] L. Mamushiane, A. Lysko, and S. Dlamini, "A comparative evaluation of the performance of popular SDN controllers," in *2018 Wireless Days (WD)*, IEEE, 2018, pp. 54–59.

[203] D. Houcque *et al.*, "Introduction to Matlab for engineering students," *Northwestern University*, no. 1, 2005.

[204] W. McKinney *et al.*, "Pandas: A foundational Python library for data analysis and statistics," *Python for high performance and scientific computing*, vol. 14, no. 9, pp. 1–9, 2011.

[205] N. Ari and M. Ustazhanov, "Matplotlib in python," in *2014 11th International Conference on Electronics, Computer and Computation (ICECCO)*, IEEE, 2014, pp. 1–6.

[206] N. Ketkar and E. Santana, *Deep learning with Python*. Springer, 2017, vol. 1.

[207] J. C. Walrand, M. Turner, and R. Myers, "An Architecture for In-Vehicle Networks," *IEEE Transactions on Vehicular Technology*, 2021.

[208] A. Ibraheem, Z. Sheng, G. Parisis, and D. Tian, "In-Vehicle Network Delay Tomography," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*, IEEE, 2022, pp. 5528–5533.

[209] Y. Xia and D. Tse, "Inference of link delay in communication networks," *IEEE Journal on Selected areas in Communications*, vol. 24, no. 12, pp. 2235–2248, 2006.

[210] A. M. Turing, "Rounding-off errors in matrix processes," *The Quarterly Journal of Mechanics and Applied Mathematics*, vol. 1, no. 1, pp. 287–308, 1948.

[211] T. Steinbach, P. Meyer, S. Buschmann, and F. Korf, "Extending OMNeT++ towards a platform for the design of future in-vehicle network architectures," *arXiv preprint arXiv:1609.05179*, 2016.

[212] D. Thiele, J. Schlatow, P. Axer, and R. Ernst, "Formal timing analysis of CAN-to-Ethernet gateway strategies in automotive networks," *Real-time systems*, vol. 52, pp. 88–112, 2016.

[213] E. Bisong and E. Bisong, "Introduction to Scikit-learn," *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, pp. 215–229, 2019.

[214] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[215] C. Gay and T. Matsumoto, "A Study of Message Arrival Timestamps on Controller Area Networks," *International Journal of Automotive Engineering*, vol. 13, no. 1, pp. 29–37, 2022.

[216] Y. J. Kim, J. H. Kim, B. M. Cheon, Y. S. Lee, and J. W. Jeon, "Performance of IEEE 802.1 AS for automotive system using hardware timestamp," in *The 18th IEEE International Symposium on Consumer Electronics (ISCE 2014)*, IEEE, 2014, pp. 1–2.

[217] A. Ibraheem, Z. Sheng, G. Parisis, and D. Tian, "Neural network based partial tomography for in-vehicle network monitoring," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, 2021, pp. 1–6.

[218] P. Meyer, F. Korf, T. Steinbach, and T. C. Schmidt, "Simulation of mixed critical in-vehicular networks," in *Recent Advances in Network Simulation: The OMNeT++ Environment and its Ecosystem*, A. Virdis and M. Kirsche, Eds. Cham: Springer International Publishing, 2019, pp. 317–345.

[219] M. Buechel, J. Frtunikj, K. Becker, *et al.*, "An automated electric vehicle prototype showing new trends in automotive architectures," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, IEEE, 2015.

[220] R. Hegde, S. Kumar, and K. Gurumurthy, "The impact of network topologies on the performance of the in-vehicle network," *International Journal of Computer Theory and Engineering*, 2013.

[221] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, 2014.

[222] N. McKeown, T. Anderson, H. Balakrishnan, *et al.*, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM computer communication review*, 2008.

[223] P.-W. Tsai, C.-W. Tsai, C.-W. Hsu, and C.-S. Yang, "Network monitoring in software-defined networking: A review," *IEEE Systems Journal*, 2018.

[224] C. Birkinshaw, E. Rouka, and V. G. Vassilakis, “Implementing an intrusion detection and prevention system using software-defined networking: Defending against port-scanning and denial-of-service attacks,” *Journal of Network and Computer Applications*, 2019.

[225] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, “Network slicing and softwarization: A survey on principles, enabling technologies, and solutions,” *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[226] T. Ahmed, A. Alleg, R. Ferrus, and R. Riggio, “On-demand network slicing using SDN/NFV-enabled satellite ground segment systems,” in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, ieee, 2018.

[227] N. Zope, S. Pawar, and Z. Saquib, “Firewall and load balancing as an application of SDN,” in *2016 Conference on advances in signal processing (CASP)*, IEEE, 2016.

[228] W. Zhou, L. Li, M. Luo, and W. Chou, “REST API design patterns for SDN northbound API,” in *2014 28th international conference on advanced information networking and applications workshops*, IEEE, 2014, pp. 358–365.

[229] R. Tagyo, D. Ikegami, and R. Kawahara, “Network tomography using routing probability for virtualized network,” in *2018 IEEE International Conference on Communications (ICC)*, IEEE, 2018.

[230] P. Fussey and G. Parisis, “Poster: an in-vehicle software defined network architecture for connected and automated vehicles,” in *Proceedings of the 2nd ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services*, 2017.

[231] A. Ibraheem, “Cross Network Slicing in Vehicular Networks,” in *Intelligent Technologies for Internet of Vehicles*, Springer, 2021, pp. 151–189.

[232] S. R. Pokhrel, “Software defined internet of vehicles for automation and orchestration,” *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[233] T. Mekki, I. Jabri, A. Rachedi, and L. Chaari, “Software-defined networking in vehicular networks: A survey,” *Transactions on Emerging Telecommunications Technologies*, 2022.

[234] T. Häckel, A. Schmidt, P. Meyer, F. Korf, and T. C. Schmidt, “Strategies for Integrating Control Flows in Software-Defined In-Vehicle Networks and Their Impact on Network Security,” in *2020 IEEE Vehicular Networking Conference (VNC)*, IEEE, 2020.

[235] R. E. Tarjan and T. RE, “A Note on Finding the Bridges of a Graph.,” 1974.