STATUS OF THESIS

Title of thesis | Formal Specification Language for Vehicular Ad-Hoc Networks

I MAYTHEM KAMAL ABBAS

Hereby allow my thesis to be placed at the information Resource Center (IRC) of Universiti Teknologi PETRONAS (UTP) with the following conditions:

1. The thesis becomes the property of UTP
2. The IRC of UTP may make copies of the thesis for academic purposes only.
3. The thesis is classified as:

☐ Confidential

☑ Non-confidential

If this thesis is confidential, please state the reason:

The contents of the thesis will remain for _____ years.

Remarks on Disclosure:

Endorsed by

MAYTHEM KAMAL ABBAS
Universiti Teknologi PETRONAS
Malaysia.

Date: _____

AZWEEN ABDULLAH
Universiti Teknologi PETRONAS
Malaysia.

Date: _____

UNIVERSITI TEKNOLOGI PETRONAS

Approval by Supervisor (s)

The undersigned certify that they have read, and recomand to the Postgraduate Studeies Programme for acceptance, a thesis titled "**Formal Specification Language for Vehicular Ad-hoc Networks**" submitted by (**Maythem Kamal Abbas**) for the fulfillment of the requirements for the degree of Master of Science in Information Technology.

_____

Date

Signature : _____

Main Supervisor : _____

Date : _____

Co-Supervisor : _____

TITLE PAGE

UNIVERSITI TEKNOLOGI PETRONAS

Formal Specification Language for Vehicular Ad-hoc Networks

By

Maythem Kamal Abbas

A THESIS

SUBMITTED TO THE POSTGRADUATE STUDIES PROGRAMME

AS A REQUIREMENT FOR THE

DEGREE OF MASTER OF SCIENCE

INFORMATION TECHNOLOGY

BANDAR SERI ISKANDAR PERAK

APRIL, 2009.

DECLARATION

I hereby declare that the thesis is based on my original work except for quotations and citations which have been duly acknowledge. I also declare that it has not been previously or concurrently submitted for any other degree at UTP or other institutions.

Signature : _____

Name : _____

Date : _____

FORMAL SPECIFICATION LANGAUGE FOR VEHICULAR AD-HOC NETWORK

Abstract

Vehicular Ad-Hoc Network (VANET) is a form of Mobile Ad-Hoc Network (wireless Network), originally used to provide safety & comfort for passengers, & currently being used to establish Dedicated Short Range Communications (DSRC) among near by Vehicles (V2V Communications) and between vehicles and nearby fixed infrastructure equipments; Roadside equipments (V2I Communications). VANET was used also to warn drivers of collision possibilities, road sign alarms, auto-payment at road tolls and parks. Usually VANET can be found in Intelligent Transportation Systems (ITS).

VANET is the current and near future hot topic for research, that has been targeted by many researchers to develop some applications and protocols specifically for the VANET. But a problem facing all VANET researchers is the unavailability of a formal specification language to specify the VANET systems, protocols, applications and scenarios proposed by those researchers.

A specification language is a formal language that is used during the systems design, analysis, and requirements analysis. Using a formal specification language, a researcher can show "What his system does", Not How.

As a contribution of our research we have created a formal specification language for VANET. We made the use of some Romans characters & some basic symbols to represent VANET Systems & Applications. In addition, we have created some combined symbols to represent actions and operations of the VANET system and its participating devices. Our formal specification language covers many of the VANET aspects, and offers Validity Test and Consistency Test for the systems.

Using our specification language, we have presented three different case studies based on a VANET system model we have created and put them into the system validity and consistency tests and showed how to describe a VANET system and its applications using our formal specification language.

FORMAL SPECIFICATION LANGUAGE FOR VEHICULAR AD-HOC NETWORK

ABSTRAK

Vehicular Ad-Hoc Network (VANET) terhasil dari Mobile Ad-Hoc Network (wireless Network), pada dasarnya digunakan untuk penyediaan keselamatan dan keselesaan kepada penunggang, dan pada masa kini ia dipertingkatkan dengan penggunaan komunikasi jarak dekat (DSRC) di antara kenderaan-kenderaan (V2V Communications) dan infrastruktur yang berhampiran; kemudahan di tepi jalan((V2I Communications). VANET juga digunakan untuk memberi amaran kepada pemandu tentang kemungkinan berlakunya perlanggaran, amaran tanda jalan, pembayaran automatik di tol dan di tempat meletakkankan kenderaan. Selalunya VANET boleh didapati di ITS (intelligent transportation Systems).

VANET adalah merupakan topik yang 'panas' pada masa kini dalam penyelidikan dan ia menjadi sasaran kepada penyelidik untuk memperkembangkan beberapa aplikasi dan protokol khususnya dalam VANET. Tetapi masalah yang dihadapi oleh penyelidik VANET adalah ketidakmampuan bahasa rasmi yang khusus untuk pengkhususan sistem VANET, protokol, aplikasi dan senario yang diusulkan oleh penyelidik-penyelidik.

Bahasa khusus adalah bahasa rasmi yang digunakan semasa merekacipta sistem, analisis, dan keperluan analisis. Penggunaan bahasa rasmi yang khusus, penyelidik boleh mempamerkan apa yang sistem itu hasilkan, bukannya bagaimana ia dihasilkan.

Sebagai penyumbang kepada penyelidikan, kami telah menghasilkan bahasa rasmi yang khusus untuk VANET. Kami menggunakan beberapa karakter Romans dan beberapa simbol asas untuk dipersembahkan kepada sistem VANET dan aplikasi. Tambahan pula, kami telah menghasilkan beberapa kombinasi simbol untuk aksi terkini dan operasi sistem VANET serta alat-alat yang terlibat dengannya. Bahasa rasmi khusus kami meliputi banyak aspek VANET dan menawarkan Ujian Pengesahan and Ujian Konsistensi untuk sistem.

Dengan menggunakan bahasa khusus kami, kami telah membentangkan tiga perbezaan kajian kes berdasarkan dari model sistem VANET yang telah terhasil dan

diletakkan ke dalam pengesahan sistem dan ujian konsistensi serta mempamerkan bagaimana untuk menggambarkan sistem VANET dan aplikasinya menggunakan bahasa rasmi yang khusus dihasilkan oleh kami.

ACKNOWLEDGEMENTS

First of all, I cannot thank enough to my family for always supporting me in my personal and academic endeavors. Their relentless encouragement has enabled me to persevere even in times of duress. As well as, I would like to thank my dear fiancée for her continuous support to me, that she was my inspirer, always.

I would like to thank my supervisor Dr. Azween Bin Abdullah for his supervision and guidance throughout the whole work with this thesis. I greatly appreciate his encouragement to go ahead with my work especially in time of lake of myself confidence.

I would like to thank Dr. Ahmad Kamil Mahmood, for his support and leading as the head of the computer and Information Science department, UTP. In addition, I'd like to thank Dr. Etienne Schneider for his friendly advices and helping me to get a source book related to my work I needed.

Special thanks for all of Dr. Baharum Baharudin, Mr. Low Tan Jung and Dr. Etienne Schneider for their helpful comments when reviewing my research papers.

My sincere thanks to all the administrative staff in UTP for their assistance. Special thanks for all of my colleagues under the supervision of Dr. Azween for sharing their experiences and knowledge with me. Finally, I would like to address my appreciation to all of my colleagues in CIS department for being brothers and sisters to me.

TABLE OF CONTENTS

STATUS OF THESIS……. i
SUPERVISOR APPROVAL…….. ii
TITLE PAGE……. iii
DECLARATION……. iv
ABSTRACT……. v
ABSTRAK……. vi
ACKNOWLEDGEMENTS……..... viii
TABLE OF CONTENTS……. ix
LIST OF TABLES……..... xiii
LIST OF FIGURES……. xiv
LIST OF ABREVIATIONS……. xiv
CHAPTER ONE: INTRODUCTION…… 1
1.1 VANET …… 1
1.2 DSRC…… 2
1.3 VANET Characteristics…… 3
1.4 VANET Communication promises…… 4
1.5 DSRC – VANET Applications…… 4
1.6 VANET Applications…… 5
1.7 Formal Specification Language…… 5
1.8 Objective of the Study…… 7
1.9 Motivation…… 7
1.10 VANET challenges…… 8
1.11 Scope of the Study…… 7
1.12 Case Studies…… 9
1.13 Thesis Layout…… 10
1.14 Abbreviations…… 10
CHAPTER TWO: RELATED WORK…… 11
2.1 Literature Review …… 11

2.1.1 Services through VANET……..…….... 11
2.1.2 Data Dissemination……..… 12
2.1.3 VANET Modeling & Simulation…….…… 13
2.1.4 VANET Communication Enhancement..……..……... 13
2.2 VANET Security Issues……... 14
2.3 State of the Art…….…… 15
2.4 Related Work Summary…….……..…… 16
CHAPTER THREE: METHODOLOGY…….……...…..…… 17
3.1 Proposed Used Basic symbols / Notation……..……17
3.1.1 Device Ability……..… 17
3.1.2 Device Movement…… 18
3.1.3 Messages exchangeability…… 18
3.1.4 Device Internal behavior logic…….…… 20
3.1.5 Security…… 22
3.1.6 Validity Test …… 23
3.1.7 Consistency Test……..…… 26
3.1.8 Design and Configuration…… 28
3.1.9 Environment description……..… 30
3.1.10 Connections description…….…… 32
3.2 Layered System Model for VANET Environment & Application Layer Protocol……... 34
3.2.1 System Devices…… 35
3.2.2 Assumptions…….…… 41
3.2.3 Devices Communications……..…….. 41
3.2.4 The Application Layer Protocol Abilities…… 43
3.2.4.1 Remote Speed limit phase……... 44
3.2.4.2 Sending a service message to a Layer-3 Device phase……... 47
3.2.4.3 Getting access from a mobile node into another node phase…….. 52
CHAPTER FOUR: CASE STUDIES……... 70
4.1 Introduction……...… 70

4.2 Case study -1: Speed control & Highway monitoring…… 70
4.2.1 Participated devices definitions ……..… 71
4.2.2 Communications between participating devices……...……... 83
4.2.3 Participating messages……...……..…… 85
4.2.4 Messages Flow for Case Study-1……..… 88
4.2.5 A Lower-Level Scenario Specification…… 90
4.3 Case Study–2: Remote car locating & sending a Service request / Function message……...… 99
4.3.1 Participated devices definitions …… 100
4.3.2 Communications between participated devices……..… 101
4.3.3 Participated messages…… 103
4.3.4 The Highest-Level Message flow Specification…… 107
4.3.5 Message Flow of Case Study-2……..…… 107
4.3.6 A Lower-Level Scenario Specification…….. 110
4.4 Case Study – 3: Suspect car instant termination……..…… 120
4.4.1 Participated devices definitions …… 121
4.4.2 Communications between participated devices…….…… 124
4.4.3 Participated messages…… 125
4.4.4 Messages Flow of Case Study-3…… 134
CHAPTER FIVE: RESULTS AND DISCUSSION……...…… 138
5.1 Introduction……..……...… 138
5.2 Case studies Differences…….…… 138
5.2.1 Validity test…… 138
5.2.2 Consistency Test…… 147
5.3 Comparative Study……..… 151
CHAPTER SIX: CONCLUSION AND FUTURE WORK……... 155
6.1 Introduction…… 155
6.2 Conclusion……...…… 155
6.3 Future Work…… 156
6.3.1 Tool Development……..… 156

6.3.2 Language Enhancement…….…… 156
6.3.3 Application layer protocol enhancement……...… 157
6.3.4 Developing a simulator for VANET systems…… 157
6.3.5 Developing more of VANET applications…… 157
REFERENCES……...… 158
BIBLOGRAPHY…… 161
REFEREED COPNFERENCES PROCEEDINGS……..…..…… 164

LIST OF TABLES

Table 3.1: Validity Truth Table…… 24
Table 3.2: Consistency Truth Table……... 26
Table 3.3: Layered System Specific functions descriptions ……..……..... 33
Table 5.1: First Scenario's Validity Truth Table……..……..…….. 139
Table 5.2: Second Scenario's Validity Truth Table……..…..…... 141
Table 5.3: Op-AB Validity Truth Table……... 143
Table 5.4: Op-CF Validity Truth Table……..…….. 144
Table 5.5: Op-GH Validity Truth Table……... 145
Table 5.6: Op-IL Validity Truth Table……. 145
Table 5.7: Op-MP Validity Truth Table……... 146
Table 5.8: First Scenario's Consistency Truth Table…… 147
Table 5.9: Second Scenario's Consistency Truth Table……...…… 148
Table 5.10: Op-AB Consistency Truth Table……... 149
Table 5.11: Op-CF Consistency Truth Table…… 150
Table 5.12: Language Type…….. 151
Table 5.13: Specification Languages - Application Area……..... 152
Table 5.14: Covered aspects Comparison……. 154

LIST OF FIGURES

Figure 1.1: VANET Integrated Infrastructure…… 1
Figure 1.2: Interface Objects…….…….. 6
Figure 3.1: Validity / Consistency Test Example……. 23
Figure 3.2: System operational grouping……..…… 25
Figure 3.3: Standards Directions…….…….. 30
Figure 3.4 – Our proposed Layered system…….. 35
Figure 3.5: Remote Speed limit phase Message Flow…….. 44
Figure 3.6: Sending a service message to a Layer-3 Device phase Message Flow…….. 48
Figure 3.7: Getting access from a mobile node into another node phase Message Flow……. 53
Figure 4.1: Case Study 1: Speed Control & High Way Monitoring……. 70
Figure 4.2: Coordinator device internal architecture…… 73
Figure 4.3: Road side equipment internal architecture……..... 77
Figure 4.4 : Vehicle's VANET Device (VVD)…… 80
Figure 4.5: Speed Control & High Way Monitoring – messages flow…… 89
Figure 4.6: Speed Control & High Way Monitoring – messaging algorithm…….. 90
Figure 4.7: Case Study 1 – Scenario setup……... 92
Figure 4.8: Speed Control & High Way Monitoring – Vehicle's Report Creation……. 98
Figure 4.9: Case Study 2: Remote car locating & sending a service / order message….. 99
Figure 4.10: Remote car locating & sending a Service / Function message – Messages Flow…… 109
Figure 4.11: Speed Control & High Way Monitoring – Messaging Algorithm…… 110
Figure 4.12: Study Case 2 - Scenario setup…… 113
Figure 4.13: Case Study 3: Suspect car instant termination…….….. 120
Figure 4.14: Police vehicle's VANET Device (PVD)…….……... 122
Figure 4.15: Suspect car instant termination – messages flow……... 136
Figure 4.16: Suspect car instant termination – messages flow – After getting the SVD Information……..….. 137

LIST OF ABREVIATIONS

Co_D	Coordinator Device
DSRC	Dedicated Short Range Communication
GF	A set of “Get Functions”
Mtc	Message type checker
OBU / OBE	On Board Unit / On Board Equipment
PAz	Packet Analyzer function
PCr	Packet Creator
PFr	Packet Forwarder
PSC	Propagating a Speed Code
RSU / RSE	Road Side Unit / Road Side Equipment
SF	A set of “Set Functions”.
SOP	Sub-Operation
SQL	SQL Query statement Creator
SVSR	Sending a Vehicle’s Status Report
V2I	Vehicle-To-Infrastructure communication
V2V	Vehicle-To-Vehicle communication
VVD	Vehicle’s VANET Device

CHAPTER ONE: INTRODUCTION

1.1 VANET

Vehicular Ad-Hoc Network (VANET) is a form of Mobile Ad-Hoc Network (wireless Network), originally used to provide safety & comfort for vehicle users. Currently it is being used to establish a dedicated short range of communication (DSRC) among nearby Vehicles (V2V Communications) and between vehicles and nearby fixed infrastructure equipment; Roadside equipment (V2I Communications). VANET is also used to warn drivers from any collision possibilities, road sign alarms, auto-payment at road tolls and parks. VANET can be usually found at Intelligent Transportation Systems (ITS).[SCHROTH C., 2006].

The VANET works ideally in an integrated environment which is shown in Figure 1.1.[COPS M., 2006].

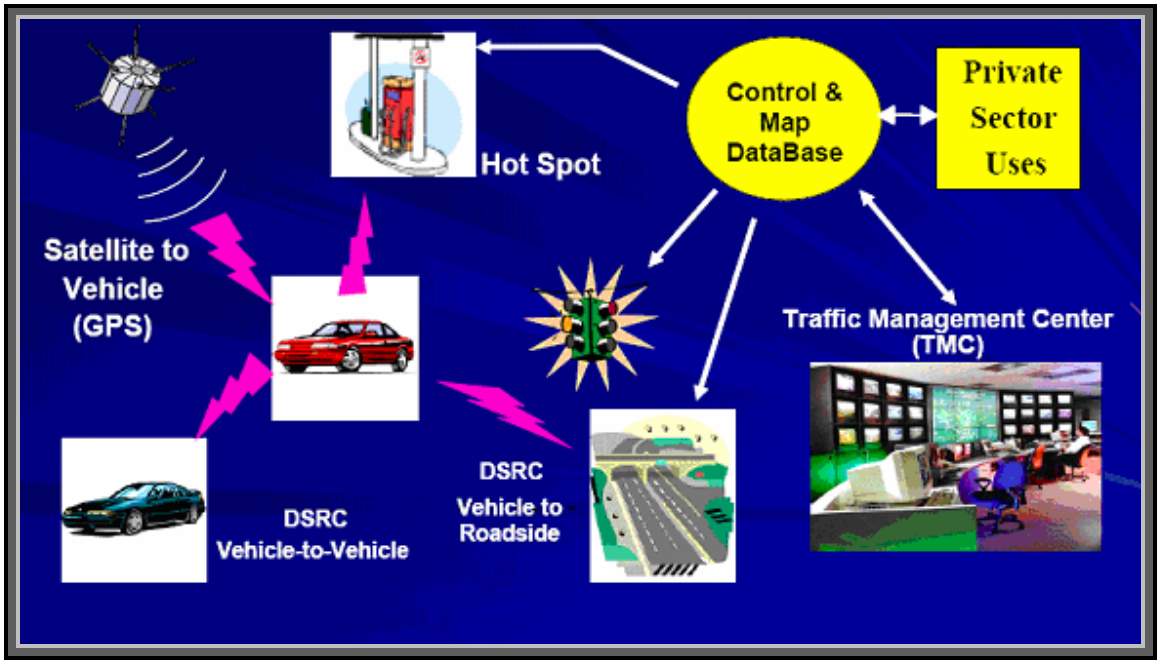

Figure 1.1: VANET Integrated Infrastructure

Ad-hoc Networks do not use centralized administration, and do not rely on any pre-established infrastructure; its nodes rely on each other to keep the network connected. Although, VANET is not a pure Ad hoc Network since it does rely on a fixed infrastructure when a V2I communication happens, it uses the DSRC technology to connect the vehicles with the existing infrastructure.

1.2 DSRC

5.9 GHz DSRC (Dedicated Short Range Communication) is a short to medium range communication service that supports both Public Safety and Private operations from roadside to vehicle and from one vehicle to another vehicles communication environment. DSRC is meant to be a complement to cellular communication by providing very high data transfer rates in circumstances where, minimizing latency in the communication link and isolating relatively small communication zones are important.[LEEARMSTRONG, 2008]

DSRC is a short range radio created to serve as a transportation specific technology. It can provide a half duplex connection between the participating nodes with a high bit rate of (6 Mbps) up to (27 Mbps) and a coverage area radius of up to 1000m. IEEE 802.11p will be based on the standard IEEE 1609 which is a higher layer standard it is also used as the groundwork for DSRC which defines enhancement to 802.11 which requires supporting ITS Applications. 802.11p project still in progress and the approved 802.11p amendment is scheduled to be published on December 2007.[IEEE GROUPER, 2008]

1.3 VANET Characteristics

The features of a vehicular ad hoc network are totally different compared to other mobile ad hoc networks. The unique properties of a VANET has pushed the researchers to make use of these characteristics to increase network performance. At the same time it can be considered as challenges. A VANET is fundamentally different [BALON N., (2006)] from other MANETs:

1. The VANET coverage area diameter is relatively small compared to other types.
2. Disintegration of the network into smaller segments often occurs. The reason behind the short life for the link in a VANET is because of the high speed mobility of the vehicles which might reach up to 200 km/h. In order to lengthen the life time of a link, we should increase the transmission power of the vehicles antennas, which on the other hand will decrease the throughput of the network.
3. A VANET rapidly changes into some predictable topology, because vehicle movements are limited by the road itself. The high mobility of vehicles cause the topology to change frequently and due to that the existence time of a communication link between two vehicles is short.
4. A VANET sparks up many unique security challenges.
5. Since VANET is using DSRC technology, it has low transmission latency, about 50ms when the vehicle's speed is at 120 Km/h.

1.4 VANET Communication promises

VANET communication promises a lot of services and easiness for the drivers, these promises resides within the following:

1- VANET promises safer roads
2- VANET promises more efficient driving
3- VANET promises more fun driving
4- VANET promises easier maintenance of the vehicles

1.5 DSRC – VANET Applications

Many researches were done to create applications based on the usage of DSRC. The VANET applications of DSRC are categorized into four classes: [BALON N., (2006)]

- **Vehicle-to-Vehicle** applications.
- **Vehicle-to-Infrastructure** applications
- **Vehicle-to-Home** applications.
- **Routing Based** applications.

Each of the above four application categories can be further categorized into safety and non-safety applications.

1.6 VANET Applications

There are many applications and others are still to be created for the Vehicular networks to satisfy all the VANET promises. These applications are categorized into:

1- **E-safety applications**
   (e.g.; turn left assistant, urgent situation vehicle approaching warning, vehicle safety examination, stolen vehicles tracing, rail accident warning, etc.)
2- **Traffic management applications**
   (e.g.; Highway merges assistant, electronic toll payment, hazardous material shipment, etc.)
3- **Enhanced driver comfort**
   (e.g.; Download or update road maps, instant messaging between vehicles, hot spot notification, parking spot locator, etc.)
4- **Maintenance**
   (e.g.; notice on safety recall, just in time repair notification, etc.)

1.7 Formal Specification Language

Formal Specification is a strict description of a system, describing the details needed by the system to perform its job properly. The formal specification language is the set of notations and rules used to write the formal specification for the system.

The main use of the formal specification language is to describe any system mathematically and then apply different scenarios on it to reveal any inconsistency, incompleteness, or ambiguity in the system's operation.

Formal methods are still hard to scale up for large systems. So the system is disintegrated into smaller sub-systems which interact with each other through interfaces to do the main system's job. The layer VANET Model has many devices which work individually. Each of them has its own interfaces to interact with other devices around. Each device is considered as the sub-systems.

Figure 1.2 shows a system which has been decomposed into three smaller Sub-Systems, A, B, and C. Each Sub-System has at least one interface which interacts with the surrounding Sub-Systems. The interface has two functions that control its operation, those are; MtC (Message Type Checker) and Pfr (Packet forwarder). The function of MtC is to be responsible for the incoming data stream whereas the Pfr is responsible for the outgoing data. For more details about both of MtC and Pfr, refer to chapter 3.

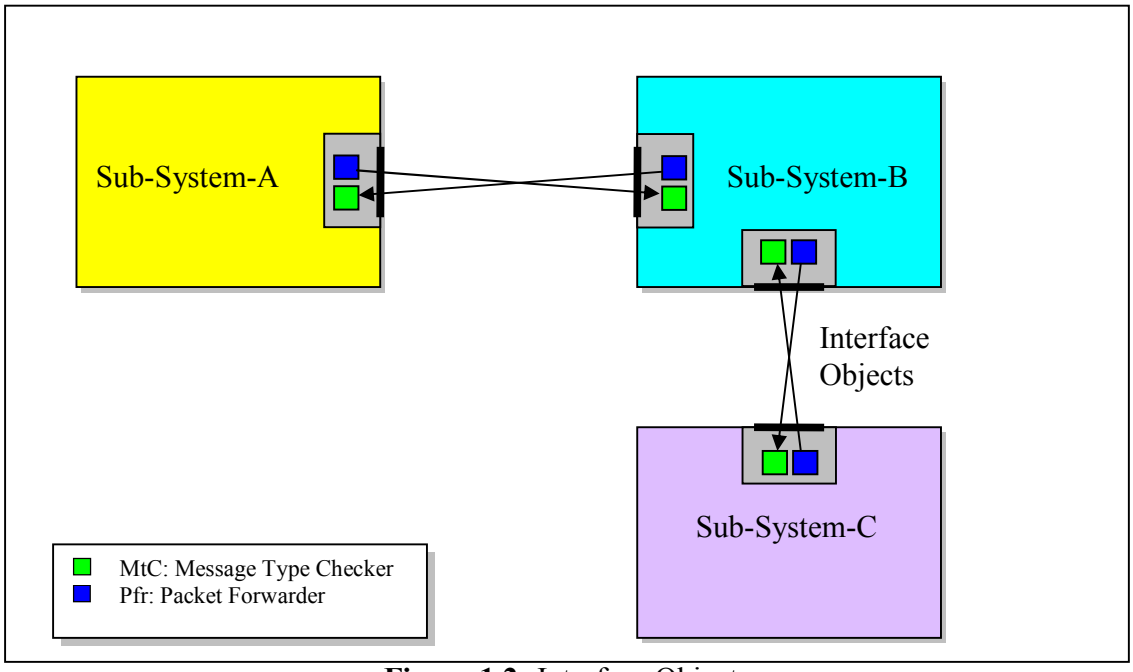

Figure 1.2: Interface Objects

The scopes of the formal methods are limited. They are not well suited to specify and analysis use interface and user interaction. However, our language does solve the user interaction problem as we will see later on this report.

The major benefits of formal methods are in reducing the number of faults in the systems. As a result, critical systems engineering is the main spot of formal methods applicability. The usage of formal methods is presumably cost-effective because high system failure costs can be avoided.

The main benefits of using the formal specification are; it obliges a comprehensive analysis of system's requirements, incompleteness and inconsistencies

can be discovered and resolved. It is true that there are larger costs to be spent up front and efforts are exhausted in developing the specification for a system. However, implementation and validation costs must be compacted as the specification process reduces errors and vagueness in the requirements.

1.8 Problem Statement

Three problems we have specified and we are going to focus on in this thesis, those are:

1- There is no common formal specification language for VANET researchers to use so they specify their systems.

2- There is no Mini Model of VANET Environment to be used by researchers as part of their test bed.

3- There is no suggested scheme to control and located vehicles remotely via VANET.

1.9 Objective of the Study

The main objective is creating a formal specification language that can be used for describing VANET aspects theoretically and specify by proving VANET's System and their applications. The second objective is creating a layered system that can be used to model VANET environment for case studies scenarios. Finally outline an application layer protocol on a layered system to deliver services to and access the remote mobile nodes.

1.10 Motivation

Car manufacturers are about to take a quantum leap in terms of enhancing driving safety but they are waiting for new technologies and applications to be created and be experimented before they could use with their products. Our proposed model of layered VANET system and the Application layer protocol can be an important as well as a unique part of that future leap. In the meantime, our proposed VANET specification language can be used by those going for work during that future period of time on related researches to present and prove the consistency and the completeness of their ideas and theories to reveal an ambiguity, incompleteness and inconsistency in them.

The main incentive behind creating a formal specification language and the layered model system for VANET environment is needed by the researchers who are going to work on VANET area during the future leap. Hence the inducement after lining out the application layer protocol is to draw a starter line for creating a common protocol which can be used on VANET environment and also to create more applications based on the protocol.

1.11 VANET challenges

VANET challenges can be listed as:

1- Devices Communication efficiency with High Mobility
2- Packet Delivery ratio
3- Routing reliability – Dynamically changing network Topology map
4- Dead-lock management
5- High traffic management
6- Participated devices Coverage Area & Energy
7- Communication Events Synchronization

1.12 Scope of the Study

VANET has many aspects that need to be plunged into and also to find the solutions. In order to do so, we need to have a specification language to describe the solutions. VANET aspects can be listed as:

1- Nodes Mobility (Node's movement direction, speed, …etc)
2- Nodes Visibility
3- Nodes Abilities
4- Security
    a. Authentication & key management
    b. Privacy
    c. Trust & Revocation
    d. Secure positioning
5- Messages Delivery/exchanging
6- Environment description (Nodes positions, environment limitations, …etc)
7- Data Dissemination
8- Routing

Our proposed specification language covered many of the VANET aspects (listed above) except some such as routing and security (Only the simple Authentication & key management have been covered).

Our proposed layered system contains N number of layers, which can be grouped into three groups:

- Coordination group
- Distribution group
- Host group

Each layer has its own features and devices to operate at. Our proposed system hires 25 different messages to implement its functions. (*See Chapter-3*)

1.13 Case Studies

We have implemented three case studies (*See Chapter-4*) in our system to show its abilities and at the same time, we highlighted that they would be specified using our specification language. These case studies are listed:

- Vehicles Speed control on highway
- Stolen car locating & remotely stop it.
- Fast ending for police car chasing.

1.14 Thesis Layout

This report consists of 6 chapters, the first two chapters explain some general concepts of VANET and Formal Specification Languages, and also tell the state of the art. Chapter 3 describes our methodology to design the proposed layered system Model and the application of layered protocol and illustrates our proposed specification language and how to make use of it. Chapter 4 describes mathematically (using our proposed specification language shown in chapter 3) three case studies. Chapter 5 discusses the validity and the consistency test for the three case studies and depicts three comparisons between our formal specification language and other languages. Chapter 6 concludes the whole report and indicates what hereafter work can be done.

CHAPTER 2: RELATED WORK

2.1 Literature Review

With the latest inventions in wireless technologies, automobile manufacturers are about to take an enormous step to enhance the driving safety and comfort by allowing vehicles to talk with each other along with the roadside equipment infrastructure explicitly Vehicular Ad-hoc Networks or VANET[FARKAS C. 2007]. In order that, many projects are in progress aiming for more safe highways.

2.1.1 Services through VANET

A project was conducted (September 2007) by three computer scientists at University of South Carolina, The title of the project was: "Application Level Protocol for Accident reconstruction in VANETs", which was aiming to investigate the possibilities of leveraging inter-vehicle communications within VANET framework. This is to analyze the crash data for accurate accident reconstruction, collecting data after an accident happens, which will help in solving out the problem but not saving the lives. The National science Foundation in South Carolina (NFS) sponsored the project which costs about $838,000 [FARKAS C. 2007].

Two French motorway companies; SAFESPOT and CVIS (Cooperative Vehicle-Infrastructure Systems) with different participants of the ITS business are involved in projects related to infrastructure/vehicle communication. SAFESPOT had started in February 2006 to develop a project which was mainly aimed to understand and assess, trough test in real condition. It was also to identify the potential of the cooperative approach in terms of transport safety improvement, which had addressed 12 problems that can be solved through this project. This project has cost about € 38 Million and is expected to be completed in four years time. Moreover, about 51 partners from 12 different European countries had supported this project.[FREMONT G., 2007]

The CVIS had started a project in February 2006. FP6 Integrated Project was aimed to develop and experiment new technologies which allowed road vehicles to communicate with other roadside infrastructure. Based on real-time road and traffic information, many novel applications can be produced. The consequence will increase road safety and efficiency, and reduce the environmental impact. Sixty-one partners from 12 countries had cooperated to work and sponsor this project which costed about € 41 Million. They had planned to finish it within 4 years time. [FREMONT G., 2007]

In [SONG H., 2008], some researchers have proposed a sensor-network-based security system for vehicles. This would detect unauthenticated movement and keep track of the stolen vehicles while alarming a near by base station by sending warning messages to the security office at that parking lot. However, it has some limitations especially with the extreme case of the none-existence of neighbors although a sensor has tried its maximum level of power.

2.1.2 Data Dissemination

In [BAUMANN R., 2004], a research project was done in 2004 titled as: "Engineering and simulation of mobile ad-hoc routing protocols for VANET on highways and in cities", which was aiming at creating two new broadcasting mechanisms for VANET networks. It was suggested to minimize the number of broadcasting messages and to get more stable routes: the Secure Ring Broadcasting (SRB) and the Directed Route Node Selection (DRNS). This will help to speed up the communication within VANET environment.

Also there was a master's project aiming to improve the broadcasting in VANET Environment, which was done in 2006 at University of Michigan, "*Increasing Broadcast Reliability in Vehicular Ad-Hoc Networks*". Basically they proposed a new technique to improve the reception rate of broadcast messages by a dynamic adjustment for the contention window size done by the VANET nodes. The size of adjustment is based on the number of successfully received packets per the last few seconds.[BALON N., (2006)]

There were many researches done on Information Dissemination within VANET environment. Some of them were very beneficial and successful, while others were not studied in detail. An example of Information Dissemination project is; "Optimizing Dissemination of Alarm Messages in Vehicular Ad-Hoc Networks (VANET)"; which was done in 2004 at the University of Avignon – France. In that project, mainly they proposed a scheme for alarm messages dissemination of accidents to warn other vehicles about the accident in a more efficient way by restricting rebroadcast to only special nodes, named "relays".[ABDERRAHIM B., 2004]

2.1.3 VANET Modeling & Simulation

In [SOMMER C., et.al, 2008], some researchers have created an integrated tool for graphical modeling. A bidirectional coupling of network simulation, road traffic micro simulation, and also a comprehensive library that can be used for communication networks (using OMNET++ simulator).

2.1.4 VANET Communication Enhancement

Vehicular networks are highly mobile and often disconnected therefore the multi-hop data delivery in VANET environment is so complex. Hence many researchers targeted this complexity to solve and find efficient suggestions to jump over those complexities. In [ZHAO J., 2006], a group of researchers proposed a bunch of data delivery protocols which are outperforming the existing solutions (at that time) in terms of packet delivery ratio, data packet delay, and protocol overhead.

Due to the vehicle's high speed and the limitation of the access point's coverage area, getting an IP address from a DHCP (Dynamic Host configuration Protocol) might not be guaranteed. This might consume up to 100 percent of the vehicle's available connection time. So In [ARNOLD T., 2008], an IP address passing protocol was suggested to reduce the overhead of obtaining an IP address to under one-tenth of a second (Without modifying either DHCP or Address Passing (AP) software).

In [BYCHKOVSKY V., 2006], a group of researchers targeting the Internet access service from vehicles, tried to prove that unplanned network service can provide reasonable performance to network clients moving in vehicles at vehicular speeds. Finally they found some measurement results which can improve transport protocols in VANET networks.

In [ZHAO J., 2008], a scheme was proposed to extend the service range of roadside access points, which allow drive-thru vehicles to maintain high throughput within an extended coverage range, by using a vehicle-to-vehicle reply scheme.

From the personal point of view, Vehicle-To-Vehicle based services are inefficient especially with standalone vehicle scenarios. For example, a single vehicle on a highway would like to send a message to a faraway vehicle while there are no vehicles between them to relay and deliver the message then they will never be able to contact each other. While Vehicle-To-Infrastructure based services can be more efficient because the services are guaranteed to be provided by the Road Side Equipments (RSEs) all the way long. So that we need to develop and improve protocols and services for the V2I communication based applications. However, we still need the V2V connections to provide either different types of services or to serve as a redundancy or a supportive solution for the infrastructure.

2.2 VANET Security Issues

VANET environment is very vulnerable to hacking attempts, which will result in disaster if it happened. For that reason; robust security architecture should be created to protect VANET environment and its members from such disasters. With the intention of creating such robust architecture, it should cover the list of requirements below:

- Vehicles Privacy
- V2V & V2I Authentication
- Trust and revocation

2.3 State of the Art

VANET is a fast-moving research area which attracts the intention of diverse people from both academia and industrial background so as to make different workshops and conferences in the direction of promoting communication amid them for advance further research interests and actions to enable new transportation services and products, e.g., advanced traffic management, vehicle control, safety control, and networking and information services for users on the road.

Moreover, many of major automotive companies have also explicated an attention to sponsor or to participate in VANET researches, such as:

- Routing protocols for active safety in VANET.
- Challenges of V2V and V2I wireless communication.
- Wireless technology usage within cars.
- Propagation issues.
- Security issues in VANET and trustworthy networking
- Service applications Infotainment, content distribution, internet access, etc.
- Traffic management, vehicle control, and safety related applications for ITS systems.
- Wireless Connection Quality-Of-Service in ITS systems.
- Vehicular network performance modeling and analysis, network flow & congestion control, Architecture & communication Protocols, Medium Access Control and Routing Protocols.
- Mobility management and intersystem handovers
- Simulation models and test-beds for VANET
- Implementation and field tests of VANET Systems
- Potential modifications needed to improve the DSRC standard.
- Incentive, cooperation, and reputation systems.

2.4 Related Work Summary

According to our literature review, none of the researchers have created a common specification language for VANET. If we have to have a look at [BLUM J., 2004], [FARKAS C. 2007], [SOMMER C., 2008], [ZHAO J., 2008], [GUEMARI L., 2001], [BALON N., (2006)], [BAUMANN R., 2004], [SCHROTH C., 2006], and [LARSSON T., 1998], all of them have worked on VANET Projects or related to VANET, but none of them has specified the system or the application created. Instead of that, they just jumped into the simulation part by using couple of simulators such as Ns2, OMNeT++, or OPNeT, with the help of programming languages such as C++. This means they have almost involved in the implementation phase of a system development, and this what we (Researchers) would like to avoid because if an error occurred then fixing it would cost much more than what it does when we discover the error at the formal specification phase. The reason for not using a formal specification language by the researchers is the inexistence of any formal specification language precisely created to represent VANET systems. Many formal specification languages have been created for other purposes rather than specifying VANET systems. For example, LOTOS (Language of Temporal Ordering Specifications), CASL (Common Algebraic Specification Language), Larch (a set of languages), Z Notations (Zermelo-Frankel Specification language) and many others, all of those languages are for specifying different types of computer systems, but none of them is able to specify VANET systems, and that is because of the unique features for VANET systems. Refer to section 5.3 for more detailed comparison between our language and more than other 12 formal specification languages.

In the next chapter we are going to illustrate the three methodologies we followed to achieve our three objectives.

CHAPTER 3: METHODOLOGY

In this section, we are going to demonstrate the methodology we have used to create our Formal Specification Language. Then we will show how to use and how effective is the language that we have created by showing some VANET system examples which we have created to solve the message delivery via VANET by referring to its abilities in high speed driving & hard highway monitoring problems, stolen car locating problem, and long police car chasing problem.

3.1 Proposed Used Basic symbols / Notation

Our specification language consists of symbols and notations which are categorized into six groups. Each group can be used to show or present different aspect; Device Ability (DA), Device Movement (DM), Messages Exchangeability (ME), Device Internal Behavior (IB) logic, Security (Sc), and Design & Configuration (D&C) rules. In the next subsections, we will describe each category and its symbols.

3.1.1 Device Ability

The following symbols can be used to express the abilities of a device:

A ➔ B A can reach B, while B can NOT.

A ⇔ B both of A & B can reach each other.

A Ξ B A controls B – A has the jurisdiction over B

A θ B A sees B

3.1.2 Device Movement

The following symbols can be used to illustrate the movement of the mobile nodes with the system (To show scenarios):

A ► B A moves to/reaches/towards area B

A ◄ B A moves away from B

A ▲ B A speeds up within area B

A ▼ B A slow down within area B

A ⌂ B A stops within B

A ↺ A U-turns

A ╬ A reaches an intersection

A ╦ Reaches a T-Blocked road

A ↴ A Turns-Right/South

A ↥ A Turns- Left/North

A ~ (x,y) A is moving from Lane-x to Lane-y

3.1.3 Messages exchangeability

The following notations can be used to show the message flow between the system's devices:

A ~x> B A sends a message type-x to B

A ~~> B A forwards the incoming message into B

A ~x○ B A Propagates a message type-x to layer-B

A ○x~ B From area A, many devices are sending message type-x to Device B

A ○~~ B From area A, many devices are sending different messages of different types to Device B

A <x~ B A receives back a reply message type-x from B

Mtx^{-1} Waiting for a reply message (Type-x Message)

To define the message of a system:

Mt*X*: *Source*, *Destination*, *Message_Flow_Type* , *<<message_Fields>>*

Where:

Source: The source device that creates this kind of messages.

Destination: The destination device that receives and reacts to such message type.

Message_Flow_Type: The packet type as attributes (e.g.; Multicast, Unicast, Broadcast, Multi-hop, Single hope)

Message_Fields: all the data fields of that message type in an ordered sequence.

In case there is more than one probability of source or destination devices, then we list the devices in one field by separating them by or sign " | ", as following:

Mt*X*: *Source-1* | *source-2* |…. | *Source-n*, *Destination-1* | … *Destination-n*, *Message_Flow_Type* , *<<message_Fields>>*

Where $0 < n < \infty$.

To express a message at different layers:

<X> An Application layer packet of type X.

[X] A Network layer packet of type X.

For any block of algorithm can be presented as follow:

Label: *<Label>*

<Body>

Example:

Label: Block-A

Statement 1

Statement 2

…etc.

To call any block, we can just write the key word **Lbl:** followed by the label name.

Example:

Lbl: Block-A

3.1.4 Device Internal behavior logic

The following notations and expressions can be used to show how exactly the internal behavior for the devices is:

$F_1 \nearrow F_2$ Function-1 Sparks / Calls Function-2

$F_1(\_, X) \approx> O(Y), (X)$ Function 1 forwards value X as an input to O (where O is a Device or a Function) to get Y as the output from O.

SQL(a, b, c,…) ≈> X insert a record into table X

SQL(a, b, c,…) <≈ X Get a record values (a,b,c,…) by SQL Query from table X on the Database server.

Gf (*Output*),(*input*) *[<≈ souce]*

Gf (x, y, z,...),() <≈ A Or Gf (x, y, z,...) <≈ A Means: Get (x,y,z,…) by Get_function from A. Where A can be a register or a table.

Both of the following two lines mean: Get n values by Get_function from the user interface of device X, then assign these values into (x,y,z,…), respectively. Where n is a positive integer number. No input.

Gf (x, y, z,...),() <≈ X_Int

Or

Gf (x, y, z,...) <≈ X_Int

Meanwhile the following line means: Get the output of the Get_function of table Y when sending the values (a, b, c,…), and assign the result to variable X.

Gf (X), (a, b, c,…) <≈ Y

The following is a list of some basic comparative operators, mathematical and logic operations:

A Vs. B Compare A with B

< Less than

> Greater than

= Equal

! Not

& And

| Or

(A & B & …) → C, D if (A & B) then C & D

(A & B & …) !→ C if Not (A & B & …) then C

3.1.5 Security

The following symbols concern the systems security but have a note that we did not cover all the security aspects, these notations can be used to show a symbol authentication between devices:

A ₸ B A trusts B without authentication or already authenticated

A <₸> B A and B trust each other without authentication or already authenticated

A ₸ B^k A trusts B with authentication Key (k)

A <₸> B^k A trusts B with authentication Key (k), while B trusts A without a Key.

A^g <₸> B^k A trusts B with authentication Key (k) and B trusts A with authentication Key (g).

A !₸ B A Does NOT trust B

A !<₸> B A and B do not trust each other.

A <!₸> B A does not trust B, while B trusts A.

A <₸!> B A trusts B, while B does not trust A.

3.1.6 Validity Test

The formal specification language will be used to prove any system or scenario is valid. This validity test will make sure that all operations are valid by proving that all of their steps are valid. See the following example figure:

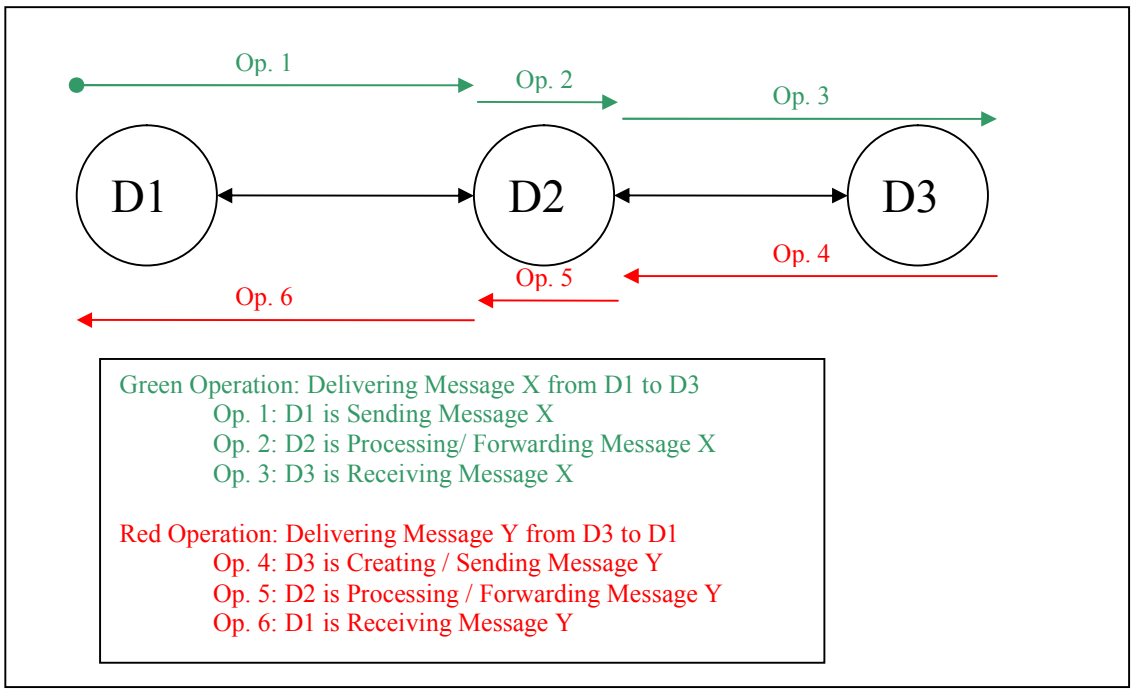

Figure 3.1: Validity / Consistency Test Example

We have come out with a system that has three devices (D1, D2, D3). The main aim of this system is sending a signal message X from D1 and receiving a reply message (Message Y) for that signal. For validating the system, we need to validate its main operation. This can be done by starting the validation from the least operations going up. In other words, a whole system's operation can be divided into smaller groups of sub-operations; each group has a set of sequenced mini operations. Each of these mini operations can be divided into smaller units depending on the system's complexity.

In Figure 3.1, there are two sub-operations; Green Operation and Red Operation, which will composite the main operation. These two operations themselves are compounded by more sub-operations (Op.1 through Op.6). These are assumed to be the lowest level operations to be validated.

According to logic, the validity rule is; "*A **set of sentences are considered valid, if and only if there is no line in the system's operation truth table having all of its statements are True while the conclusion is False.***". Thus we have to go through the entire truth table and check all the lines whether there is or not a line that breaks the validity rule. Refer Table 3.1, the first line; the Op.1, Op.2, and Op.3 are True and their conclusions (Message X was delivered) was true as well, then the Green operation is valid. In the meantime, the Red Operation is valid as well for the same reason. Finally, since both of the Operations are valid, then the whole system is valid, but not quite yet, we have to check the rest of the lines in the truth table to make sure there is no such line that breaks the validity rule. See the following sentences to understand how each lines of the truth table were constructed:

Green Operation $Validity_x$ = !(!*Msg_X_Delivered* **&** *Op.1* **&** *Op.2* **&** *Op.3)*

Red Operation $Validity_x$ = !(!*Msg_Y_Delivered* **&** *Op.4* **&** *Op.5* **&** *Op.6)*

Whole System $validity_x$= $Green\_Operation\_Validity_x$ **&** $Red\_Operation\_Validity_x$

Where x is an integer, $0 < x < \infty$, the number of the case (One line in the truth table).

Table 3.1: Validity Truth Table (NOT ALL CASES)

Op.1	Op.2	Op.3	Msg X Delivered	Green Op. Validity	Op.4	Op.5	Op.6	Msg Y Delivered	Red Op. Validity	Validity
T	T	T	T	T	T	T	T	T	T	T
T	F	F	F	T	F	F	F	F	T	T
T	T	T	T	T	T	T	T	F	F	F

For the second line in Table 3.1, it cannot be considered as the validity measurement because there is at least one false statement (Op.2 is False and caused the faultiness for the rest of the sentences). For third line in the truth table, the Green Operations are valid because all of the sentences and the conclusion are True, while for the Red Operation, all the statements are true but the conclusion for some reason is false, that would cause the Invalidity for the whole system.

A whole system operation would be divided into levels of groups as shown in Figure 3.2 below. Each level is a group of its units, e.g. the lowest level represents groups of steps, and the second lowest level has groups of algorithms. Each level's unit is a set of the units of the lower level. For example, Algorithm is a set of steps, and a sub-operation is a set of algorithms and so on. Both of the validity and the consistency tests start from the least level going up.

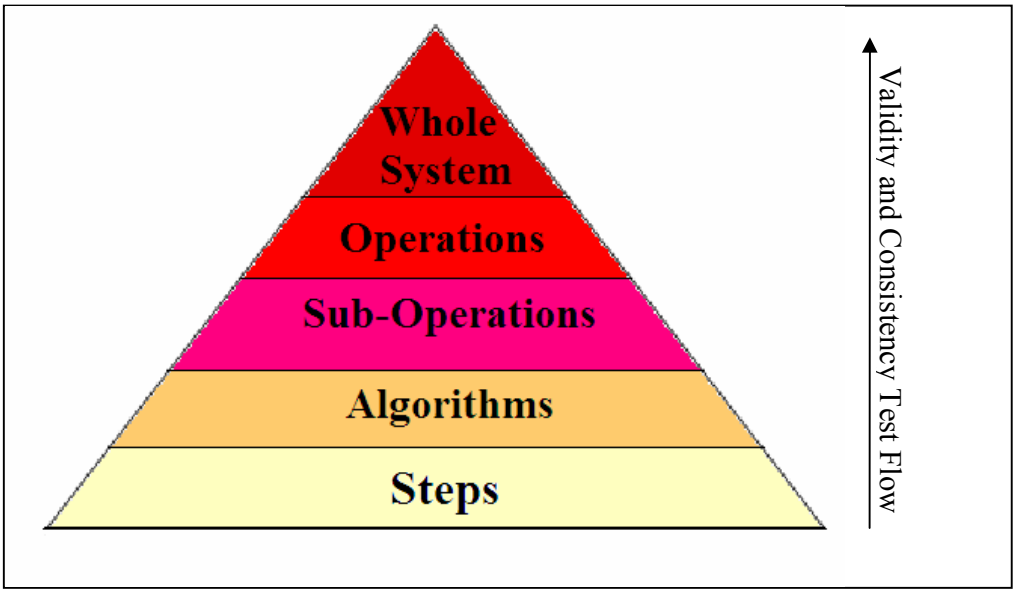

Figure 3.2: System operational grouping

The following set of equations is applicable on each pair of levels in Figure 3.2, when each level's unit is a WHOLE to the lower level:

OpV_{a1} = !(Low-Op.1 & Low-Op.2 & … & Low-Op.$_n$ & !Conclusion)

Where:

OpV_{a1}: Operation_a (e.g. Green Operation) Validity with the first set of parameters, where OpVa is a set of n of the Lower level operations (Low-Op), where n is a positive integer number $0 < n < \infty$.

OpV_{am}: Operation_a (e.g. Green Operation) Validity with the n'th set of parameters, where m is a positive integer number $0 < m < \infty$.

Whole System Validity$_1$ (WSV$_1$)= OpV$_{a1}$ & OpV$_{b1}$ & … & OpV$_{n1}$

Where:

Whole System Validity$_1$: the whole system validity (at a level) with the first set of the parameters.

The final result for the validity test is:

WSV$_T$ = WSV$_1$ & WSV$_2$ & … & WSV$_n$

Where:

WSV$_T$: The final result for the whole system validity, (T) for Total.

n: is a positive integer number, $0 < n < \infty$.

3.1.7 Consistency Test

In order to prove that any system or scenario is consistent then we have to make sure that there is at least one full line in the system truth table that all of its operations and the conclusion are true.

According to logic, the consistency rule is; "***A set of sentences are considered consistent, if and only if there is at least one line in the system's operation truth table having all of its statements and the conclusion True.***"

In Table 3.2, we can see the first line satisfies the consistency rule then we can conclude that the whole system is consistent.

Table 3.2: Consistency Truth Table (NOT ALL CASES)

Op.1	Op.2	Op.3	Msg X Delivered	Green Op. Consistency	Op.4	Op.5	Op.6	Msg Y Delivered	Red Op. Consistency	Consistency
T	T	T	T	T	T	T	T	T	T	T
T	F	F	F	-	F	F	F	F	-	-
T	T	T	T	T	T	T	T	F	F	-

If we have a look to at the third line, we find all the operations are True just like the first line but the conclusion is False, this means there are two ways for the system to implement and to achieve the same goal at two different sets of parameters. The system passed the consistency test with the first set of the parameters but failed with the seconds set. To find the consistency of a Whole-Operation, we have to make sure that all the sub-Operations are Consistent. See the following equation:

CT_x = Op.1 & Op.2 & … & $Op._n$ & Conclusion

Where:

CT_X: Consistency Test for the Whole-Operation X

There are some systems which have both compulsory and optional operations. Therefore the consistency test should be repeated at least twice with two different cases to make sure that the system is consistent with such a scenario. The tests will be applied as the following:

- First time we apply it on the system without considering the optional operation groups.
- Secondly, we apply the test on the system with considering the first group of the optional operation.
- The whole system consistency test will be repeated depends on the number of the Optional operations groups, see the following:

Number of the CTs = Number of the Optional Operations Groups + 1

Where the value (one) added refers to the first test when we apply the test on the system (Compulsory Operations only) without considering any of the optional operations.

Now the consistency test final result will be found from:

Whole system consistency = CT_1 | CT_2 | … | CT_n

Where:

CT_1: Consistency Test for the system with the first set of parameters

n: is a positive integer number, $0 < n < \infty$.

And the consistency probability (robustness) can be found by:

Consistency Probability = (Number of Successful Tests / Total Number of the Tests) * 100 %

3.1.8 Design and Configuration

In order to show the design of a system and set its devices configuration, first, we need to show the architecture entities and the internal relations between all the entities of the participated devices. Then explain each of them, by showing the setup of the whole system (how entities can be related/connected to each other). Finally we will display the configuration of the devices. For that purpose we can use the following syntaxes:

Let A = Device-a

A = {set of components}

A set of components can be any set of hardware components, functions, and/or properties.

Or if we want to talk about a group of devices or devices clustering, then:

Let A_a = Group (a) of Device-A

To set the properties of device A_x values:

A_x_Property: < ***Propert Value*** >

(e.g.; A_x_ID: Rsq56)

To show the ports of each device, we can present them as a set of ports:

Let SP = {Set of Ports}

(e.g.; SP= {P_1, P_2, …})

To set the configuration of each port:

P_x_Property: ***< Property Value >***

e.g.; P_1_IP: *< Ip address >*

P_1_MAC: *< MAC address >*

To show the functions of each device, we can present them as a set of functions:

Let SF = {Set Of Functions}

e.g; SF = {F_1, F_2,…}

Then show a brief explanation about each of the functions listed in the set, we can use one of the two methods; the first is by using a plain text to describe the function:

Let F_1: <Plain Text Description>

e.g.; F_{Mtc}: Checks the type of the incoming message & decides the destiny of the incoming message.

Or we can use the second method, which tells what kind of inputs should be given to the function, and what kind of outputs we expect of that function:

Let F_1: I(List of Input Parameters), O(List of Outputs)

Where 'I' stands for "Input"

And 'O' stands for "Output"

e.g.; F_{Mtc}:I(incoming message), O(incoming Message's Data, next function code)

Finally, to write the body of any function, all we have to do is follow the function's header through the steps which combine the function's body included within two brackets. See bellow:

Let F_1: I(List of Input Parameters), O(List of Outputs)

{

Step 1

Step 2

…. Step n

}

Where: F1 is a function that has n of steps, $0 < n < \infty$.

3.1.9 Environment description

We can use these symbols to describe the scenario's roads directions, their status (Crowded, Terminated,…etc.) and the position of the vehicles on the road (over which lane the vehicle is, or to which direction it will go). See Figure 3.3.

Let $\mathbf{R_N}$: Northern Road (towards the North)

$\mathbf{R_S}$: Southern Road (towards the South)

$\mathbf{R_E}$: Eastern Road (towards the East)

$\mathbf{R_W}$: Western Road (towards the West)

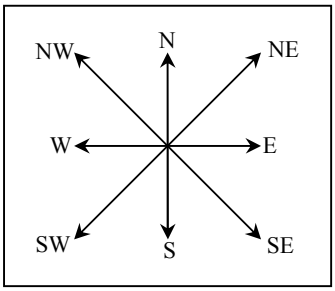

Figure 3.3: Standards Directions

We can combine two directional symbols to show the rest of the directions, as bellow:

$\mathbf{R_{SE}}$: South-East Road

$\mathbf{R_{NE}}$: North-East Road

$\mathbf{R_{SW}}$: South-West Road

$\mathbf{R_{NW}}$: North-West Road

To specify the instant lane the vehicle is over, we can specify it by adding the Symbol (Lx) to the Road direction, see bellow:

Let $\mathbf{R_{NL1}}$: Northern Road – LANE 1

$\mathbf{R_{NL2}}$: Northern Road – LANE 2

… and So on.

Moreover for the right- hand side driver's road system, lanes counting starts from the left side of the road. e.g.; For Malaysia, lane 1 is the one on the left of the road, while Lane n (Where n is a number $0 < n < \infty$) is the one on the right side of the road.

For the left-hand side driver's road systems, lanes counting starts from the right side of the road. e.g.; For Iraq, lane 1 is the one on the right side of the road, while Lane n (where n is a number $0 < n < \infty$) is the one on the left side of the road.

To specify the features for road, we just add the feature keyword to the Road name, as bellow:

Let R_N_feature-keyword = <*Value-Of-The-Feature*>

Example:

Let R_N_Allowed_Speed = 70 Km/h

Let R_{SE}_Crowded = True

Let R_{NW}_Terminated = False

To tell on which road the vehicle is, we use:

A --: Rx Means: Vehicle A is driving on Road Rx. The symbol (--) was derived from the shape of the street lining. And the second part (:) was derived from Traffic light stops.

And to tell how big the coverage area of the device is, we use:

Let A Ø <*m*> Means: The Coverage radius of device-A is m, where m is a positive integer number represents the radius in Meters.

To tell how far is the device from another, we use:

Let A ○-○ B: <*m*> The distance between device-A and device-B is m, where m is a positive integer number represents the distance in Meters.

To tell the Three-Dimensional position of an object or another device with reference to a device, we use:

Let A –xyz B: <#x> <#y> <#z> with reference to Device-A, Device/Object-B is located at position <#x> <#y> <#z>, where #x, #y, #z are three numbers refer to the Axis-position.

3.1.10 Connections description

To illustrate the type of connection between two devices, we use:

A >--< B: <*Value*>

Where *Value* is the connection type (e.g., Wired, Wireless, Wi-Fi, …etc.)

We can use one of the many options given to show the connection between devices which is then separated by Comma:

B >--< A: <*Value_1*>, <*Value_2*>,…, or <*Value_n*>

Where: $0 < n < \infty$.

We might need to tell about the status of the connection at a specific moment (e.g.; Active, Inactive, Listening,…etc.) then we use this sentence:

A >--< B_Status: <*Value_1*>

For the description of connection at the system definition we might want to tell the range of status for values that can be taken by the connection which is being used:

A >--< B_RStatus: <*Value_1*>, <*Value_2*>,…, or <*Value_n*>
Where: $0 < n < \infty$.

For any other property of the connection that we want to illustrate, we can follow this sentence:

A >--< B_<*Property_Keyword*>: <*Value*>

Example:
A >--< B_<*Bandwidth* >*:* 1 Mb

In the next sections we are going to illustrate the details of each category and its combined notations by showing how to use them. In order to do that, we will show some examples depending on a layered system model which we designed to solve the message delivery among the participated devices within a VANET environment. Therefore, the first next section we will use to illustrate the proposed layered system model. Then we carry out with the details of the categories.

3.2 Layered System Model for VANET Environment & Application Layer Protocol

In this section, we illustrate our proposed layered system and formally specify the messages exchanged within its algorithms using our own logic notations (Refer Section 3.1).

Table 3.3: Layered System Specific functions description

Mtc	Is a function that checks the Message type (Mt) & decides the destiny of the incoming message.
PAz	Is the incoming packet analyzer function, it analyses the packet according to the packet type.
SF	Is a set of "Set functions".
GF	Is a set of "Get functions".
SQL	Is the function that creates SQL statements, their type (Query or Setting type) depends on the Mtc function's output.
PFr	Is a function that is responsible of forwarding packets to ports.
PCr	Is a function used to create Packets, their type (Unicast / Broadcast, speed code or searching packets) depends also on the output of the Mtc function.

Our layered model was derived from the Object Oriented Programming (OOP) Idea when each class has many objects those would get the functions and the features from that parent class in addition to their unique functions and features. So we have mapped the OOP idea to fit the VANET environment and create a Model that has many layers (each layer just like a Class), each has some features and functions those to inherit to the devices (objects) operate at that specific layer. Each device has functions and interfaces to communicate and interact among the devices. See [BOOCH G., 1994].

Our system has N number of layers but for the example purpose we are showing three layers with four devices we have specified, see Figure 3.4, the three layers are.

1- Layer one: Co-ordination layer (L1)
2- Layer two: Distribution Layer (L2)
3- Layer three: Host Layer (L3)

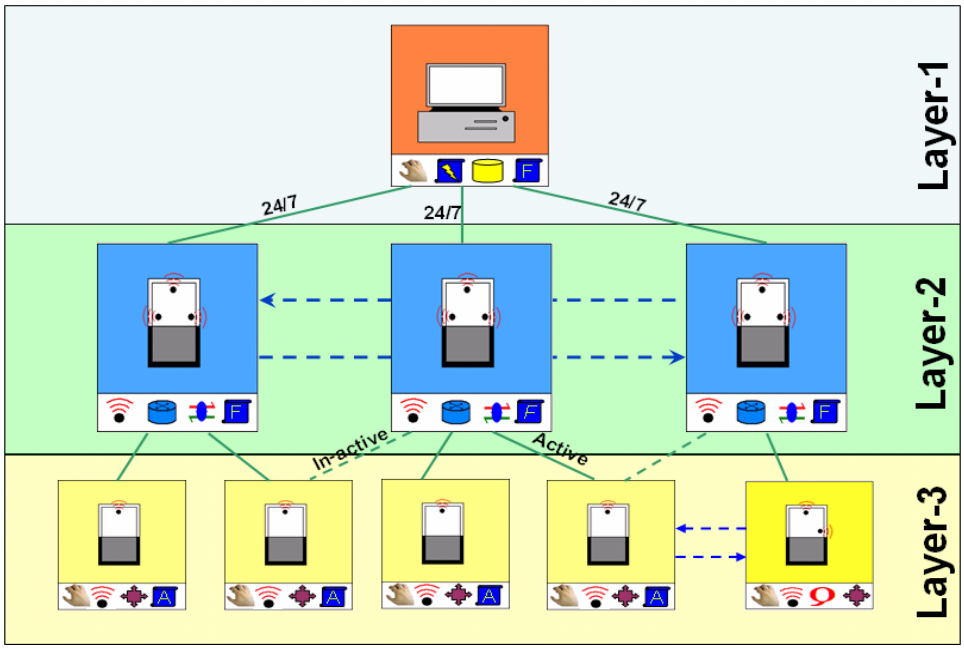

Figure 3.4 – Our proposed Layered system

3.2.1 System Devices

Each layer has its own n number of devices to work in. Each of those devices has its own functions, tables, number of ports, variables (registers that keep the configuration in). For further description of the device, we can list its components as a set, and then describe each component separately as a sub-set of other components and so on. See the following device descriptions:

1- Layer 1 Devices (α):

Using our Specification language words set to describe the layer α:

Let $\alpha = \{\alpha_A, \alpha_B, \ldots, \alpha_n\}$ // Layer- α has n number of devices

Where:

α_A: L1 Device-A

α_B: L1 Device-B

n: a number, where $0 < n < \infty$

Now to describe each device of layer-α that has its own components, we have to use a set of rules by listing its components:

Let $\alpha_A = \{ \alpha_A F, \alpha_A T, \alpha_A P\}$ // Leyer α Device-A has three major components

Where:

$\alpha_A F$: L1 Device-A Functions

$\alpha_A T$: L1 Device-A Tables

$\alpha_A P$: L1 Device-A Ports

Finally, we have to describe each component in each device by saying:

And let:

$\alpha_A F = \{ \alpha_{Mtc}, \alpha_{Paz}, \alpha_{Sf}, \alpha_{Gf}, \alpha_{SQL}, \alpha_{Pcr}, \alpha_{Pfr},$ ***** MORE FUNCTIONS*****$\}$

// the above sentence lists the functions set of the α_A device. The list might be extended into a longer one depending on the device design (This is what ***** MORE FUNCTIONS***** refers to). See Table 3.1 to know the duty of each of the functions in the list.

$\alpha_A T = \{\alpha_{Spc}[a][b], \alpha_{SMT}[c][d], \alpha_{VT}[e][f], \alpha_{SCT}[g][h], \alpha_{ToR}[i][j]\}$

// the above sentence lists the tables set of the α_A device.

$\alpha_A P = \{\alpha_{P1}, \alpha_{P2}\}$ // Tells the set of ports in device the α_A.

Finally, we combine all of the sets to present the device as one single set as shown below:

$\alpha_A = \{ \alpha_{Mtc}, \alpha_{Paz}, \alpha_{Sf}, \alpha_{Gf}, \alpha_{SQL}, \alpha_{Pcr}, \alpha_{Pfr}, \alpha_{Spc}[a][b], \alpha_{SMT}[c][d], \alpha_{VT}[e][f], \alpha_{SCT}[g][h], \alpha_{ToR}[i][j], \alpha_{P1}, \alpha_{P2}\}$

2- Layer 2 Devices (β):

We repeat the same procedure with layer-β:

Let $\beta = \{\beta_{A,} \beta_{B,} \ldots, \beta_n\}$ // Layer-β has n number of devices

Where:

β_A: L2 Device-A

β_B: L2 Device-B

n: a number, where $0 < n < \infty$

Let $\beta_A = \{ \beta_A F, \beta_A T, \beta_A V, \beta_A P\}$

Where:

β_A: L2 Device-A

$\beta_A F$: L2 Device-A Functions

$\beta_A T$: L2 Device-A Tables

$\beta_A V$: L2 Device-A Variables

$\beta_A P$: L2 Device-A Ports

And Let:

$\beta_A F = \{ \beta_{Mtc}, \beta_{Paz}, \beta_{Sf}, \beta_{Gf}, \beta_{Pcr}, \beta_{Pfr},$ ***** MORE FUNCTIONS*****$\}$

// The above sentence lists the functions set of the β_A device. The list might be extended into a longer one depending on the device design (This is what ***** MORE FUNCTIONS***** refers to). Refer Table 3.1 to know the duty of each of the functions in the list.

$\beta_A T = \{\beta_{VT}[e][f], \beta_{SCT}[g][h]\}$

// the above sentence lists the tables set of the β_A device.

$\beta_A V = \{\beta_{Spc}, \beta_{Rsq}\}$

// the above sentence lists the registers set of the β_A device. These registers keep the device configuration values in it.

$\beta_A P = \{\beta_{P1}, \beta_{P2}, \beta_{P3}\}$ // Display the set of ports in device the β_A.

By combining all the previous sets together in one single set we get:

$\beta_A = \{ \beta_{Mtc}, \beta_{Paz}, \beta_{Sf}, \beta_{Gf}, \beta_{Pcr}, \beta_{Pfr}, \beta_{VT}[e][f], \beta_{SCT}[g][h], \beta_{Spc}, \beta_{Rsq}, \beta_{P1}, \beta_{P2}, \beta_{P3}\}$

3- Layer 3 Devices (Γ):

Using our Specification language words set to describe the layer Γ:

Let $\Gamma = \{ \Gamma_A, \Gamma_B, \ldots, \Gamma_n\}$ // Layer- Γ has n of number devices

Where:

Γ_A: L1 Device-A

Γ_B: L1 Device-B

n: a number, where $0 < n < \infty$

In layer-Γ, we got two devices, so we present them separately:

For device-A of layer3 (Γ_A):

Let $\Gamma_A = \{ \Gamma_A F, \Gamma_A T, \Gamma_A P\}$

Where:

Γ_A: L3 Device-A

$\Gamma_A F$: L3 Device-A Functions

$\Gamma_A T$: L3 Device-A Tables

$\Gamma_A P$: L3 Device-A Ports

And Let:

$\Gamma_A F = \{ \Gamma_{Mtc}, \Gamma_{Paz}, \Gamma_{Sf}, \Gamma_{Gf}, \Gamma_{Pcr}, \Gamma_{Pfr},$ ***** MORE FUNCTIONS*****$\}$

// the above sentence lists the functions set of the Γ_A device. The list might be extended into a longer one depending on the device design (This is what ***

MORE FUNCTIONS* refers to). See Table 3.1 to know the duty of each of the functions in the list.

$\Gamma_A T = \{\Gamma_{RT}[e][f], \Gamma_{ST}[g][h]\}$

// the above sentence lists the tables set of the Γ_A device.

$\Gamma_A P = \{\Gamma_{P1}, \Gamma_{P2}\}$ } // Tells the set of ports in device the Γ_A.

Then:

$\Gamma_A = \{ \Gamma_{Mtc}, \Gamma_{Paz}, \Gamma_{Sf}, \Gamma_{Gf}, \Gamma_{Pcr}, \Gamma_{Pfr}, \Gamma_{RT}[e][f], \Gamma_{ST}[g][h], \Gamma_{P1}, \Gamma_{P2}\}$

For device-B of layer3:

Let $\Gamma_B = \{ \Gamma_B F, \Gamma_B T, \Gamma_B V, \Gamma_B P\}$

Where:

Γ_B: L3 Device-B

$\Gamma_B F$: L3 Device-B Functions

$\Gamma_B T$: L3 Device-B Tables

$\Gamma_B V$: L3 Device-B Variables

$\Gamma_B P$: L3 Device-B Ports

And Let:

$\Gamma_B F = \{\Gamma_{Mtc}, \Gamma_{Paz}, \Gamma_{Sf}, \Gamma_{Gf}, \Gamma_{Pcr}, \Gamma_{Pfr},$ *** **MORE FUNCTIONS*****$\}$

// the above sentence lists the functions set of the Γ_B device. The list might be extended into a longer one depending on the device design (This is what *** **MORE FUNCTIONS*** refers to). See Table 3.1 to know the duty of each of the functions in the list.

$\Gamma_B T = \{\Gamma_{RT}[e][f]\}$

// the above sentence lists the tables set of the Γ_B device.

$\Gamma_B V = \{\Gamma_{SCT\_Flag}\}$

// the above sentence lists the registers set of the Γ_B device. These registers keep the device configuration values in it.

$\Gamma_B P = \{\Gamma_{P1}\}$ // Tells the set of ports in device the Γ_B.

Then:

$\Gamma_B = \{\Gamma_{Mtc}, \Gamma_{Paz}, \Gamma_{Sf}, \Gamma_{Gf}, \Gamma_{Pcr}, \Gamma_{Pfr}, \Gamma_{RT}[e][f], \Gamma_{SCT\_Flag}, \Gamma_{P1}\}$

3.2.2 Assumptions

- All connections are assumed to be Bi-Directional & existent connections when ever needed.
- An existent Routing Protocol has already been configured by the system administrator.
- The connection between α & β devices is assumed to be existing 24 hours a day for 7 days a week.

3.2.3 Devices Communications

- The connections between α & β devices can be wired, wireless or any other communication media type, however the connection between β & Γ devices should be wireless.

  α >--< β: Wire, Wireless, *or others*

  B >--< Γ: Wireless

- Connection between β & Γ devices can be Active or Inactive connections.
- Γ_B can have only one active connection, either with Γ_A or β_A devices.
- β devices can communicate with each other.
- α – To – β – To - β connections supposed to have IP/MAC address filters to prevent any hacking attempts. Hence, with the existence of such filters, the trustiness between the devices does not need for the authentication any more. The sentence below shows that device α and β trust each other completely without the need for an authentication key. Please notice that our Specification language, so far, doesn't support the full specification notations and the rules for the security aspect.

  α <**Ŧ**> β

  and:

  β_x <**Ŧ**> β_{x+1}

- β – To - Γ – To – β connections should be secured using some kind of authentication process. The sentence below shows that device β trusts device Γ with an authentication key k, while device Γ trusts device β with the authentication ket g.

  β^g <**Ŧ**> Γ^k

- Layer-1 devices can communicate with Layer-2's & Layer-2 devices can communicate with Layer-1 devices when required:

  $\alpha_A \Leftrightarrow \beta_A$

- At the same time, Layer-1 devices have a full control on Layer-2 & Layer-3 devices:

  $\alpha_A \ \Xi \ \beta_A$

And:

$$\alpha_A \Xi \Gamma_x$$

- Layer-2 devices are connected to each other so they can reach each other:

$$\beta_{Ax} \Leftrightarrow \beta_{Ay}$$

- Layer-2 Devices can reach Layer-3 devices when required and Layer-3 devices can reach layer-2 devices:

$$\beta_A \Leftrightarrow \Gamma_x$$

- So, Layer-1 Devices can reach Layer-3's only through layer-2 devices.

$$\alpha_A \Leftrightarrow \beta_A \Leftrightarrow \Gamma_x$$

- Layer-3 device-B, has limited abilities compares to Layer-1 Device-A, Γ_B can reach layer-2 devices only, while Γ_A can reach Layer-2 devices as well as Layer-3 devices.

$$\Gamma_B \Leftrightarrow \beta_A$$

While:

$$\Gamma_A \Leftrightarrow \beta_A$$

And:

$$\Gamma_A \rightarrow \Gamma_B$$

- As well as, Layer-3 Device-A can control Layer-3 device-B:

$$\Gamma_A \Xi \Gamma_B$$

3.2.4 The Application Layer Protocol Abilities

Our proposed protocol has the following abilities:

- Issuing & delivering orders to remote nodes.
- Receiving and saving reports from remote nodes.
- Issuing & delivering services to remote nodes.
- Locating a remote node.
- Provide secured access to devices.

In the next sections, we show the abilities of our protocol through showing its 24 messages abilities and categorize them as how they interact with each other as three phases.

3.2.4.1 Remote Speed limit phase

These are the messages in remote speed limit phase:

1- Issuing Speed code Message

2- Broadcast Speed Code Message

3- Issue/send L1 Node status report

4- Unicast L1 Node status report

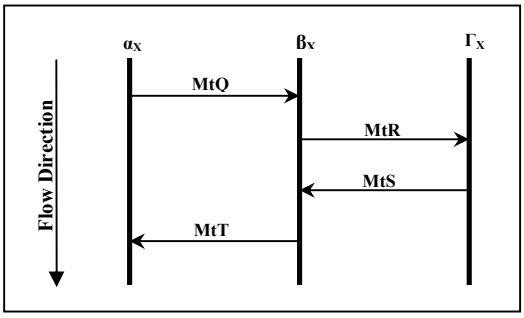

FIGURE 3.5: Remote Speed limit phase Message Flow

Starting by Message 1, a layer-1 device-A generates and sends a speed code to a layer-2 device that will forward the same message type (MtQ) into the next Layer-2 device and broadcast the contents of the incoming message in MtR to Layer-3 devices (Message 3).

When a layer-3 device-A receives the MtR, it starts to create and sends out a message that has the device's status at that moment (Message 4 - MtS) to the Layer-2 device which in turn will uncast the status report to the layer-1 device (Message 5 - MtT).

Let's illustrate the system's abilities to limit the speed of remote vehicles in details:

1- Issuing Speed code Message

When Layer-2 Device-A receives a message, it checks its type to decide what and how would it be analyzed:

α_A ~Q> β_{Ax}: <MtQ, #Hp, #SpC>

Let IMsg = <MtQ, #Hp, #SpC>

Where: IMsg: Incoming Message

Let $NMsg_1$= <MtQ, #Hp-1, #SpC>

Where: $NMsg_1$ = New Message-1

β_{Mtc} (IMsg) ↜ β_{Paz} (#Hp, #SpC) , β_{Pcr}($NMsg_1$)

β_{Pcr} ↜ β_{Pfr} ($NMsg_1$, β_{P3})

When the Mtc function finds out that the incoming message type is MtQ, it will call Paz function to start analyzing the rest of the message as (#Hp, #SpC), as well as, Mtc will call Pfr function to forward the message to the next Layer-2 device-A.

2- Broadcast Speed Code Message

After Layer-2 device analyzes the incoming message into its values, Paz would sparks Pcr function to create a new message typed-R (MtR) that will be propagated to Layer-3 Devices (Γ) through port-2 (β_{P2}) by β_{Pfr} function:

β_A ~R> Γ_B: <MtR, Rsq(β_{Ax}), #SpC>

Let $NMsg_2$ = β_{Pcr} (MtR, Rsq(β_{Ax}), #SpC)

Where: $NMsg_2$ = New Message-2

β_{Pcr} ↜ β_{PFr}($NMsg_2$, β_{P2})

3- Issue/send L1 Node status report

When a layer-3 Device-B receives the propagated message, it will first try to identify the message type by using $\Gamma_{Mtc:}$

Γ_B ~S> β_A: <MtS, CR>

Let IMsg = <MtR, Rsq(β_{Ax}), #SpC>

Where: IMsg: Incoming Message

Now, Mtc's function is to recognize the message type as MtR, it will call Paz function to analyze the rest of the incoming message as (Rsq(β_{Ax}), #SpC):

Let $NMsg_3$ = <MtS, CR>

Γ_{Mtc} (IMsg) ↜ Γ_{Paz} (Rsq(β_{Ax}), #SpC) , Γ_{Pcr}($NMsg_3$)

Γ_A will keep the information of the incoming message source device in RT table:

(Rsq(β_{Ax}), RSE_IP) ≈> Γ_{RT}[][]

As we can see, Mtc sparked Pcr function to create a message type-S that contains the report (CR) that will be unicasted to the RSE that is currently in touch with (this can be found in RT table).

Γ_{Pcr} ↜ Γ_{Pfr}($NMsg_3$, Γ_{P1})

4- Unicast L1 Node status report

After receiving an incoming message by a Layer-2 Device-A and recognized its type as MtS, its Mtc function will call Paz to analyze the rest of the message as (CR), and saves a copy of that report in its VT. Then creates a unicast message that includes the report and the RSE's Rsq.

β_A ~T> α_A: <MtT, CR, Rsq>

Let IMsg = <MtS,CR>

Let $NMsg_4$ = <MtT, CR, Rsq>

β_{Mtc} (IMsg) ↜ β_{Paz} (CR), $\beta_{Pcr}(NMsg_4)$

(CR) ≈> β_{VT}[e][f]

β_{Pcr} ↜ $\beta_{Pfr}(NMsg_4)$

Now, a layer-1 Device-A receives a message, its Mtc function will adopt that message up to recognize its type. After recognizing the type is S, the Mtc function will sparks Paz to read the next bits of the incoming message as the report and where does it come from (CR,Rsq).

Let IMsg = <MtS, CR>

α_{Mtc} (IMsg) ↜ α_{Paz} (CR)

Now, the report is ready to be saved into database:

α_{Paz} ↜ α_{SQL}(CR, Rsq) ≈> α_{VT}[e][f]

3.2.4.2 Sending a service message to a Layer-3 Device phase

First of all, a manual human interaction should be done by the system administrator to query the details of the Layer-1 device from its own VT table. However, if could not find them then try to query the central DB. By assuming the data were found and the layer-3 Device still within the coverage of Layer-3 Device-A. Then using the following messages, we can approach the Layer-3 Device and deliver a service message to:

1- Sending a service Message

2- Broadcast the service Message

3- Sending a Reply of the service

4- Unicast the Reply of the service to Layer-1 Device-A

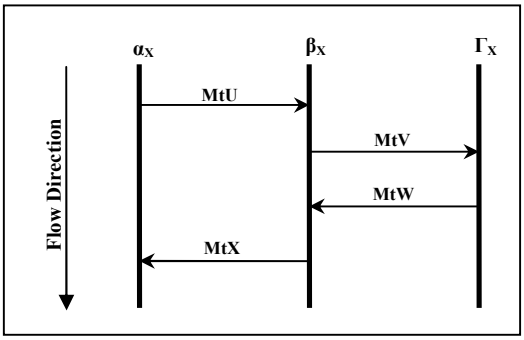

FIGURE 3.6: Sending a service message to a Layer-3 Device phase Message Flow

By sending the service message (MtU) to a layer-2 device-A, the last will broadcast it to all Layer-2 devices (MtV) and forward it to the next Layer-2 devices. When a layer-3 device receives that service message, it will apply the service and reply the application result (Successful, not successful, or some other predefined value). The reply can reach from Layer-3 device to layer-1's, through two messages; the first will deliver the reply from Layer-3 to layer-2 device (MtW), while the second is (MtX) which will be unicasted by layer-2 device to Layer-1's.

Let's illustrate the abilities of the layered system at the Sending a service message to a Layer-3 Device phase in more detailed explanation:

1- Sending a service Message

When Layer-2 Device-A receives a message, it checks its type to decide what how would it be analyzed:

α_A ~U> β_A: <MtU, Hp, SPN#, Value, XXX >

Let IMsg = <MtU, Hp, SPN#, Value, XXX >

Where: IMsg: Incoming Message

Let $NMsg_1$= <MtV, SPN#, Value, XXX>

Where: $NMsg_1$ = New Message-1

β_{Mtc} (IMsg) ↝ β_{Paz} (Hp, SPN#, Value, XXX) , β_{Pcr}($NMsg_1$)

β_{Pcr} ↝ β_{Pfr} ($NMsg_1$, β_{P3})

When its Mtc function finds out that the incoming message's type is MtU. It will call Paz function to start analyzing the rest of the message as (Hp, SPN#, Value, XXX), where XXX can be a useful information for the required service, as well as, Mtc will call Pfr function to forward the message to the next Layer-2 device.

2- Broadcast the service Message

After Layer-2 device analyzes the incoming message into its values, Paz will sparks Pcr function to create a new message typed-V (MtV) that will be propagated to Layer-3 Devices (Γ) through port-2 (β_{P2}) by β_{Pfr} function:

β_A ~V> Γ_B: <MtV, SPN#, Value, XXX>

Let $NMsg_2$ = β_{Pcr}(MtV, SPN#, Value, XXX)

Where: $NMsg_2$ = New Message-2

β_{Pcr} ↝ β_{PFr}($NMsg_2$, β_{P2})

3- Sending a Reply of a service

When a layer-3 Device-B receives the propagated message, it will first try to identify the message type by using Γ_{Mtc}.

Γ_B ~W> β_A: <MtW, SPN#, Value>

Let IMsg =<MtV, SPN#, Value, XXX>

Where: IMsg: Incoming Message

Now, Mtc's function is to recognize the message type as MtV, it will call Paz's function to analyze the rest of the incoming message as (SPN#, Value, XXX):

Let $NMsg_3$ = <MtW, SPN#, Value>

Γ_{Mtc} (IMsg) ↝ Γ_{Paz} (SPN#, Value, XXX), $\Gamma_{Pcr}(NMsg_3)$

Γ_{Paz} ↝ Γ_{FX}

Where Γ_{FX} is any service function

After the service implementation has been done, an output of that function will appear; that will be presented by a value that could be successful, unsuccessful, or any other predefined value. This value will be inserted into an MtW message and sent through Port 1 of Layet-3 Device-A to the RSE (Currently in touch with).

Γ_{Pcr} ↝ $\Gamma_{Pfr}(NMsg_3, \Gamma_{P1})$

4- Unicast the Reply of a service to Layer-1 Device-A

After receiving an incoming message by a Layer-2 Device-A, and recognizing its type as MtW, its Mtc function will call Paz to analyze the rest of the message as (Rsq, SPN#, Value). Then create a unicast message (MtX) which includes the service reply, Layer-3 device ID, and the RSE's Rsq this message is unicasted from, to the layer-1 Device-A.

β_A ~X> α_A: <MtX, Rsq, SPN#, Value>

Let IMsg = <MtW, SPN#, Value>

Let $NMsg_4$ = <MtX, Rsq, SPN#, Value>

β_{Mtc} (IMsg) ↜ β_{Paz} (Rsq, SPN#, Value), β_{Pcr}($NMsg_4$)

β_{Pcr} ↜ β_{Pfr}($NMsg_4$, β_{P1})

Now, a layer-1 Device-A receives a message, its Mtc function will adopt that message up to recognize its type. After finding the type is X, the Mtc function will sparks Paz to read the next bits of the incoming message as the service reply and where does it come from (Rsq, SPN#, Value).

Let IMsg = <MtX, Rsq, SPN#, Value>

α_{Mtc} (IMsg) ↜ α_{Paz} (Rsq, SPN#, Value)

3.2.4.3 Getting access from a mobile node into another node phase

This phase starts by the user of Layer-3 device-A inputs an ID of a layer-3 Device-B through its user interface and send a request message to a layer-2 Device-A. It tries to get the full information of that device-B before it can get access to it.

The following are the messages used in the phase of getting access from a mobile node into another mobile node:

1- Access Authentication Request (MtB^{-1})

2- Reply - Access Authentication Request

3- Layer-3 Device-A info. Request (MtD^{-1})

4- Reply – Layer-3 Device-A info. Request (MtC, MtF)

5- Copy of Layer-3 Device-A info

6- Layer-3 Device-A info. Request (MtD^{-1})

7- Layer-3 Device-B SPN# info Request (MtH^{-1})

8- Reply - Layer-3 Device-B SPN# info Request (MtG)

9- Layer-3 Device-B SPN# info Request (MtJ^{-1})

10- Reply – Layer-3 Device-B SPN# info Request (MtK, MtI)

11- Layer-3 Device-B SPN# info Request

12- Copy of Layer-3 Device-B SPN#'s info

13- Access Request & Speed information (MtP^{-1})

14- PID availability enquiry (MtO^{-1})

15- Reply - PID availability enquiry (MtN)

16- Reply for Access request (MtM)

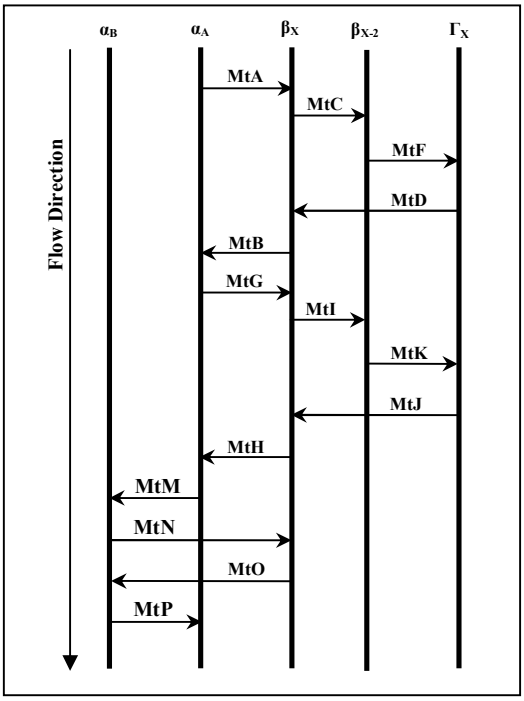

FIGURE 3.7: Getting access from a mobile node into another node phase Message Flow

By sending an Access Authentication request message (MtA) to a layer-2 device-A, it will check for the existence of the source node (which is Layer-3 Device-A) in its SCT Table. If the data exists, then it will reply with a positive Access authentication reply (MtB). Otherwise, it will try to ask for those data from the two previous layer-2 devices (MtC), which will reply with a positive reply message if the requested data is exist in the SCT table of any of them (MtD). If it is not found, then it will send a request to the Layer-1 Device-A (MtF) asking for the same information, that will also check for the availability for that information. If it has been found, it will unicast it (MtD) to the layer-2 device-A which first start with the query, while a negative reply will be issued in case of the none existence.

When the (destination) layer-2 device-A receives the MtD message, it will unicast it to the Layer-3 device-A (which asked for the information) and forwards a copy of those information (MtE) to the next ten layer-2 devices to guarantee the information delivery to people who have requested for layer-2 device. In case of receiving the requested data by L2DA, it will grant the access for the layer-3 Device-A, so now it can read from / write to

L2DA. According to L3DA, it might need to get information for another L3DB so it can get access to the last, which can query the L2DA (MtG) to get from them.

The same sequence will be repeated after looking at L3DB's information, but this time L2DA searches in its ST table, the messages are different, and the result reply will be (MtH) sent to the L3DA.

Finally, assume that L3DA has gotten the information of the other L3DB, now it can try to authenticate into that device (MtM) using these information. L3DB will check the trustworthy of the source node of MtM message by asking the L2DA (MtN). Lastly, it will look for a match of the node's ID in its SCT table. Once it has been found, then it will reply through a positive MtO to the L3DB, which consequently sends out a unicast of positive reply to the L3DA telling that now it can read from / write on it.

Let's illustrate the abilities of the layered system at getting access from a mobile node into another node phase in detailed manner:

1- Access Authentication Request (MtB^{-1})

Message type-A is a created by Γ_A and sent to β_A as a unicast message.

When β_A receives the message, it checks the type to decide what how would it be analyzed:

Γ_A ~A> β_A: <MtA, PID, Pswd>

Let IMsg = <MtA, PID, Pswd>

Where: IMsg: Incoming Message

Let $NMsg_1$= <MtB, Value>

Where: $NMsg_1$ = New Message-1

Let $NMsg_2$ = <MtC, PID, Pswd, Rsq(x), Hp>

Where: $NMsg_2$ = New Message-2

β_{Mtc} (IMsg) ↗ β_{Paz} (PID, Pswd) ↗ (PID_S, $Pswd_S$) <≈ β_{SCT}

Where PID_S: PID Value stored in the SCT table

$Pswd_S$: The stored Pswd value in SCT

PID Vs. PID_S & Pswd Vs. $Pswd_S$ → $\beta_{Pcr}(NMsg_1)$ ↗ β_{Pfr} ($NMsg_1$, β_{P2})

PID Vs. PID_S & Pswd Vs. $Pswd_S$!→ $\beta_{Pcr}(-NMsg_1)$, $\beta_{Pcr}(NMsg_2)$

Where: " -NMsg " means Negative New Message

$\beta_{Pcr}(-NMsg_1)$ ↗ β_{Pfr} ($-NMsg_1$, β_{P2})

$\beta_{Pcr}(NMsg_2)$ ↗ β_{Pfr} ($NMsg_2$, β_{P1})

When the Mtc function finds out that the incoming message's type is MtA. It will call Paz function to start analyzing the rest of the message as (PID, Pswd). At the same time, it looks for the PID_S & $Pswd_S$ in the SCT table. If it has been found, then it will compare PID with PID_S and Pswd with $Pswd_S$. If they matched, then β_A will create a positive reply message (MtB) and forward it to port to as a unicast message to Γ_A. If the values are not matching to each other, then a negative MtB message will be sent to Γ_A.

While in case of none existence of PID's information in the SCT Table, β_A will create a query message (MtC) and forward it to the two previous Layer-2 devices by asking them for those missed information, probably, those information are exists in the first previous β Device.

2- Reply - Access Authentication Request

There are four different types of MtB messages to be sent by layer-2 devices and be received by layer-3 devices. When the last layer receives the message it will check the type of the message, this would be done by Mtc.

Γ_A <B~ β_A: <MtB, Value>

Let IMsg =<MtB, Value>

Where: IMsg: Incoming Message

Let $NMsg_1$= <MtG, PID, SPN#>

Where: $NMsg_1$ = New Message-1

Γ_{Mtc} (IMsg) ↜ Γ_{Paz} (Value)

After Mtc has discovered the message type, it will call Paz function to analyze the rest of the incoming message as (Value):

(Value=0) → $\Gamma_{Pcr}(NMsg_1)$ ↜ Γ_{Pfr} ($NMsg_1$, Γ_{P1})

(Value=1) → A text Message appears on the user interface "Access Denied! wrong Password"

(Value=2) → A text Message appears on the user interface "Waiting …!"

(Value=3) → A text Message appears on the user interface "Access Denied! Invalid PID's Info."

3- Layer-3 Device-A info. Request (MtD^{-1})

In case of none existence of PID's information in the SCT table of the Layer-2 device, it will call for the help of the two previous layer-2 devices to find out the information in their own SCT tables:

β_A ~C> $\beta_{A\text{-}1/2}$: <MtC, PID, Pswd, Rsq(x), Hp>

Let IMsg =<MtC, PID, Pswd, Rsq(x), Hp>

Where: IMsg: Incoming Message

And Let Hp = 2

Let $NMsg_1$= <MtC, PID, Pswd, Rsq(x), Hp-1>

Where: $NMsg_1$ = New Message-1

Let $NMsg_2$= <MtF, PID, Pswd, Rsq(x)>

Where: $NMsg_2$ = New Message-2

Let $NMsg_3$= <MtD, Value, PID's info>

Where: $NMsg_3$ = New Message-3

β_{Mtc} (IMsg) ↗ β_{Paz} (PID, Pswd, Rsq(x), Hp) ↗ (PID_S, $Pswd_S$) <≈ β_{SCT}

Where PID_S: PID Value stored in the SCT table

$Pswd_S$: The stored Pswd value in SCT

PID Vs. PID_S & Pswd Vs. $Pswd_S$ → β_{Pcr}($NMsg_3$)↗ β_{Pfr} ($NMsg_3$, β_{P1})

PID Vs. PID_S & Pswd Vs. $Pswd_S$ & Hp=0 !→ β_{Pcr}(-$NMsg_1$) ↗ β_{Pfr} (-$NMsg_1$, β_{P3})

PID Vs. PID_S & Pswd Vs. $Pswd_S$ & Hp!=0 !→ β_{Pcr}($NMsg_2$) ↗ β_{Pfr} ($NMsg_2$, β_{P3})

Where: "-NMsg" means Negative New Message

4- Reply – Layer-3 Device-A info. Request (MtC, MtF)

When a message type-D is sent by $\beta_{A-1/2}$ or α_A to β_A, it will be issued as a unicast message, that its value will be forwarded as the unicast message (MtB) to the Layer-3 device-A (Γ_A) that carries the PID as its ID.

β_A <D~ $\beta_{A-1/2}$ or α_A: <MtD, Value, PID's info>

Let IMsg= <MtD, Value, PID's info>

Where: IMsg = Incoming message

Let $NMsg_1$=<MtB, Value>

Where: $NMsg_1$ = New Message-1

Let $NMsg_2$=<MtE, Hp, PID's info>

Where: $NMsg_2$ = New Message-2

Hp = 10

β_{Mtc} (IMsg) ↜ β_{Paz} (Value, PID's info)

$\beta_{Pcr}(NMsg_1)$ ↜ β_{Pfr} ($NMsg_1$, β_{P2})

(Value = 0) → $\beta_{Pcr}(NMsg_2)$ ↜ β_{Pfr} ($NMsg_2$, β_{P1})

5- Copy of Layer-3 Device-A info

When a layer-2 device-A receives a positive Message type-D (that has the PID information), it will forward a copy of those information to the next ten layer-2 Devices as a MtE with Hp value of 10.

β_A ~E> $\beta_{A+1/10}$: <MtE, Hp, PID's info>

Let IMsg= <MtE, Hp, PID's info>
Where: IMsg = Incoming message
Hp = 10
Let $NMsg_1$=<MtE, Hp-1, PID's info>
Where: $NMsg_1$ = New Message-1
Let $NMsg_2$=<MtB, Value>
Where: $NMsg_2$ = New Message-2

β_{Mtc} (IMsg) ↗ β_{Paz} (Hp, PID's info) ↗ (PID's info) ≈> β_{SCT}
and (Hp-1 > 0) → β_{Pcr} ($NMsg_1$) ↗ β_{Pfr} ($NMsg_1$, β_{P3})
β_{Pcr} ($NMsg_2$) ↗ β_{Pfr} ($NMsg_2$, β_{P2})

6- Layer-3 Device-A info. Request (MtD^{-1})

When a layer-1 device-A receives the MtF message, it will analyze it into its prime fields looking for the PID identification whether its information are missed and requested. So Layer-1 Device-A will look for it in its SCT table. If it has been found, then it will issue a MtD message and unicast it to the RSE(Rsq). On the other hand, if it has not been found, then it will query the Database server storage.

$\beta_{A\text{-}2}$ ~F> α_A: <MtF, PID, Pswd, Rsq(x)>

Let IMsg= <MtF, PID, Pswd, Rsq(x)>
Where: IMsg = Incoming message
Let $NMsg_1$=<MtD, Value, PID's info, Rsq(x)>
Where: $NMsg_1$ = New Message-1
Let $NMsg_2$=(Database Server Query) ** QUERY FROM THE SQL SERVER**
Where: $NMsg_2$ = New Message-2

α_{Mtc} (IMsg) ↗ α_{Paz} (PID, Pswd, Rsq(x)) ↗ (PID's info) <≈ α_{SCT}
found: α_{Pcr} ($NMsg_1$) ↗ α_{Pfr} ($NMsg_1$, α_{P1})
not found: α_{Pcr} ($NMsg_2$) ↗ α_{Pfr} ($NMsg_2$, α_{P2})

7- Layer-3 Device-B SPN# info Request (MtH^{-1})

When the layer-3 device-A receives a positive reply (+MtB) to access a layer-2 device-A, it will create an information request message (MtG) asking for the full set of Node's (numbered SPN#) information and waits for the reply (MtH).

When the MtG message reaches the layer-2 device-A, it will be analyzed by the Mtc & Paz functions respectively. Layer-2-device-A looks for the requested information (using SPN# value) in its VT table. If it has been found, it will reply by a positive MtH reply. If it has not been found, then the device tries to ask the two previous layer-2 devices for those information (MtI).

Γ_A ~G> β_A: <MtG, PID, SPN#>

Let IMsg= <MtG, PID, SPN#>

Where: IMsg = Incoming message

Let $NMsg_1$= <MtH, Value, Node SPN#'s info>

Where: $NMsg_1$ = New Message-1

Let $NMsg_2$= <MtI, PID, SPN#, Rsq(x), Hp)

Where: $NMsg_2$ = New Message-2

Hp = 2

β_{Mtc} (IMsg) ↗ β_{Paz} (PID, SPN#) ↗ (SPN#'s info) <≈ β_{VT}

((SPN#'s info) <≈ β_{VT}) → β_{Pcr} ($NMsg_1$) ↗ β_{Pfr} ($NMsg_1$, β_{P2})

((SPN#'s info) <≈ β_{VT}) !→ β_{Pcr} ($NMsg_2$) ↗ β_{Pfr} ($NMsg_2$, β_{P1})

8- Reply - Layer-3 Device-B SPN# info Request (MtG)

MtH is a service reply message sent by the layer-2 device-A and received by the layer-3 device-A, responding to the MtG's service request message.

When MtH reaches the layer-3 device-A, it will pass through Mtc and Paz functions to be analyzed into its prime fields values. MtH might be a positive reply or a negative:

Γ_A <H~ β_A: <MtH, Value, Node SPN#'s info>

Let IMsg = <MtH, Value, Node SPN#'s info>

Where: IMsg: Incoming Message

Let $NMsg_1$= <MtM, PID, SPN#'s info, Opt, SPc>

Where: $NMsg_1$ = New Message-1

Γ_{Mtc} (IMsg) ↜ Γ_{Paz} (Value, Node SPN#'s info)

After Mtc discovers the message type, it will call Paz function to analyze the rest of the incoming message as (Value, Node SPN#'s info):

(Value=0) → $\Gamma_{Pcr}(NMsg_1)$ ↜ Γ_{Pfr} ($NMsg_1$, Γ_{P2})

And (Node SPN#'s info) ≈> Γ_{ST}

(Value=1) → A text Message appears on the user interface "Access Denied! wrong Password"

(Value=2) → A text Message appears on the user interface "Waiting …!"

(Value=3) → A text Message appears on the user interface "Access Denied! Invalid SPN#'s Info."

9- Layer-3 Device-B SPN# info Request (MtJ^{-1})

In case of none existence of SPN#'s information in a VT table of the Layer-2 device, it will call for the help of the two previous layer-2 devices to find out the information in their own VT tables:

β_A ~I> $\beta_{A-1/2}$: <MtI, PID, SPN#, Rsq(x), Hp>

Let IMsg =<MtI, PID, SPN#, Rsq(x), Hp>

Where: IMsg: Incoming Message

Hp = 2

Let $NMsg_1$= <MtI, PID, SPN#, Rsq(x), Hp-1>

Where: $NMsg_1$ = New Message-1

Let $NMsg_2$= <MtK, PID, SPN#, Rsq(x)>

Where: $NMsg_2$ = New Message-2

Let $NMsg_3$= <MtJ, Value, PID, Car SPN#'s info>

Where: $NMsg_3$ = New Message-3

β_{Mtc} (IMsg) ↗ β_{Paz} (PID, SPN#, Rsq(x), Hp) ↗ (SPN#'s info.) <≈ β_{VT}

((SPN#'s info.) <≈ β_{VT}) → β_{Pcr}($NMsg_3$) ↗ β_{Pfr} ($NMsg_3$, β_{P1})

PID Vs. PID_S & Pswd Vs. $Pswd_S$ & Hp=0 !→ β_{Pcr}(-$NMsg_1$) ↗ β_{Pfr} (-$NMsg_1$, β_{P3})

PID Vs. PID_S & Pswd Vs. $Pswd_S$ & Hp!=0 !→ β_{Pcr}($NMsg_2$) ↗ β_{Pfr} ($NMsg_2$, β_{P3})

Where: "-NMsg" means Negative New Message

10- Reply – Layer-3 Device-B SPN# info Request (MtK, MtI)

When a message type-J sent by $\beta_{A-1/2}$ or α_A to β_A, it will be issued as the unicast message, that its value will be forwarded as the unicast message (MtH) to the Layer-3 device-A (Γ_A) that carries the PID as its ID.

β_A <J~ $\beta_{A-1/2}$ or α_A: <MtJ, Value, PID, Car SPN#'s info>

Let IMsg= <MtJ, Value, PID, Node SPN#'s info>

Where: IMsg = Incoming message

Let $NMsg_1$=<MtH, Value, Node SPN#'s info>

Where: $NMsg_1$ = New Message-1

Let $NMsg_2$= <MtL, Hp, SPN#'s info>

Where: $NMsg_2$ = New Message-2

Hp = 10

β_{Mtc} (IMsg) ↜ β_{Paz} (Value, PID, Node SPN#'s info)

$\beta_{Pcr}(NMsg_1)$ ↜ β_{Pfr} ($NMsg_1$, β_{P2})

(Value = 0) → $\beta_{Pcr}(NMsg_2)$ ↜ β_{Pfr} ($NMsg_2$, β_{P1})

11- Layer-3 Device-B SPN# info Request

When the layer-1 device-A receives a MtK message, it will analyze it into its prime fields looking for the SPN# number that its information are missed and requested. Therefore Layer-1 Device-A will look for it in its VT table. If it has been found then, it will issue the MtJ message and unicast it to the RSE(Rsq). Meanwhile if it has not been found, then it will query the Database server storage.

β_{A-2} ~K> α_A: <MtK, PID, SPN#, Rsq(x)>

Let IMsg= <MtK, PID, SPN#, Rsq(x)>

Where: IMsg = Incoming message

Let $NMsg_1$= <MtJ, Value, PID, Node SPN#'s info>

Where: $NMsg_1$ = New Message-1

Let $NMsg_2$=(Database Server Query) ***** Standard SQL Sentence*****

Where: $NMsg_2$ = New Message-2 ** QUERY FROM THE SQL SERVER**

α_{Mtc} (IMsg) $\nearrow$ α_{Paz} (PID, SPN#, Rsq(x)) $\nearrow$ (SPN#'s info) <≈ α_{VT}

(SPN#'s info) <≈ α_{VT} → α_{Pcr} ($NMsg_1$) $\nearrow$ α_{Pfr} ($NMsg_1$, α_{P1})

(SPN#'s info) <≈ α_{VT} !→ α_{Pcr} ($NMsg_2$) $\nearrow$ α_{Pfr} ($NMsg_2$, α_{P2})

12- Copy of Layer-3 Device-B SPN#'s info

When the layer-2 device-A receives a positive Message type-J (that has the SPN#'s information), it will forward a copy of these information to the next ten layer-2 Devices as a MtL with Hp value of 10.

β_A ~L> $\beta_{A\text{-}1/10}$: <MtL, Hp, SPN#'s info>

Let IMsg = <MtL, Hp, SPN#'s info>

Where: IMsg = Incoming message

Hp = 10

Let $NMsg_1$ = <MtL, Hp-1, SPN#'s info>

Where: $NMsg_1$ = New Message-1

Let $NMsg_2$ = <MtH, Value, Node SPN#'s info>

Where: $NMsg_2$ = New Message-2

β_{Mtc} (IMsg) $\nearrow$ β_{Paz} (Hp, SPN#'s info) $\nearrow$ (SPN#'s info) ≈> β_{VT}

and (Hp-1 > 0) → β_{Pcr} ($NMsg_1$) $\nearrow$ β_{Pfr} ($NMsg_1$, β_{P1})

β_{Pcr} ($NMsg_2$) $\nearrow$ β_{Pfr} ($NMsg_2$, β_{P2})

13- Access Request & Speed information (MtP^{-1})

When the layer-3 device-A receives a MtH message with a value of zero, it means this message has the required information to access the other layer-3 deives-B. However, it should ask for the permission to get that access (MtM).

Γ_A ~M> Γ_B: <MtM, PID, SPN#'s info, Opt, SPc>

Let IMsg = <MtM, PID, SPN#'s info, Opt, SPc>

Where: IMsg = Incoming message

Hp = 10

Let $NMsg_1$ = <MtN, PID>

Where: $NMsg_1$ = New Message-1

Let $NMsg_2$ = <MtP, Value>

Where: $NMsg_2$ = New Message-2

Γ_{Mtc} (IMsg) ↯ Γ_{Paz} (Hp, PID, SPN#'s info, Opt, SPc) ↯ (SPN#s info) <≈ Γ_{SCT}

Where: SPN#s info: stored SPN# information in the node's registers

(SPN#'s info Vs. SPN#s info) → Γ_{Pcr} ($NMsg_1$) ↯ Γ_{Pfr} ($NMsg_1$, Γ_{P1})

(SPN#'s info Vs. SPN#s info) !→ Γ_{Pcr} ($NMsg_2$) ↯ Γ_{Pfr} ($NMsg_2$, Γ_{P2})

When Value of $NMsg_2$ = 2

14- PID availability enquiry (MtO^{-1})

MtN message is a validation message, it checks whether a PID node is exist or not (is it a real PID node, or a fake one). So, when the layer-2 device-A receives the MtN message, it analyzes and start looking for a PID match in its SCT table. If the match was found, then it will reply with a positive MtO message to the requester node. If it was not found then it will send a negative MtO reply.

Γ_B ~N> β_A: <MtN, PID>

Let IMsg = <MtN, PID>

Where: IMsg = Incoming message

Let $NMsg_1$ = <MtO, Value>

Where: $NMsg_1$ = New Message-1

β_{Mtc} (IMsg) ↗ β_{Paz} (PID) ↗ (PIDs) <≈ β_{SCT}

Where: PIDs: a PID Match stored in the node's SCT Table.

(PID Vs.PIDs) → β_{Pcr} ($NMsg_1$) ↗ β_{Pfr} ($NMsg_1$, β_{P2})

When Value of the NMsg1= 0

(PID Vs.PIDs) → β_{Pcr} ($NMsg_1$) ↗ β_{Pfr} ($NMsg_1$, β_{P2})

When Value of the NMsg1= 1

15- Reply - PID availability enquiry (MtN)

When the layer-3 device-B receives a positive MtO reply message from a layer-2 device-A, it means that PID node is a valid one and it can trust it.

Γ_B <O~ β_A: <MtO, Value>

Let IMsg = <MtO, Value>

Where: IMsg = Incoming message

Let $NMsg_1$ = <MtP, Value>

Where: $NMsg_1$ = New Message-1

Γ_{Mtc} (IMsg) ↯ Γ_{Paz} (Value) ↯ Γ_{Pcr} ($NMsg_1$) ↯ Γ_{Pfr} ($NMsg_1$, Γ_{P1})

16- Reply for Access request (MtM)

Finally, when the Layer-3 device-A receives a positive reply from another layer-3 device-A, it means access is granted by the other side to access it. However, if it is a negative reply, then it means the access was denied by the other side.

Γ_A <P~ Γ_B: <MtP, Value>

Let IMsg = <MtP, Value>

Where: IMsg = Incoming message

Let $NMsg_1$ = <MtM, PID, SPN#'s info, Opt, SPc>

Where: $NMsg_1$ = New Message-1

(Value=0) → Γ_{Pcr}($NMsg_1$)↯ Γ_{Pfr} ($NMsg_1$, Γ_{P1})

(Value=1) → A text Message appears on the user interface "Access Denied! Invalid PID's Info."

(Value=2) → A text Message appears on the user interface "Access Denied! wrong Password"

Now, after showing the design of our proposed system using our specification language and the abilities of both of them, in the next chapter we are going to show some case studies done on our system using the same formal specification language that we have proposed.

CHAPTER 4: CASE STUDIES

4.1 Introduction

Using our Formal Specification Language, we are going to disclose three case studies; Speed control & Highway monitoring, Remote car locating & sending a Service request / Function message, and the last case is on Suspect car instant termination. Then we formalize their designs, and show how they work with some scenarios then we prove their validity and consistency.

Each case studies in this chapter can be laid out by describing it with flowcharts and algorithms (except for the third case study), then we set up the case study devices and their configuration using our specification language and show the scenario description on each cases. Finally, we apply the validity and consistency tests on each case study separately using the formal specification language that we have created.

4.2 Case study -1: Speed control & Highway monitoring

For the first case study "Speed Control & High Way Monitoring", See Figure 4.1, we got

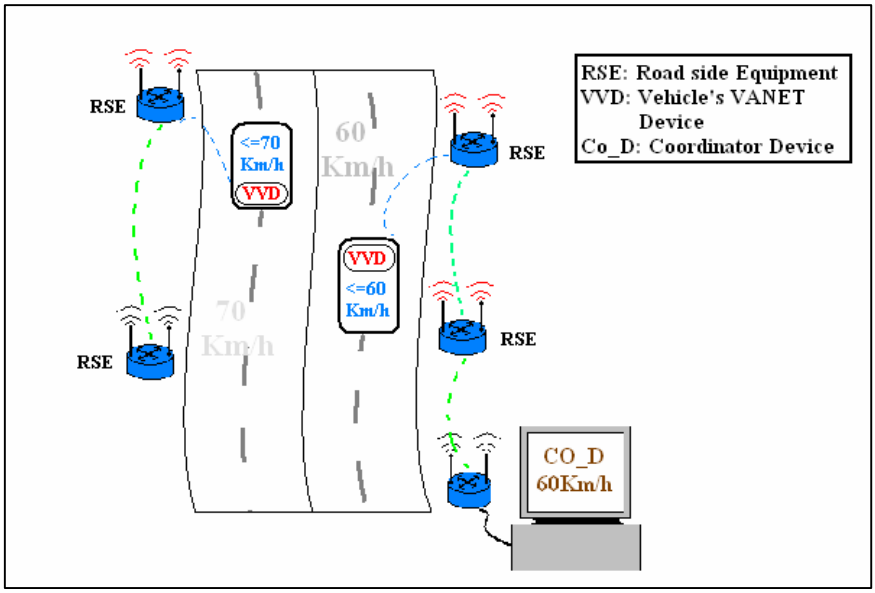

Figure 4.1: Case Study 1: Speed Control & High Way Monitoring

a Highway of two sides, the one on the right is going north while the other is going the opposite way. Each of the two ways has two lanes. The allowed speed on the right road is 60 Km/h, while 70 Km/h is the allowed speed for the left road.

On each side of the road there is a set of Equipment (Road side Equipment), those are connected to each other and have connected to an administrative device (Coordinator device) which has the ability to reach and access any of those Road side equipments (RSEs), as well as the vehicles' devices (VVD).

In this scenario, the coordinator device officer inputs the allowed speed on the road he is responsible about as a number (in our scenario, the number should be 60 for the right road; The right car driver is allowed to drive at the maximum of 60 Km/h, not more than that). This will be converted into an equivalent code which will be forwarded to the RSE devices that will propagate the code to the vehicles within its coverage area.

If the right side was driving at a higher speed more than 60 Km/h, then the vehicle's communication device will receive the code propagated by the RSEs. It will convert it into an analog signal that controls the vehicle's speed to reduce its speed and peaks at 60 km/h not more than that.

To present this scenario mathematically, we use our formal specification language to define each device and the connection type between them and their configuration. Finally we show how exactly the messaging processes between the devices.

4.2.1 Participated devices definitions

In this scenario, three devices are participating in; these are; Coordination device (which is the master of everything happening on the road it's responsible for), Road Side Equipment (which has forwarded and propagated the messages between the coordinator and vehicle devices.) Vehicles VANET Devices which resides inside the vehicles, its job is to receive the messages propagated by the RSE and translate it to take the right required action.

For the coordinator device, it has three contents; they are; set of functions, set of table, and set of ports. The set of functions contain seven different functions; α_{Mtc} (checks the type of the incoming message and decides the destiny of the carried data), α_{Paz} (Analyzes the incoming data stream according to the message type into usable data), α_{Sf} (Sets some values as required), α_{Gf} (Get some values for some functions when required), α_{SQL} (Creates SQL syntaxes), α_{Pcr} (Creates different messages types as required), α_{Pfr} (Forwards the messages to the required ports). A set of tables composes of 4 tables; SMT (Speeds Mapping table), VT (lists all the vehicles instantly r within the coverage area of the system and those were in since the last X hour(s), where X is a number ($0 < X < \infty$)), SCT (Lists all the cars those were reported as suspect cars), ToR (Lists the IDs of all the road side units under the responsibility of the coordinator device). Finally, the set of the coordinator device ports, which has two main ports; P1 & P2. P1 (Port one, which connects the coordinator device to the first road side unit on the highway it's responsible to manage). On the other side we have P2 (Port two, which connects the coordinator device to a database server).

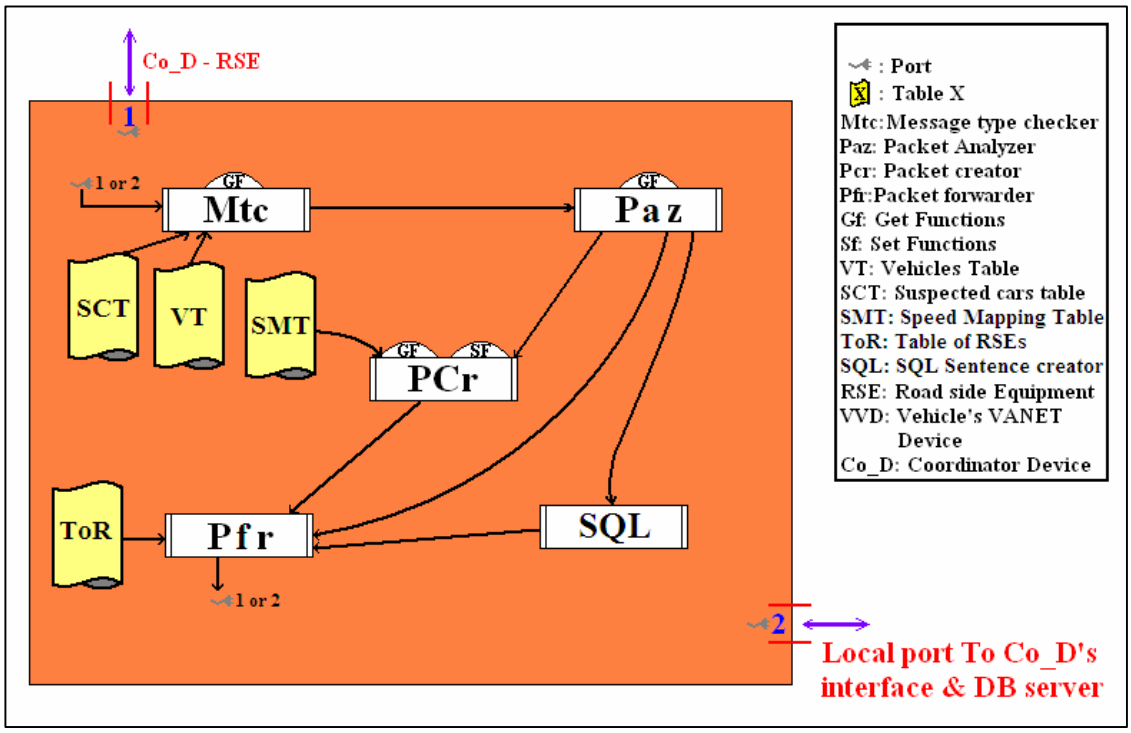

Figure 4.2: Coordinator device internal architecture

Let $\alpha_A = \{ \alpha_A F, \alpha_A T, \alpha_A P, \alpha_A\_INT\}$

Where:

α_A : Coordinator device (Co_D)

$\alpha_A F$: Co_D's Functions

$\alpha_A T$: Co_D's Tables

$\alpha_A P$: Co_D's Ports

α_A_INT: Co_D's User Interface

And let:

$\alpha_A F = \{ \alpha_{Mtc}, \alpha_{Paz}, \alpha_{Sf}, \alpha_{Gf}, \alpha_{SQL}, \alpha_{Pcr}, \alpha_{Pfr}\}$

Where:

α_{Mtc}: Message type checker function

α_{Paz}: Packet / Massage Analyzer

α_{Sf}: set of SET functions

α_{Gf}: Set of GET Functions

α_{SQL}: SQL syntax creator function

α_{Pcr}: Packet / Message Creator

α_{Pfr}: Packet / Message forwarder (specifies to which port)

And Let:

$\alpha_A T = \{\alpha_{SMT}[c][d], \alpha_{VT}[e][f], \alpha_{SCT}[g][h], \alpha_{ToR}[i][j]\}$

Where:

$\alpha_{SMT}[c][d]$: Speed Mapping table, maps speeds as numbers into its equivalent code. It has c of rows and d of columns, where both of c and d are positive integers.

$\alpha_{VT}[e][f]$: Table of all the vehicles access the coverage area for the last x hour(s). Where x is a number ($0 < x < \infty$). It has e of rows and f of columns, where both of e and f are positive integers.

$\alpha_{SCT}[g][h]$: Contains the full information about Special vehicles table instantly within the coverage area. It has g of rows and h of columns, where both of g and h are positive integers.

$\alpha_{ToR}[i][j]$: a Table contains the full information about the RSEs, those are under the coordinator device responsibility. It has i of rows and j of columns, where both of i and j are positive integers.

$\alpha_A P = \{\alpha_{P1}, \alpha_{P2}\}$

Where:

α_{P1}: The first port of Co_D device, connected to the first RSE.

α_{P2}: The second port of Co_D device, connected to a database server.

Then:

$\alpha_A = \{ \alpha_{Mtc}, \alpha_{Paz}, \alpha_{Sf}, \alpha_{Gf}, \alpha_{SQL}, \alpha_{Pcr}, \alpha_{Pfr}, \alpha_{SMT}[c][d], \alpha_{VT}[e][f], \alpha_{SCT}[g][h], \alpha_{ToR}[i][j], \alpha_{P1}, \alpha_{P2}\}$

α_A_Features:

α_A_Mobile = False // The Co_D is not a mobile device

α_A_SQL_Server = True // The Co_D has the ability to query an SQL server directly

α_A_User_Interface = True // The Co_D device can be managed by a user interface

α_A_initialize_Operations = True // The Co_D has the ability to start messaging other devices.

α_A_Info = True // Co_D can be considered as an information center, that because it's connected to a database server which stores the information in its storage.

α_A_Routing = True, <*Routing Protocol(s) used*> // Co_D has the routing ability by using the protocols listed <Routing Protocol(s) used>.

For the Road side Equipment (RSE), it has 4 main contents; Set of Functions, Set of Tables, Some registers, and set of ports.

The functions set contains six different functions; β_{Mtc} (checks the type of the incoming message and decides the destiny of the carried data), β_{Paz} (Analyzes the incoming data stream according to the message type into usable data), β_{Sf} (Sets some values as required), β_{Gf} (Get some values for some functions when required), β_{Pcr} (Creates different messages types as required), β_{Pfr} (Forwards the messages to the required ports). The set of tables has 2 members; VT (lists all the vehicles instantly r within the coverage area of the system and the ones were in since the last X hour(s), where X is a number ($0 < X < \infty$)), SCT (Lists all the cars those were reported as suspect cars). The set of registers has two members, these are; Speed code register (which keeps the last speed code received from the coordinator device to propagate to the vehicles within its coverage) and Road Side Equipment ID register (which keeps the configuration information of the RSE), and finally, the set of the RSE ports, which has three main ports; P1, P2, & P3. P1 (Port one, this is connected to the next RSE), P2 (Port 2, this is the propagation port, it propagates the messages received from the coordinator device to the vehicles, as well as, it receives messages from the vehicles to be forwarded to the Co_D), P3 (Port 3, Connected to Co_D or to the previous RSE).

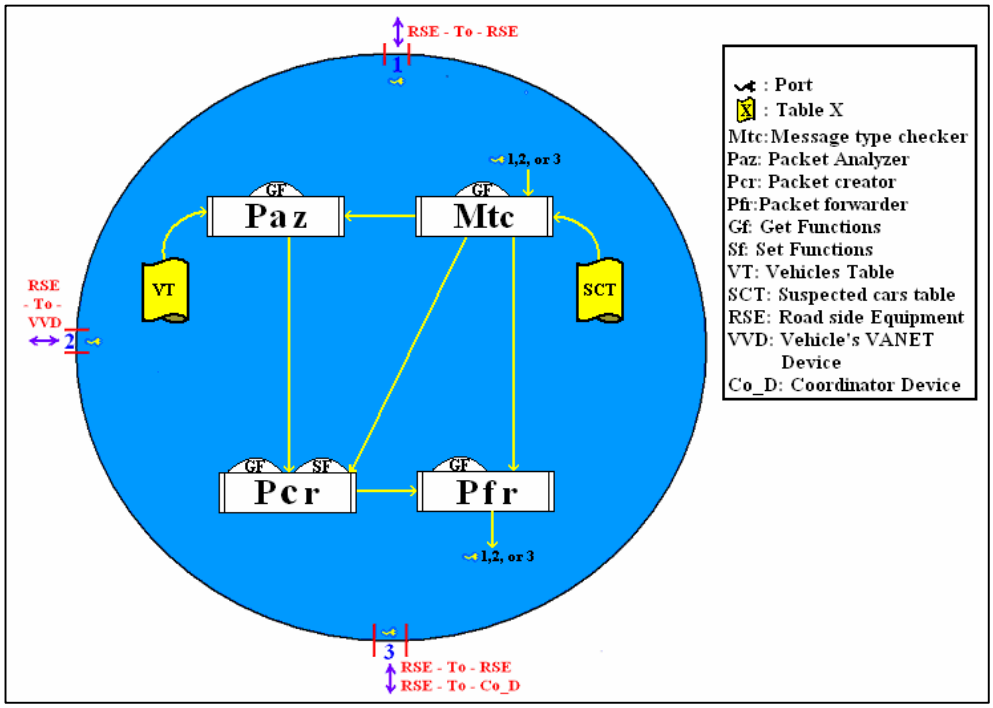

Figure 4.3: Road side equipment internal architecture

Let $\beta_A = \{ \beta_A F, \beta_A T, \beta_A V, \beta_A P\}$

Where:

β_A: Road Side Equipment (RSE)

$\beta_A F$: RSE's Functions

$\beta_A T$: RSE's Tables

$\beta_A V$: RSE's Variables

$\beta_A P$: RSE's Ports

And Let:

$\beta_A F = \{ \beta_{Mtc}, \beta_{Paz}, \beta_{Sf}, \beta_{Gf}, \beta_{Pcr}, \beta_{Pfr}\}$

Where:

β_{Mtc}: Message type checker function

β_{Paz}: Packet / Massage Analyzer

β_{Sf}: set of SET functions

β_{Gf}: Set of GET Functions

β_{Pcr}: Packet / Message Creator

β_{Pfr}: Packet / Message forwarder (specifies to which port)

And Let:

$\beta_A T = \{\beta_{VT}[e][f], \beta_{SCT}[g][h]\}$

Where:

$\beta_{VT}[e][f]$: Table of all the vehicles access the coverage area for the last x hour(s). Where x is a number ($0 < x < \infty$). It has e of rows and f of columns, where both of e and f are positive integers.

$\beta_{SCT}[g][h]$: Contains the full information about Special vehicles table instantly within the coverage area. It has g of rows and h of columns, where both of g and h are positive integers.

And Let:

$\beta_A V = \{\beta_{Spc}, \beta_{Rsq}\}$

Where:

β_{Spc}: A register keeps the current allowed speed to be driven at on the road.

β_{Rsq}: A register keep the configuration of the RSE (e.g., its ID, IP, …).

And Let:

$\beta_A P = \{\beta_{P1}, \beta_{P2}, \beta_{P3}\}$

Where:

β_{P1}: The first port of an RSE device, connected to the next RSE.

β_{P2}: The second port of an RSE device, it's the junction point between the vehicles and the system infrastructure.

β_{P3}: The third port of an RSE device, connected to the previous RSE or To the Co_D.

Then:

$\beta_A = \{ \beta_{Mtc}, \beta_{Paz}, \beta_{Sf}, \beta_{Gf}, \beta_{Pcr}, \beta_{Pfr}, \beta_{VT}[e][f], \beta_{SCT}[g][h], \beta_{Spc}, \beta_{Rsq}, \beta_{P1}, \beta_{P2}, \beta_{P3}\}$

β_A_Features:

β_A_Mobile = False // An RSE is not a mobile device

β_A_SQL_Server = False // The RSE does not have the ability to query an SQL server directly, but it can reach the database server through Co_D

β_A_User_Interface = False // The RSE device does not have a user interface to be managed by.

β_A_initialize_Operations = False // The RSE does not have the ability to start messaging other devices, it's just a forwarder.

β_A_Info = True // The RSE can be considered as an information center, that because it stores information of other devices. But it doesn't have the full information of the SQL server, RSE stores the information for the last x hour(s), where $0 < x < \infty$.

β_A_Routing = True, <Routing Protocol(s) used> // Co_D has the routing ability by using the protocols listed <Routing Protocol(s) used>.

For the Vehicle's VANET Device (VVD), it has 4 main contents; Set of Functions, Set of Tables, Some registers, and set of ports.

The functions set contains six different functions; Γ_{Mtc} (checks the type of the incoming message and decides the destiny of the carried data), Γ_{Paz} (Analyzes the incoming data stream according to the message type into usable data), Γ_{Sf} (Sets some values as required), Γ_{Gf} (Get some values for some functions when required), Γ_{Pcr} (Creates different messages types as required), Γ_{Pfr} (Forwards the messages to the required ports). The set of tables has one member; RT (lists all the RSEs instantly receiving messages from), SCT (Lists all the cars those were reported as suspect cars). The set of registers has one member, that is; SCT_Flag (indicates this vehicles is considered as a suspected vehicle), and finally, the set of the Vehicle's VANET Device ports, which has one main port; P1 (Port one, Connected to one of the RSEs or to a police vehicle).

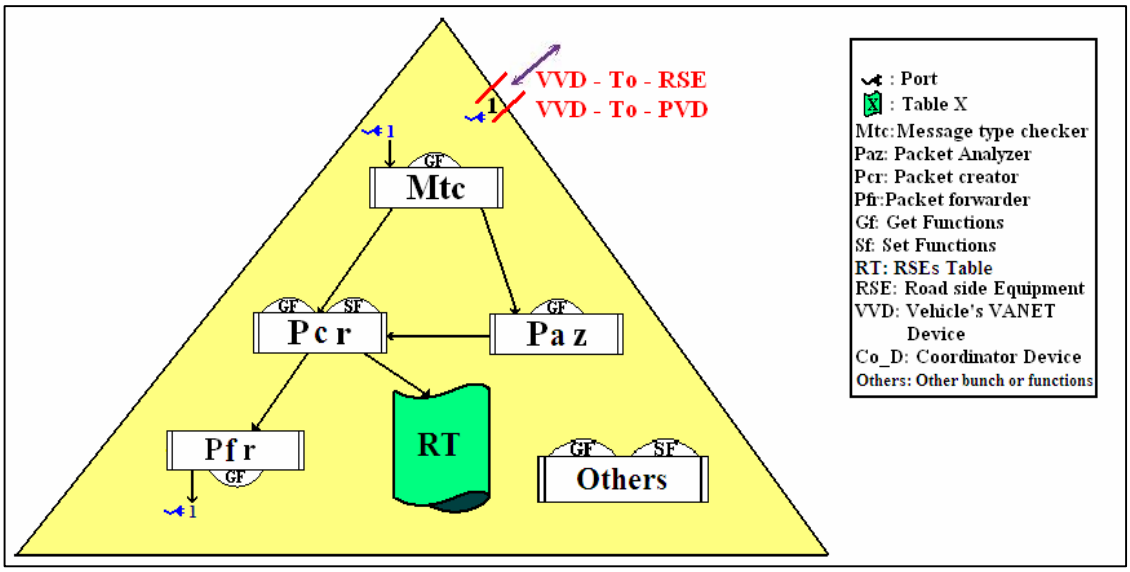

Figure 4.4 : Vehicle's VANET Device (VVD)

Let Γ_B = { Γ_BF, Γ_BT, Γ_BV, Γ_BP, Γ_BDAC}

Where:

Γ_B: Vehicle's VANET Device (VVD)

Γ_BF: VVD's Functions

Γ_BT: VVD's Tables

Γ_BV: VVD's Variables

Γ_BP: VVD's Ports

Γ_BDAC: VVD's Digital to Analog Converter (attached device)

And Let:

Γ_BF = {Γ_{Mtc}, Γ_{Paz}, Γ_{Sf}, Γ_{Gf}, Γ_{Pcr}, Γ_{Pfr}, Car_Ping() , Speed_Limit(SpC), Car_Stop()}

Where:

Γ_{Mtc}: Message type checker function

Γ_{Paz}: Packet / Massage Analyzer

Γ_{Sf}: set of SET functions

Γ_{Gf}: Set of GET Functions

Γ_{Pcr}: Packet / Message Creator

Γ_{Pfr}: Packet / Message forwarder (specifies to which port)

Car_Ping():a functions can be used to spark the car to report its status

Speed_Limit(SpC): a function to limit the speed of the vehicle to the required speed (SpC).

Car_Stop(): a function to stop the car.

And Let:

Γ_BT = {Γ_{RT}[e][f]}

Where:

Γ_{RT}[e][f]: lists all the RSEs instantly receiving messages from. It has e of rows and f of columns, where both of e and f are positive integers.

And Let:

$\Gamma_B V = \{\Gamma_{SCT\_Flag}, \Gamma_{FI}, \Gamma_{II}\}$

Where:

Γ_{SCT_Flag}: A flag register that indicates the suspicion of the vehicle.

Γ_{FI}: A register that keeps the vehicle's Fixed Information

Γ_{II}: A register that keeps the vehicle's Instant Information

And Let:

$\Gamma_B P = \{\Gamma_{P1}\}$

Where:

Γ_{P1}: The connection port of device $\Gamma_{B,}$ connected to the active RSE.

Then:

$\Gamma_B = \{\Gamma_{Mtc}, \Gamma_{Paz}, \Gamma_{Sf}, \Gamma_{Gf}, \Gamma_{Pcr}, \Gamma_{Pfr}$, Car_Ping() , Speed_Limit(SpC), Car_Stop(), $\Gamma_{RT}[e][f], \Gamma_{SCT\_Flag}, \Gamma_{P1}\}$

Γ_B_Features:

Γ_B_Mobile =True // A VVD is a mobile device

Γ_B_SQL_Server = False // The VVD does not have the ability to query an SQL server directly, but it can reach the database server through RSE from the Co_D.

Γ_B_User_Interface = True // The VVD device has a user interface to be managed by.

Γ_B_initialize_Operations = False // The VVD does not have the ability to start messaging other devices; it's just a receiver and a replier.

Γ_B_Info = False // The VVD can not be considered as an information center.

Γ_B_Routing =False // Co_D has the routing ability by using the protocols listed <Routing Protocol(s) used>.

4.2.2 Communications between participating devices

- Connections between Co_D and RSE devices can be with wire, wireless or any other communication media type. Meanwhile connection between RSE & VVD devices should be wireless.

  α_A >--< β_A: Wire, Wireless, or others

  β_A >--< Γ_B: Wireless

- Connections between Co_D-To-RSE & between RSE-To-RSE devices should be available 24 hours a day for 7 days a week.

  α_A >--< β_A_Availibility: 24/7

  β_{Ax} >--< β_{Ax+1}_Availibility: 24/7

- Connection between RSE & VVD devices can be Active or In-Active connections.

  β_A >--< Γ_B_RStatus: Active or Inactive

- VVD can have only one active connection with one RSE device at a time.

  Γ_B >--< β_A_RStatus: Active

- RSE devices can communicate each other.

  $\beta_A \Leftrightarrow \beta_A$

- Co_D – To – RSE – To - RSE connections supposed to have IP/MAC address filters to prevent any hacking attempts. As a result of that, we do not need for authentication between Co_D & RSE, or between RSE and another.

  Γ_B >--< β_A_IPFilter = True

  Γ_B >--< β_A_MACFilter = True

  α_A <**T**> β_A

  β_{Ax} <**T**> β_{Ax+1}

- RSE – To - VVD – To – RSE connections should be secured using some kind of authentication process.

  $\beta_A^{g} <\mathbf{T}> \Gamma_B^{k}$

- Co_D can communicate RSE & RSE devices can communicate Co_D devices when required:

  $\alpha_A \Leftrightarrow \beta_A$

- So, Co_D can reach VVD devices only through RSE devices on the side of the highway.

  $\alpha_A \Leftrightarrow \beta_A \Leftrightarrow \Gamma_B$

- At the same time, Co_D devices have a full control on RSE & VVD devices:

  $\alpha_A \Xi \beta_A , \Gamma_A$

- RSEs are connected to each other so they can reach each other:

  $\beta_{Ax} \Leftrightarrow \beta_{Ay}$

4.2.3 Participating messages

In this scenario, four messages are needed to implement the full procedure of the experiment, these are; MtQ (Speed Code Message), MtR (Last Destination - Speed Code Message), MtS (Mobile node's report), and MtT (Last Destination - Mobile node's report).

MtQ: α_A | β_{Ax-1}, β_{Ax}, MU, <MtQ, Hp, SpC>

Where:

- MtQ – Message type Code (10001) - Speed Code Message
- MU: Multi-hop Unicast
- Hp – Hop count
- SpC – Speed Code

MtR: β_A, Γ_B , SB, <MtR, Rsq, SpC>

Where:

- MtR – Message Type (10010) - Last Destination - Speed Code Message
- SB: Single-hop Broadcast
- Spc – Speed Code
- RSq – RSE Sequence of which is propagating the received Speed code message.

MtS: Γ_B, β_A, SU, <MtS, CR>

Where:

- Mt - Message Type (10011) - Mobile node's report
- SU: Single-Hop Unicast
- CR – Car's Report

MtT: β_A, α_A, MU, <MtT, CR, Rsq>

Where:

- MtT - Message Type (10100) - Last Destination - Mobile node's report
- MU: Multi-hop Unicast
- CR – Car's Report
- RSq – RSE's sequence Number of which the vehicle lies within its coverage area.

On the other hand, still user of the system needs more detailed information about the messages. Hence we need to create what is called Messages Cards, which will tell every detail about the messages. These cards should be attached as an appendix with the formal specification for the system we are describing, but we are going to show them here.

1- Message type-Q Definition Card

Message Name: **MtQ**	Message ID: **M#17**
Description: **Speed Code Message**	
Packet Type: **Multi-Hop-Unicast**	
Source: **α_A or β_{x-1}**	Destination: **β_x**
Message length: **3 (Fields)**	
Message Fields: **M#17, Hp, SpC**	
Details: • **Hp** – Decides how many β Device after the first β device should receive this message.	

2- Message type-R Definition Card

Message Name: **MtR** Message ID: **M#18**
Description: **Last Destination - Speed Code Message**

Packet Type: **Broadcast**
Source: $\boldsymbol{\beta_A}$ Destination: **Γ**

Message length: **3 (Fields)**
Message Fields: **M#18, Rsq, SpC**

Details:

- **Rsq** – Tells the Γ receiver device this is the ID for the β_A device which sent this message.

3- Message type-S Definition Card

Message Name: **MtS** Message ID: **M#19**
Description: **Mobile node's report**

Packet Type: **Unicast**
Source: **Γ** Destination: $\boldsymbol{\beta_A}$

Message length: **2 (Fields)**
Message Fields: **M#19, CR**

Details:

- None.

4- Message type-T Definition Card

Message Name: **MtT** Message ID: **M#20**
Description: **Last Destination - Mobile node's report**

Packet Type: **Unicast**
Source: $\boldsymbol{\beta_A}$ Destination: $\boldsymbol{\alpha_A}$

Message length: **3 (Fields)**
Message Fields: **M#20, CR, Rsq**

Details:

- None.

4.2.4 Messages Flow for Case Study-1

Figure 4.5 illustrates the four messages flow between the participated devices starting by the Co_D sends the allowed speed code out as Multi-Hop Unicast to the RSE that carries the ID: x (RSE(x)), that message type is MtQ which will travel through all the RSE devices between the two of them. When the RSE(x), it analyzes the MtQ message to get the carried information (SpC - Speed Code), encapsulate the information within an MtQ message type and propagates it to all the mobile VVDs within its coverage area.

For any VVD, when receiving an MtR message is the spark to do two major operations; the first is applying the comparison between the current vehicle's speed and the incoming speed code, if the last is less than the instant vehicles speed, then it will limits the vehicle's speed to peak at the incoming speed. The second operation is to start creating an instant report about the vehicle's status at that moment, encapsulate it by an MtS message type and unicast it to the active RSE at that moment. The RSE will unicast the MtS message to the Co_D which will analyze & keep a copy of that vehicle's report in its database. See Figure 4.6 which shows the flow chart for the whole operation starting by the Co_D sends out an MtQ message till the speed code been received by the vehicle.

The Figure 4.6 shows more details about what is going on inside each device when it receives a specific message, how will the message been processed by that device's functions, how those devices functions are making the use of the data carried by the incoming message, and what actions will be taken as a result of the incoming message.

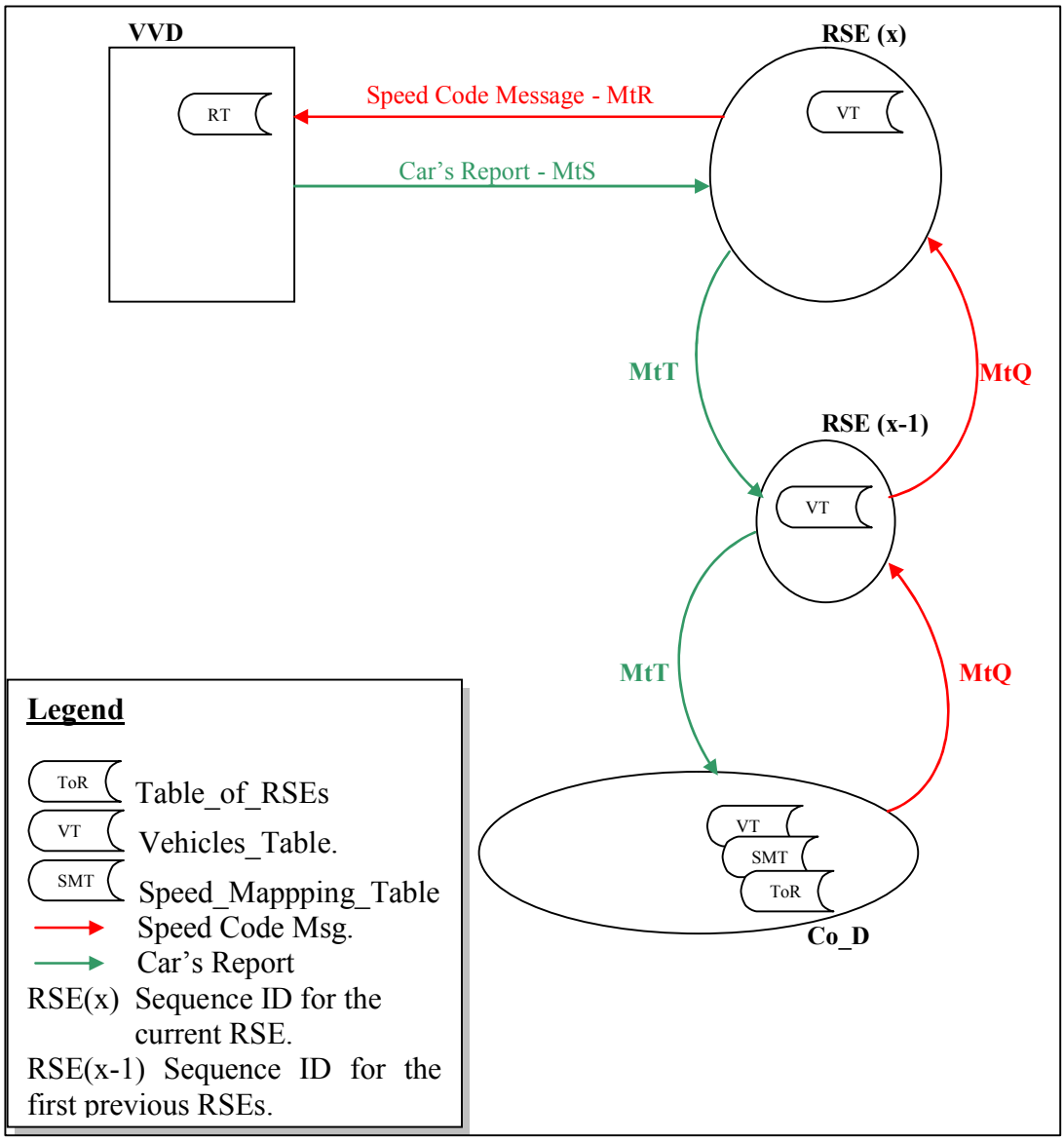

Figure 4.5: Speed Control & High Way Monitoring – messages flow

4.2.5 A Lower-Level Scenario Specification

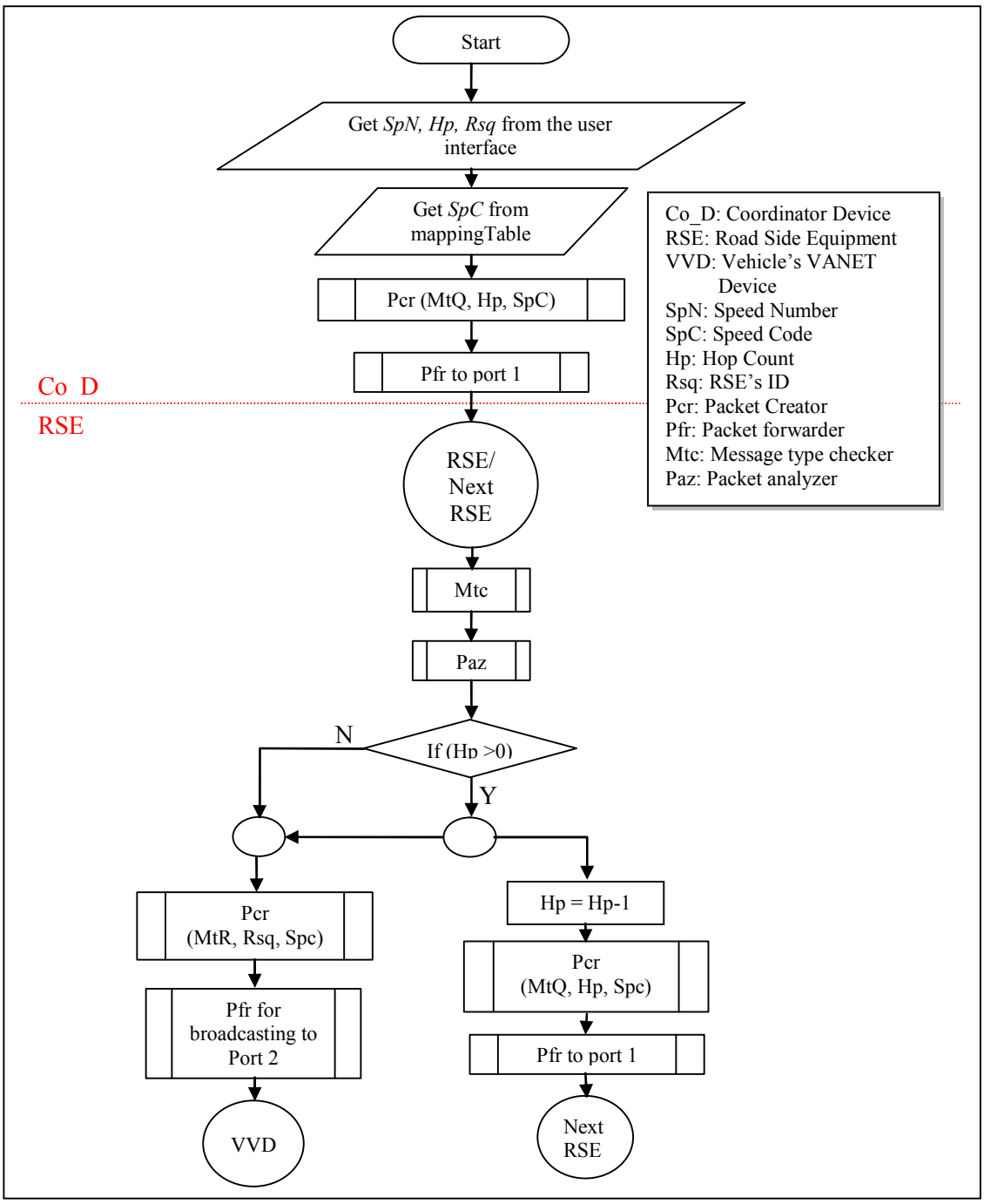

Figure 4.6: Speed Control & High Way Monitoring – messaging algorithm

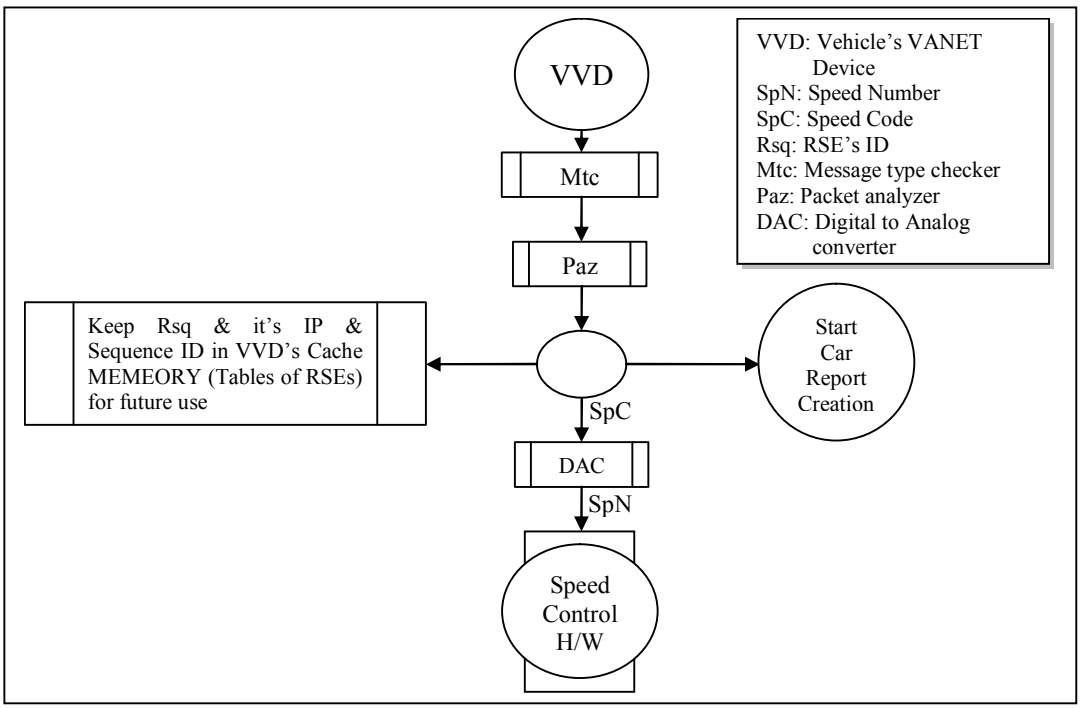

Figure 4.6.Continue: Speed Control & High Way Monitoring – messaging algorithm

Let's assume that we have the two highways shown before in Figure 4.1, the allowed speed on the right side is 60Km/h, so the Co_D administrator enters, through the Co_D interface; the allowed speed as a number, the Rsq (RSE's ID) that specifies which RSE the message should be started to propagate the vehicles from the opposite lane, and the number of hops, this number tells how many RSEs after the first one should receive the allowed speed message to propagate. See the following Scenario initial setup:

Let α_{A1} = Co_D1

α_{A1}_P1_IP: *192.168.100.1/30*

Let β_{A1} = RSE1

β_{A1}P = {P1, P2, P3}

P1_IP: *192.168.100.3/30*

P2_IP: *192.168.1.1/24*

P2_DHCP: *True – Full range*

P3_IP: *192.168.100.2/30*

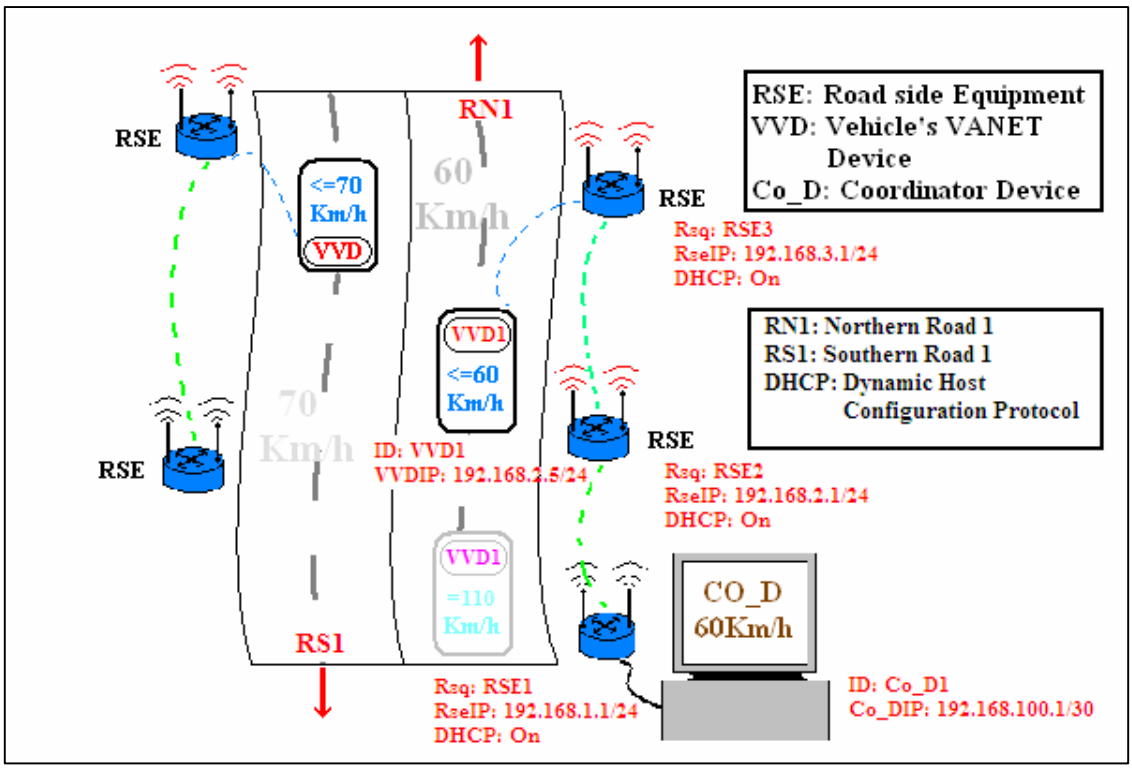

Figure 4.7: Case Study 1 – Scenario setup

Let β_{A2} = RSE2

β_{A2}P = {P1, P2, P3}

P1_IP: *192.168.100.5/30*

P2_IP: *192.168.2.1/24*

P2_DHCP: *True – Full range*

P3_IP: *192.168.100.4/30*

Let RN1_Allowed_Speed: *60Km/h*

Let RS1_Allowed_Speed: *70Km/h*

Let β_{A3} = RSE3

β_{A3}P = {P1, P2, P3}

P1_IP: *NULL*

P2_IP: *192.168.3.1/24*

P2_DHCP: *True – Full range*

P3_IP: *192.168.100.6/30*

Let Γ_{B1} = VVD 1

Γ_{B1}P = {P1}

P1_IP: 192.168.2.5/24

Γ_{B1} --: RN1

Γ_{B1} ◄ RSE1

Γ_{B1}_Speed: 110 Km/h

Γ_{B1} ► RSE3

α_{Gf} (SPN, Rsq, Hp) <≈ α_{A1}_Int

After the administrator enters those 3 values, they would be forwarded to the functions of Co_D to be processed. The first value named the SpN (Speed Number) will be mapped (by the *Get_SpC* function) into its equivalent code SpC (Speed Code), that can be found in the SMT (Speed Mapping Table). In other words, Co_D would forward the SpN to the *Get_SpC* of the SMT Table. The second value entered by the administrator is the Rsq value which will be useful to get the IP address of the RSE device that Co_D should unicast the MtQ message to, that IP can be found in ToR (Table of RSEs), and this can be done by forwarding the Rsq to the *Get_RSEIP* function of the ToR Table that will get the equivalent RSE IP.

α_{Gf} (SPc),(SPN) <≈ α_{A1_SMT}

α_{Get_RSEIP} (RseIP) , (Rsq) <≈ α_{A1_ToR}

Now the Co_D has two values; Hp, and the SpC, these will be forwarded to the Pcr function to create an MtQ message and forward it to the Pfr function that would encapsulate it within a packet that has the RSE IP as its destination IP, and then forward it to Port 1 of the Co_D device.

α_{Pcr} (<MtQ>) , (Hp, SpC) $\nearrow$ α_{Pfr} ([MtQ]) , (<MtQ>, $RseIP_{A1}$, P1)

Where $RseIP_{A1}$: The Ip address for RSE1

α_{A1} ~Q> β_{A1}

The MtQ message on the network as a data stream flow, a group of ones and zeros, traveling from RSE to another till it reaches the destination one, there it will be processed. When the destination RSE receives the data stream, it checks the first bits of that data stream to check the type of the incoming message, and this is done by the Mtc (Message Type Checker) function.

After specifying the message type, Mtc function forwards the rest of the message's data fields (which still as data stream only) to the Paz (Packet Analyzer) function, this will analyze the data stream into specific data values to make the use of them. So for the MtQ message, it will be analyzed and understood as 2 fields; Hp, and Spc, respectively.

RSE checks the value of the incoming Hp whether it is equal to zero and not to decide what to do with the incoming data. If Hp more than Zero, then it will propagate the SpC within an MtR message to the vehicles through port 2, as well as, it would forward through port one a copy of the SpC by an MtQ message it creates after decreasing the Hp by 1. While if Hp equals to Zero, then it would not forward the SpC to the next RSE, just keeps propagating MtR messages through port 2 to the vehicles within its coverage area.

β_{A1} <Q~ α_{A1}

β_{Mtc} (MtQ , <_data>) , (<DATA>) $\nearrow$ β_{Paz} (MtQ, Hp, SpC) , (MtQ, <_data>)

Where: <DATA> is the full incoming data stream

<_data> is the rest of the unanalyzed data of the incoming stream.

β_{Pcr} (<MtR>) , (Rsq , SpC) $\nearrow$ β_{Pfr} ([MtR]) , (<MtR> , P2)

β_{A1} ~R○ Γ_{B1}

(Hp > 0) → Pcr (<MtQ>) , (Hp-1 , SpC) $\nearrow$ Pfr ([MtQ]) , (<MtQ> , $RseIP_{A2}$, P1)

Where $RseIP_{A2}$: The Ip address for RSE2

β_{A1} ~Q> β_{A2}

Now a vehicle receives an MtR message, it analyzes it and understands it as Rsq & SpC fields. The Rsq tells from which RSE device the packet came, which will be kept and the RSE's IP in the RT (Table of RSEs). The SpC value will be sent to a Digital to Analog converter to get the analog equivalent for it, which is SpN. SpN will be sent to the speed controller to limit the maximum peak the car can reach. And finally as a result or receiving the MtR message, the vehicle will start creating an instant vehicle status report. See Figure 4.8.

Γ_{B1} ○R~ β_{A1}

Γ_{Mtc} (MtR , <_data>) , (<DATA>) $\nearrow$ Γ_{Paz} (MtR, Rsq, SpC) , (MtR, <_data>)

SQL (Rsq, RseIP) ≈> Γ_{B1_RT}

Γ_{B1}DAC (SpN) , (SpC)

The car's report (CR) consists of two parts, fixed information and instant information. The fixed information contains the device & vehicles information such as it's registration ID for the car, Engine ID,…etc. while the instant information has the instantaneous status of the vehicle at the moment of the report creation.

Figure 4.8 shows the flow chart of the operation starting from the vehicle‘s report creation till it has been saved into the database. The report creation sparks by receiving the MtR message by the VVD which will put both of the fixed information and the instant status information within an MtS message type frame and unicast it to the current active RSE connection. The IP for the destination RSE can be found in the Table of RSEs (RT).

Let CR = {FI , II}

Where:

CR: Car’s Report

FI: Fixed information

II: Instant Information

Γ_{Gf} (FI , II) <≈ $\Gamma_{FI,}$ Γ_{II}

Γ_{Pcr} (<MtS>), (FI , II) ↜ Γ_{Pfr} ([MtS]), (<MtS> , P1)

// To get the IP address of the Rse currently has an active connection with the Vehicle

Γ_{Get_RSEIP} (RseIP), (*) <≈ Γ_{B1_RT}

Where:

*: means the active connection RSE

Γ_{B1} ~S> β_{A1}

When an RSE receives an MtS message, it would analyze message and would get the CR to keep a copy of it in the RSE's VT table and to insert it into an MtT message frame and unicast it to the Co_D.

β_{A1} <S~ Γ_{B1}

β_{Mtc} (MtS , <_data>) , (<DATA>) $\nearrow$ β_{Paz} (MtS, Hp, SpC) , (MtS, <_data>)

β_{Pcr} (<MtR>) , (Rsq , SpC) $\nearrow$ β_{Pfr} ([MtR]) , (<MtR> , P2)

β_{Pcr} (<MtT>) , (CR, Rsq) $\nearrow$ β_{Pfr} ([MtT]) , (<MtT> , P3)

β_{A1} ~T> α_{A1}

Finally, the Co_D receives the vehicle's report and immediately it will create a SQL syntax which will be forwarded to the Database server. The purpose is to keep a copy of that CR and the RSE's ID for that it was sent from, in the database server's storage.

α_{A1} <T~ β_{A1}

α_{SQL} (CR,Rsq) ≈> α_{A1_VT}

α_{SQL} (<SQL_Ins_Stmnt>), (CR, Rsq) $\nearrow$ α_{Pfr} ([SQL_Ins_Msg]), (<SQL_Ins_stmnt>)

Where:

SQL_Insert_stmnt: SQL statement to insert a record into a table.

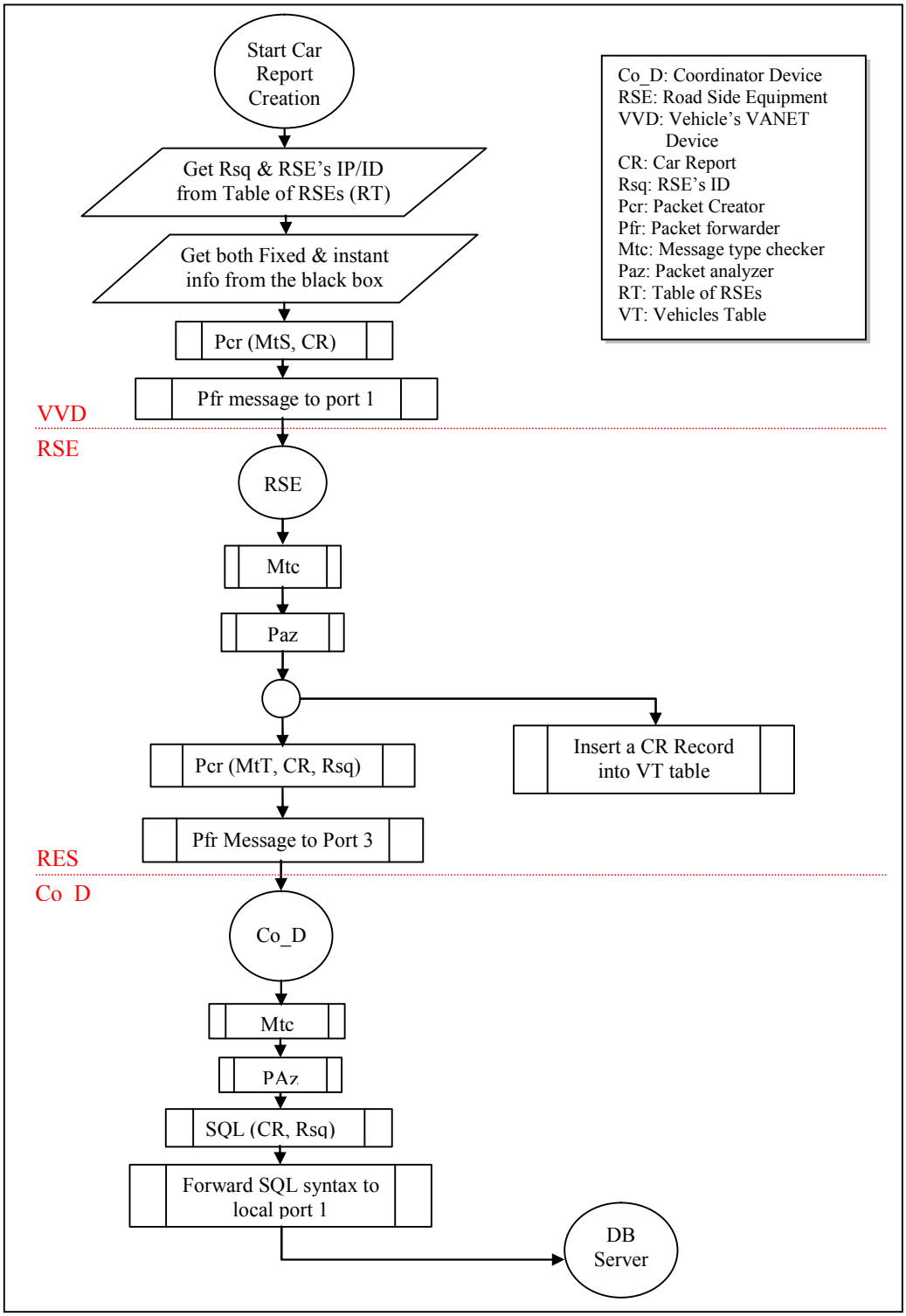

Figure 4.8: Speed Control & High Way Monitoring – Vehicle's Report Creation

4.3 Case Study–2: Remote car locating & sending a Service request / Function message

As for the Second case study "Remote car locating & sending a service / order message", See Figure 4.9, we got a Highway of two sides, the one on the right is going north while the other is going the opposite way. Each of the two ways has two lanes.

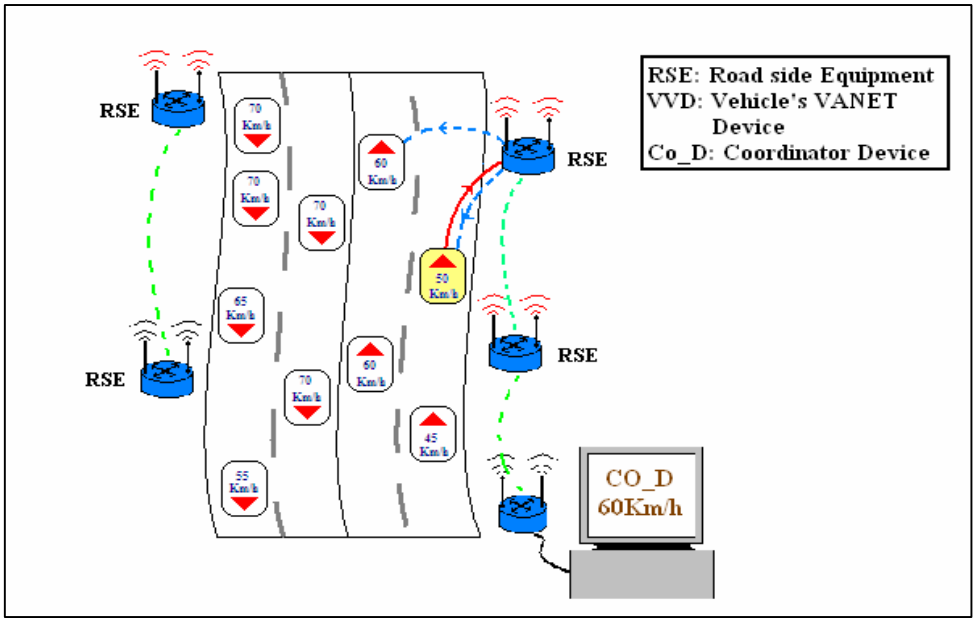

Figure 4.9: Case Study 2: Remote car locating & sending a service / order message

On the side of each road there is a set of Equipments (Road side Equipments), those are connected to each other and have connected to an administrative device (Coordinator device) that has the ability to reach and access any of those Road side equipments (RSEs) as well as the vehicles' devices (VVD).

In this scenario, assume that someone has reported to a police station or to any Coordinator office, that his car (The Yellow in Figure 4.9) was stolen, providing them the details of his car (Some of the fixed information of his vehicle's device - VVD) so they can locate the car.

In order to locate the car, the coordinator device officer uses some of the information provided by the victim to find the car either by looking in the common database for the last report received from the car or broadcast a Car_Ping message within a specific system on a specific highway. So the car will declare about itself by reporting its status to the coordinator device. The last solution which can be used only if the victim is sure that his car is still on the same road (within the coverage of the system).

The first solution is the most efficient in case that the time of the accident was unknown because the victim will not be able to know whether his vehicle still on the same road or passed already.

After locating the car, the coordinator device officer would send a Unicast function message to the car to stop it or just to limit its speed or do any other functions.

To present this scenario mathematically, we use our formal specification language to define each device and the connection type between them and their configuration. Finally we show how exactly the messaging between the devices.

4.3.1 Participated devices definitions

In this scenario, three devices are participating in; these are; Coordination device (which is the master of everything happening on the road that it's responsible about), Road Side Equipment (Which is forwarding and propagating the messages between the coordinator and vehicle devices), and Vehicles VANET Devices (these resides inside the vehicles, its job is to receive the messages propagated by the RSE and translate it to take the right required action). All of the three devices details, are just the same as the ones used in the first case study.

4.3.2 Communications between participated devices

- Connections between Co_D & RSE devices can be wire, wireless or any other communication media type, while Connection between RSE & VVD devices should be wireless.

  α_A >--< β_A: Wire, Wireless, or others

  β_A >--< Γ_B: Wireless

- Connections between Co_D-To-RSE & between RSE-To-RSE devices should be available 24 hours a day for 7 days a week.

  α_A >--< β_A_Availibility: 24/7

  β_{Ax} >--< β_{Ax+1}_Availibility: 24/7

- Connection between RSE & VVD devices can be Active or In-Active connections.

  β_A >--< Γ_B_RStatus: Active or Inactive

- VVD can have only one active connection with one RSE device at a time.

  Γ_B >--< β_A_RStatus: Active

- RSE devices can communicate each other.

  $\beta_A \Leftrightarrow \beta_A$

- Co_D – To – RSE – To - RSE connections supposed to have IP/MAC address filters to prevent any hacking attempts. As a result of that, we do not need for authentication between Co_D & RSE, or between RSE and another.

  Γ_B >--< β_A_IPFilter = True

  Γ_B >--< β_A_MACFilter = True

  α_A <**T**> β_A

  β_{Ax} <**T**> β_{Ax+1}

- RSE – To - VVD – To – RSE connections should be secured using some kind of authentication process.

  $\beta_A^{g} <\mathbf{T}> \Gamma_B^{k}$

- Co_D can communicate RSE & RSE devices can communicate Co_D devices when required:

  $\alpha_A \Leftrightarrow \beta_A$

- So, Co_D can reach VVD devices only through RSE devices on the side of the highway.

  $\alpha_A \Leftrightarrow \beta_A \Leftrightarrow \Gamma_B$

- At the same time, Co_D devices have a full control on RSE & VVD devices:

  $\alpha_A \Xi \beta_A , \Gamma_A$

- RSEs are connected to each other so they can reach each other:

  $\beta_{Ax} \Leftrightarrow \beta_{Ay}$

4.3.3 Participated messages

In this scenario, four messages are needed to implement the full procedure of the experiment, these are; MtU (Function Message), MtV (Last Destination - Function Message), MtW (Reply – Function Message), and MtX (Last Destination - Reply – Function Message).

MtU: α_A | β_{Ax-1} , β_{Ax} , MU, < MtU, Hp, SPN#, Value, XXX >

Where:

- MtU – Message type Code (10101) - Function Message
- MU: Multi-hop Unicast
- Hp – Hop count
- SPN# – Only the stolen car's Plate number
- Value:
    - Value = 0 → run function "Stop_Car"
    - Value = 1 → XXX represents the SpC.
    - Value = 2 → Run function "Ping_Car"
    - Value = x → For more functions
- XXX: see Value = 1.

MtV: β_A, Γ_B , SU, < MtV, SPN#, Value, XXX >

Where:

- MtV – Message type Code (10110) - Last Destination - Function Message
- SB: Single-hop Broadcast
- SPN# - Stolen car's Plate number
- Value:
    - Value = 0 → run function "Stop_Car"
    - Value = 1 → XXX represents the SpC.
    - Value = 2 → Run function "Ping_Car"
    - Value >= x → For more functions
- XXX: see Value = 1.

MtW: Γ_B, β_A, SU, < MtW, SPN#, Value >

Where:

- MtW - Message Type (10111) - Reply – Function Message
- SU: Single-Hop Unicast
- SPN# - Stolen car's Plate number
- Value:
    - Value = 0 → function "Stop_Car" was done successfully
    - Value = 1 → Access was denied. *It's the negative answer to any service.*
    - Value = 2 → Positive Reply - function "Ping_Car"
    - Value = x → Reply for other functions

MtX: β_A, α_A, SU, < MtX, Rsq, SPN#, Value >

Where:

- MtX - Message Type (11000) - Last Destination - Reply – Function Message
- SU: Single-hop Unicast
- Rsq – the RSE's sequence which directly received the reply from the VVD
- SPN# - Stolen car's Plate number
- Value:
    - Value = 0 → function "Stop_Car" was done successfully
    - Value = 1 → Access was denied.
    - Value = 2 → Positive Reply - function "Ping_Car"
    - Value = x → Reply for other functions

For more details about the used messages in this scenario, we write the following Messages Cards, these will tell every detail about the messages. These cards should be attached as an appendix with the formal specification for the system we describe. However we are going to show them here.

1- Message type-U Definition Card

Message Name: MtU Message ID: M#22
Description: **Function Message**

Packet Type: **Multi-Hop-Unicast**
Source: $\boldsymbol{\alpha_A}$ Destination: $\boldsymbol{\beta_A}$

Reply: **MtX**
Message length: **5 (Fields)**
Message Fields: **M#21, Hp, SPN#, Value, XXX**

Details:

- Its meaning depends on the **Value**:
  - Value = 0 → run function "Stop_Node"
  - Value = 1 → XXX represents the SpC.
  - Value = 2 → Run function "Ping_Node"
  - Value = x → For more functions

2- Message type-V Definition Card

Message Name: **MtV** Message ID: **M#23**
Description: **Last Destination - Function Message**

Packet Type: **Single-Hop Unicast**
Source: $\boldsymbol{\beta_A}$ Destination: **Γ**

Reply: **MtW**
Message length: **4 (Fields)**
Message Fields: **M#22, SPN#, Value, XXX**

Details:

- Its meaning depends on the **Value**:
  - Value = 0 → run function "Stop_Node"
  - Value = 1 → XXX represents the SpC.
  - Value = 2 → Run function "Ping_Node"
  - Value = x → For more functions

3- Message type-W Definition Card

Message Name: **MtW** Message ID: **M#24**
Description: **Reply to MtV – Last Destination - Function Message**

Packet Type: **Single-Hop Unicast**
Source: **Γ** Destination: $\boldsymbol{\beta_A}$

Message length: **3 (Fields)**
Message Fields: **M#23, SPN#, Value**

Details:

- Its meaning depends on the **Value**:
  - Value = 0 → function "Stop_Node" was done successfully
  - Value = 1 → Access was denied. *It's the negative answer to any service.*
  - Value = 2 → Positive Reply - function "Ping_Node"
  - Value = x → Reply for other functions

4- Message type-X Definition Card

Message Name: **MtX** Message ID: **M#25**
Description: **Reply to MtU – Function Message**

Packet Type: **Single-Hop Unicast**
Source: $\boldsymbol{\beta_A}$ Destination: $\boldsymbol{\alpha_A}$

Message length: **4 (Fields)**
Message Fields: **M#24, Rsq, SPN#, Value**

Details:

- Its meaning depends on the **Value**:
  - Value = 0 → function "Stop_Node" was done successfully
  - Value = 1 → Access was denied. *It's the negative answer to any service.*
  - Value = 2 → Positive Reply - function "Ping_Node"
  - Value = x → Reply for other functions

4.3.4 The Highest-Level Message flow Specification

Let's specify the scenario at the highest level (Black-Boxes Level).

α_{A1} ~U> β_{Ax} // Co_D sends a *Function message* to RSE that carries the ID x.

β_{Ax} ~V> Γ_{B1} // RSE(x) passes the *Function message* to the Required vehicle-1.

$\Gamma_{B1} \triangle \beta_{Ax}$ // Vehicle-1 Stops within the coverage area of β_{Ax}.

Γ_{B1} ~W> β_{Ax} // Vehicle-1 replies the status of the *function* implementation to the instant active RSE.

β_{Ax} ~X> α_{A1} // RSE forwards the function implementation results report to Co_D.

Till now, we expressed and showed the highest level specification for the scenario. Now let's show how exactly the messages are being exchanged among the different devices and illustrate how they been analyzed inside each device.

4.3.5 Message Flow of Case Study-2

Figure 4.10 illustrates the stage after finding the location of stolen car's from the Co_D's database or the common database. It shows the four messages flow between the participated devices starting by the Co_D sends out the function message with a value of Zero (Stop_Car) as a Multi-Hop Unicast to the RSE that carries the ID: x (RSE(x)), that message type is MtU which will travel through all the RSE devices between the two of them. When the RSE(x) receives an incoming message, it analyzes that message to find its type is MtU then get the carried information (Car's ID & Value) , encapsulates the information within an MtV message type and unicast it to the stolen car within its coverage area.

When the stolen car receives a MtV message with the value zero, two major operations will be sparked; the first operation will be calling the function *Stop_Car* which will reduce the car's speed till it stops completely then the vehicle will start to create a reply message telling whether the Car_Stop function was done successfully (Value = 0) or not (Value = 1), encapsulates it by an MtW message type and unicast it to the active

RSE at that moment. The RSE will unicast the MtW message contents (as MtX) to the Co_D which will analyze the incoming MtX message and prints the reply message result on the user interface. See Figure 4.11 which shows the flow chart for the whole operation starting by the Co_D sends out an MtU message till the function message been received by the vehicle. The second sparked operation is creating a vehicle report and unicast it to the active RSE. The seconds operation can be seen in case study-1 Figure 4.8.

The Figure 4.11 shows more details about what is going on inside each device when it receives a specific message, how will the message been processed by that device's functions, how those devices functions are making the use of the data carried by the incoming message, and what actions will be taken as a result of the incoming message.

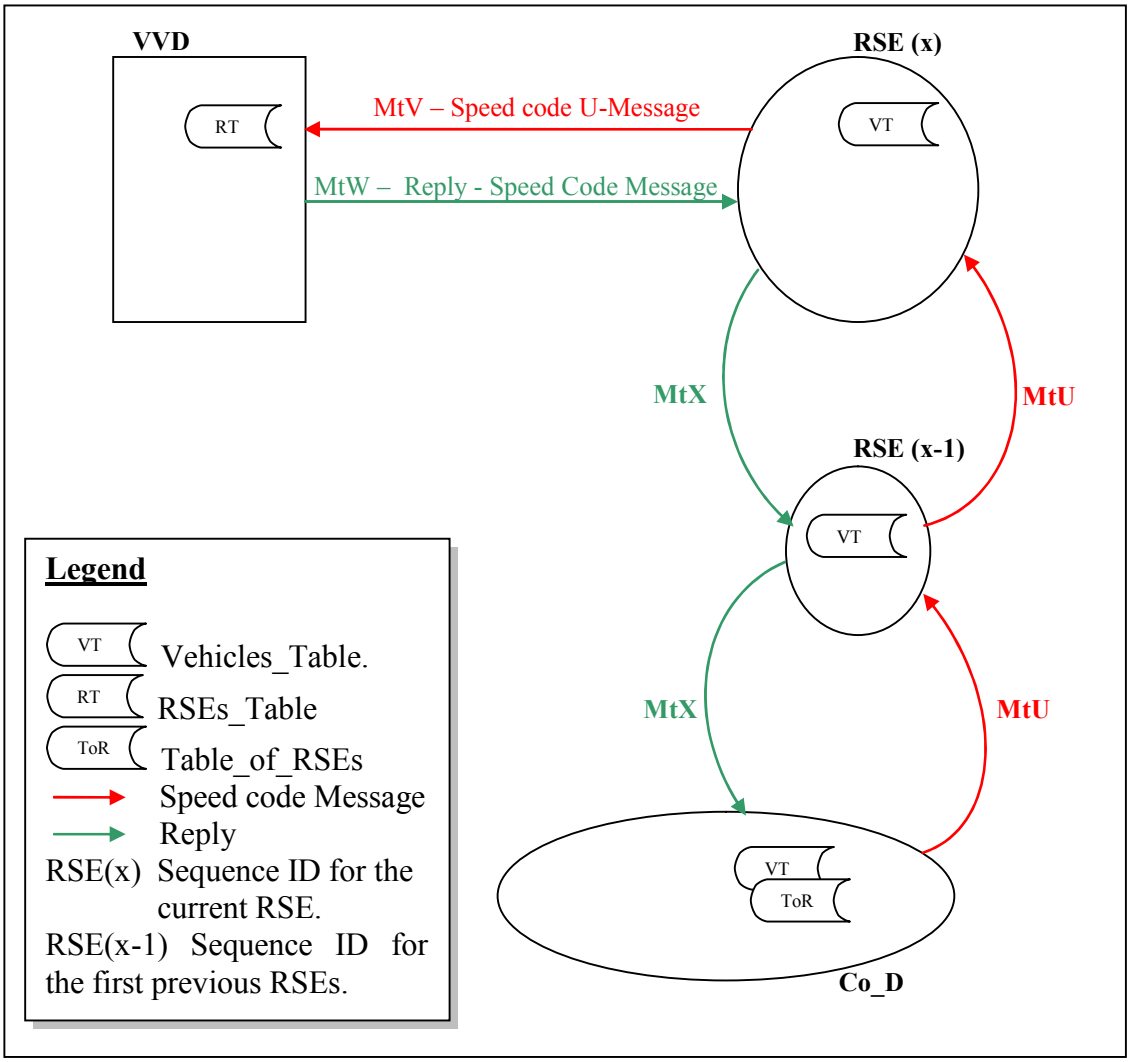

Figure 4.10: Remote car locating & sending a Service / Function message – Messages Flow

4.3.6 A Lower-Level Scenario Specification

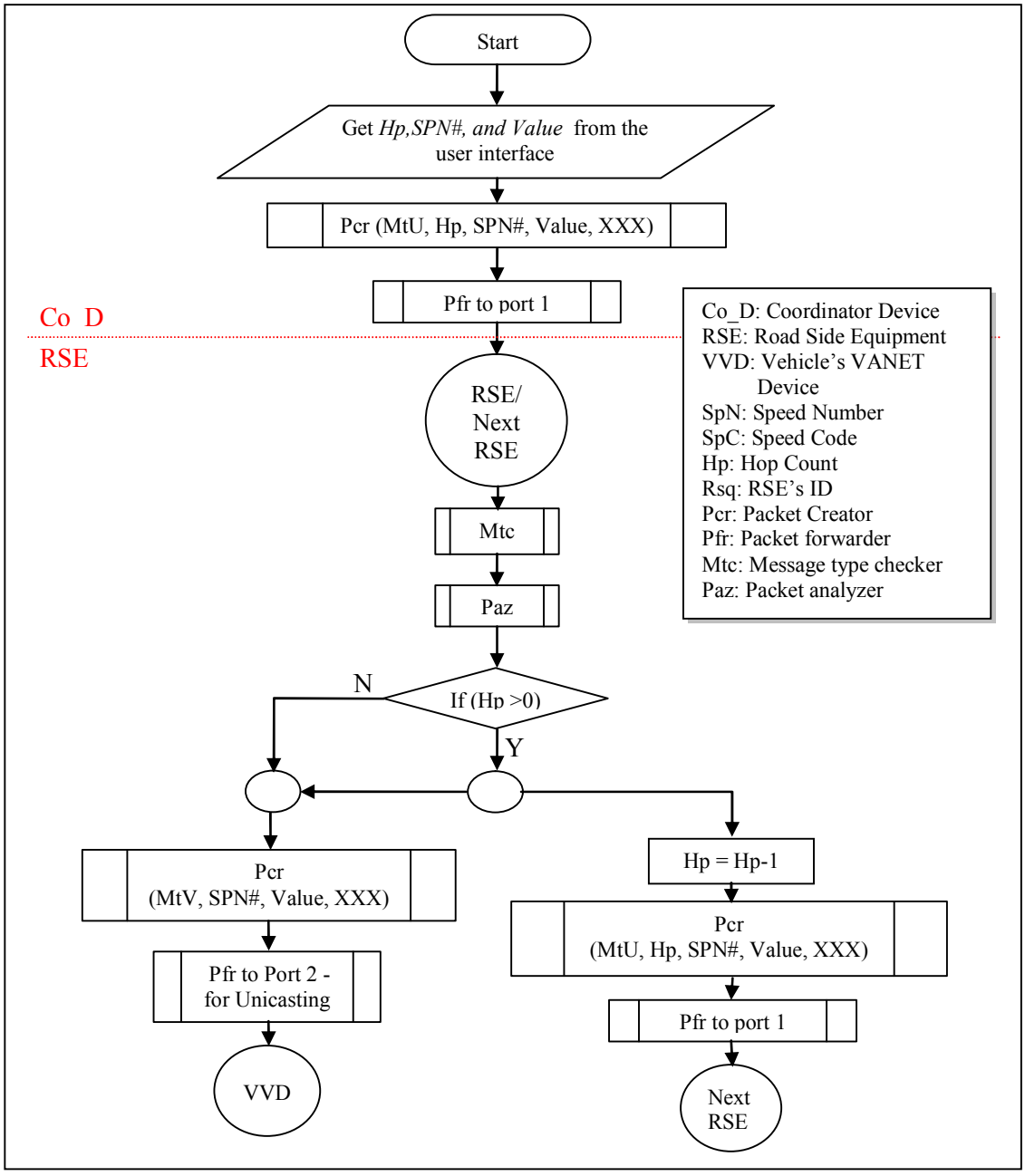

Figure 4.11: Speed Control & High Way Monitoring – Messaging Algorithm

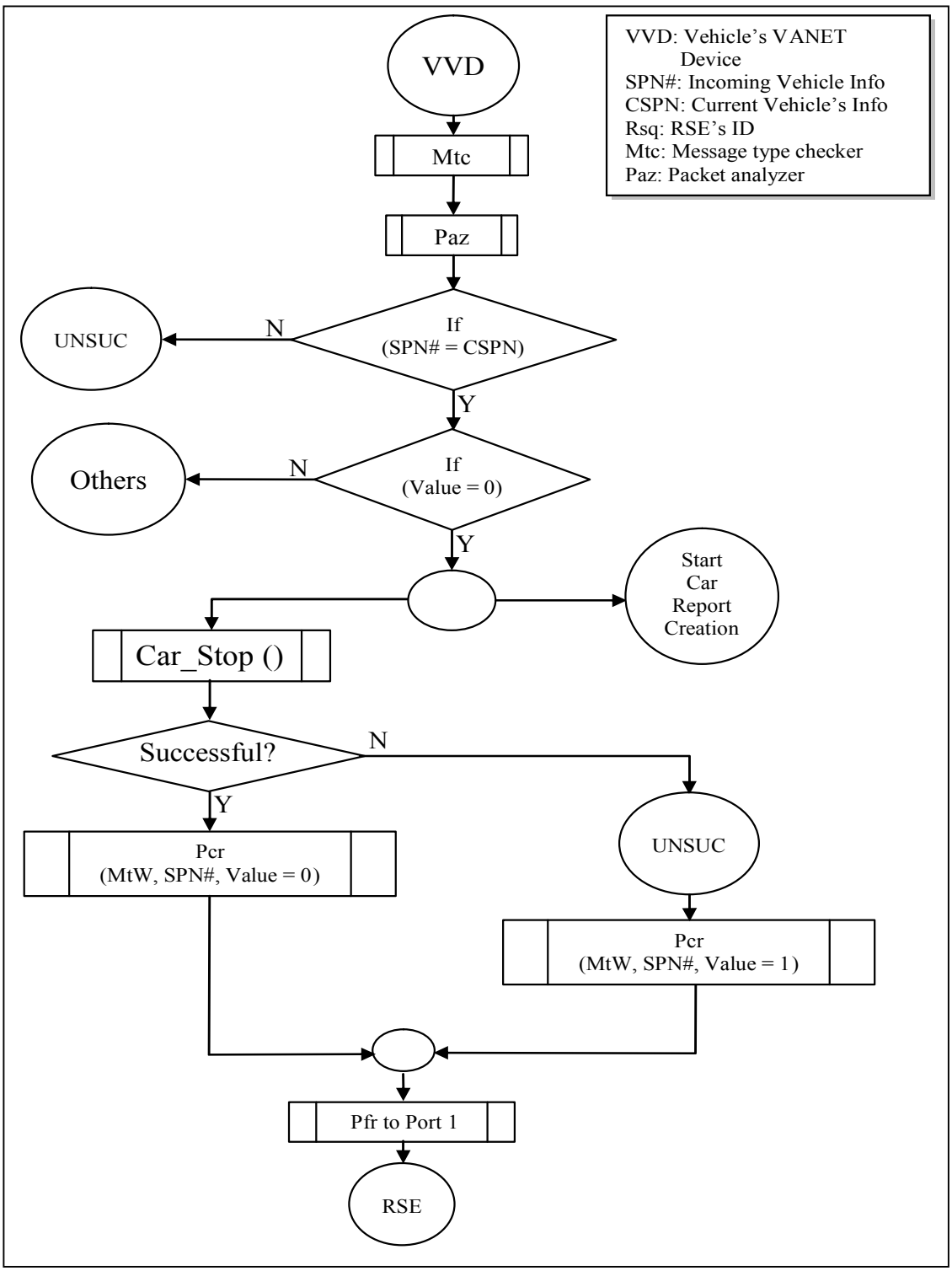

Figure 4.11.Continue 1: Speed Control & High Way Monitoring – Messaging Algorithm

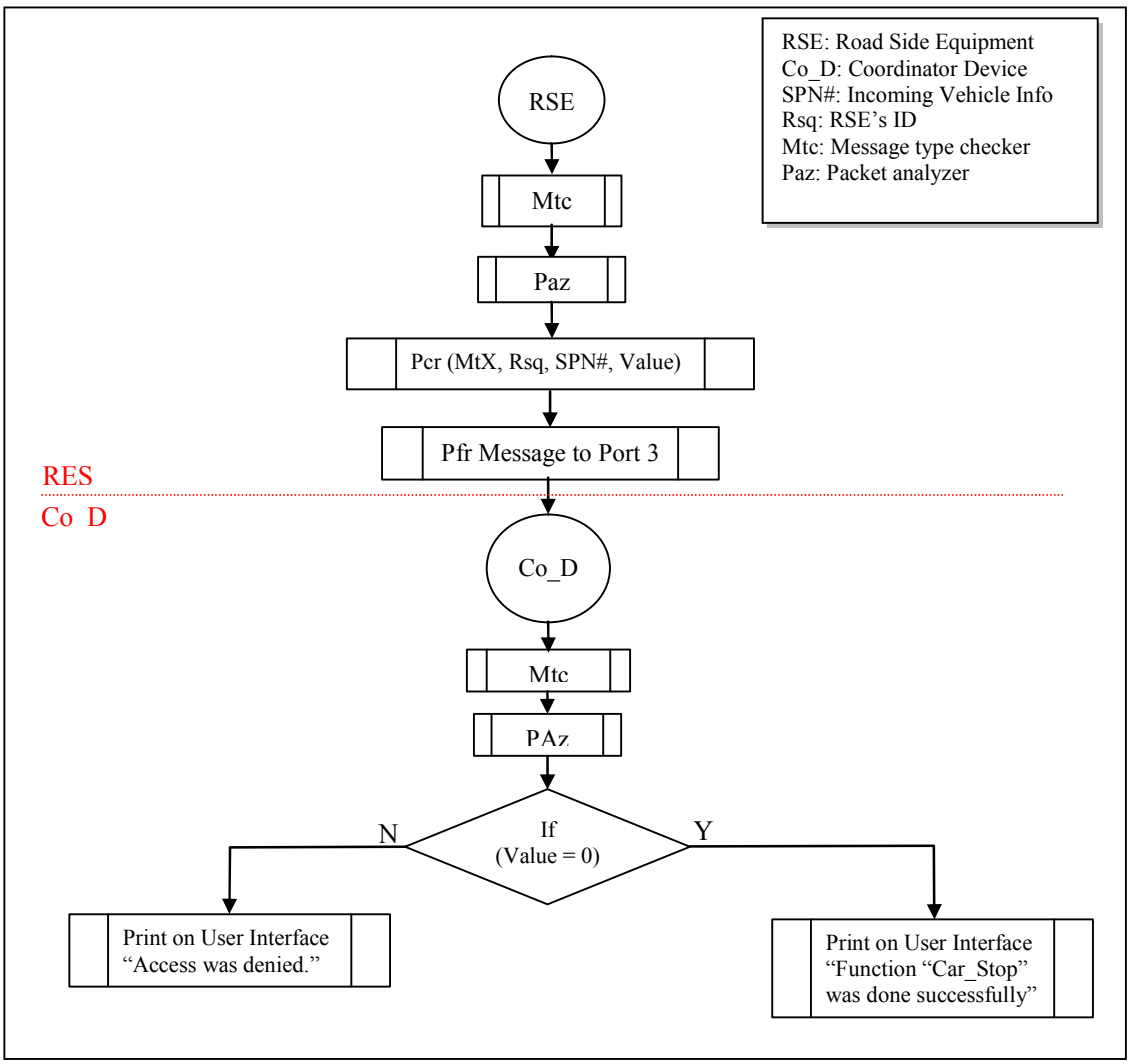

Figure 4.11.Continue 2: Speed Control & High Way Monitoring – messaging algorithm

Let's assume that we have the two highways shown before in Figure 4.9, the allowed speed on the right side is 60Km/h, so the Co_D administrator enters, through the Co_D interface; the allowed speed as a number, the Rsq (RSE's ID) that specifies which RSE the message should be start propagated to vehicles from, and the number of hops, this number tells how many RSEs after the first one should receive the allowed speed message to propagate. See the following Scenario initial setup:

Let α_{A1} = Co_D1

α_{A1}_P1_IP: *192.168.100.1/30*

Let β_{A1} = RSE1

β_{A1}P = {P1, P2, P3}

P1_IP: *192.168.100.3/30*

P2_IP: *192.168.1.1/24*

P2_DHCP: *True – Full range*

P3_IP: *192.168.100.2/30*

Let β_{A2} = RSE2

β_{A2}P = {P1, P2, P3}

P1_IP: *192.168.100.5/30*

P2_IP: *192.168.2.1/24*

P2_DHCP: *True – Full range*

P3_IP: *192.168.100.4/30*

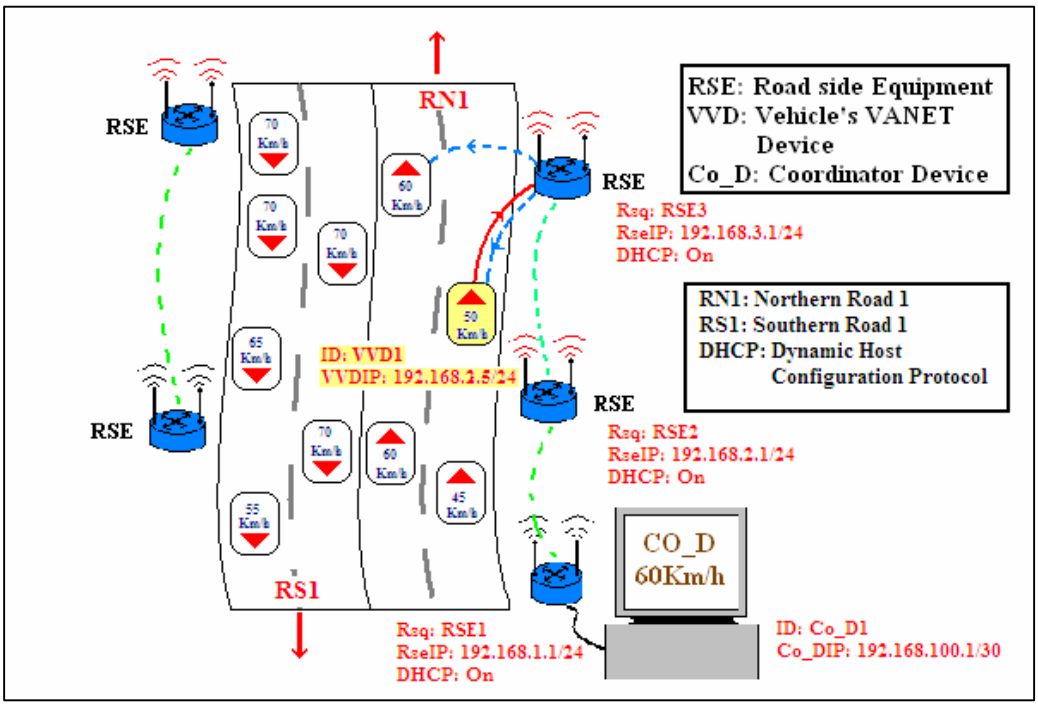

Figure 4.12: Study Case 2 - Scenario setup

Let RN1_Allowed_Speed: *60Km/h*

Let RS1_Allowed_Speed: *70Km/h*

Let β_{A3} = RSE3

β_{A3}P = {P1, P2, P3}

P1_IP: *NULL*

P2_IP: *192.168.3.1/24*

P2_DHCP: *True – Full range*

P3_IP: *192.168.100.6/30*

Let Γ_{B1} = VVD 1 // The stolen car

Γ_{B1}P = {P1}

P1_IP: 192.168.2.5/24

Γ_{B1} --: RN1

Γ_{B1} ◄ RSE2

Γ_{B1}_Speed: 50 Km/h

Γ_{B1} ► RSE3

The Scenario starts when a car's owner declares to any Co_D office that his car was stolen and provides the office with his car information such as his vehicle's plate number, his VVD's serial number, …etc.

The Co_D administrator starts looking for the car in his Co_D's native tables to check whether the car is within his system's coverage or not. If not, then he sends a query to the common database asking for the current location or the last report was received from the stolen car.

After specifying under which system coverage the stolen car is, the administrator and the car's owner decide to stop the car remotely. The administrator enters 4 entries into the Co_D device through its interface:

α_{Gf} (SPN#, Rsq, Hp, value) <≈ α_{A1}_Int

After the administrator enters those 4 values, they will be forwarded to the Co_D's internal functions to be processed. The first value SPN# (Stolen car Plate Number) can be any of the car's information other than the plate number. The second value entered by the administrator is the Rsq value which will be useful to get the IP address of the RSE device that Co_D should unicast the MtU message to, that IP can be found in the native table ToR (Table of RSEs) or in one ToR table of another Co_D device, and this can be done by forwarding the Rsq to the *Get_RSEIP* function of the ToR Table that will get the equivalent RSE IP.

In case the RSE resides within the responsibility of the same requester Co_D:

α_{A}_Get$_{_RSEIP}$ (RseIP) , (Rsq) $<\approx \alpha_{A\_ToR}$

While, if the RSE resides under the responsibility of a system other than the requester's, then we have to mention the requester's ID and the requested from ID.

α_{A1}_Get$_{_RSEIP}$ (RseIP) , (Rsq) $<\approx \alpha_{Ax\_ToR}$

The third input is the operation option value; this will decide what kind of operation is requested to be done. In MtU message type, if Value = 0 then the desired operation is to stop the car, and this what we are going to do, stop the stolen car remotely. But if we want to limit the car's speed only, then the administrator should use the value 1 and enter the required speed as well.

Now the Co_D has three values; Hp, SPN#, and the Operation option Value, these would be forwarded to the Pcr function to create an MtU message and forward it to the Pfr function that would encapsulate it within a packet that has the RSE IP as its destination IP, and then forward it to Port 1 of the Co_D device.

α_{Pcr} (<MtU>) , (Hp, SPN#, Value) $\nearrow$ α_{Pfr} ([MtU]) , (<MtU>, $RseIP_{Ax}$, P1)

Where

$RseIP_{Ax}$: The Ip address for RSE under which's coverage area the stolen car is under.

α_{A1} ~U> β_{Ax}

The MtU message on the network as a data stream flow, a group of ones and zeros, with the help of the routing protocols used in the VANET network, traveling from RSE to another till it reaches the destination one, there it will be processed. When the destination RSE receives the data stream, it checks the first bits of that data stream to check the type of the incoming message, and this is done by the Mtc (Message Type Checker) function.

After specifying the message type, Mtc function forwards the rest of the message's data fields (which still as data stream only) to the Paz (Packet Analyzer) function, this will analyze the data stream into specific data values to make the use of them. So for the MtU message, it will be analyzed and understood as 4 fields; Hp, and SPN#, Value, and XXX respectively. XXX is a value which's type depends on the operation option value. In our scenario, the value will be equal to zero (Stop the car) so XXX's value will be NULL (No need for it). But if we want to limit the car's speed, then the administrator needs to give the required speed, which will be represented by XXX.

RSE checks the value of the incoming Hp whether it's equal to zero or not to decide what to do with the incoming data. If Hp more than Zero, then it would encapsulate the SPN# and the value within an MtV message, get's the IP of the stolen vehicle by mapping the SPN# into its equivalent IP (from its native VT) and starts unicating to the stolen vehicle through port 2, as well as, it would forward through port one a copy of the incoming MtU message after decreasing the Hp by 1 to the next RSE, and this to make sure that the message would reach the required vehicle. While if Hp equals to Zero, then it would not forward anything to the next RSE, just keeps unicasting the MtV messages through port 2 to the vehicle hopping it's still within its coverage area.

β_{Ax} <U~ α_{A1}

β_{Mtc} (MtU , <_data>) , (<DATA>) ↯ β_{Paz} (MtU, Hp, SPN#, Value, XXX) , (MtU, <_data>)

Where: <DATA> is the full incoming data stream

<_data> is the rest of the unanalyzed data of the incoming stream.

β_{Pcr} (<MtV>) , (SPN#, Value, XXX) ↯ β_{Pfr} ([MtV]) , (<MtV> , P2)

β_{Ax} ~V> Γ_{B1}

(Hp > 0) → Pcr (<MtU>) , (Hp-1, SPN#, Value, XXX) ↯ Pfr ([MtU]) , (<MtU> , $RseIP_{Ax}$, P1)

Where $RseIP_{Ax}$: The Ip address for the RSE which has the last the report received from the stolen vehicle

β_{Ax} ~U> β_{Ax+1}

Now a vehicle receives an MtV message, it analyzes it and understands it as SPN#, Value, XXX fields. The vehicle's VANET device would compare the incoming SPN# value (carried by the MtV message) with its own (they probably the same but just for double check), if they are equal, then it would read the rest of the incoming data and check for the next field, The Value, if Value = 0 then it would start creating an instant vehicle status report and send it back to the Co_D, See Figure 4.8 in case study 1, and call the *Car_Stop* Function to stop the car.

In case the incoming SPN# information are not matching the vehicle's, or the Car_Stop function wasn't successfully done (For some failure reason), then the vehicle's VANET Device would start to create a negative reply (MtW message with a Value of 1) and send it to the current active RSE. Other wise it would send a positive reply (MtW message with a Value of 0) to the current active RSE.

Γ_{B1} <V~ β_{Ax}

Γ_{Mtc} (MtV , <_data>) , (<DATA>) ⇝ Γ_{Paz} (MtV, SPN#, Value, XXX) , (MtV, <_data>)

Let CSPN# = The Stolen Vehicle Information (Γ_{B1})

(SPN# = CSPN#) → (Value = 0) → **Label: Start Car Report Creation**, ⇝Car_Stop()

(Car_Stop()) → Pcr (<MtW>) , (SPN#, Value = 0) ⇝ Pfr ([MtW]) , (<MtW> , $RseIP_{Ax}$, P1)

!(Car_Stop()) → **Label: UNSUC**

(SPN# = CSPN#) → **!**(Value = 0) → **Label: Others**

!(SPN# = CSPN#) → **Label: UNSUC**

Label: UNSUC

Pcr (<MtW>) , (SPN#, Value = 1) ⇝ Pfr ([MtW]) , (<MtW> , $RseIP_{Ax}$, P1)

// To get the IP address of the Rse which currently has an active connection with the Vehicle:

Γ_{Get_RSEIP} (RseIP), (*) <≈ Γ_{B1_RT}

Where:

*: means the active connection RSE

Γ_{B1} ~W> β_{Ax}

When an RSE receives an MtW message, it would analyze the message and would encapsulate the values: Rsq, SPN#, and the operation status reply value within an MtX message frame and unicast it to the Co_D.

β_{Ax} <W~ Γ_{B1}

β_{Mtc} (MtW , <_data>) , (<DATA>) $\nearrow$ β_{Paz} (MtW, SPN#, Value) , (MtW, <_data>)

β_{Pcr} (<MtX>) , (Rsq, SPN#, Value) $\nearrow$ β_{Pfr} ([MtX]) , (<MtX> , P3)

β_{Ax} ~T> α_{A1}

Finally, the Co_D receives the unicast MtX message and analyze it its fields to get the Operation status reply (Value). If Value equals to Zero, then it would print a message on its user interface screen "From <*SPN#*>; Function 'Car_Stop' was done successfully." While if the value was equal to one, then the message that would be shown on the user interface screen is "From <*SPN#*> Access was denied."

α_{A1} <T~ β_{Ax}

α_{Mtc} (MtX , <_data>) , (<DATA>) $\nearrow$ α_{Paz} (MtX, Rsq, SPN#, Value) , (MtX, <_data>)

(Value = 0) $\rightarrow$ α_{Print} ("From <*SPN#*>; Function 'Car_Stop' was done successfully.")

!(Value = 0) $\rightarrow$ α_{Print} ("From <*SPN#*> Access was denied.")

4.4 Case Study – 3: Suspect car instant termination

For the third case study "Suspect car instant termination", See Figure 4.13, we got a Highway of two sides, the one on the right is going north while the other is going the opposite way. Each of the two ways has two lanes.

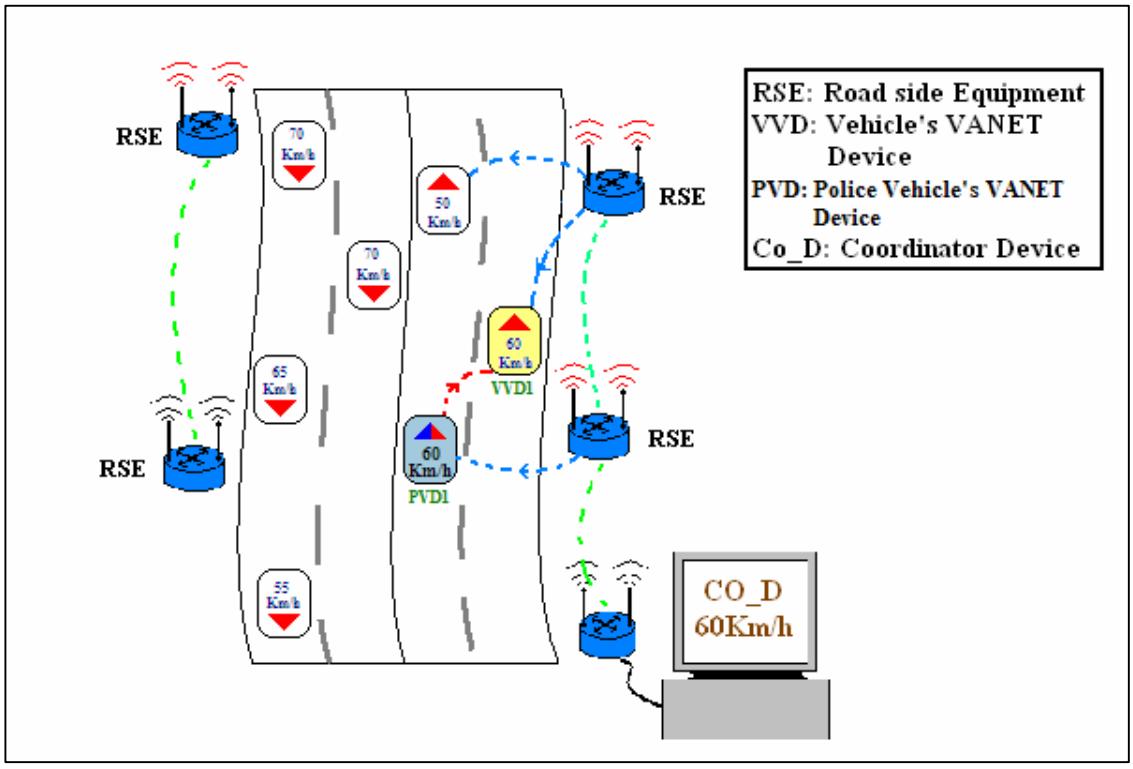

Figure 4.13: Case Study 3: Suspect car instant termination

On the side of each road there is a set of Equipments (Road side Equipments), those are connected to each other and have connected to an administrative device (Coordinator device) that has the ability to reach and access any of those Road side equipments (RSEs) as well as the vehicles' devices (VVD).

In this scenario, assume that we have a suspected car (The Yellow in Figure 4.13), the police want to stop that suspect vehicle peacefully by accessing the suspect's VVD and passes an order to stop it. The suspect's VVD needs some credentials to be accessed another device. That information can be gotten from the RSE instantly the suspect's VVD within coverage area.

To present this scenario mathematically, we use our formal specification language to define each device and the connection type between them and their configuration. Finally we show how exactly the messaging between the devices.

4.4.1 Participated devices definitions

In this scenario, four devices are participating in; they are; Coordination device (which is the master of everything happening on the road that it's responsible about), Road Side Equipment (Which is forwarding and propagating the messages between the coordinator and vehicle devices), and Vehicles VANET Devices (these resides inside the vehicles, its job is to receive the messages propagated by the RSE and translate it to take the right required action). These were the same devices used in the last two case studies, while in this case study we have an additional device that is called Police vehicle's VANET Device (which is located inside any police car and has some administration abilities).

For the first three devices details, we illustrated in the first case study. For the Police vehicle's VANET Device (PVD), it has 4 main contents; Set of Functions, Set of Tables, Some registers, and set of ports. The functions set contains six different functions; Γ_{Mtc} (checks the type of the incoming message and decides the destiny of the carried data), Γ_{Paz} (Analyzes the incoming data stream according to the message type into usable data), Γ_{Sf} (Sets some values as required), Γ_{Gf} (Get some values for some functions when required), Γ_{Pcr} (Creates different messages types as required), Γ_{Pfr} (Forwards the messages to the required ports). The set of tables has one member; RT (lists all the RSEs instantly receiving messages from), SCT (Lists all the cars those were reported as suspect cars). The set of registers has one member, that is; SCT_Flag (indicates this vehicles is considered as a suspected vehicle), and finally, the set of the Vehicle's VANET Device ports, which has one main port; P1 (Port one, Connected to one of the RSEs or to a police vehicle).

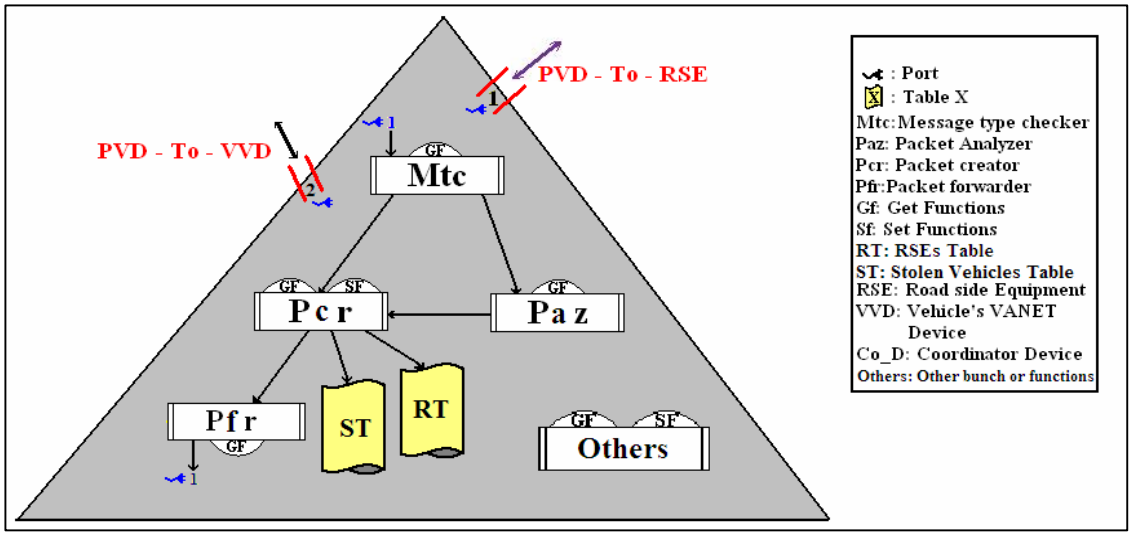

Figure 4.14 : Police vehicle's VANET Device (PVD)

Let $\Gamma_A = \{ \Gamma_A F, \Gamma_A T, \Gamma_A V, \Gamma AP, \Gamma_A DAC\}$

Where:

Γ_A: Police vehicle's VANET Device (PVD)

$\Gamma_A F$: PVD's Functions

$\Gamma_A T$: PVD's Tables

$\Gamma_A V$: PVD's Variables

$\Gamma_A P$: PVD's Ports

$\Gamma_A DAC$: PVD's Digital to Analog Converter (attached device)

And Let:

$\Gamma_A F = \{ \Gamma_{Mtc}, \Gamma_{Paz}, \Gamma_{Sf}, \Gamma_{Gf}, \Gamma_{Pcr}, \Gamma_{Pfr}$, Car_Ping(),Speed_Limit(SpC), Car_Stop()$\}$

Where:

Γ_{Mtc}: Message type checker function

Γ_{Paz}: Packet / Massage Analyzer

Γ_{Sf}: set of SET functions

Γ_{Gf}: Set of GET Functions

Γ_{Pcr}: Packet / Message Creator

Γ_{Pfr}: Packet / Message forwarder (specifies to which port)

Car_Ping():a functions can be used to spark the car to report its status

Speed_Limit(SpC): a function to limit the speed of the vehicle to the required speed (SpC).

Car_Stop(): a function to stop the car.

$\Gamma_A T = \{\Gamma_{RT}[e][f], \Gamma_{ST}[g][h]\}$

Where:

$\Gamma_{RT}[e][f]$: lists all the RSEs instantly receiving messages from. It has e of rows and f of columns, where both of e and f are positive integers.

$\Gamma_{ST}[g][h]$: List all the suspected cars. It has g of rows and h of columns, where both of g and h are positive integers.

$\Gamma_A V = \{ \Gamma_{SCT\_Flag}, \Gamma_{FI}, \Gamma_{II}\}$

Where:

Γ_{SCT_Flag}: A flag register that indicates the suspicion of the vehicle.

Γ_{FI}: A register that keeps the vehicle's Fixed Information

Γ_{II}: A register that keeps the vehicle's Instant Information

$\Gamma_A P = \{\Gamma_{P1}, \Gamma_{P2}\}$

Where:

Γ_{P1}: The first connection port of device $\Gamma_{A,}$ connected to the active RSE.

Γ_{P2}: The second connection port of device $\Gamma_{A,}$ connects to another Γ devices.

Then:

$\Gamma_A = \{ \Gamma_{Mtc}, \Gamma_{Paz}, \Gamma_{Sf}, \Gamma_{Gf}, \Gamma_{Pcr}, \Gamma_{Pfr}$, Car_Ping(),Speed_Limit(SpC), Car_Stop(), $\Gamma_{RT}[e][f], \Gamma_{ST}[g][h], \Gamma_{SCT\_Flag}, \Gamma_{FI}, \Gamma_{II,} \Gamma_{P1}, \Gamma_{P2}\}$

Γ_A_Features:

Γ_A_Mobile =True // A PVD is a mobile device

Γ_A_SQL_Server = False // The PVD does not have the ability to query an SQL server directly, but it can reach the database server through RSE and the Co_D.

Γ_A_User_Interface = True // The PVD device has a user interface to be managed by.

Γ_A_initialize_Operations =True // The PVD does not have the ability to start messaging other devices; it's just a receiver and a replier.

Γ_A_Info =True // The PVD can be considered as an information center.

Γ_A_Routing =True // PVD has the routing ability by using the protocols listed <Routing Protocol(s) used>.

4.4.2 Communications between participated devices

In addition to the communications illustrated in the previous case studies, we got some more of them to explain regarding to the PVD device.

- Connection between RSE & PVD devices can be Active or In-Active connections.

  β_A >--< Γ_A_RStatus: Active or Inactive

- PVD can have more than one active connection; one with an RSE device, and another with any other device (In our scenario, this connection would be with a VVD).

  Γ_A >--< β_A_RStatus: Active

  Γ_A >--< X_RStatus: Active or Inactive

  Where: X is any device or any layer.

- RSE – To - PVD – To – RSE connections should be secured using some kind of authentication process.

  $\beta_A{}^g <\mathbf{T}> \Gamma_A{}^k$

- PVD– To - VVD – To – PVD connections should be secured using some kind of authentication process.

  $\Gamma_B{}^g <\mathbf{T}> \Gamma_A{}^k$

- PVD devices can have a full control on VVD devices:

  $\Gamma_A \Xi \Gamma_B$

4.4.3 Participated messages

In this scenario, sixteen messages are needed to implement the full procedure of the experiment, these are; MtA(Access Authentication Request), MtB(Reply - Access Authentication Request), MtC(PVD info. Request From RSE), MtD(Reply – PVD info. Request (MtC, MtF)), MtE(Copy of PVD info), MtF(PVD info. Request From Co_D), MtG(Car SPN# info Request), MtH(Reply - Car SPN# info Request (MtG)), MtI(Car SPN# info Request – RSE from RSE), MtJ(Reply – Car SPN# info Request (MtK, MtI)), MtK(Car SPN# info Request from Co_D), MtL(Copy of SPN#'s info), MtM(PVD – to – SVD Access Request & Speed information), MtN(PID availability enquiry), MtO(Reply - PID availability enquiry), MtP(Reply - PVD–To–SVD Access Request & Speed information).

For more details about the used messages in this scenario, we write the following Messages Cards, these would tell every detail about the messages. These cards should be attached as an appendix with the formal specification for the system we are describing. But we are going to show them here.

1- Message type-A Definition Card

Message Name: **MtA** Message ID: **M#01**
Description: **Access Authentication Request**

Packet Type: **Unicast**
Source: $\boldsymbol{\Gamma_A}$ Destination: $\boldsymbol{\beta_A}$

Reply: **MtB**
Message length: **3 (Fields)**
Message Fields: **M#01, PID, Pswd**

Details:

- Created and being sent by Γ_A when it wants to access any device on β Layer.

2- Message type-B Definition Card

Message Name: **MtB** Message ID: **M#02**
Description: **Reply to MtA – Access Authentication Request**

Packet Type: **Unicast**
Source: $\boldsymbol{\beta_A}$ Destination: $\boldsymbol{\Gamma_A}$

Reply: **None**
Message length: **2 (Fields)**
Message Fields: **M#02, Value**

Details:
Its meaning depends on the **Value**:

- **Value = 0** Access Permitted
- **Value = 1** Access Denied – Wrong Password
- **Value = 2** Waiting – PID's info was not found in Special_cars_Table (Asking Two previous **β** devices for the info, Sending **MtC**).
- **Value = 3** Access Denied – Invalid PID's info.

3- Message type-C Definition Card

Message Name: **MtC** Message ID: **M#03**
Description: **Advanced - Γ_A Information Request**

Packet Type: **Multi-Hop-Unicast**
Source: **β_{Ax}** Destination: **$\beta_{x-1/2}$**

Reply: **MtD**
Message length: **5 (Fields)**
Message Fields: **M#03, PID, Pswd, Rsq(x), Hp**

Details:

- Created by **β_A** Device and being sent to the two previous **β** devices asking them for PID's information in case of none existence in **β_A** SCT table.

4- Message type-D Definition Card

Message Name: **MtD** Message ID: **M#04**
Description: **- Reply to MtC – PVD information Request**
- Reply to MtF – PVD information Request

Packet Type: **Unicast**
Source: **α_A** or **$\beta_{x-1/2}$** Destination: **β_{Az}**

Message length: **3 (Fields)**
Message Fields: **M#04, Value, PID's info**

Details:
Its meaning depends on the **Value**:

- **Value = 0** Information found, read next bits as (PID's info)
- **Value = 1** Information found, BUT Access Denied
- **Value = 2** No information were found in Special_Cars_Table

- When a β device receives an MtD reply with values (MtD, 0, PID's info), then it would send MtB to Γ & sends a copy of the PID's info to the next 10 β devices (MtE).

5- Message type-E Definition Card

Message Name: **MtE** Message ID: **M#05**
Description: **Copy of PVD's information**

Packet Type: **Multi-hop-Unicast**
Source: $\boldsymbol{\beta_{Ax}}$ Destination: $\boldsymbol{\beta_{x+1/10}}$

Message length: **3 (Fields)**
Message Fields: **M#05, Hp, PID's info**

Details:

- Being sparked and created when a β_A device receives a positive MtD reply holding the PID's information. MtE would be sent to the next ten β devices.

6- Message type-F Definition Card

Message Name: **MtF** Message ID: **M#06**
Description: $\boldsymbol{\Gamma_A}$ **information Request**

Packet Type: **Unicast**
Source: $\boldsymbol{\beta_{x-2}}$ Destination: $\boldsymbol{\alpha_A}$

Reply: **MtD**
Message length: **4 (Fields)**
Message Fields: **M#06, PID, Pswd, Rsq(x)**

Details:

- In case of none existence of PID's ($\boldsymbol{\Gamma_A}$) info in $\boldsymbol{\beta_{x-1/2}}$**'s SCT table**, then the last would send an MtF request message to $\boldsymbol{\alpha_A}$ asking for those information.

7- Message type-G Definition Card

Message Name: **MtG** Message ID: **M#07**
Description: **Mobile Node SPN# info Request**

Packet Type: **Unicast**
Source: $\boldsymbol{\Gamma_A}$ Destination: $\boldsymbol{\beta_A}$

Reply: **MtH**
Message length: **3 (Fields)**
Message Fields: **M#07, PID, SPN#**

Details:

- PID requests full information for the mobile node SPN#.

8- Message type-H Definition Card

Message Name: **MtH** Message ID: **M#08**
Description: **Reply to MtG - mobile node SPN# info Request**

Packet Type: **Unicast**
Source: $\boldsymbol{\beta_A}$ Destination: $\boldsymbol{\Gamma_A}$

Message length: **3 (Fields)**
Message Fields: **M#08, Value, SPN#'s info**

Details:
Its meaning depends on the **Value**:

- **Value = 0** Mobile node SPN# info found, read it from Mobile node SPN#'s info field.
- **Value = 1** Waiting – SPN# was not found in Vehicles_Table (asking two previous β devices for Mobile node (Γ Device) SPN#'s info, sending **MtI**)
- **Value = 2** Mobile node SPN#'s info invalid.

9- Message type-I Definition Card

Message Name: **MtI** Message ID: **M#09**
Description: **Advanced - Mobile node SPN# info Request**

Packet Type: **Multi-Hop-Unicast**
Source: β_{Ax} Destination: $\beta_{x-1/2}$

Reply: **MtJ**
Message length: **5 (Fields)**
Message Fields: **M#10, PID, SPN#, Hp, Rsq(x)**

Details:

- This message will be created by β_{Ax} in the case of none existence of the mobile node SPN#'s information in its VT table, asking the two previous β devices to look in their VT tables.

10- Message type-J Definition Card

Message Name: **MtJ** Message ID: **M#10**
Description: **- Reply to MtI – Advanced - Mobile node SPN# info Request**
- Reply to MtK – Last resort - Mobile node SPN# info Request

Packet Type: **Unicast**
Source: α_A **or** $\beta_{x-1/2}$ Destination: β_{Ax}

Message length: **4 (Fields)**
Message Fields: **M#10, Value, PID, Mobile node SPN#'s info**

Details:

- Its meaning depends on the **Value**:
  - **Value = 0** Information found, read next bits as (Rsq(x), Car SPN#'s info)
  - **Value = 1** Waiting - No information were found in VT table
  - **Value = 2** SPN#'s info. Are invalid.

11- Message type-K Definition Card

Message Name: **MtK** Message ID: **M#11**
Description: **Last resort - Mobile node SPN# info Request**

Packet Type: **Unicast**
Source: β_{x-2} Destination: α_A

Message length: **4 (Fields)**
Message Fields: **M#11, PID, SPN#, Rsq(x)**

Details:

- This message will be created by β_{x-2} in the case of none existence of the mobile node SPN#'s information in its VT table, asking α_A to look in its own VT table & the common database storage.

12- Message type-L Definition Card

Message Name: **MtL** Message ID: **M#12**
Description: **Copy of SPN#'s info**

Packet Type: **Multi-Hop-Unicast**
Source: β_{Ax} Destination: $\beta_{x+1/10}$

Message length: **3 (Fields)**
Message Fields: **M#12, Hp, SPN#'s info**

Details:

- None.

13- Message type-M Definition Card

Message Name: **MtM** Message ID: **M#13**
Description: **Γ-Γ Access Request & a Service**

Packet Type: **Unicast**
Source: $\mathbf{\Gamma_A}$ Destination: $\mathbf{\Gamma_B}$

Reply: **MtP**
Message length: **5 (Fields)**
Message Fields: **M#13, PID, SPN#'s info, Opt, SPc**

Details:

- Its meaning depends on the **Opt** (Option), which is a value that decides whether to run a specific operation or receives a speed code.
  - Opt = # Any other specific services

14- Message type-N Definition Card

Message Name: **MtN** Message ID: **M#14**
Description: **PID ($\mathbf{\Gamma_A}$) availability enquiry**

Packet Type: **Unicast**
Source: $\mathbf{\Gamma_B}$ Destination: $\mathbf{\beta_A}$

Reply: **MtO**
Message length: **2 (Fields)**
Message Fields: **M#14, PID**

Details:

- None.

15- Message type-O Definition Card

Message Name: **MtO** Message ID: **M#15**
Description: **Reply to MtN - PID (Γ_A) availability enquiry**

Packet Type: **Unicast**
Source: β_A Destination: Γ_B

Message length: **2 (Fields)**
Message Fields: **M#15, Value**

Details:

- Its meaning depends on the **Value**:
  - Value = 0 PID available in SCT Table
  - Value = 1 PID invalid

16- Message type-P Definition Card

Message Name: **MtP** Message ID: **M#16**
Description: **Reply to MtM - Γ-Γ Access Request & a service**

Packet Type: **Unicast**
Source: Γ_B Destination: Γ_A

Message length: **2 (Fields)**
Message Fields: **M#16, Value**

Details:

- Its meaning depends on the **Value**:
  - Value = 0 Operation has done successfully
  - Value = 1 Access Denied – PID Invalid
  - Value = 2 Access Denied – SPN#'s info doesn't match the sent information.

NOTE:- According to the limitation of the thesis pages number, in the next sections we are not going to show the scenario's lower level specification using our language.

4.4.4 Messages Flow of Case Study-3

Figure 4.15 illustrates the whole operation when a police vehicle PVD-1 suspects the vehicle SVD-1 and would like to stop it. PVD-1 sends a message of type A (MtA) to the current RSE asking for the permission to access the database of that RSE to look for the full information of that suspected car. MtA has the Police car ID (PID) and the password (Pswd) for accessing the RSE. When the RSE receives the MtA message, it would analyze it and check whether it carries the correct password or not, if it's the correct password then it would check of the PID availability in its own SCT, if a match found there then it would reply with a positive MtB (Value = 0). Otherwise a Waiting MtB message (Value = 2) would be issued and sent back to the PVD-1 while creating and passing an MtC message to the two previous RSEs asking them for the PVD's Information. If any of them has those information, then it would send them within an MtD message to the Original RSE that will recheck to make sure the 2 PIDs are matching then would unicast a positive MtB to the PVD-1. But, if both of the two previous RSEs do not have that information, then the second previous RSE would create and send a unicast message (MtF) to the Co_D asking for that information. The reply to the MtF message would be and MtD message unicasted to the original RSE (Rse(x)).

Now PVD-1 got authenticated into the RSE, so now it can access and query the tables of that RSE. The police man inside PVD-1 uses the user-interface to enter the Suspected car's Plate Number (SPN#) and send an MtG message to the RSE to look for the full details for that suspected car to access it and stop it.

So the RSE receives that query message and start looking for the SPN# match in its own VT table. If a match found, then it will enclose the full information of that SPN within an MtH message then send it back to the PVD to make the use of it. While in case of missing the match, the RSE would try to get that information from the two previous RSEs or from the Co_D (as the last resort) using the query messages MtI and MtK respectively and it would get the reply within an MtJ message.

Supposedly by now, PVD-1 has the full information of SVD-1 so it can use them to authenticate into and access the suspected car and terminates it to just limit its speed. For our scenario purpose, the PVD-1 would stop SVD-1.

All that will be started when the PVD-1 receives a positive MtH message that carries the full information of the SVD-1. The police car will create an MtM message and unicast it to the suspected car. MtM message has the required service option (in our case study Opt = 0 which means Stop_the_car) that will not been implemented till the authentication process takes place.

The authentication process has two stages. The first stage is done by sending a query message (MtN) to the RSE asking for the validity of the police car that holds the PID, the MtM message came from. If the PID was a valid, then a positive MtO reply message would be created inside the RSE and unicasted back to SVD-1.

Here the second stage starts, a matching operation between the incoming SPN's information (came within the MtM message) and the SVD's. if they are matched, then the authentication is approved, SCT_Flag set into 1 (Which means now the SVD-1 is officially a suspected car), Stop_Car Function been called and finally a reply message MtP would be sent to the PVD-1 telling the result of the MtM request sent earlier.

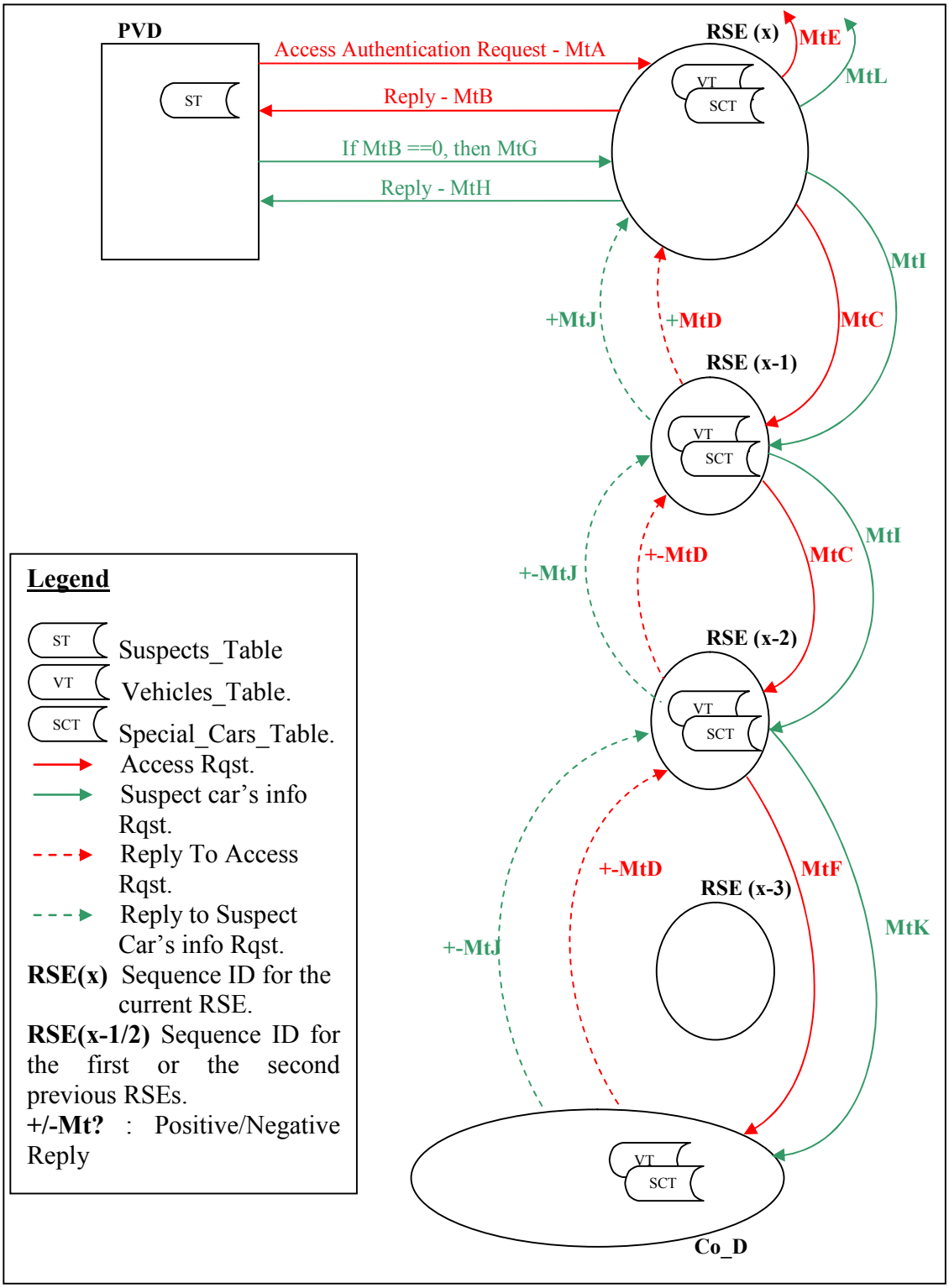

Figure 4.15: Suspect car instant termination – messages flow

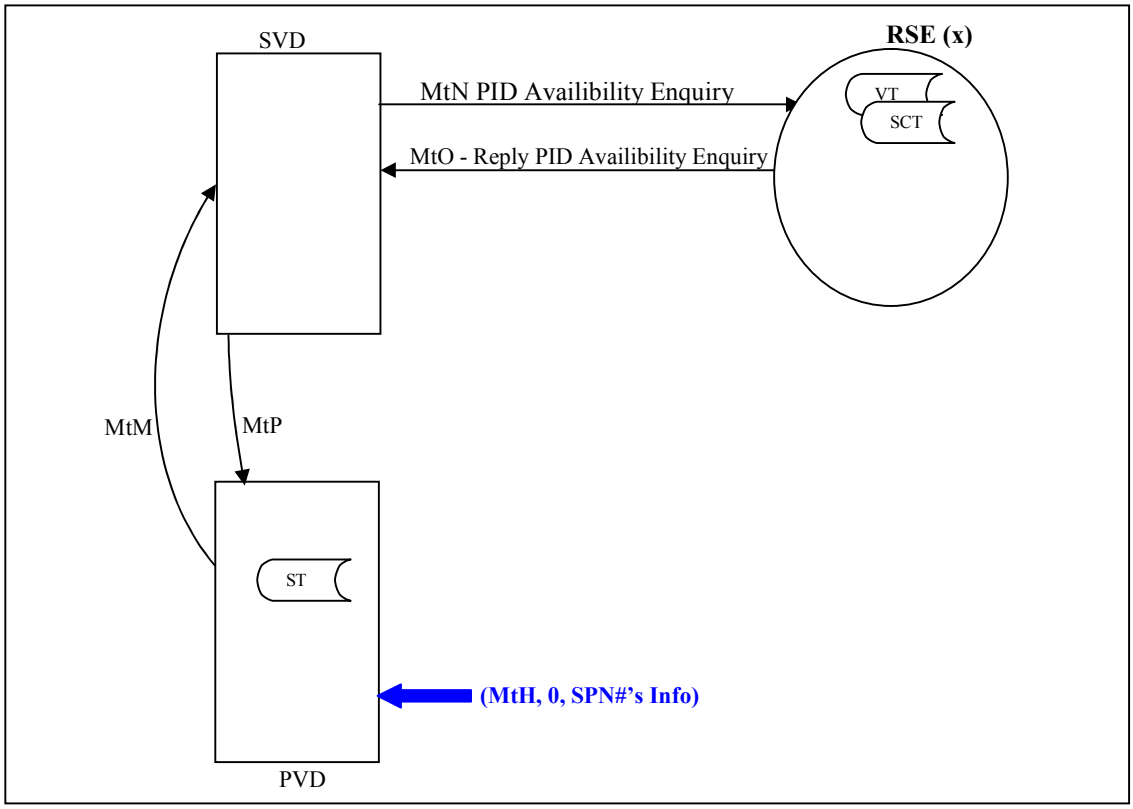

Figure 4.16: Suspect car instant termination – messages flow – After getting the SVD information

CHAPTER 5: RESULTS AND DISCUSSION

5.1 Introduction

In the previous chapter we've shown three case studies with three different scenarios using our formal specification language. Mathematically using our language, we have described the layered system used as the VANET Environment Model and showed each of the system's devices architectures and its sets of operations, components, and relations.

5.2 Case studies Differences

By looking to the three case studies in chapter Four, each of them is different from the other, but they all were applied on the same system hardware. So we have 3 different applications work on the same hardware.

In this section, we are going to show both of the validity and the consistency tests on each of the three case studies. The first two case studies, their tests might look similar but they are not, that we followed two different approaches to apply the test on.

5.2.1 Validity test

The first case study has four main steps to be done:

$\alpha_{A1} \sim Q> \beta_{A1}$ …(1) // Co_D sends a Speed code *message* to RSE that carries the ID x.

$\beta_{A1} \sim R\circ \Gamma_{B1}$ …(2) // RSE(x) passes the Speed Code *message* to the Required vehicle-1.

$\Gamma_{B1} \sim S> \beta_{A1}$ …(3) // Vehicle-1 replies with the vehicle's instant status report to the instant active RSE.

$\beta_{A1} \sim T> \alpha_{A1}$ …(4) // RSE forwards the vehicle's instant status report to Co_D.

We can divide the whole operation into two sub-operations; propagating a speed code, and sending a vehicle's status report, and then we check for their validity and consistency separately. Or we can just consider all the four steps shown above as on operation with one conclusion because this scenario is simple. But we will illustrate the first way.

The first two steps are under the first sub-operation; Propagating a Speed Code (PSC) and their conclusion should be:

$$\Gamma_{B1} \circ R \sim \beta_{A1} \quad \ldots(12)$$

While the second two steps (3 and 4) reside under the second sub-operation; Sending a Vehicle's Status Report (SVSR), which's conclusion should be:

$$\alpha_{A1} < T \sim \beta_{A1} \quad \ldots(34)$$

Table 5.1: First Scenario's Validity Truth Table

Possibilities	(1)	(2)	(12)	PSC Validity	(3)	(4)	(34)	SVSR Validity	Validity
1	F	F	F	T	F	F	F	T	-
0	F	T	F	T	F	T	F	T	-
1	T	F	F	T	T	F	F	T	-
1	T	T	T	T	T	T	T	T	T

By looking at the possibility flag column in the truth table above, the second line is impossible to happen, so we can exclude it from the testing. The reason behind the impossibility of the second line is because our system is a sequential system, then Operation (1) is False which makes Operation (2) impossible to be True. All the impossible lines would be excluded from the validity test.

There are some lines are possible to happen but can be excluded from the validity test because they are not related to the validity proof. The first and the third lines are not related to the validity proof because they do not satisfy the first part of the validity rule which is saying "*A set of sentences are considered Valid, if and only if there is no line in*

the system's operation truth table having ***all of its statements are True*** *while the conclusion is False.*". So only one line left for the validity test process to check on, that is the forth.

Assuming that all the connections and messaging were done properly with no problems occurred then according to the validity test:

Sub-Operation Validity = !(Op.1 & Op.2 & … & $Op._n$ & !Conclusion)

SOP validity = !((1) **&** (2) **&** !(12))

= !(T **&** T **&** !(T))

= !(F) = T

SVSR Validity = !(!(34) & (3) & (4))

= !(!(T) & T & T)

= !(F) = T

Whole System Validity = OpV_{a1} & OpV_{b1} & … & OpV_{n1}

Whole System validity = SOP Validity & SVSR Validity

= T & T = T

Then we conclude that the system is valid with such a scenario.

For the second scenario, it has four main steps to be done:

α_{A1} ~U> β_{Ax} …(1) // Co_D sends a *Function message* to RSE that carries the ID x.

β_{Ax} ~V> Γ_{B1} …(2) // RSE(x) passes the *Function message* to the Required vehicle-1.

Γ_{B1} ~W> β_{Ax} …(3) // Vehicle-1 replies the status of the *function* implementation to the instant active RSE.

β_{Ax} ~X> α_{A1} …(4) // RSE forwards the function implementation results report to Co_D.

Because this scenario is simple, We are going to Not divide the whole operation into two sub-operations then check their validity and consistency, as we did in case study One. We are going to consider there are four operations and one conclusion that would be:

$\Gamma_{B1} \triangle \beta_{Ax}$ …(1234) // Vehicle-1 Stops within the coverage area of β_{Ax}.

Table 5.2: Second Scenario's Validity Truth Table

No.	Possibilities	(1)	(2)	(3)	(4)	(1234)	Validity
1	**1**	**F**	**F**	**F**	**F**	**F**	-
2	0	F	F	F	T	F	-
3	0	F	F	T	F	F	-
4	0	F	F	T	T	F	-
5	0	F	T	F	F	F	-
6	0	F	T	F	T	F	-
7	0	F	T	T	F	F	-
8	0	F	T	T	T	F	-
9	**1**	**T**	**F**	**F**	**F**	**F**	-
10	0	T	F	F	T	F	-
11	0	T	F	T	F	F	-
12	0	T	F	T	T	F	-
13	**1**	**T**	**T**	**F**	**F**	**F**	-
14	0	T	T	F	T	F	-
15	**1**	**T**	**T**	**T**	**F**	**F**	-
16	**1**	**T**	**T**	**T**	**T**	**T**	**T**

By looking at the possibilities flag column in the truth table above, we can see that only 5 lines are possible to happen, those are: 1, 9, 13, 15, & 16. So we can exclude all the others from the validity testing.

There are some lines are possible to happen but can be excluded from the validity test because they are not related to the validity proof. Only line 16 is related to the validity test because it satisfies the first part of the validity rule.

By assuming that all the connections and messaging were done properly with no problems occurred then according to the validity test:

Operation Validity = !(Op.1 & Op.2 & … & $Op._n$ & !Conclusion)

Operation Validity = !((1) **&** (2) **&** (3) **&** (4) **&** !(1234))

= !(T **&** T **&** T **&** T **&** !(T))

= !(F) = T

Then we conclude that the system is valid with such a scenario.

For the last scenario is a bit complex, so that we have divided it into sixteen steps:

Γ_{B1} ~A> β_{Ax} …(1) // Police vehicle sends an Access Authentication Request to the RSE device.

β_{Ax} ~B> Γ_{B1} …(2) // A reply to the message MtA

β_{Ax} ~C> β_{Ax-1} …(3) // RSE requests the PVD information from the previous RSE

β_{Ay} ~D> β_{Ax} …(4a) // A reply from an RSE to Message MtC.

α_A ~D> β_{Ay} …(4b) // A reply from a Co_D to Message MtF.

β_{Ax} ~E> β_{Ax+1} …(5) // RSE sends a copy of the PVD information to the next RSE.

β_{Ax-2} ~F> α_A …(6) // RSE requests the PVD information from the Co_D

Γ_{B1} ~G> β_{Ax} …(7) // Police vehicle sends a Mobile Node information Request to the RSE.

β_{Ax} ~H> Γ_{B1} …(8) // A reply to message MtG

β_{Ax} ~I> β_{Ax-1} …(9) // RSE requests the Mobile Node information from the previous RSE

β_{Ay} ~J> β_{Ax} …(10a) // A reply from an RSE to Message MtI .

α_A ~J> β_{Ax} …(10b) // A reply from a Co_D to Message MtK.

β_{Ax-2} ~K> α_A …(11) // RSE requests the Mobile Node information from the Co_D

β_{Ax} ~L> β_{Ax+1} …(12) // RSE sends a copy of the Mobile Node information to the next RSE

Γ_{B1} ~M> Γ_{A1} …(13) // A PVD sends a request to a mobile Node to Access it

Γ_{A1} ~N> β_{Ax} …(14) // A Mobile Node try to validate a PVD in an RSE's Tables

β_{Ax} ~O> Γ_{A1} …(15) // A Reply from an RSE to a Mobile Node for the MtN request

Γ_{A1} ~P> Γ_{B1} …(16) // A Reply from a Mobile Node to a police vehicle for the MtM request

The Black steps are compulsory while the red steps are optional. So we have divided the whole operation into Five Sub-Operations; Op-AB, Op-CF, Op-GH, Op-IL, & Op-MP. Each group will have a separated validity test then we do the ANDING for their results to get the whole system's validity test result. The conclusions for each Sub-operation as follows:

- Conclusion for Op-AB is:

  $\Gamma_{B1} <B\sim \beta_{Ax}$ …(12)
- Conclsion for Op-CF is:

  $\beta_{Ax} <D\sim \beta_{Ay}$ …(36)
- Conclsion for Op-GH is:

  $\Gamma_{B1} <H\sim \beta_{Ax}$ …(78)
- Conclsion for Op-IL is:

  $\beta_{Ax+1} <L\sim \beta_{Ax}$ …(912)
- Conclsion for Op-MP is:

  $\Gamma_{B1} <P\sim \Gamma_{A1}$ …(1316)

The Red steps are optional with respect to the whole operation, so the system might still be working if there is a problem appeared with any of them, but for the validity test all of the operations, the compulsory and the optional should be valid.

Let's assume that all the sub-operations are valid except the Op.CF was invalid because its conclusion was False when all of its steps were done completely and flagged as True, then we will get the following truth tables of the system:

Table 5.3: Op-AB Validity Truth Table

Possibilities	(1)	(2)	(12)	Op-AB Validity
1	F	F	F	-
0	F	T	F	-
1	T	F	F	-
1	T	T	T	T

Table 5.4: Op-CF Validity Truth Table

Possibilities	(3)	(4a)	(4b)	(5)	(6)	(36)	Op-CF Validity
1	**F**	**F**	**F**	**F**	**F**	**F**	**-**
0	F	F	F	F	T	F	-
0	F	F	F	T	F	F	-
0	F	F	F	T	T	F	-
0	F	F	T	F	F	F	-
0	F	F	T	F	T	F	-
0	F	F	T	T	F	F	-
0	F	F	T	T	T	F	-
0	F	T	F	F	F	F	-
0	F	T	F	F	T	F	-
0	F	T	F	T	F	F	-
0	F	T	F	T	T	F	-
0	F	T	T	F	F	F	-
0	F	T	T	F	T	F	-
0	F	T	T	T	F	F	-
0	F	T	T	T	T	T	-
1	**T**	**F**	**F**	**F**	**F**	**F**	**-**
0	T	F	F	F	T	F	-
0	T	F	F	T	F	F	-
0	T	F	F	T	T	F	-
0	T	F	T	F	F	F	-
0	T	F	T	F	T	F	-
0	T	F	T	T	F	F	-
0	T	F	T	T	T	F	-
1	**T**	**T**	**F**	**F**	**F**	**F**	**-**
0	T	T	F	F	T	F	-
0	T	T	F	T	F	F	-
0	T	T	F	T	T	F	-
1	**T**	**T**	**T**	**F**	**F**	**F**	**-**
0	T	T	T	F	T	F	-
1	**T**	**T**	**T**	**T**	**F**	**F**	**-**
1	**T**	**T**	**T**	**T**	**T**	**F**	**F**

Table 5.5: Op-GH Validity Truth Table

Possibilities	(7)	(8)	(78)	Op-GH Validity
1	F	F	F	-
0	F	T	F	-
1	T	F	F	-
1	T	T	T	T

Table 5.6: Op-IL Validity Truth Table

Possibilities	(9)	(10a)	(10b)	(11)	(12)	(912)	Op-IL Validity
1	**F**	**F**	**F**	**F**	**F**	**F**	**-**
0	F	F	F	F	T	F	-
0	F	F	F	T	F	F	-
0	F	F	F	T	T	F	-
0	F	F	T	F	F	F	-
0	F	F	T	F	T	F	-
0	F	F	T	T	F	F	-
0	F	F	T	T	T	F	-
0	F	T	F	F	F	F	-
0	F	T	F	F	T	F	-
0	F	T	F	T	F	F	-
0	F	T	F	T	T	F	-
0	F	T	T	F	F	F	-
0	F	T	T	F	T	F	-
0	F	T	T	T	F	F	-
0	F	T	T	T	T	T	-
1	**T**	**F**	**F**	**F**	**F**	**F**	**-**
0	T	F	F	F	T	F	-
0	T	F	F	T	F	F	-
0	T	F	F	T	T	F	-
0	T	F	T	F	F	F	-
0	T	F	T	F	T	F	-
0	T	F	T	T	F	F	-
0	T	F	T	T	T	F	-
1	**T**	**T**	**F**	**F**	**F**	**F**	**-**
0	T	T	F	F	T	F	-
0	T	T	F	T	F	F	-
0	T	T	F	T	T	F	-
1	**T**	**T**	**T**	**F**	**F**	**F**	**-**
0	T	T	T	F	T	F	-
1	**T**	**T**	**T**	**T**	**F**	**F**	**-**

1	T	T	T	T	T	T	T

Table 5.7: Op-MP Validity Truth Table

Possibilities	(13)	(14)	(15)	(16)	(1316)	Op_MP Validity
1	F	F	F	F	F	-
0	F	F	F	T	F	-
0	F	F	T	F	F	-
0	F	F	T	T	F	-
0	F	T	F	F	F	-
0	F	T	F	T	F	-
0	F	T	T	F	F	-
0	F	T	T	T	F	-
1	T	F	F	F	F	-
0	T	F	F	T	F	-
0	T	F	T	F	F	-
0	T	F	T	T	F	-
1	T	T	F	F	F	-
0	T	T	F	T	F	-
1	T	T	T	F	F	-
1	T	T	T	T	T	T

Although, four sub-operations are valid and only one optional operation is invalid, the whole system is considered invalid according to the following equation:

$$\text{Whole System Validity} = \text{Op.AB} \;\&\; \text{Op.CF} \;\&\; \text{Op.GH} \;\&\; \text{Op.IL} \;\&\; \text{Op.MP}$$
$$= T \;\&\; F \;\&\; T \;\&\; T \;\&\; T$$
$$= F$$

When implemented, the specification language tool should show where exactly the invalidity reason is.

5.2.2 Consistency Test

Table 5.8 shows the truth table for the layered system working with the Speed control & Highway monitoring scenario, the first case Study.

What we are looking for in the consistency truth table, "is there any line has all the operations steps True and the conclusion is True as well?", In other words, we are looking for at least one bridge to move from the input side to the required output side.

Table 5.8: Consistency Truth Table

Possibilities	(1)	(2)	(12)	PSC Consistency	(3)	(4)	(34)	SVSR Consistency	Consistency
1	F	F	F	T	F	F	F	T	F
0	F	T	F	T	F	T	F	T	-
1	T	F	F	T	T	F	F	T	F
1	T	T	T	T	T	T	T	T	T

The forth line satisfies the consistency rule but let's see the following equation:

$$CT_x = Op.1 \ \& \ Op.2 \ \& \ \dots \ \& \ Op._n \ \& \ Conclusion$$

$$CT_{PSC4} = (1) \ \& \ (2) \ \& \ (12)$$

$$= T \ \& \ T \ \& \ T = T$$

$$CT_{SVSR4} = (3) \ \& \ (4) \ \& \ (34)$$

$$= T \ \& \ T \ \& \ T = T$$

$$CT_4 = CT_{PSC} \ \& \ CT_{SVSR}$$

$$= T \ \& \ T = T$$

As long as there is at least one True Consistency Test then no need to care about the rest of the tests, see below:

Whole system consistency = $CT_1 \mid CT_2 \mid \ldots \mid CT_n$

= - | - | - | T = T

Then we conclude that the system is consistent.

Table 5.9 shows the truth table for the layered system working with the second scenario; Remote car locating & sending a Service request / Function message scenario.

Table 5.9: Consistency Truth Table

Possibilities	(1)	(2)	(3)	(4)	(1234)	Consistency
1	F	F	F	F	F	-
0	F	F	F	T	F	-
0	F	F	T	F	F	-
0	F	F	T	T	F	-
0	F	T	F	F	F	-
0	F	T	F	T	F	-
0	F	T	T	F	F	-
0	F	T	T	T	F	-
1	T	F	F	F	F	-
0	T	F	F	T	F	-
0	T	F	T	F	F	-
0	T	F	T	T	F	-
1	T	T	F	F	F	-
0	T	T	F	T	F	-
1	T	T	T	F	F	-
1	T	T	T	T	T	T

The forth line satisfies the consistency rule but let's see the following equation:

CT_x = Op.1 & Op.2 & … & $Op._n$ & Conclusion

CT_{All} = (1) & (2) & (3) & (4) & (1234)

= T & T & T & T & T = T

As long as there is at least one True Consistency Test then no need to care about the rest of the tests, see below:

Whole system consistency = $CT_1 \mid CT_2 \mid \ldots \mid CT_n$

= - | T = T

Then we conclude that the system is consistent with the second scenario as well.

For the third case, there are many possibilities for the consistency to be taken care of, that because it has compulsory and optional operations, so the consistency test should be repeated at least twice with two different cases to make sure that the:

- First test will be applied on the system considering the compulsory operations; Op.AB, Op.GH, and Op.MP.
- Second test we apply it on the system with considering Op.AB, Op.CF, Op.GH, and Op.MP. The parameters we use here should make the use of Op.CF.
- Finally, we apply the test on the system with the existence of all the operations; Op.AB, Op.CF, Op.GH, Op.IL, and Op.MP. the parameters we use here should make the use of Op.IL.

Let's assume that the first test was done and found the system was consistent. While when the second test was applied, at the Op.CF a problem appeared when sending a reply message from an RSE to another because of (For Example) wrong routing information or incorrect message interpretation. Then we get the following truth tables:

Table 5.10: Op-AB Consistency Truth Table

Possibilities	(1)	(2)	(12)	Op-AB Consistency
1	F	F	F	-
0	F	T	F	-
1	T	F	F	-
1	T	T	T	T

Table 5.11: Op-CF Consistency Truth Table

Possibilities	(3)	(4a)	(4b)	(5)	(6)	(36)	Op-CF Consistency
1	**F**	**F**	**F**	**F**	**F**	**F**	**-**
0	F	F	F	F	T	F	-
0	F	F	F	T	F	F	-
0	F	F	F	T	T	F	-
0	F	F	T	F	F	F	-
0	F	F	T	F	T	F	-
0	F	F	T	T	F	F	-
0	F	F	T	T	T	F	-
0	F	T	F	F	F	F	-
0	F	T	F	F	T	F	-
0	F	T	F	T	F	F	-
0	F	T	F	T	T	F	-
0	F	T	T	F	F	F	-
0	F	T	T	F	T	F	-
0	F	T	T	T	F	F	-
0	F	T	T	T	T	T	-
1	**T**	**F**	**F**	**F**	**F**	**F**	**-**
0	T	F	F	F	T	F	-
0	T	F	F	T	F	F	-
0	T	F	F	T	T	F	-
0	T	F	T	F	F	F	-
0	T	F	T	F	T	F	-
0	T	F	T	T	F	F	-
0	T	F	T	T	T	F	-
1	**T**	**T**	**F**	**F**	**F**	**F**	**-**
0	T	T	F	F	T	F	-
0	T	T	F	T	F	F	-
0	T	T	F	T	T	F	-
1	**T**	**T**	**T**	**F**	**F**	**F**	**-**
0	T	T	T	F	T	F	-
1	**T**	**T**	**T**	**T**	**F**	**F**	**-**
1	**T**	**F**	**T**	**T**	**T**	**F**	**F**

Accordingly, the following operations will never come up, so the second consistency test result will be negative. By assuming for the same reason the third consistency test fails. According to the following rule:

$$\text{Whole system consistency} = CT_1 \mid CT_2 \mid CT_3$$
$$= T \mid F \mid F$$
$$= T$$

The system still consistent but with a low robustness measure, one third, see below:

Consistency Probability = (Number of Successful Tests / Total Number of the Tests) * 100 %
= (1 / 3) * 100 % = 33%

5.3 Comparative Study

Our formal specification language is written using the algebraic approach, which means, a system can be described in terms of operations and relationships between them. See Table 5.12 which we made to compare other specification languages into ours regarding the Type of the language.

Table 5.12: Language Type

Language	Language Type
Our SL	Hybrid Systems Algebraic-Based & Scenario-Based
LOTOS	Temporal-Ordering-Based
CASL	First order Logic Based
CSP	Process Algebra Based
Alloy	First order Logic Based
Larch	Sequential System Algebraic-Based
mCRL2	Process Algebra & Abstract Equational data Types Based
VDM	Sequential System Model-Based
Z Notation	Sequential System Model-Based

B	Sequential System Model-Based
SPIN	Model-Based
OBJ	Sequential System Algebraic-Based
Petri Nets	Concurrent System Model-Based

Our language is a formal specification language written to specify precisely VANET Systems and applications. But it can specify other types of Networks such as Infrastructures Networks, Mobile Networks, and Sensors Networks. Table 5.13 shows a comparison between the different specification languages application areas.

Table 5.13: Application Area

Language	**Application Area**
Our SL	VANET & Networked systems
LOTOS	Protocol Specification
CASL	General Purpose
CSP	Interaction of Concurrent System
Alloy	Software Systems Expressing
Larch	Computing Systems
mCRL2	Concurrent Discrete Event Systems Description
RSL	Grid-Resource Discovery
RAISE	Software Development
VDM	Computer-Based Systems Development
Z Notation	Computing Systems Modeling and Describing
B	Development of Computer Software
SPIN	distributed software systems
OBJ	Computer Software Systems
Petri Nets	Discrete Distributed Systems

As we have mentioned before, our formal specification language is written to specify precisely VANET Systems and applications. According to our survey and literature review no other formal specification was created to specify VANET aspects. So if we would like to compare other Specification languages to our language regarding the covered VANET aspects, then ours provides a set of notations and rules those cover some additional aspects (VANET related), these aspects are: (See Table 5.14.)

- Device Ability - DA
- Device Movement - DM
- Messages exchangeability - ME
- Device Internal Behavior logic - IB
- Security - Sc
- Validity Test - VT
- Consistency Test - CT
- Design and Configuration – D&C
- VANET Environment Description - ED
- Connections Description - CD

Table 5.14: Covered aspects Comparison, √: Completely Covered, P√: Partially Covered

Language	DA	DM	ME	IB	Sc	VT	CT	D&C	ED	CD
Our SL	√	√	√	√	P√	√	√	√	√	√
LOTOS	√		√	√		√	√	P√		
CASL	√		√	√				√		
CSP	√		√	√	√		√	P√		
Alloy	√			√			√	√		
Larch	√			√		√	√	P√		
mCRL2	√		√	√		√	√	√		
RSL	√		√	√						
RAISE	√		√	√						
VDM	√		√	√				√		
Z	√		√	√			√	P√		
B	√			√		√	√	√		
SPIN	√		√	√			√	√		
OBJ	√						√	P√		
Petri Nets	√		√	√				√		

The last point we would like to mention, but not the least, formal specification languages have a limitation that they are not well suited to deal with the user interaction, while our language plan is to create an answer file for each user interface. That file has all the answering information required from the user to enter. This file should be filled by the researcher who is using our language and should be changed according to the scenario purpose.

CHAPTER 6: CONCLUSION AND FUTURE WORK

6.1 Introduction

In this chapter some final remarks on the comparison results in chapter Five. To begin, section 6.2 describes the results of the project. In addition, section 6.3 gives some directions for future research into the area of vehicular Ad-hoc Networks.

6.2 Conclusion

We started this project aiming to accomplish three Objectives; those are Creating a formal specification language precisely for VANET systems, creating a layered model for VANET systems, and outlining an application layer protocol for Service delivery to remote nodes. In chapter 3 we have explained in details the creation of our specification language and defined all the notations and the rules of the language. Then we described the layered system using our specification language and illustrated the protocol abilities through showing its messages abilities.

If we get back to Table 5.13 we would find that our language is the only one among the rest in the table, covers the application area of VANET systems. Our formal specification language, mainly we have created it to specify and proof the validity and verification of the VANET systems and their applications. VANET compares to other network types, has special needs such as high mobility, High level of security, and High bandwidth, so it needs a special language that can describe all those different aspects. The aspects covered by our language are VANET specific but at the same time it can be used to describe some other network types such as Infrastructures networks, Mobile nodes Networks, and Sensors Networks.

VANET has the two types of systems; Sequential and Concurrent Systems as well, so that, in Table 5.12 we can see our specification language was described as a *Hybrid Systems* specification language. On the same table, we showed that our language is *Algebraic-Based & Scenario-Based* because it was written according to the Algebraic approach and can be used based on Scenarios.

Our specification language covers many of the VANET aspects; those were not covered all by any other specification language as we can see in Table 5.14.

6.3 Future Work

Many further works can be added to the specification language or to the layered system model or to its application layer protocol.

6.3.1 Tool Development

A tool for our language is needed to be developed so other researchers who work on VANET systems and applications development can use to present their work and simulate it and prove its validity and consistency.

We have created the language notations set, rules set, and two proofing tests. What needed is a GUI tool supported with a text editor that provides the ability to print our language symbols and interprets them. Our language uses many Non-ASCII notations. The GUI tool's text editor should provide the ability to print those notations.

6.3.2 Language Enhancement

As we mentioned before, our specification language precisely specify VANET systems and their applications and it covers many aspects of the VANET area but no all of them. For example, the security aspect is a very important requirement in VANET systems because if a successful hacking attack could happen on any VANET system, it might lead to a disaster, so that we always need a robust security system besides the VANET system. Some more notations are needed in the security aspect to cover.

Our language provides two kinds of tests, Validity test & Consistency test, the first is to tell whether or not a system is available for a specific application or not and the opposite way. The consistency test tells how reliable a system is with some scenarios or an application or the other way back. Many other tests or algorithms can be added to the current language to make it more efficient to deal with faults and errors trying to refine the system by auto-fix them or just to show where and what the error is and gives some suggestions on how to fix that error.

6.3.3 Application layer protocol enhancement

The motivation for lining out the application layer protocol was to draw a starter line for creating a common protocol can be used on VANET environment and to push researchers to create more applications based on this protocol. So more works can be done to enhance and develop the protocol, such as, creating more multi-purpose messages, those can be used by different applications for different purposes.

6.3.4 Developing a simulator for VANET systems

Using our layered system model combined with the specifications language notations (instead of using programming language) we can create a model based simulator engine for VANET systems. There would be icons refer to different layers devices, a user can click and drag the icon of a device and put it on the simulator's design board, when double click the device icon, its built-in functions would appear and the user can create more functions for that device using our language notations.

The simulator would have an extension to design printable VANET systems figures. This extension would offer many views of the system. The lower level we show, the more details of the system would be shown.

6.3.5 Developing more of VANET applications

Based on our application layer protocol and the VANET environment model, many applications can be inspired from and be developed. For example, a punishment fine delivery system that will provide kind of mailing system between the central police station of a city and the vehicles within that city.

REFERENCES

ABDERRAHIM B., 2004, University of Avignon – France, "Optimizing Dissemination of Alarm Messages in Vehicular Ad-Hoc Networks (VANET)".

ARNOLD T., 2008, WYATT L., ZHAO J. and CAO G. "IP Address Passing for VANET," IEEE International Conference on Pervasive Computing and Communications (Percom), 2008.

BALON N., (2006). University of Michigan, Master's thesis "*Increasing Broadcast Reliability In Vehicular Ad-Hoc Networks*".

BAUMANN R., 2004. ETH Zurich, Master's Thesis in Computer Science; *"Vehicular Ad hoc Networks (VANET) - Engineering and simulation of mobile ad hoc routing protocols for VANET on highways and in cities"*.

BLUM J., 2004, ESKANDARIAN A., and HOFFMAN L. J., Challenges of intervehicle ad hoc networks. IEEE Trans. Intelligent Transportation Systems.

BOOCH G., 1994. Book: "Object-Oriented Analysis and Design with Applications (2nd Edition) (The Benjamin/Cummings Series in Object-Oriented Software Engineering)", ISBN 0-8053-5340-2.

BYCHKOVSKY V., 2006, B. Hull, A. Miu, H. Balakrishnan, "A Measurement study of Vehicles Internet Access Using In Situ Wi-Fi Networks," ACM MOBICOM 2006.

COPS M., 2006. Program Manager, Vehicle Infrastructure Integration Consortium, VII Strategy for Safety and Mobility Program, Sept 29 2006.

FARKAS C. 2007, KOPYLOVA Y. University of South Carolina, Master's Proposal; *"Application Level Protocol for Accident Reconstruction in VANETs"*.

FREMONT G., 2007, BOYER T. – ASF. Report; *"A co-operative vehicle / infrastructure system to improve road transport safety and provide accurate in-vehicle information - The SAFESPOT & CVIS projects", ASECAP 2007 Annual congress*.

GUEMARI L., 2001. Master Thesis, Institute National desTelecommunicatoins, "An OPNeT Model implementation for Ad-hoc On Demand Distance Vector Routing Protocol".

IEEE GROUPER, 2008. Official IEEE 802.11 Working group project Timelines: http://grouper.ieee.org/groups/802/11/Reports/802.11_Timelines.htm, Accessed before June 2008.

LARSSON T., 1998, HEDMAN N (1998). Master Thesis in Lulea University of Technology, "Routing Protocols in Wireless Ad-hoc Networks - A Simulation Study".

LEEARMSTRONG, 2008. Arm Strong consulting, Inc. web site: http://www.leearmstrong.com/DSRC/DSRCHomeset.htm, Accessed before June 2008.

SOMMER C., 2008, DIETRICH I., DRESSLER F.,DULZ W., GERMAN R. “A Tool chain for UML-Based Modeling and simulation of VANET Scenarios wityh Realistic Mobility Models”, Mobihoc 2008.

SOMMERVILLE I., 2007. Book: “Software Engineering”, Eighth Edition.

SONG H., 2008, ZHU S. and CAO G. "SVATS: A Sensor-network-based Vehicle Anti-theft System“, IEEE INFOCOM mini-conference, 2008.

SCHROTH C., 2006, STRASSBERGER M., EIGNER R., EICHLER S. (2006). "*A Framework for Network Utility Maximization in VANETs". In: Proceedings of the 3rd ACM International Workshop on Vehicular Ad Hoc Networks (VANET)* : ACM SIGMOBILE, 2006.- 3rd ACM International Workshop on Vehicular Ad Hoc Networks (VANET).- Los Angeles, USA, p. 2

TIDMAN P., 1999, and KAHANE H., Book: ”Logic and Philosophy – A Modren Introduction”.

ZHAO J., 2006, and CAO G. (2006). “VADD: Vehicle-Assisted Data Delivery in Vehicular Ad Hoc Networks”, IEEE INFOCOM, April 2006.

ZHAO J., 2008, ARNOLD T., ZHANG Y., and CAO G. (2008). “Extending Drive-thru Data access by Vehicle-to-Vehicle Reply,” ACM International Workshop on Vehicular Ad Hoc Network (VANET), 2008.

BIBLOGRAPHY

ABUELELA M., 2008, OLARIU S., and WEIGLE C. M. Department of Computer Science, Old Dominion University, IEEE Publ;ications, "NOTICE: An Architecture for the Notification of Traffic Incidents".

ARIB, 2005. Association of Radio Industries and Bussiness (ARIB), Japan. Global Standards Collaboration (GSC), France. "Study of a DSRC Basic Application Interface to Extend Application in Vehicles."

CHOUDHARY G., 2007. Department of Computer Science, Old Dominion University, Master's Project Final Report, "Providing VANET Security through Position Verification"

FESTAG A., 2008, NOECKER G., STRASSBERGER M., LUBKE A., BOCHOW B., TORRENT-MORENO M., SCHNAUFER S., EIGNER R., CATRINESCU C., and KUNISCH J.. Proceedings of 5'th International Workshop on Intelligent Transportation (WIT). "'NOW – Network On Wheels': Project Objectives, Technology and achievements."

GABBAY M., 2007 . Heriot-Watt University, Scotland, "Formal Specification Course."

GUTTAG J., 1982, HORNING J., WING J., MIT Laboratory for computer Science, "Some Notes On Putting Formal Specifications to Productive Use."

HOLFELDER W., 2004. DaimlerChrysler Research and Technology, INC. Palo Alto, CA, "Vehicle-to-Vehicle and Vehicle-to-Infrastructure Communication Recent Developments, Opportunities and Challenges."

JI L., 1999, ISHIBASHI M., CARSON M.. Institute for Systems Research, University of Maryland. “An Approach to Mobile Ad hoc Network Protocol Kernel Design.”

JONES B., 2005. ITS Joint Program Office, U.S. Department of Transportation, National VII Coalition, “DSRC – Linking the Vehicle and the Road.”

JULIUSSEN E., 2007. Principle Analyst, Telematics Research Group, INC. Detroit. “Advancements in V2V & C2I-V2R.”

L. WANG, 2004, and S. OLARIU, Department of Computer Science Old Dominion University, Book: “Hybrid Routing Protocols for Mobile Ad- hoc Networks.”

MANI P., 2003. Master’s thesis, University of Kansas, “Development and Performance Characterization of Enhanced AODV Routing for CBR and TCP Traffic.”

OLARIU S., 2008. Sensor Networks Research Group at Department of Computer Science, Old Dominion University. “An Architecture for Traffic Incident Detection”.

ROEBUCK R., 2005. SIRIT Technologies, Carlton, Texas. 5.9 GHz Prototype Development Program “DSRC Technology and the DSRC Industry Consortium (DIC) Prototype Team.”

STAMPOULIS A., 2006, CHAI Z., “A Survey of Security in Vehicular Networks”.

STOJMENOVIC I., 2008. University of OTTAWA, Lecture Notes: "Engineering Environmentally Friendly and Integrated Intelligent Transportation Systems."

WING J., 1. School of Computer Science, Carnegie Mellon University, Pittsburgh, USA. "Hints for Writing Specifications."

WING J., 2. School of Computer Science, Carnegie Mellon University, Pittsburgh, USA. "Teaching Mathematics to Software Engineers."

WING J., 1990. School of Computer Science, Carnegie Mellon, Pittsburgh, USA. IEEE, "A Specifier's Introduction to Formal Methods."

WING J., 1992. School of Computer Science, Carnegie Mellon, Pittsburgh, USA. IEEE, "Specifications in Software Development."

WING J., 1994. Massachusetts Institute of Technology, "Encyclopedia of Software Engineering – 2 Volume Set – Formal Methods."

WING J., 1996. Massachusetts Institute of Technology, "Teaching and Learning Formal Methods - Hints to Specifiers."

YONGKANG X., 2004, LIN Z., XIUMING S., YONG R., *and* ZHENGXIN M., Department of Electronic Engineering, Tsinghua University, Beijing, P.R. China. "Neighbor-Medium-Aware MAC Protocol with Fairness for Wireless Ad Hoc Networks"

YONGKANG X., 2005. Department of Electronic Engineering, Tsinghua University, Beijing, P.R. China. "Introduction of Mobile Ad hoc Network."

REFEREED CONFERENCES PROCEEDINGS

1. Maythem Kamal Abbas, Azween B. Abdullah, Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, "**Vehicles Speed Control via VANET**", ICIMU 2008.

2. Maythem Kamal Abbas, Azween B. Abdullah, Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, "**Remote Mobile Nodes Service Delivery via VANET**", NetApps 2008.

3. Maythem Kamal Abbas, Azween B. Abdullah, Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, "**A VANET Safety Application: Remote Mobile Nodes Service Delivery via VANET**", NPC 2009.