POLITECNICO DI TORINO

Master's Degree in Communications and Computer Networks Engineering

Master's Degree Thesis

TITLE

DEVELOPMENT OF A DISTRIBUTED VIRTUAL TRAFFIC LIGHT PROTOCOL FOR VEHICULAR NETWORKS

Supervisor	Candidate
Prof. CLAUDIO ETTORE CASETTI	AHMADREZA JAME

MONTH YEAR

October 2020

Abstract

Passing a junction by vehicles was a controversial issue from many years ago. Traffic lights were practiced as an approach. However, the lack of intelligence in their functionality sometimes leads to unnecessary waiting times for cars, leading to more fuel consumption and air pollution. This thesis concentrated on developing a distributed virtual traffic light protocol for vehicular networks to reduce the waiting time of vehicles at the intersections and decrease the number of stops and goes and, as a result, lessen fuel consumption. To this purpose, a Veins project was created using OMNeT++ and SUMO simulators. Finally, two algorithms were proposed to minimize the time to clear the intersection and stops and goes.

Acknowledgements

I would like to express my special gratitude to Professor Casetti, who gave me the golden opportunity of working on the topic of the Development of a Distributed Virtual Traffic Light Protocol for Vehicular Networks. This research vastly increased my knowledge. I came to know about many new things that I am thankful to them. He gave me the possibility to think out of the box and propose my ideas, but steering me in the right direction whenever it was needed.
Secondly, I would like to thank my family, especially my beloved mother and friends, who helped me and encouraged me from miles away, inspiring me to be motivated and concentrated on concluding this project within the limited time frame.
I am overwhelmed in all humbleness and gratefulness to acknowledge my depth to all those who have helped me put these ideas, well above the level of simplicity and into something concrete.
I would like to thank Marco Rapelli, who helped me gather different information, collect data, and guide me from time to time. Despite his busy schedules, he gave me different ideas in making this project unique.
In the end, I would like to especially thank Politecnico di Torino for allowing me to study there, giving me the ability to build a better future.

Ahmadreza Jame, October 2020.

Table of Contents

List of Algorithms

List of Figures

Chapter 1

Introduction

From the day that the first motor vehicles introduced, the human lifestyle has significantly changed, making them able to explore the world, expand their cities, and carry their heavy materials. With this technology, many new challenges appeared, like car accidents, air pollution, and traffic. Therefore, during each period, elite people began to find new solutions to overcome the obstacles. Today, many organizations are controlling car factories to push them to produce safer and more environment-friendly vehicles. By the appearance of computers, vehicles got smarter. In recent years, autonomous vehicles are introduced, which opened new doors for researchers to solve new challenges resulting in more efficient, greener, faster, and safer travels. Vehicle to Vehicle (V2V), Vehicle to Infrastructure (V2I), and Vehicle to Everything (V2X) are standardized approach introduced by scientists for autonomous vehicles. In this chapter, some of the reasons for using vehicular networks and new trends in vehicular networks are mentioned.

1.1 Intelligence Need

Traffic lights, as they do not communicate with their surrounding environment, for every traffic situation, they act the same. For example, the traffic light green phase remains the same despite the instantaneous traffic shape and density. This behavior causes extra waiting time for vehicles, forcing them to unnecessary stop (suppose that there is no car in other directions but the light is red) and thus more fuel consumption, which leads to air pollution and other environmental issues. Recently, new approaches used to address this problem. One of them is VTL (Virtual Traffic Lights). In this method, a traffic light can interact with other traffic lights and vehicles equipped with On-Board Unit (OBU) to change its decisions according to the traffic situation. The Travolution project introduced GLOSA (Green Light Optimal Speed Advisory) solution, which causes significant

improvement by reducing fuel consumption. However, it needs to equip all traffic lights and vehicles to communicate. Also, in unregulated intersections, GLOSA cannot be applied as there is no traffic light. In this thesis new approach, V2V-VTL (V3TL) implemented, and some improvements were introduced.

1.2 Vehicles New Technologies

According to European Commission definition [1] figure 1.1, there are five automation levels for vehicles. Each one indicates the level of autonomy of the vehicle. There are day-to-day improvements in manufacturing new vehicles, to make them not need any human control during the drive. The autonomy increases in higher levels of the definition, and the last level (level five), according to the definition, a vehicle can drive completely autonomous.

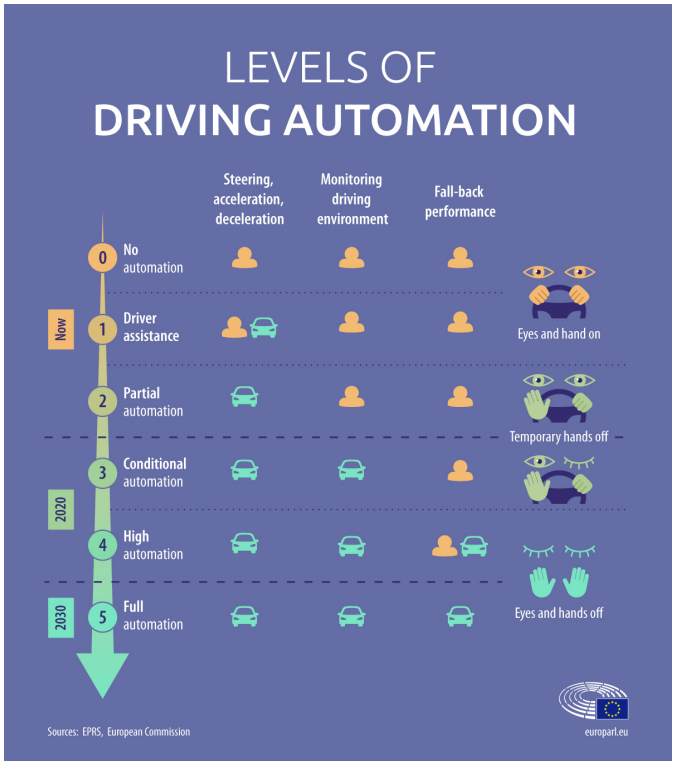

Figure 1.1: European Commission Vehicle Autonomy Definition.

According to Counter Print statistics [2], figure 1.2, it can be shown that fast improvements in vehicle autonomy would occur in the next years, which proves the importance of a Vehicle to Vehicle (V2V), Vehicle to Infrastructure (V2I), and Vehicle to Everything (V2X) communications to guarantee the safety of passengers, decrease the fuel consumption, and reduce the travel time.

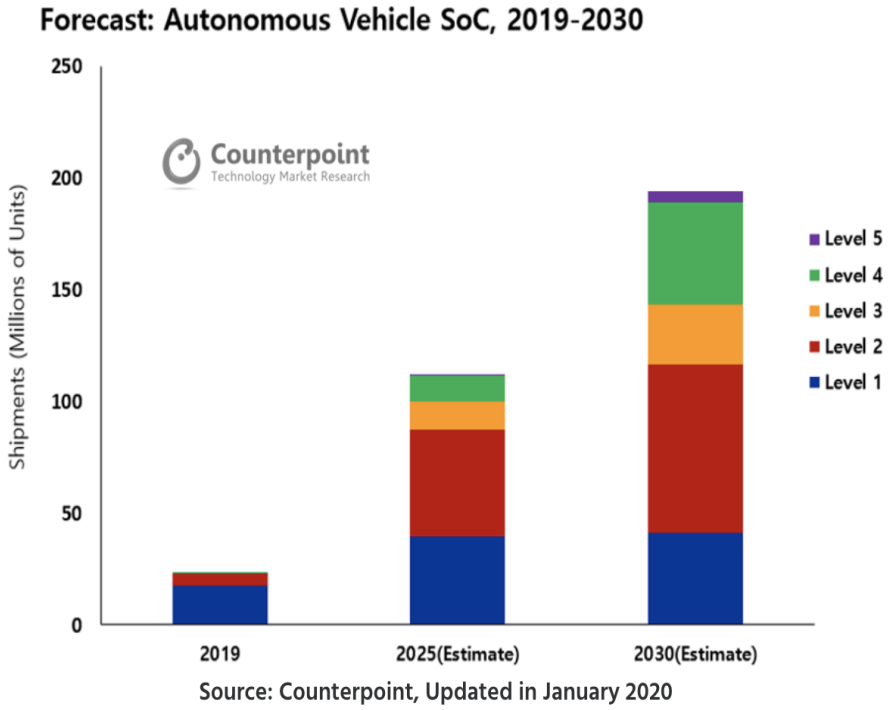

Figure 1.2: Autonomous Vehicles Production Forecast.

Applying vehicular network technologies paves the way for practicing new techniques. The new vehicles will be intelligent. Therefore, many approaches can be applied to improve their performance. This thesis focused on using V3TL and its implementation through different scenarios and tests. After explaining the whole procedure, the results and statistics are provided to address the benefits and drawbacks of using this scenario rather than traditional methods.

Chapter 2

Standards for Vehicular Communication

There are many ways for vehicular network communications; they are categorized into two methods:

- **Infrastructural Base:** Uses pre-existing infrastructures like (5G) which have defined for low latency, reliable, and high bandwidth communications.
- **Wi-Fi Base:** Uses IEEE standard 802.11p and provides V2V communication. In this approach, vehicles can communicate without any auxiliary infrastructures. In this thesis, the IEEE 802.11p was used. However, as the thesis focused on developing a distributed traffic light, which does not depend on specific lower-layer technology, it can work with other standards as well.

2.1 IEEE 802.11p

IEEE 802.11p uses two parallel technologies, the European standard Cooperative Intelligent Transport System (C-ITS) and the American standard Direct ShortRange Communication System (DSRC). The two standards have some differences in their specifications. American (PHY) layer and (MAC) layer standard is 802.11p, but the European one is ITS-G5.

2.1.1 DSRC Standard

DSRC is a standard provided by the Federal Communication Commission (FCC). The proposed bandwidth is 75 MHz in frequency (5.850 to 5925) GHz, in range, one channel is dedicated to safety and system control which name is Control Channel (CCH), and up to 6 channels are considered for non-safety messages,

Service Channel (SCH). Also, DSRC defined an onboard unit (OBU) and Road-Side Unit (RSU). The term short-range meaning coverage of hundred meters. Each channel time interval is 100 [ms] and a guard 4 [ms] considered at the beginning of each channel for compensating probable inaccuracy or delays. During CCH, every vehicle must tune to CCH frequency for safety-related data exchanges. During the SCH interval, vehicles can optionally switch to one of the SCH frequencies. For the synchronization and coordination of the channels, the system uses Coordinated Universal Time (UTC) as time reference provided by the global navigation satellite system [3].

As the figure 2.1 shows, there are seven channels on the spectrum. In the beginning, 5 MHz (5850-5855 MHz) is reserved for avoiding interference. One channel is dedicated to the control channel (channel 178), which is in charge of safety messages which carry critical safety messages. Two channels at the beginning and the end of the spectrum (5.855-5.865) and (5.915-5.925) are the Critical Safety of Life Channel and High Power Public Safety, designed for advanced applications preventing accidents. Other remaining channels are dedicated to service channels. Channels are designed to be adjacent. It is possible to join two channels using 20 MHz bandwidth, therefore using more bandwidth.

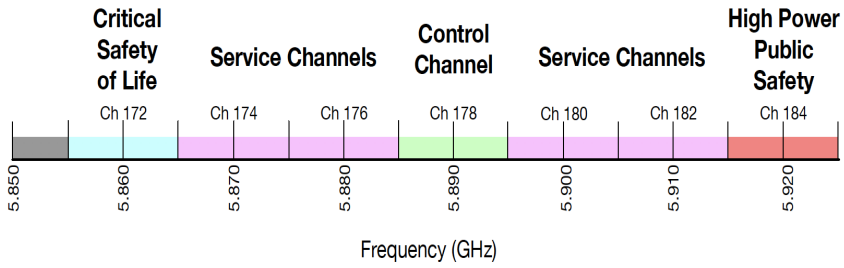

Figure 2.1: DSRC Spectrum.

For DSRC architecture, figure 2.2, each layer is defined with a different standard. However, the MAC layer and PHY layer are determined by the 802.11p standard. The interesting point is that DSRC has two parallel transport layers, one is Wireless Access in Vehicular Environments (WAVE) Short Message Protocol, and another is based on well-known transport layers (TCP/UDP) and network layer (IPV6). Depending on the application, one of the two protocols is used.

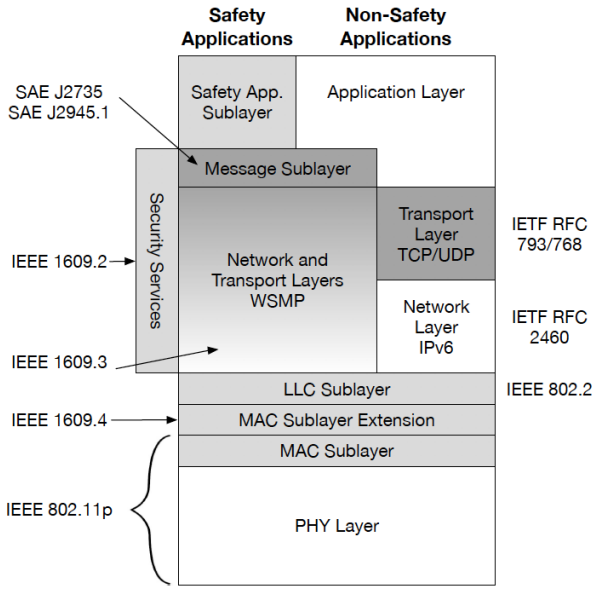

Figure 2.2: DSRC Architecture.

2.1.2 IEEE 802.11p Physical Layer

802.11p physical layer inherits from the 802.11a standard. As 802.11a works on 5.9 GHz, the inherited one also works on this frequency. There are some changes in 802.11p to make it suitable for V2X communications. 802.11p uses Orthogonal Frequency Division Multiplexing (OFDM), which divides the bandwidth to sub-carriers and sends them simultaneously. OFDM, by its characteristics, is robust against multi-path effect inter-symbol interference. As we are in vehicular networks, and inter-carrier interference exists due to vehicles' speed and mobility, 802.11p is designed to be resilient and reliable against the Doppler effect. Depending on the signal conditions, different modulations (QPSK, BPSK, 16-QAM, 64-QAM) and different bit-rates (4.5, 6, 9, 12, 18, 24, or 27 Mbps) can be used, making the signal more flexible and reliable.

2.1.3 IEEE 802.11p MAC Layer

802.11p Mac layer is also derived from other 802.11 standards, making it suitable for V2X. Like 802.11 standards, IEEE 802.11p employs the contention-based channel access EDCA as the MAC method, and an enhanced version of the primary Distributed Coordination Function (DCF) 802.11[4]. Also, 802.11p uses collision-avoidance technologies like Carrier Sense Multiple Access using Collision Avoidance (CSMA/CA) using back-off and Arbitrary Interframe Space (AIFS) and variable

Contention Windows (CW) to minimize the collision during the transmission. Supporting Quality of Service (QOS) is another advantage of this protocol.

2.1.4 DSRC Messages

Society of Automotive Engineers (SAE) provides standard (SAE J2735), which details a list of messages used in V2X. Below some important ones are listed:

Map Data

The MAP Data message is used to provide intersection and roadway lane geometry data for one or more locations (e.g., intersections and fragments of maps). Almost all roadway geometry information and roadway attributes (such as where a block region exists or what maneuvers are legally allowed at a given point) are contained in this message's "generic lane" details. MAP messages are used in intersections to numbering describe lane level details of each lane. In contrast, the SPAT message provides each signal head's current state, controlling the ability to stop/pass a given lane[5].

Signal Phase and Timing Message

The SPAT message is practiced to provide the current signal/phase timing data (times at which signals will change) for one or more signalized intersections and other time-to-day status details. SPAT messages are joined to MAP messages to provide road details.

Basic Safety Message

All equipped vehicles broadcast BSM messages at a 10Hz rate frequently for their neighboring cars. It is used for safety purposes by all safety applications. There are some essential fields inside this message that are listed below:

- **Message ID**
- **Message count**
- **Temporary ID**
- **Time**
- **Position info**
- **Motion info**
- **Break system status**

- **Vehicle safety extension**
- **Vehicle status**
- **Event flags (optional)**
- **Path history**
- **Path prediction**
- **RTCM package (optional)**
- **Other vehicle**
- **Data/Containers(optional)**

Figure 2.3 shows Basic Safety Message Format and its fields.

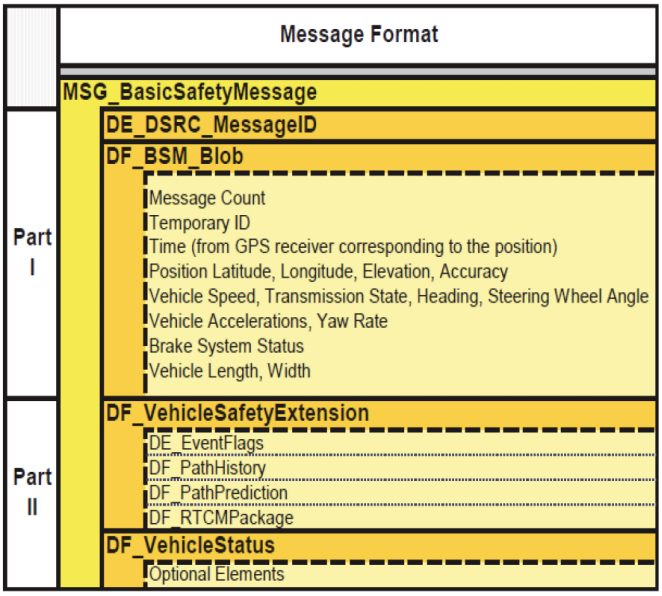

Figure 2.3: Basic Safety Message Format.

Common Safety Request

It is designed for V2V exchange safety elements.

Emergency Vehicle Alert

However, it is replaced in a new version of the standard and put in part II. It was designed to make other vehicles aware of emergency vehicles around.

Intersection Collision

The ICA message (Intersection Collision Announcement) is intended to inform other nearby vehicles that a hazardous situation exists or is possible to exist in the immediate future (e.g., it can be used to detect a red light/stop line violation and inform others). It can be sourced by both vehicles and by the infrastructure. The ICA message wraps a BSM message (real or virtual) with additional details to reflect the offending vehicle's estimated path.

Probe Vehicle Data

In the DSRC, vehicles can act as anonymous probes to collect traffic flow data, local weather data, and other conditions. The PDM message is used to control the probe data's details that the vehicle will collect and then report back to RSU devices using the PVD message.

Probe Data Management

The PDM message is used to control the probe data's details that the vehicle will collect and then report back to RSU devices using the PVD message. In the DSRC environment, vehicles can act as anonymous probes to collect traffic flow data, local weather data, and other conditions.

Road Side Alert

The RSA message is a more primitive version of the TIM message. Its original design intent was to support creating simple, quick, ad hoc messages for ATIS-like informational use from public safety vehicles by a responder in the field (such as a DOT vehicle parked to remove a roadway obstruction) without the need for coordination with others. By contrast, the TIM message was intended to support more complex pre-planned incident and work zone uses. This distinction has blurred in more recent times, but the simplicity of deploying the RSA message is still considered of value.

RTCM Corrections

The RTCM messages are used to "wrap" RTCM corrections messages for sending over the DSRC channel. Corrections data allows the GNSS of each DSRC device to maintain lane level accuracy under various conditions. These messages' type/content varies with the precise kind of corrections data needed by local deployments.

Signal Request Message

The SRM messages are used by authorized parties to request services from an intersection signal controller. Vehicles approaching an intersection use this message to affect the signal operation. This is whereby traditional preemption and priority requests are handled for intersection safety in DSRC.

Signal Status Message

The SSM messages, which are sent by the local DSRC / Signal Controller, reflect the intersection signal control's current operational state. Any prior request services (SRM messages) and their outcomes are reflected here as well. This message, therefore, serves as a means to acknowledge signal requests.

Traveler Information Message

The TIM message is used to contain a variety of traffic conditions and "advanced traveler" messages. It provides the means to inform the public about both incidents (traffic accidents) and pre-planned roadwork events. The TIM message can alert the public to severe weather conditions and other local or regional emergencies. It can also be used for different speed warnings, traffic signage, road conditions, and additional general information. The ITIS codes (J2540) are prominently used in this message.

Personal Safety Message

The PSM message is used to convey similar BSM-like for pedestrian users.

Test Message 00-15

These sixteen messages are considered for regional deployment. It can be used in a variety of demands in several ways.

2.2 C-ITS Standard

According to the European Commission that has organized the C-ITS Deployment Platform, Cooperative Intelligent Transport Systems (C-ITS) uses ad-hoc communications and other technologies like 4G and 5G. That allows V2X communications, traffic signals, roadside infrastructure, and other road users to communicate. V2X supports information, warning, and assistance services that will be gradually deployed in coordinated innovation phases during the approaching years[6]. Figure 2.4 shows the European frequency allocation for C-ITS.

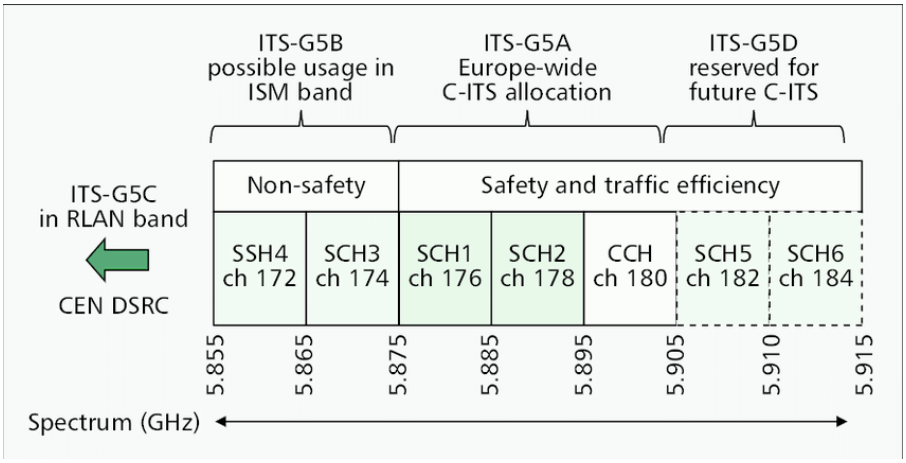

Figure 2.4: C-ITS Frequency Allocation.

2.2.1 C-ITS Protocol Stack

The access technologies layer uses specifications of the IEEE 802.11 standard are ITS-G5 (where G5 stands for the 5 GHz frequency band). The networking and transport layer has two columns: Geo-Networking and Basic Transport Protocol (BTP). The other column employs Internet protocols, particularly IPv6 with UDP, TCP, or other transport protocols such as SCTP and IP mobility extensions (Mobile IPv6 and its extensions for network mobility, NEMO). The choice of the communication profile, whether GeoNetworking or IPv6, lies in the application. Typically, the GeoNet, the working stack, is used for ad hoc communication over ITS-G5 utilizing geo-addressing, and IPv6 for communication with an IP-based infrastructure over cellular networks. IPv6 packets can also be transmitted over GeoNetworking, for which the adaptation sublayer GN6 has been designed [7]. Figure 2.5 shows the release one version of C-ITS protocol stack.

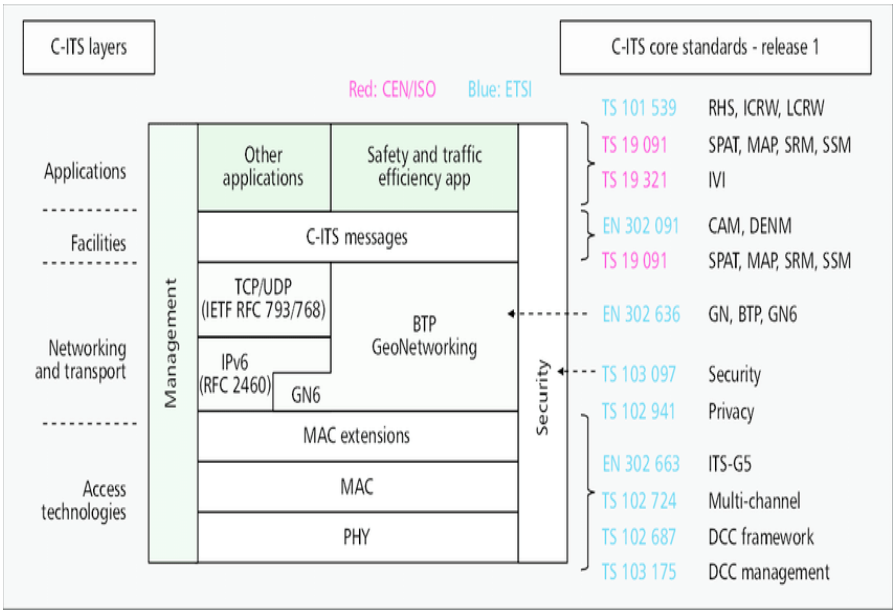

Figure 2.5: C-ITS Protocol Stack.

2.2.2 C-ITS Messages

The facility layer standard of C-ITS protocol stack defines the messages. Some of them are similar to DSRC messages, make the European standard compatible with the American one. Basic Set of Applications (BSA) is a set of obligatory applications used in the ITS station. The BSA describes a set of traffic and transport use cases. Below, the two most important C-ITS messages are described:

Cooperative Awareness Message (CAM)

CAM functionality is like BSM. CAM should be generated and broadcast every 0.1[s] to 1[s]. This frequency is defined considering the environment like congestion and application.

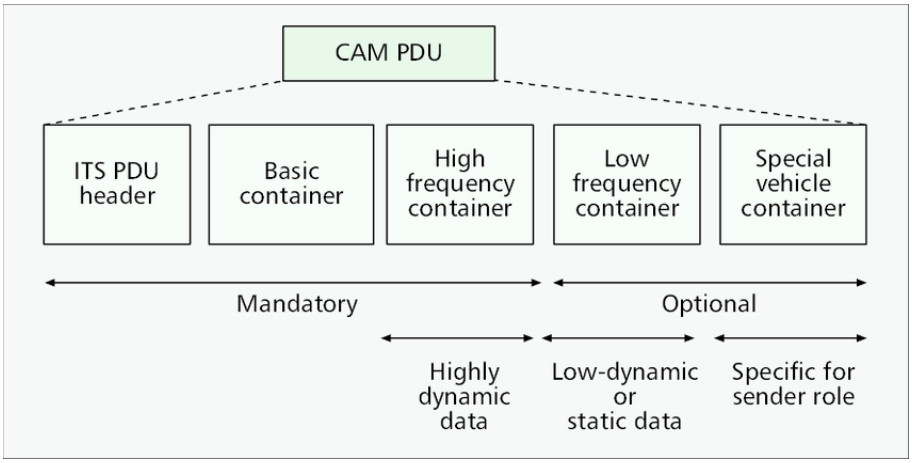

Figure 2.6: CAM Format.

Figure 2.6 shows the format of CAM message.In mandatory part there are three parts:

- **ITS PDU Header:** It is CAM protocol information. It contains:
  - Protocol Version
  - Message ID
  - Sender Address
- **Basic Container:** It contains:
  - Station Type
  - Position
- **High Frequency Container:** It contains:
  - Heading
  - Speed
  - Acceleration

Optional CAM parts:

- **Low Frequency Container:** It contains non-safety information like vehicle path history.
- **Special Vehicle Container:** It is used by special vehicles Like emergency Vehicles or for vehicles carry high-risk material.

DENM (Decentralized Environmental Notification Message)

DENM message is broadcast in a critical situation to make other vehicles aware of a potential road safety risk. Figure 2.7 shows DENM PDU format.

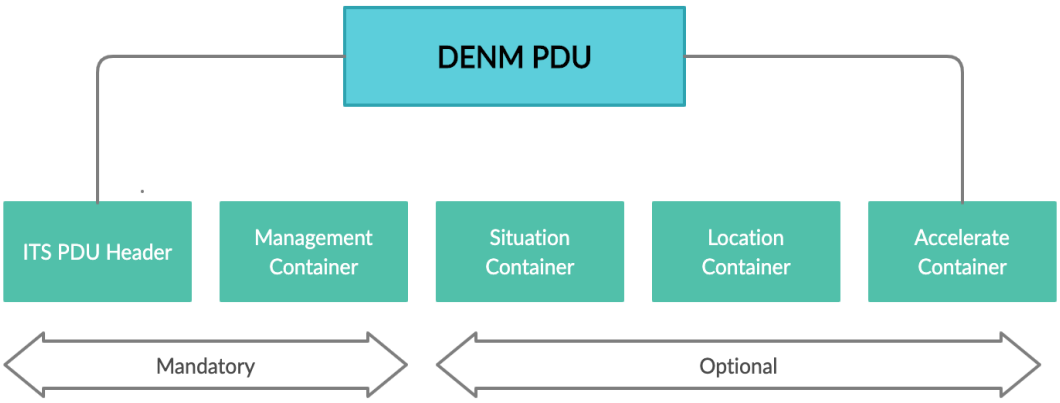

Figure 2.7: DENM Format.

DENM mandatory part is composed by two fields:

- **ITS PDU Header:** It contains:
  - Protocol Version
  - Message ID
  - Sender Address
- **Management Container:** Contains event information.

DENM Optional Part:

- **Situation Container:** It contains special event situation data and its severity.
- **Location Container:** It contains vehicle location data like:
  - Position
  - Speed
  - Heading
  - Trace
- **Accelerate Container:** It is used to transmit application specific contents, like lane position, impact reduction, road works, etc.

Chapter 3

The Virtual Traffic Light System

3.1 Simulation Tools

For developing simulation software, SUMO, VEINS, and OMNET++ were employed to build the whole scenario among different tools. Each has its characteristics for simulating networks' urban mobility, resulting in a useful and powerful tool for simulating microscopic urban mobility scenarios.

3.1.1 OMNET++

Omnet++ is a powerful simulator that simulates networks, including a variety of wired and wireless networks. Omnet++ has either a graphical and command-based interface. Also, using many libraries makes is more powerful [8].

3.1.2 SUMO

German Aerospace Center developed the Simulation of Urban Mobility. It is a microscopic, open-source simulator to simulate different traffic models, using different tools to design, modify, and implement mobility environments in different scales [9].

3.1.3 VEINS

VEINS is an inter-vehicular simulator interacting with Omnet++ as a discrete event simulator and Sumo as an urban mobility simulator, making it suitable to use

during the experiment. Figure 3.1 shows the relation between three tools, making them a powerful tool for simulating urban mobility scenariosm[10].

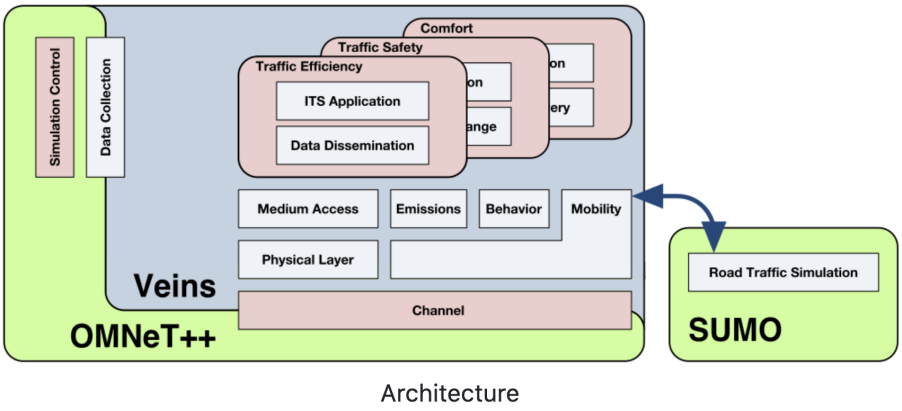

Figure 3.1: Simulator Architecture.

3.2 Software Development Method

To develop simulator software for the scenario, Bohem's spiral software development model was simplified and practiced. After each increment, the software tested, debugged and sort of simulations ran. Then, a new cycle started to apply new features and improvements. The final software was the result of many incremental layers. Figure 3.2 shows the schematic of the spiral model [11].

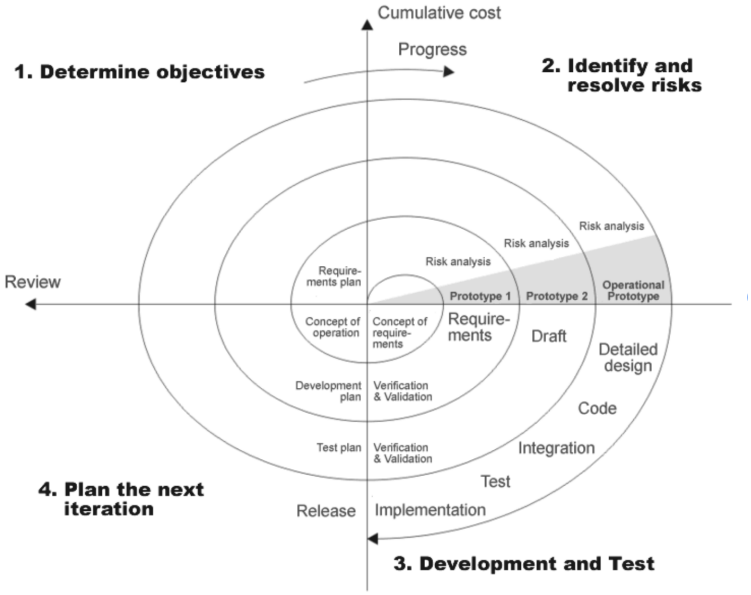

Figure 3.2: Bohem's Spiral Software Development Model.

3.3 Simulation Environment

For the first part of the scenario, a four-way junction was designed. Each direction has two lanes opposite to each other with disallowed turn over maneuver at the intersection. Therefore three allowed movements for each vehicle at the intersection were:

1. **Turning right**

2. **Turning left**

3. **Straight on**

As any movement had to be defined to build the scheduling algorithm, which will be explained in section 3.5.1, the fourth movement, which was *Stop at the Intersection* added. Four traffic lights were placed, and no roadside unit (RSU) was used; therefore, the only way of communication was the interconnection between the vehicles.

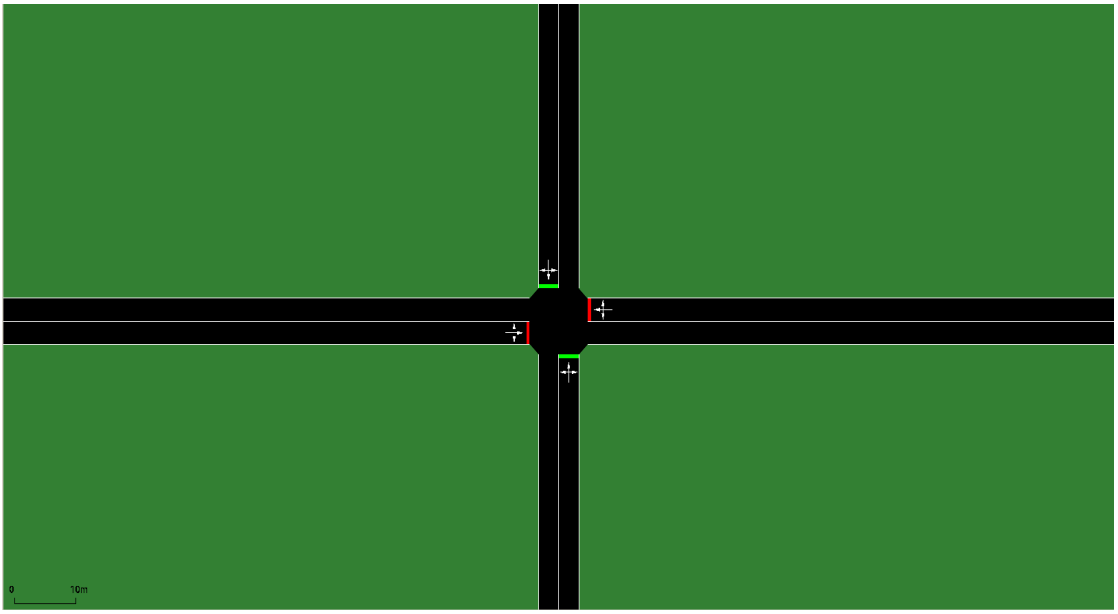

Figure 3.3: Simulation Environment (Four Traffic Lights, Two are In Green Phase and Two are In Red Phase).

In the list below the environment variables are listed:

- **TxPower:** 20 [mW]
- **Bitrate:** 6 [Mbps]
- **MinPowerLevel:** -110 [dBm]
- **NoiseFloor:** -98 [dBm]
- **AntennaOffsetZ:** 1.895 [m]
- **BeaconInterval:** 0.1 [s]
- **WhereAmIInterval:** 0.1 [s]
- **Nc:** 6 (Number of cars in each group)
- **Leader:** 3, 4 (Number of leaders to be seen to trigger scheduler algorithm)

3.3.1 Vehicles

To model vehicles, an XML file was prepared. The maximum speed of 50 Km/h was considered for each vehicle with an acceleration of 2.6 m/s and a deceleration of 5.0 m/s. The sample of the vehicle shown below:

<vType accel="2.6" decel="5.0"id="CarA" maxSpeed="13.9" minGap="2.0">

3.3.2 Attenuation

The junction was surrounded by buildings, adding additional attenuation to the signals sending by vehicles. In total, three varieties of attenuation were employed:

- **Simple Path Loss Model**: It models and computes the signal's attenuation in spaces in an obstacle-free environment. It attenuates the transmitted signal concerning the distance between the sender and the receiver, scales the transmission power into a logarithmic way of a certain amount of dB per meter.
- **Simple Obstacle Shadowing Model**: The transmission power is reduced due to the obstacles isolation between the sender and the receiver, which severely affects the transmitted signal.
- **Nakagami Fading Model**: It is used to model attenuation due to fading interference. The fading is a variation of the signal attenuation caused by fast divergence of the signal produced by multi-path propagation. Nakagami model represents these variations of the signal with a random attenuation created according to the Gamma distribution.

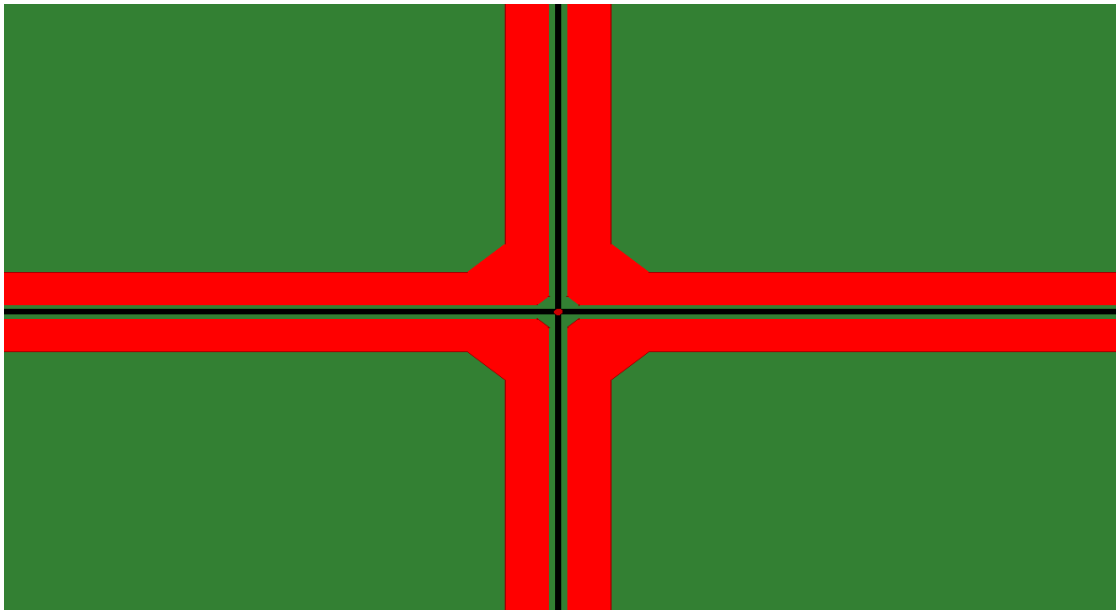

Figure 3.4: Applying Attenuation Effect by Adding Buildings.

3.4 Scenario

The scenario aimed to design, implement and test a model to simulate the V3Tl (V2V-based Virtual Traffic Light) [12]. Three objectives considered during architecture and implementation of the scenario:

- Interconnection between vehicles is the only way of communication, meaning that no RSU is used. It was done to minimize the cost and complexity of the project's real implementation, which is one of the pillars in project design.
- The model should pass more vehicles in a minute than other scenarios (unregulated intersection and intersections with traffic lights). Therefore, the time efficiency of cars should be increased.
- The Stop and Go rate per vehicle should reduce to increase fuel efficiency and consequently reduce air pollution and CO2 emission.

  Many functions were implemented to reach all objectives that played important roles in the project, and the simulator was responsible for orchestrating all functions.

3.4.1 Basic Safety Message

The first step for each vehicle is to learn about its position periodically and also to know about other vehicles of its surrounding by sending and receiving *BSM* The information which is carried by modified BSM which was developed to satisfy project needs is shown below:

- **Sending Time :** Sending time added by transmitter vehicle.
- **Vehicle ID**: Each vehicle has its own unique ID.
- **Vehicle Position**: Vehicle's geographical position.
- **Vehicle Velocity**: Indicates the speed of the vehicle.
- **Vehicle Direction**: Indicates the direction of the vehicle approaching the intersection.
- **Vehicle Leader String**: A file in the leader vehicle contains the information of all group members.
- **Vehicle Intersection String**: A file contains all information of vehicles approaching the intersection that are subscribed in the groups.

- **Vehicle Solution Dataset**: Output file of scheduler which has scheduling information.
- **Vehicle Leader Election Flag**: Its value changes to true if the vehicle is elected as a leader.
- **Vehicle Intersection Flag**: Its value changes to true if the vehicle passes the intersection.
- **Vehicle Scheduled Flag**: Its value changes to true if the vehicle is scheduled.
- **Vehicle Road ID**: The ID of the road the vehicle is passing through.
- **Vehicle Position In Queue**: The position of the vehicle in the group.
- **Vehicle Turn For Pass**: Indicates when the vehicle should pass the intersection according to Solution Dataset information.

3.4.2 OnWhereAmI()

The *OnWhereAmI()* function is in charge of update vehicle environmental information. After each 0.1 [s], the new information overwrites to the previous values saved in the vehicle storage.

3.4.3 OnReceiveBSM()

The *OnReceiveBSM()* function is in charge of receiving BSMs. If there is no record of the sender's vehicle in the storage, it stores data as a new record and keeps it. Otherwise, it just overwrites the previous information with the updated data. Therefore, if there is any vehicle in the radio visibility, the information is recorded and updated periodically, which is the key point for the process's next steps.

3.4.4 LeaderElection()

The next step is *Leader Election*. The *Nc* defines how many cars can exist in each group of vehicles. For this experiment, the *Nc* fixed to 6. Each vehicle checks its position with the nearest vehicle ahead, and after receiving their information via BSM, the vehicle understands that it should join the group ahead (if the group is not full) or should become a new leader and waits for other members to join. If the vehicle could not find any vehicle ahead, it becomes a new leader until it finds an incomplete group ahead. In this situation, the vehicle withdraws its leadership and joins to the incomplete group. As a consequence, the vehicles behind the mentioned vehicle becomes a new leader, and the procedure repeats until the incomplete group becomes full.

The dynamic behavior of leader election freezes near the intersection to give the already elected leaders enough time to collect their group information and participate in scheduling procedures in a stable way. Generally, the behavior is like a sliding window between cars in which every Nc vehicle is in the same group (window) until the leader election situation changes and group members updated. Figure 3.5 shows in the west direction, vehicles are doing leader election procedures.

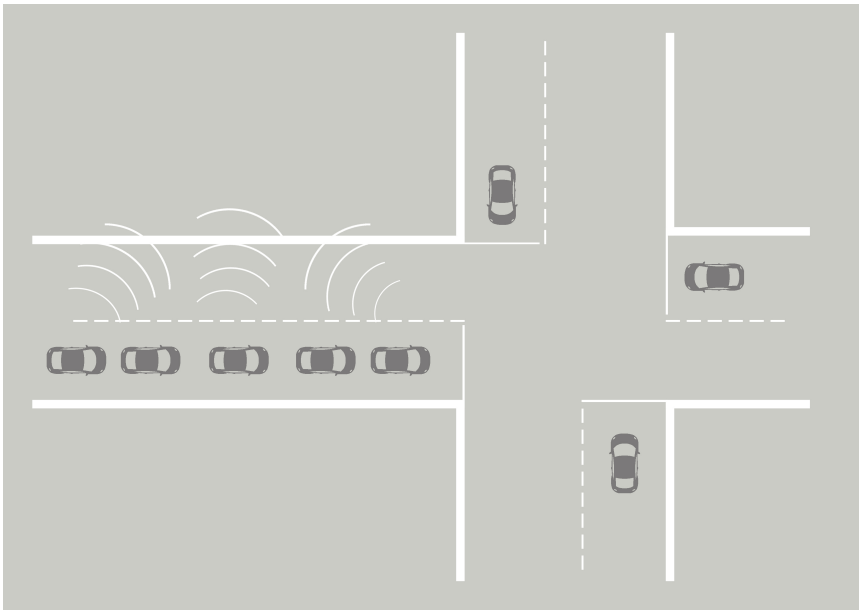

Figure 3.5: Leader Election Procedure.

Algorithm 1 shows the procedure of the leader election.

Algorithm 1: Leader Election Algorithm

```
if (There is no vehicle ahead OR The group ahead is full) then
    Leader Election Flag=TRUE;
     Queue Position=1;
else
    Queue Position=Queue Position of Vehicle ahead+1;
end
```

3.4.5 LeaderString()

The leaders are in charge of collecting important data from their group members. After the leadership election, each leader starts to recognize its group members and collect their necessary information for the rest of the procedure. The leader of each group broadcasts the unified group information stores in a string variable and is

put in BSM. As figure 3.6 shows, the orange car is elected as Direction Leader (DL) and starts to collect Direction Data set (DD). The result is Leader String (LS).

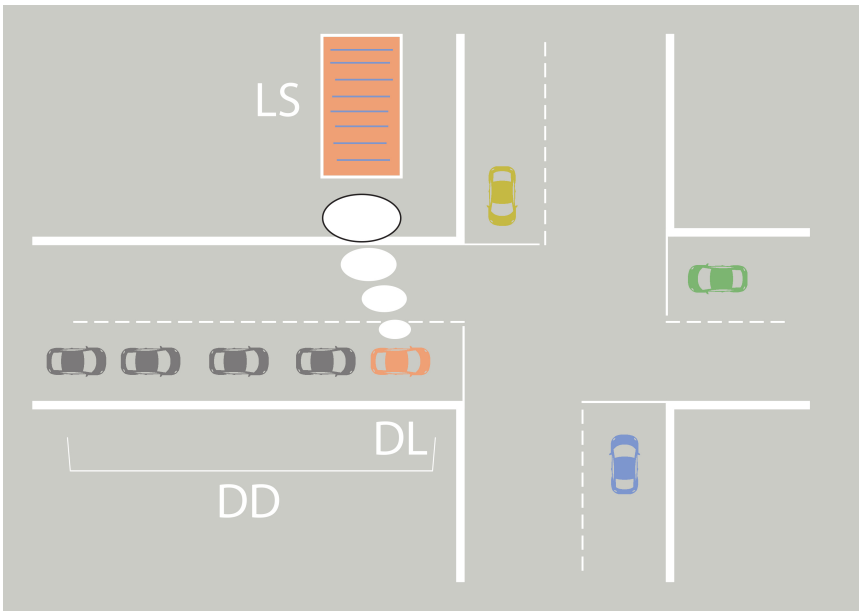

Figure 3.6: Leader String Procedure.

Algorithm 2 shows the Procedure of the leader string algorithm.

Algorithm 2: Leader String Algorithm

```
if (I am leader) then
    for (Each vehicle in my group) do
        LeaderString+=VehicleInfo;
    end
end
```

For each group member a list of variables collected and stored:

- **Vehicle ID**
- **Geographical position**
- **Direction**
- **Speed**
- **Heading**
- **Distance to the intersection**
- **Position in Queue**

3.4.6 IntersectionString()

When a leader gets close to the intersection and sees at least another leader from different directions, it starts to exchange its leader-string with other leaders and merge their leader-strings with its own. Then it makes the intersection string; the intersection string has essential information for scheduling. Therefore, each leader has the same view of all vehicles, followed by each leader in different directions. It should be mentioned that the leaders collect the nearest leader strings of each direction but not the other leaders, therefore at each tier maximum of four leaders mutually exclusive concerning their directions, participate in the intersection string. Figure 3.7 shows the schematic of the intersection string.

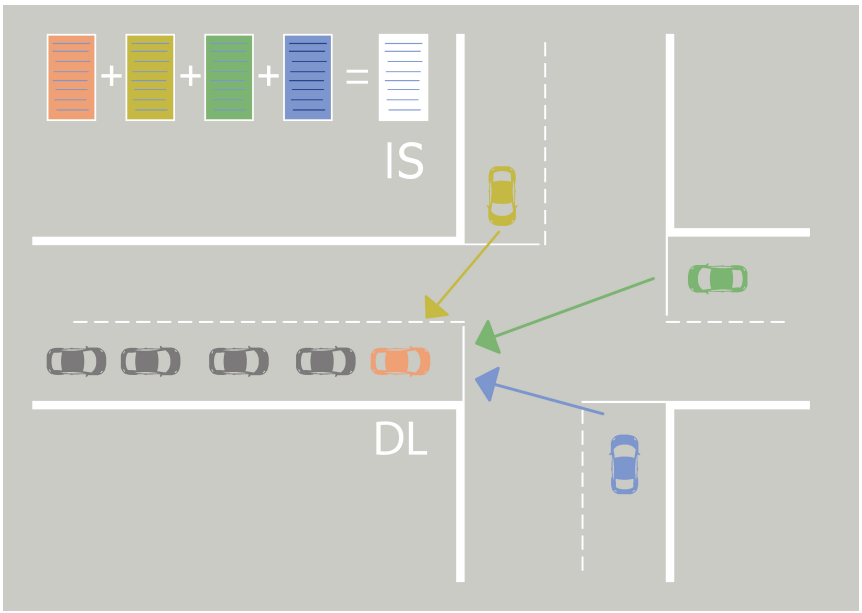

Figure 3.7: Intersection String Procedure.

Algorithm 3 shows the procedure of intersection string.

Algorithm 3: Intersection String Algorithm

```
if (I am a leader) then
    for (Each vehicle stored in database) do
        if (The vehicle is a leader of another direction) then
            if (The leader is nearest to intersection in its direction) then
                IntersectionString+=LeaderString;
            end
        end
    end
end
```

3.5 Intersection Modeling

From this point to start scheduling, each leader should know in prior the *allowed* movements in the intersection; *allowed* movement is a move that satisfies two conditions:

1. Being one of the four movements which mentioned in 3.3 (Turn right, Turn left, Straight on or Stop).
2. The movement should not cause a collision with another vehicle passing the junction.

To meet these two conditions, modeling the junction which considers all different *allowed* movements was necessary, therefore, four combinations of *allowed* movements considered.

- **Four movements in parallel**
- **Three movements in parallel**, (therefore in one direction there should be no vehicle or it should stop).
- **Two movements in parallel and two stops**
- **One movement and three stops**

A bitmasking model was used to find the different possible allowed conditions using the Python programming language. As an output, a file with forty-nine possible allowed movements generated. As shown in figure 3.8, the east side vehicle, and the west side vehicle are stopping at the intersection, and the north side and the south side vehicles are turning right. If there is just one movement that changes the bit to one at each time in the yellow area, the movement is safe. Otherwise, it means that there is a collision, and other movements are discarded to come back

to a safe condition. To model the situations in which there is no vehicle at one of the directions, an imaginary vehicle with *always stop* movement situation added to the hypothesis to help the scheduler have a comprehensive view of the intersection during the scheduling process.

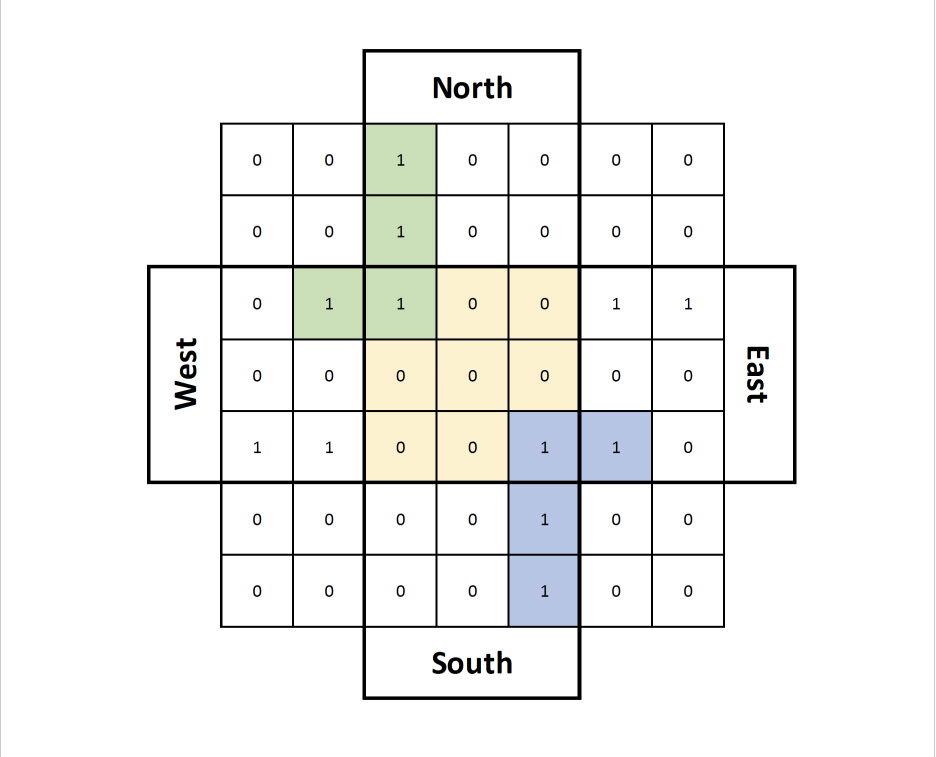

Figure 3.8: Movements Bit Masking: (South is turning right; North is turning right, East and West stop).

In figure 3.9, the first group, which is indicated with green arrows, satisfies both mentioned conditions and could be the first tier of simultaneous movements. After that, the second tier (orange arrows) can pass the intersection, and eventually, the third tier, blue arrows, pass the intersection.

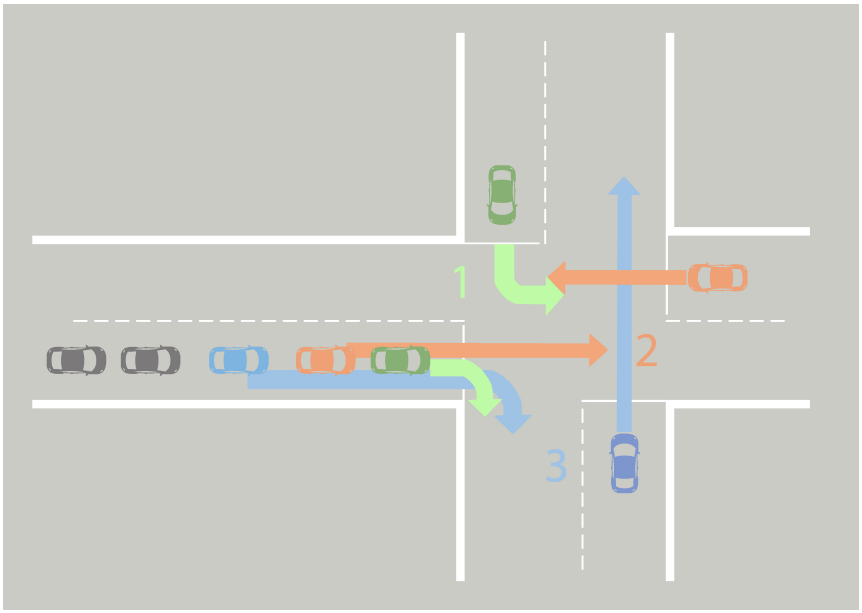

Figure 3.9: A Sample of Legal Moves.

3.5.1 Legal Movements Output

As a convention, each row of the output file represents one set of legal movements, and each column represents a direction with the following order:

1. **East**
2. **West**
3. **South**
4. **North**

The fifth column represents the number of parallel moves in each row. Each number in first four columns represents a movement at this order:

0. **Stop**
1. **Turning right**
2. **Turning left**
3. **Straight on**

As it is shown in figure 3.10, for example, the row number one, indicates four parallel following movements: east turning right, west turning right, south turning

right, and north turning right. The other example: row number ten indicates: east turning left, west stops/there is no car on the west, south turning right, and north turning right.

1	11114
2	11103
3	11203
4	13103
5	11013
6	11023
7	31013
8	10113
9	10313
10	20113
11	01113
12	01133

13	02113
14	11002
15	13002
16	31002
17	33002
18	10102
19	10202
20	10302
21	20102
22	10012
23	10022
24	20012

25	30012
26	01102
27	01202
28	02102
29	03102
30	01012
31	01022
32	01032
33	02012
34	00112
35	00132
36	00312

37	00332
38	10001
39	20001
40	30001
41	01001
42	02001
43	03001
44	00101
45	00201
46	00301
47	00011
48	00021
49	00031

Figure 3.10: All possible Movements Output Format.

3.6 Scheduler

The scheduling process starts as soon as enough leaders from different directions communicate and make the intersection string. The minimum leader's value is a variable that can be defined at the beginning of the scenario, indicating the scheduler's triggering condition. Its range is between two and four. The first leader, which satisfies the conditions, starts to run the scheduler process. The scheduler considers two main goals to schedule the vehicles:

1. **Maximize the parallel allowed movements for each tier**
2. **Minimize the stop and goes per vehicle**

The other important factor in designing the scheduler is processing time. The algorithm should produce the output file before the leaders pass the intersection. After several tests, the minimum distance of a leader to the intersection that satisfies the conditions was 5.13 meters, and the maximum was 96.38 meters. Therefore, For the worst-case scenario, the algorithm should be fast enough to overcome the minimum time limit. Equation 3.1 shows the worst-case scenario of minimum time for the scheduler to prepare the output.

$$T_{Output} + BSMbroadcastDelay <= \frac{MinimumDistance}{Vehicle'sMaximumSpeed} \tag{3.1}$$

Therefore,

$T_{Output} <= \frac{5.13m}{13.9m/s} - 0.1[s]$,

as a result: $T_{Output} <= 0.269[s]$.

Two algorithms were designed and developed to meet the deadlines. The first one used a decision tree and considered one level in-depth at the tree. Therefore, it found the results in a reasonable time. The second approach, which named *Look-Ahead* algorithm, was slower as it considers two levels in the depth of the decision tree. However, it could reach better results in maximizing the number of passing cars in the same situation concerning the first algorithm.

3.6.1 Scheduler (First Algorithm)

The algorithm starts with receiving as an input the intersection string, and the possible movements file which is placed in the vehicle storage. Then, the algorithm begins to build a decision tree according to the intersection string. The tree is dynamic, meaning that after scheduling each tier, the remaining unscheduled vehicles are updated to determine which cars are in the next first tier if the previously scheduled vehicles pass the intersection. This procedure continues until the scheduler processes the last unscheduled vehicle of the last level.

According to figure 3.11, to maximize the number of passing cars (P), the scheduler at each level searches for the maximum available parallel movements; therefore, for the first level, path A is the best candidate (P of path A > P of path B). Consequently, for the second decision, path A1 is selected. The third decision, A1-1, and A1-2 have the same (P); therefore, the algorithm searches to find the minimum Stop and Goes (S) between them, resulting in choosing A1-2. The algorithm observes the same situation for two options A1-2-1 and A1-2-2, as both P and S of two candidates are equal; therefore, the scheduler randomly chooses one. When the algorithm reaches the decision tree's last level, remaining unscheduled vehicles are collected with their new positions in their queues, and a new decision tree is built. This procedure repeats until all the vehicles are being scheduled.

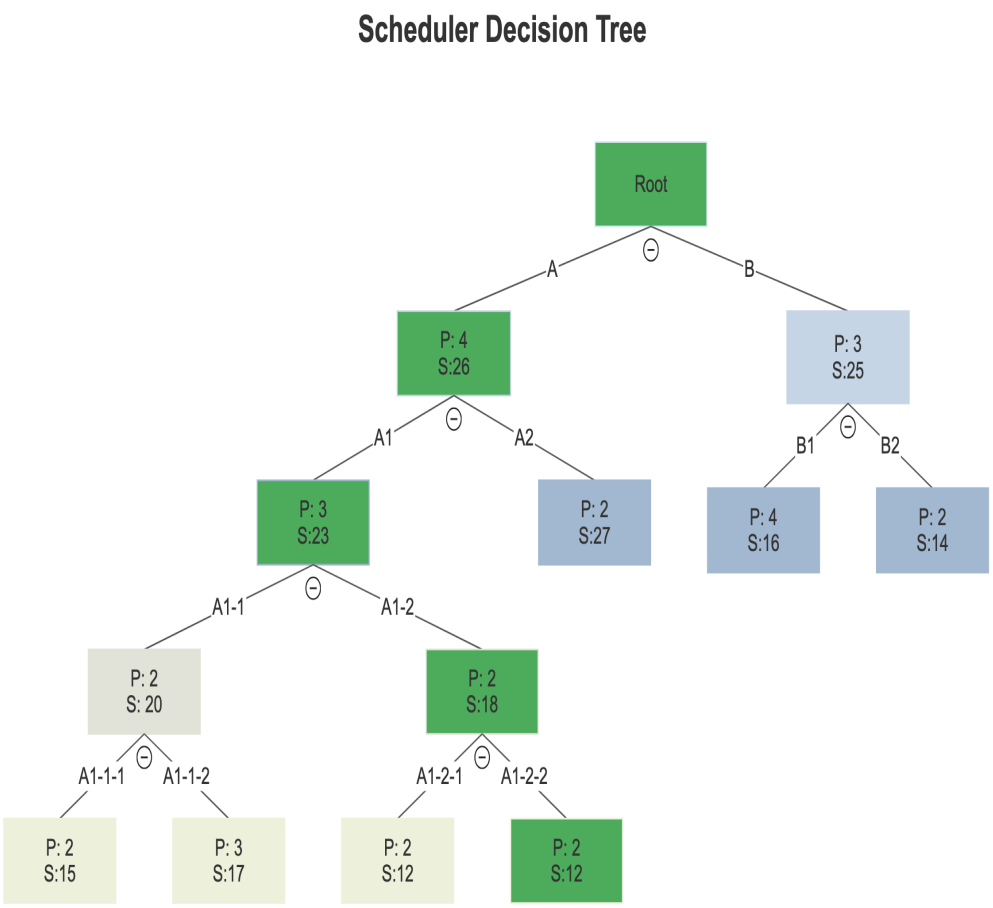

Figure 3.11: Decision Tree Sample, (P: Number of possible parallel passes, S: Number of Stop and Goes).

Figure 3.12 shows the algorithm processing time vs. the maximum time limit allowed for the scheduler. There is a sufficient gap between the algorithm processing time and its time limit.

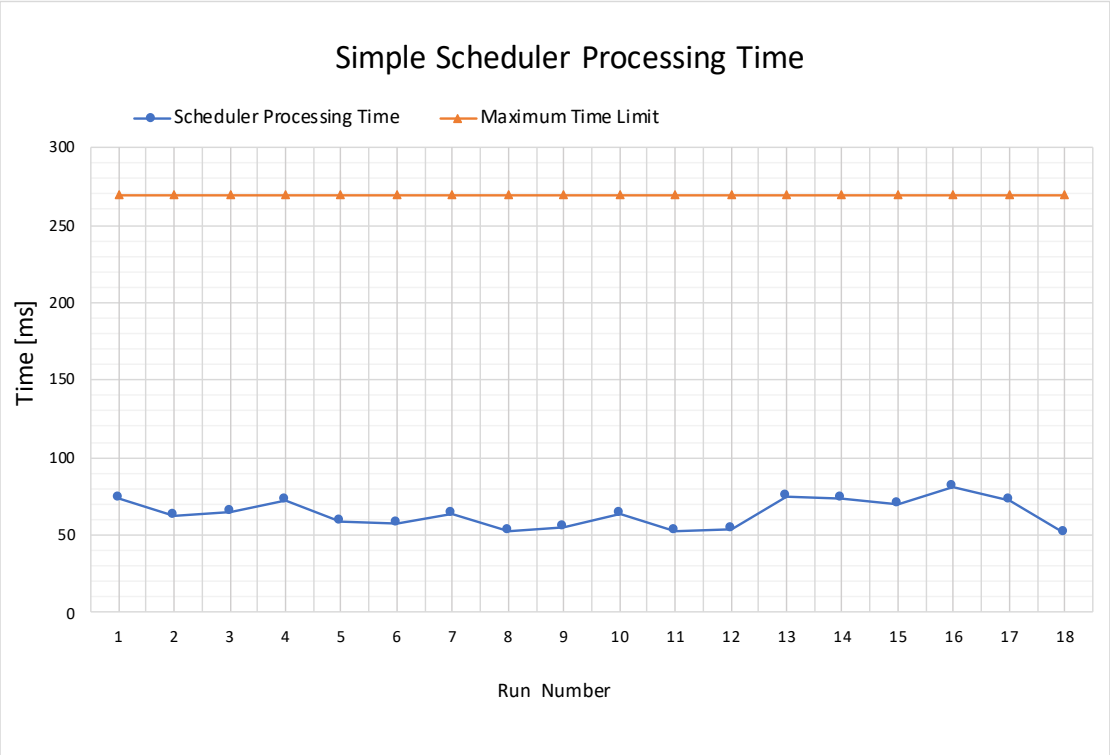

Figure 3.12: Simple Scheduler Processing Time.

Algorithm 4: Scheduler Algorithm (Simple)

```
if (The number of leaders observing is grater than the defined value) then
    VehicleCount= The number of vehicles in the intersection string;
    while VehicleCount>0 do
        Make all possible combinations with the first tier of the vehicles;
        Compare with legal movements to determine which ones are legal;
        Find the maximum possible movements among the all legal moves;
        if (Two moves cause the same number of passing cars) then
            Check the number of stop and goes if the movements happen;
            Choose the minimum value of stop and goes;
            if (Number of Stop and Goes are the same) then
                Randomly choose between the candidates;
            end
        end
        Store the selected movement;
        Update the number of Stop and Goes considering the movements;
        Update the pointers with respect to selected move;
        VehicleCount-= Number of vehicles can pass in the selected
         movements;
    end
    Store the full solution in SolutionDataset in BSM;
end
```

After the algorithm finishes, a complete set of solutions is provided for all of the vehicles involved in the scheduling process. Solution Data set will save to BSM and, as a consequence, starts to broadcast.

3.6.2 Scheduler (Look-Ahead Algorithm)

To reach better results, *Look-Ahead Algorithm* proposed, this algorithm is an improvement of the previous one, with the more computational complexity. However, as shown in the next section, it achieves better results by proposing better combinations resulting in more passing vehicles at each tier, which was the most critical pillar in the algorithm design. The *Look-Ahead Algorithm* uses a decision tree like the previous algorithm, but at each time, it goes one more level in-depth and considers the possible future combinations concerning the recent decision; therefore, at each time, the algorithm can analyze the consequences of the recent decision in the remaining combinations and suggest a smarter solution for each level of the scheduling.

As the figure 3.13 shows, in the Look-Ahead algorithm, at each level, the *Sum* variable is the sum of the node's P-value and the maximum value among node's children P-values. Therefore, for the same tree, for the first decision, because the

Sum value of path A and B are the same, the algorithm looks for the smaller number of Stop and Goes (S), and path B is selected. Then, because path B1 has the greater *Sum* value than B2, B1 is selected, and for the last level, B1-2 is chosen due to the equal value of *Sums* but the smaller value of S.

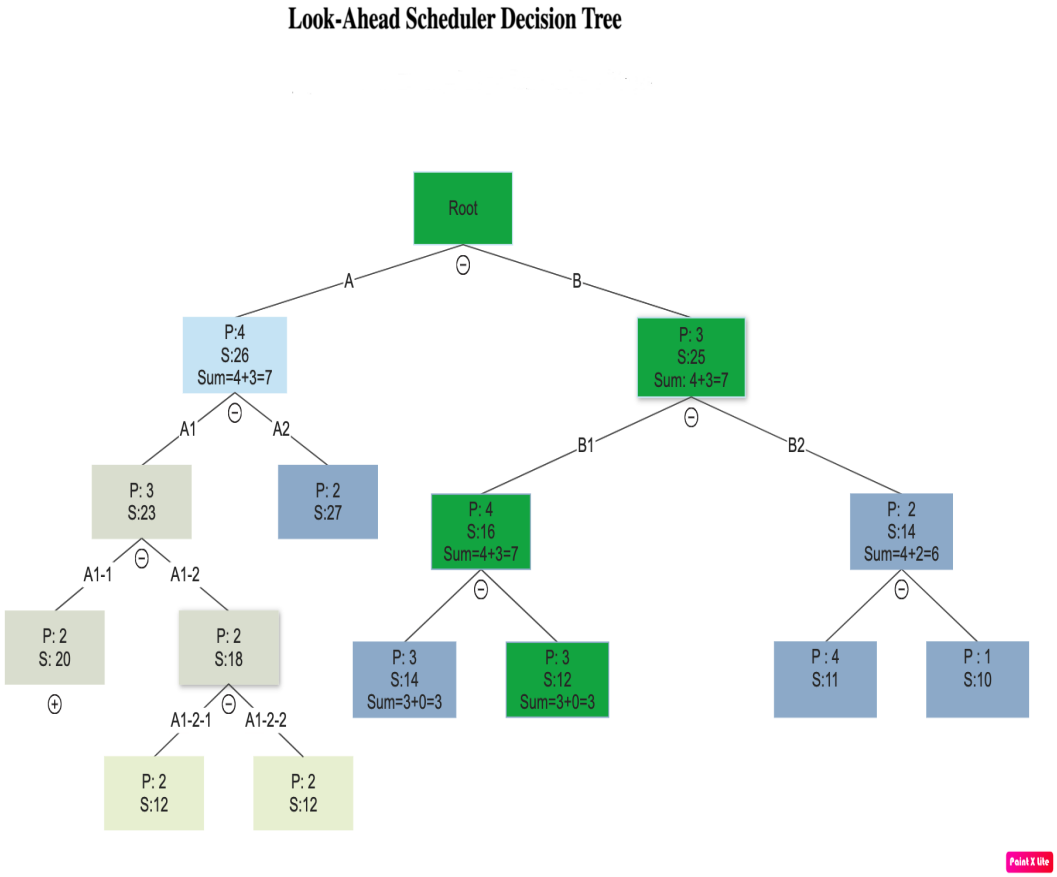

Figure 3.13: Look-Ahead Decision Tree Sample,(P: Number of possible parallel passes, S: Number of Stop and Goes, Sum: Sum of the node's P value and the maximum value among node's children P values).

The Look-Ahead algorithm has to consider at each time two levels and calculate the P values of all its children; therefore, the processing time is longer, however as it is shown in 3.14, there is still a good gap between the maximum time limit and the response of the scheduler, making it possible to use the Look-Ahead scheduler in the simulator.

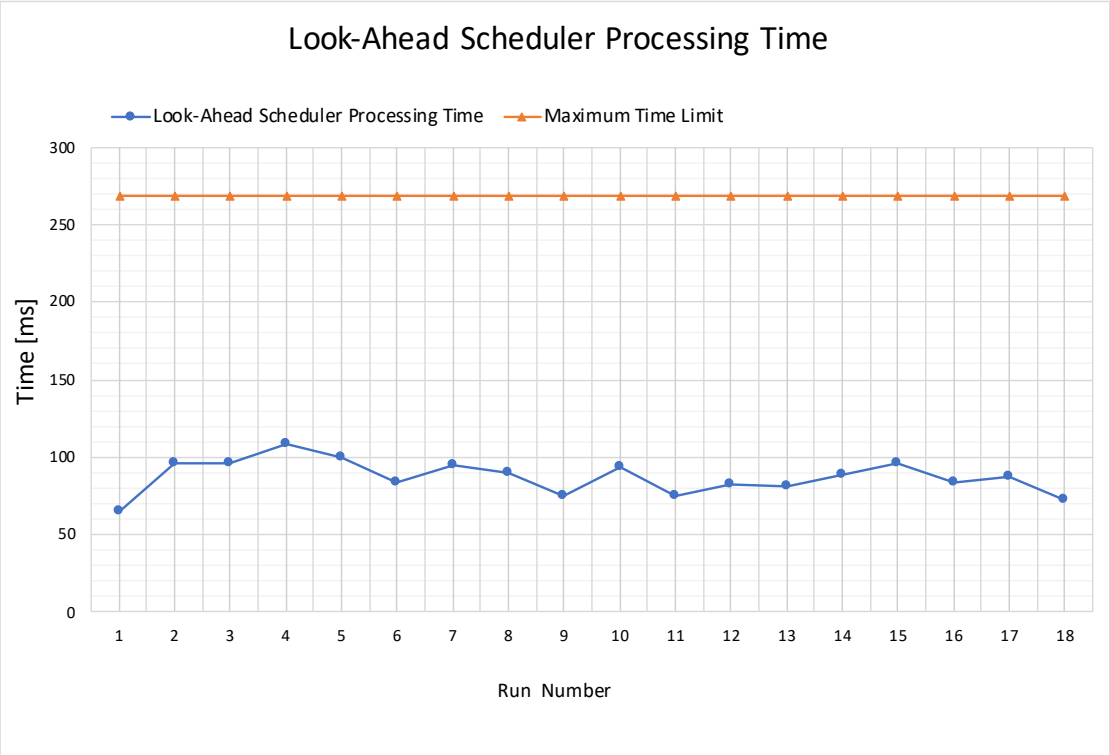

Figure 3.14: Look-Ahead Scheduler Processing Time.

Algorithm 5: Scheduler Algorithm (Look-Ahead)

```
if (The number of leaders observing is grater than defined value) then
    VehicleCount= The number of vehicles in the intersection string;
    while VehicleCount>0 do
        Make all possible combinations with the first tier of the vehicles;
        Compare combinations with legal movements to determine which
         ones are legal;
        Consider the next move with respect to all legal movements;
         Temporarily update the decision tree with respect to each
         movement to find out the next tier possible movements;
        Sum the number of simultaneous moves of each movements and its
         greatest child;
        Choose the greatest sum in the tree;
        if (Two moves cause the same number of sum) then
            Check the number of Stop and Goes if the movements happens;
            Choose the minimum value of Stop and Goes;
            if (The number of Stop and Goes are the same) then
                Randomly choose between the candidates;
            end
        end
        Store the selected movement;
        Update the number of stop and goes;
        Update the pointers with respect to selected move;
        VehicleCount-= Number of vehicles passing in the selected
         movements;
    end
    Store the full solution in SolutionDataset in BSM;
end
```

3.6.3 Comparison of Two Scheduling Algorithms

The comparison between two algorithms was focused on two main aspects:

- **Algorithm processing time**
- **Levels to clear the intersection**

For the algorithms' processing time, as there is enough margin between the time limit and the algorithms processing time, both algorithms outputs are acceptable because they both met the deadline. Otherwise, it was too late for any result, and it was useless to prepare the output due to the speed of vehicles that they may pass the intersection before receiving the solution data set.

As figure 3.15 shows, obviously, the first algorithm has a better processing time than the Look-Ahead algorithm. However, both algorithms perform well enough to prepare the output before the deadline. Therefore, both algorithms are good candidates for the scenario.

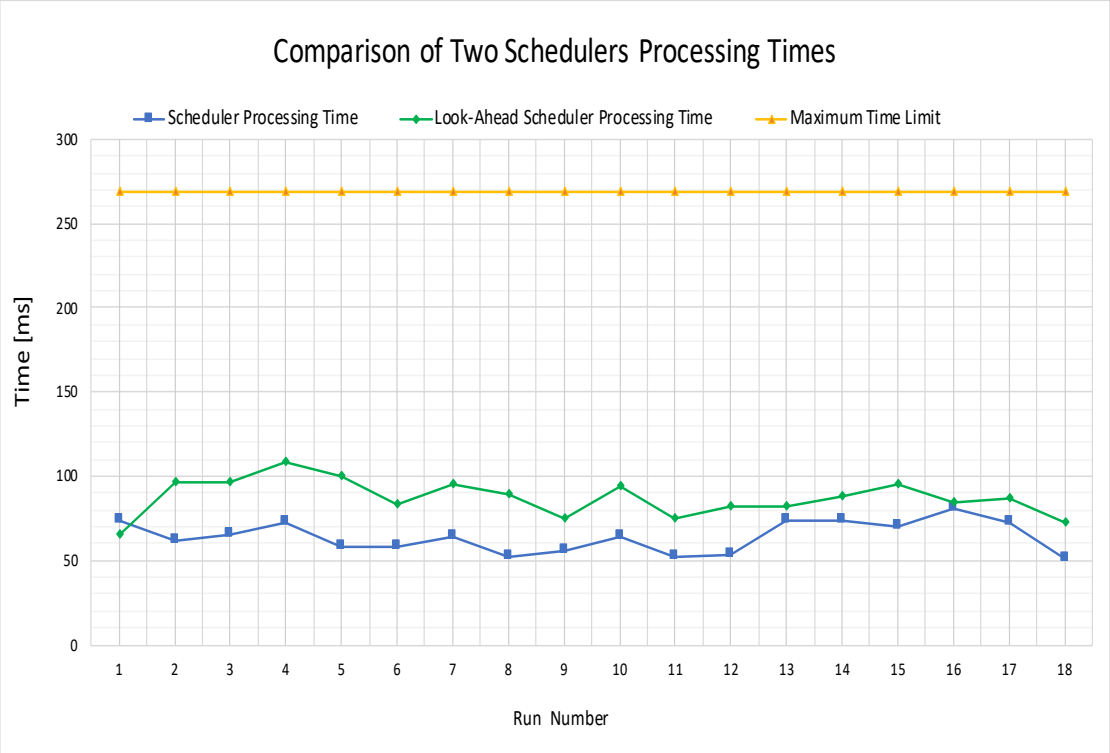

Figure 3.15: Comparison of Processing Times Between Two Algorithms.

Concerning the second criteria, levels to clear the intersection meaning for each group of 24 vehicles (as Nc=6 and there are four directions) are scheduled in how many levels all of them can pass the intersection. As figure 3.16, the Look-Ahead algorithm performs 14.6% better in scheduling cars. Therefore for the same situation, the output of the Look-Ahead algorithm is more efficient. As a result, the Look-Ahead algorithm was selected for the rest of the simulations.

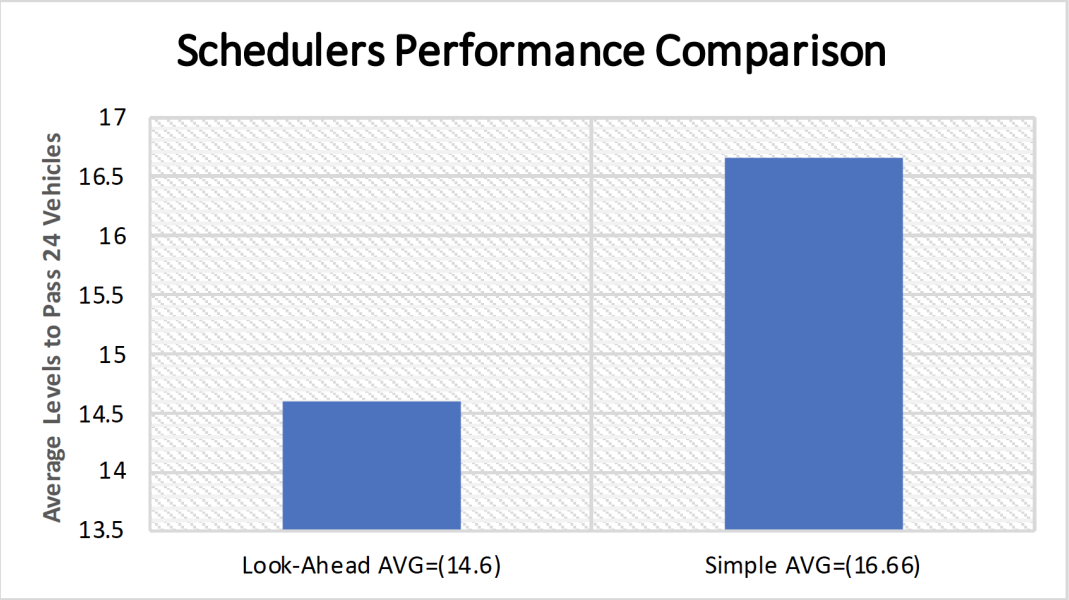

Figure 3.16: Performance Comparison of the Two Scheduling Algorithms.

3.6.4 Scheduler Output

Scheduler output is a matrix that indicates at each time which vehicle can pas the intersection satisfying all conditions. To make the result more efficient, each row of the two-dimensional matrix is the priority level of passing, each column is a direction, and each member in the matrix is a vehicle indicates by its unique id. Therefore, only with a two-dimensional integer matrix, the scheduler can show all the necessary information for the involving vehicles. Figure 3.15 shows the SolutionDataset (SD) broadcast by the leader sharing with other vehicles.

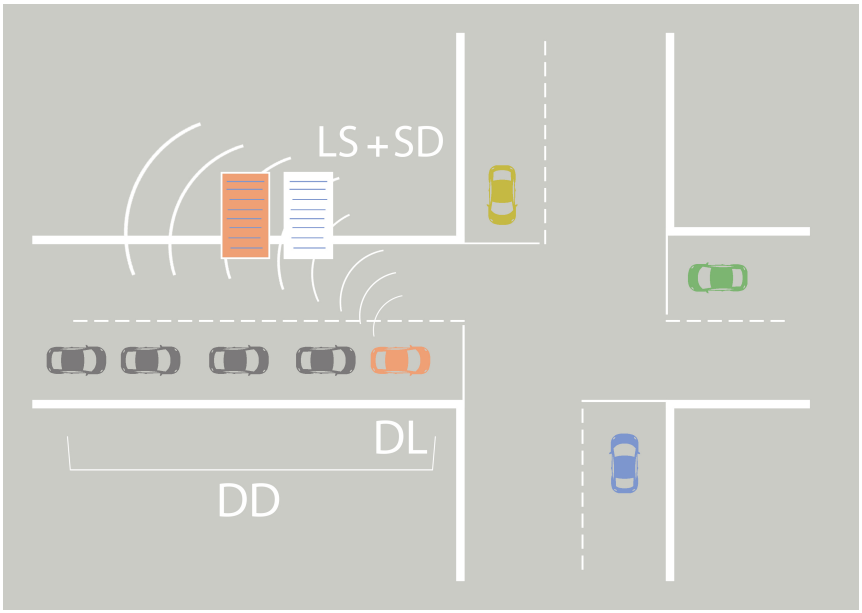

Figure 3.17: Solution Dataset Procedure.

As figure 3.18 shows, the output file indicates vehicles from each direction can pass the intersection at which time. For example, at time zero, vehicle id forty-three from the south and vehicle thirty-one from the north can simultaneously pass the intersection. In two other directions, vehicles are forced to stop, indicated by zero in the output. At time one, the first two vehicles have passed the intersection, re-positioning happens, and now vehicles seventeen, thirteen, forty-five, and thirty-four are in the first tier. The scheduler decides to pass vehicle forty-five from the south and thirty-four from the north, and other vehicles are forced to stop, and so on. At this example, twenty-four vehicles could pass the intersection in 15 levels.

	East	West	South	North
Time 0	0	0	43	31
Time 1	0	0	45	34
Time 2	0	0	47	39
Time 3	0	0	0	40
Time 4	0	0	49	44
Time 5	0	0	52	46
Time 6	0	13	57	0
Time 7	0	16	0	0
Time 8	17	19	0	0
Time 9	29	21	0	0
Time 10	33	23	0	0
Time 11	0	26	0	0
Time 12	37	0	0	0
Time 13	50	0	0	0
Time 14	53	0	0	0

Figure 3.18: Sample of Scheduler Output Format.

3.7 SolutionSetter()

After the scheduler sets the solution data set with appropriate data, the leader starts to share the solution data set with other leaders. Then, they share the solution with their members. Therefore, each vehicle read the solution data set. If it finds itself at the solution data set, it set its turn for pass according to the solution and changes the value of isScheduledFlag to True, meaning it is scheduled. Other vehicles behind understand that if they are not in the solution data. If so, they will participate in the next scheduling process following their leaders. During scheduling, no vehicle can join, leave, or change its group.

Algorithm 6: Solution Setter Algorithm

```
if (The vehicle receive a solution dataset via BSM) then
    if (The vehicle id is indicated in the dataset) then
        IsScheduledFlag=TRUE;
        TurnToPass=Row number that the id was indicated in the solution
         data set;
    else
        Discard the message;
    end
end
```

3.8 MoveAllower()

The function *MoveAllower()* is in charge of managing vehicles to pass in their turn. Each vehicle checks its TurnTOPass value with other vehicles approaching the junction, and according to its turn, does one of the following:

- **Pass the intersection**: If the other vehicles have the same values or greater values of TurnTOPass, or there is no other vehicle approaching the intersection at that time.
- **Stop**: If a vehicle with a smaller TurnToPass value still not passes the intersection.

If the vehicle stops, at each cycle (0.1[s]), it checks the situation by calling the function. The vehicle passes the intersection only if the conditions are satisfied.

Algorithm 7: Move Allower Algorithm

```
for (Each vehicle in the database) do
    Check TurnToPass value of the vehicle with other vehicles at the
     junction;
    if (TurnToPass of vehicle is equal or less than other vehicle OR there is
     no vehicle approaching) then
        Pass the intersection;
    else
        Stop;
    end
end
```

3.9 Two Intersections Scenario

After the first part of the scenario has been finished, the simulation environment extended to two connected intersections. Each intersection inherited the characteristics of the previous scenario. All possible routes increased to 30 routes as there are more paths that each vehicle can go. For applying the attenuation, buildings were designed and added to the simulation environment. Some fields added to BSM to help vehicles to realize at which intersection they are. Each vehicle can participate in one of the two scheduling processes on the intersections according to its position. If the vehicle have to pass two intersections, all necessary variables reset after passing the first junction, and the vehicle starts to participate in a new leader election and farther procedures. Then, if the situation satisfies, the vehicle participates in a new scheduling process. Figure 3.19 shows the simulation environment.

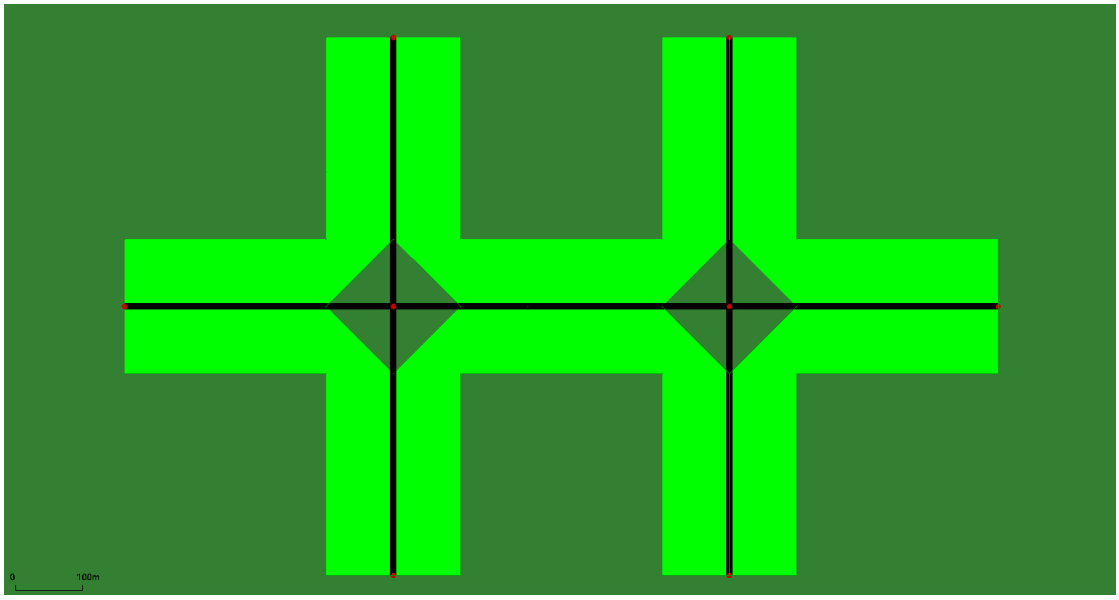

Figure 3.19: Two Intersections Simulation Environment.

This environment was created for future implementations like applying congestion control to the schedulers, synchronizing two intersection schedulers, and other further implementations.

3.9.1 Future Works

A congestion control procedure can be applied for two intersections scenario. There are no traffic lights used, and if many vehicles were are coming to the intersection, aiming the same direction, traffic jams might happen. In a congestion situation, two signals can be produced:

- **Forward congestion signal**
- **Backward congestion signal**

The last vehicle at each group is in charge of checking if the number of cars around is more than a defined threshold. Then, concerning the difference with threshold, three situations can be assumed:

- **High-Intensity congestion**
- **Medium-Intensity congestion**
- **Low-Intensity congestion**

Therefore in a congestion situation, the responsible vehicle will broadcast a five-bit message putting on BSM. The first two bits indicate the directions, for example, 00: north, 01: south, 10: east, and 11: west, the other three bits mean: 000: No Congestion, 001: Low Congestion Forward, 010: Medium Congestion Forward, 011: High Congestion Forward,101: Low Congestion Backward, 110: Medium Congestion Backward, and 111: High Congestion Backward. Each vehicle receiving the message will check its position with the sender signal, set the forward or backward signal, and hand it over to the vehicles near the intersections. The responsible leader for scheduling will check the signal and, according to the signal, changes the scheduler's decisions. The leader that wants to do the scheduling first checks for any congestion signal. If there is any signal, the leader decodes the message and discovers at which direction with which density there is a congestion. Then, it changes the scheduling process according to the situation. As a result, in correspondence, the next intersection serves more vehicles from the congested path, and the junction before the congested path sends fewer vehicles to the blocked area until everything backs to the normal situation. The pseudo-code 8 shows the procedure of the algorithm.

Algorithm 8: Congestion Avoidance Algorithm

```
if (The vehicle is a leader who is running the scheduler) then
    if (Congestion signal != 00000) then
        Decode the signal to find out the severity and the direction of
         congestion;
        if (The congestion is ahead) then
            Serve the vehicles who are going into the congested area with
             lower probability;
        else
            Serve the vehicles who are going out of the congested area with
             higher probability;
        end
    else
        Run Look-Ahead Scheduler;
    end
end
```

Chapter 4

Performance Evaluation

After developing the simulation software, many different scenarios with different criteria were applied, including the two main categories, which were finally selected. Each selected scenario was designed in three different situations and then was tested several times to reach confident results. The main two categories were:

a) Homogeneous traffic: The model that all four directions experience the same rate of traffic on average. These three scenarios were selected for collecting the results:

(i) Unregulated Scenario

(ii) Traffic Lights Scenario

(iii) Scheduler Scenario

b) Non homogeneous traffic(40-40-20): In two directions, the model experience forty percent of traffic each, and another twenty percent of traffic is distributed between two other directions. For this category again, these three scenarios were selected:

(i) Unregulated Scenario

(ii) Traffic Lights Scenario

(iii) Scheduler Scenario

Unregulated intersection, by default, is implemented in VEINS. Therefore two other scenarios had to be implemented. For the scheduler scenario, the simulation software developed. Therefore the only remaining test environment which was needed to be implemented was *Traffic Lights* scenario, which needed a method to precisely calculate the green phase time, yellow phase time, and red phase time of each of the four traffic lights.

4.1 Webster Method

Webster method introduced by F.V Webster [13] is an approach for processing the traffic lights signal times in an optimum way. This method is used vastly in civil engineering. All duration which traffic light changes from different signals to the signal it starts the process named a *cycle time*; there are the definitions necessary for computing the optimum cycle length and each signal phase:

- **Cycle length:**
  - Green Interval
  - Red Interval
  - Clearance Interval: Time to clear the intersection after changing a signal to red.
  - Change Interval: Duration of yellow signal.
- **Saturation Flow (S):** Number of vehicles passing the intersection in a homogeneous nonstop traffic per hour.
- **Lost time (L):** Stands for all possible time losses vehicle can face while passing the intersection.
- **Critical Flow Rate (y_i):** The rate of the maximum estimated flow rate between directions in one phase over the saturation flow.
- **Optimum Cycle Length (C_0):** Indicates the optimum value of the cycle to experience the least delay for all directions.

$$C_0 = \frac{1.5 * L + 5}{1 - \sum y_i} \tag{4.1}$$

- **Green Time Signal Length (G_i):** Describes green time duration.

$$G_i = \frac{y_i}{\sum y_i} * (C_0 - L) \tag{4.2}$$

4.2 Fixed Traffic Generation Rate Analysis

In these series of the simulation, the vehicle generation rate was fixed to one, means in total at each second one car was generated to pass the intersection. As it was explained at the beginning of the chapter 4, in total, six different situations were considered to these series of the simulation. For the calculation of the traffic lights scenario with the Webster method, a series of analysis was done, which are explained below:

4.2.1 Homogeneous Traffic Analysis

For the fixed-rate homogeneous traffic, after applying the Webster method, the following results were achieved. After feeding the intersection with vehicle generation rate 1[s], 3600 vehicles passed the intersection in one hour. Therefore:

$S = 3600 \frac{[Vehicle]}{[Hour]}$;

$v_N = v_S = v_W = v_E = 900[Vehicles]$;

$y_{N,S} = y_{W,E} = \frac{1800}{3600} = 0.5$;

Considered loss time for each phase= 2[s] .

Therefore the optimum cycle time obtained:

$C_0 = \frac{(1.5*4)+5}{1-0.5} = 22[s]$;

And the green phase time for each of the two phases:

$G_i = \frac{0.5}{1} * (22 - 4) = 9[s]$.

Finally, one second assigned to each yellow phase.

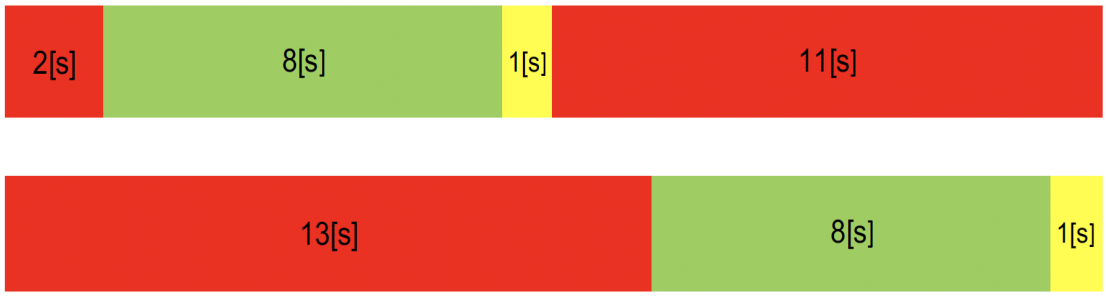

Figure 4.1: Optimum Cycle Time (Homogeneous Scenario).

4.2.2 40-40-20 Scenario Traffic Analysis

For this scenario, the generation rate was one for the intersection, distributed in the following way: 40 percent of traffic was from the east, 40 percent from the west, and 20 percent of traffic uniformly distributed between north and south directions; therefore, there was 80 percent of the traffic in one phase and 20 percent in another

phase. These results obtained:

$S = 3600 \frac{[Vehicle]}{[Hour]}$;

$v_N = v_S = 360[Vehicles]$;

$v_W = v_E = 1440[Vehicles]$;

$y_{W,E} = \frac{2880}{3600} = 0.8$;

$y_{N,S} = \frac{720}{3600} = 0.2$;

Considered loss time for each phase= 2[s] .

Therefore, the optimum cycle time obtained:

$C_0 = \frac{(1.5*4)+5}{1-(0.4+0.1)} = 22[s]$;

$G_{W,E} = \frac{0.8}{1} * (22 - 4) = 14.4[s]$;

$G_{N,S} = \frac{0.2}{1} * (22 - 4) = 3.6[s]$.

Eventually, one second assigned to each yellow phase.

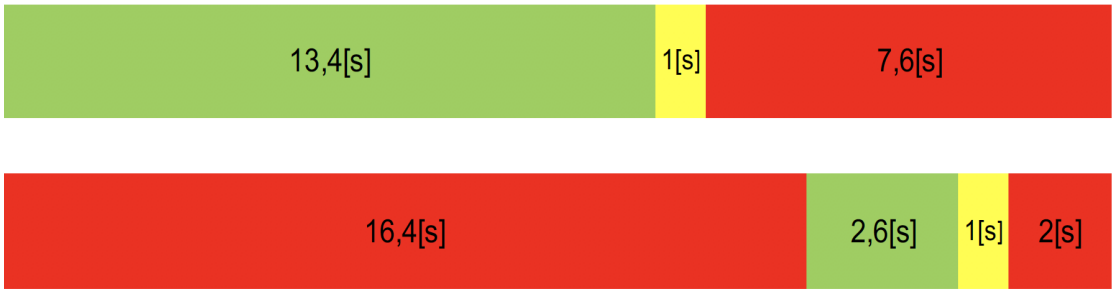

Figure 4.2: Optimum Cycle Time (40-40-20 Scenario).

4.3 Variable Traffic Rate Analysis

As there is a variable traffic generation rate for this part of the scenario, different variables should be considered. However, in the traffic lights scenario, for different rates, the result of the Webster method was the same as the previous

scenario. As an example for the homogeneous part, if we conider $\lambda = 2$, we will have:

$S = 1800 \frac{[Vehicle]}{[Hour]}$;

$v_N = v_S = v_W = v_E = 450[Vehicles]$;

$y_{N,S} = y_{W,E} = \frac{900}{1800} = \frac{1}{2}$;

Considered loss time for each phase= 2[s];

Therefore the optimum cycle time obtained:

$C_0 = \frac{(1.5*4)+5}{1-0.5} = 22[s]$;

And the green phase time for each of the two phases:

$G_i = \frac{0.5}{1} * (22 - 4) = 9[s]$.

Which is the same result we had for another generation rate equal to $\lambda = 1$. This hypothesis is also true for non-homogeneous scenario.

4.4 Results

This section dedicates to the simulations' results, showing the performance of the new approach and comparisons. Two categories and their subcategories are discussed, providing graphs and other statistics at the end of each section, a result summary is shown to show the better view of the statistics.

4.4.1 Fixed Generation Rate

After trying several generation rates, $\lambda = 1$ was selected for the series of simulations. The input file prepared using Python programming language, the vehicle paths selected randomly between twelve different paths, and the Poisson process to generate vehicle departure time was employed. The output format ready-to-place in the simulator was a file in *xml* format defining the vehicle type, paths, and departure times. In the figure 4.3 a sample of input file is shown.

```
<routes>
 <vType accel="2.6" decel="5.0" id="CarA" maxSpeed="13.9" minGap="2.0" sigma="0.5"/>
<!-- routes -->
 <route id="route1" edges="D5 D6" /> <!-- W2E -->
 <route id="route2" edges="D7 D8" /> <!-- E2W -->
 <route id="route3" edges="D1 D2" /> <!-- S2N -->
 <route id="route4" edges="D3 D4" /> <!-- N2S -->
 <route id="route5" edges="D5 D4" /> <!-- W2S -->
 <route id="route6" edges="D5 D2" /> <!-- W2N -->
 <route id="route7" edges="D7 D2" /> <!-- E2N -->
 <route id="route8" edges="D7 D4" /> <!-- E2S -->
 <route id="route9" edges="D1 D6" /> <!-- S2W -->
 <route id="route10" edges="D1 D8" /> <!-- S2E -->
 <route id="route11" edges="D3 D8" /> <!-- N2W -->
 <route id="route12" edges="D3 D6" /> <!-- N2E -->
<vehicle depart="0" id="0" route="route1" type="CarA"/>
<vehicle depart="0" id="1" route="route4" type="CarA"/>
<vehicle depart="0" id="2" route="route1" type="CarA"/>
<vehicle depart="0" id="3" route="route7" type="CarA"/>
<vehicle depart="1" id="4" route="route10" type="CarA"/>
<vehicle depart="2" id="5" route="route8" type="CarA"/>
<vehicle depart="3" id="6" route="route12" type="CarA"/>
<vehicle depart="3" id="7" route="route2" type="CarA"/>
<vehicle depart="3" id="8" route="route5" type="CarA"/>
<vehicle depart="4" id="9" route="route4" type="CarA"/>
<vehicle depart="5" id="10" route="route3" type="CarA"/>
<vehicle depart="7" id="11" route="route4" type="CarA"/>
<vehicle depart="8" id="12" route="route10" type="CarA"/>
<vehicle depart="11" id="13" route="route11" type="CarA"/>
<vehicle depart="13" id="14" route="route5" type="CarA"/>
<vehicle depart="15" id="15" route="route7" type="CarA"/>
<vehicle depart="16" id="16" route="route10" type="CarA"/>
<vehicle depart="18" id="17" route="route7" type="CarA"/>
```

Figure 4.3: Traffic Input File Sample.

After applying inputs, there were several raw output files generated each in text format and measuring different variables like the number of stops and goes per minute, the number of passing cars, scheduler processing time, and the number of times that scheduler triggered. Post data processing was applied using Python programming language and Microsoft Excel to highlight the correlations, compare and visualize the results.

Homogeneous Scenario

For this part three different scenarios were considered:

(a) Unregulated Intersection Scenario : Using unregulated intersection.

(b) Scheduler Scenario : Using the developed simulator.

(c) Traffic Light Scenario : Using traffic lights which at each time two directions are in green phase simultaneously.

The assumptions of the simulation are listed below:

- **Simulation Time:** 13 Minutes (780 [s])
- **Transient Period:** 3 Minutes
- **Number of Runs:** 5

– **Lambda(Generation Rate):** 1

– **Data collection period:** Every 1 minute

– **Number of samples:** 10

Figures 4.4, 4.5 and 4.6 show the confidence interval 95% of the passing cars during the simulation.

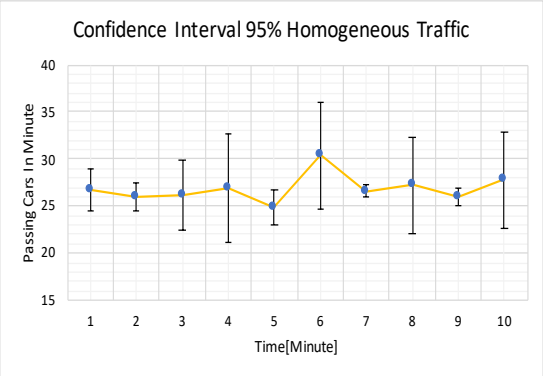

Figure 4.4: Confidence Interval 95%, Passing Cars, Unregulated Intersection Scenario.

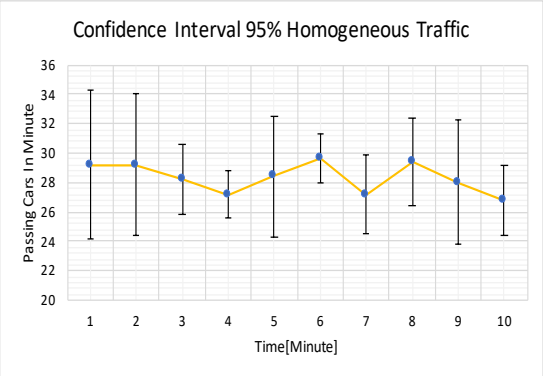

Figure 4.5: Confidence Interval 95%,Passing Cars, Traffic Lights Scenario.

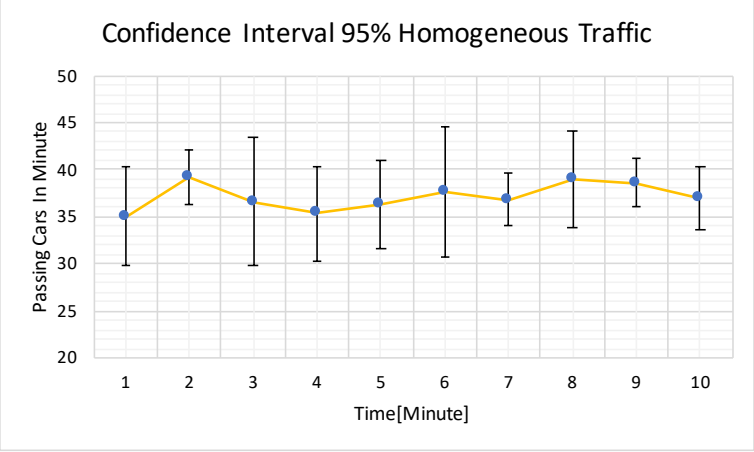

Figure 4.6: Confidence Interval 95% , Passing Cars, Scheduler Scenario.

Figures 4.7, 4.8 and 4.9 show the confidence interval 95% of the stop and go ratio per car during the simulation.

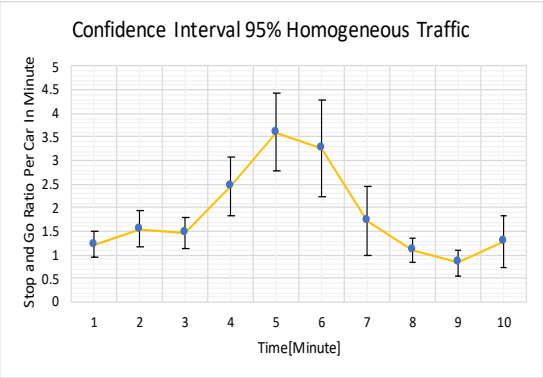

Figure 4.7: Confidence Interval 95%, Stop and Go per Car, Unregulated Intersection Scenario.

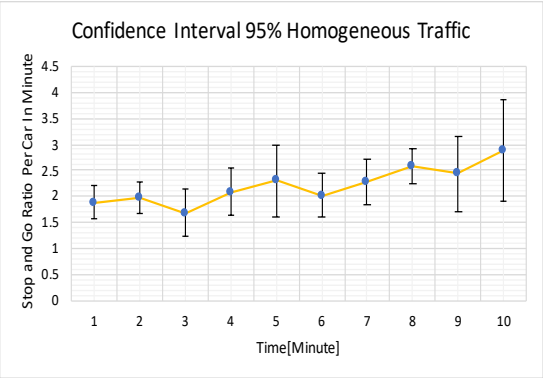

Figure 4.8: Confidence Interval 95% , Stop and Go per Car, Traffic Lights Scenario.

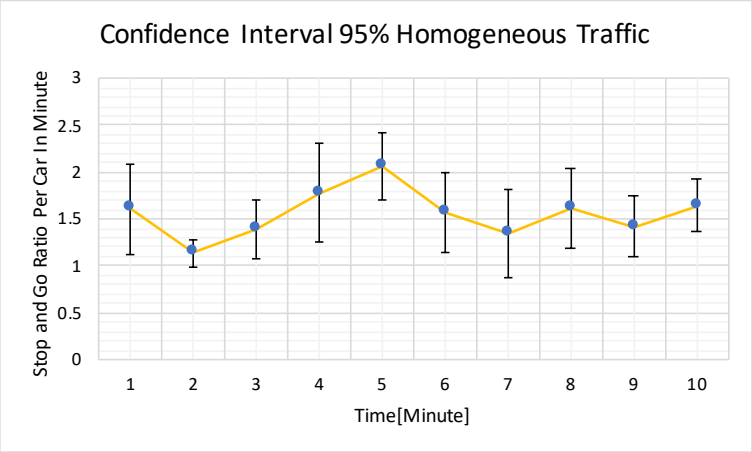

Figure 4.9: Confidence Interval 95%, Stop and Go per Car, Scheduler Scenario.

As the figure 4.10 shows, the average number of passing cars using the scheduler is significantly higher than two other scenarios, (respectively 41% and 32% better than Unregulated and Traffic Lights scenarios), which causes better time efficiency for the vehicles to pass the intersection in less time than other two methods.

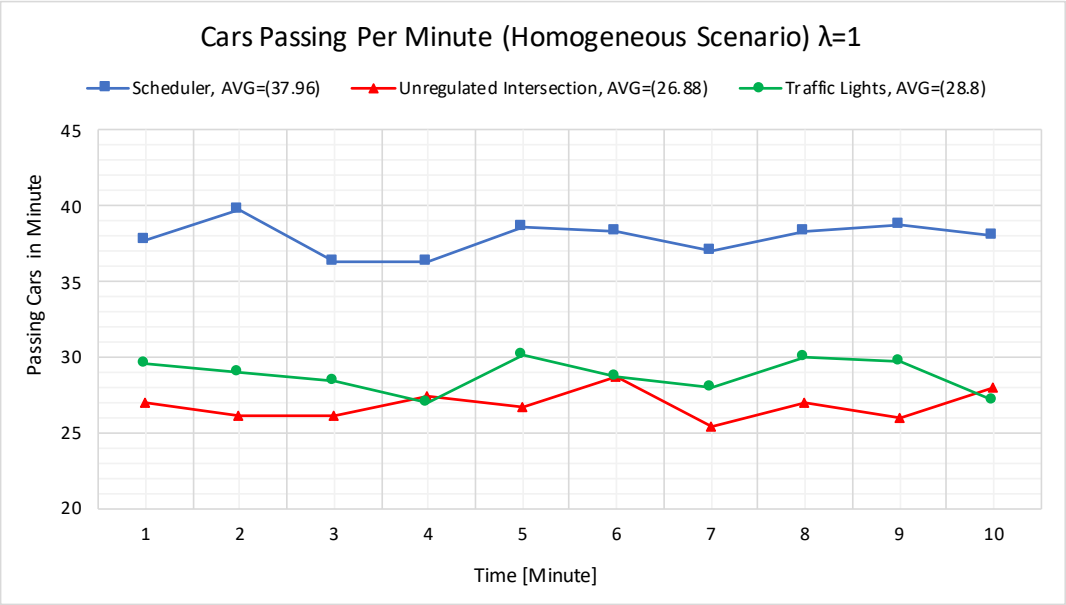

Figure 4.10: Comparison of Number of Passing Cars Per Minute.

Figure 4.11 shows the comparison of stop and go per car for three scenarios. As the figure shows the average ratio of stop and goes per car for scheduler scenario is less than two other methods, as a result, we expect less fuel consumption and less air pollution.

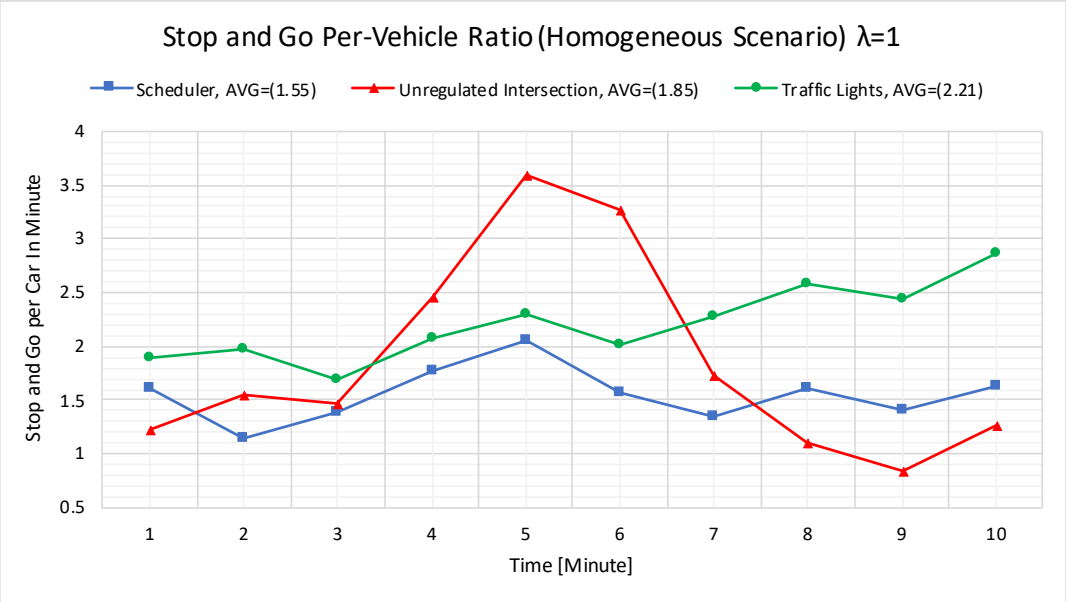

Figure 4.11: Comparison of Stop and Go Per Car Ratios.

Non-Homogeneous Scenario

For this part of the experiment, the general conditions were the same as the previous experiment (4.4.1), and again car generation rate (λ) was equal to one. In contrast, the vehicles' distribution over each direction was 40 percent from the west, 40 percent from the east, and 20 percent distributed equally to the north and south directions. The simulation assumptions are listed below:

- Simulation time: 13 minutes (780 [s])
- Transient period: 3 minutes
- Number of runs (different seeds): 5
- Lambda: 1
- Data collecting period: every 1 minute
- Number of samples for each simulation : 10

Figures 4.12, 4.13 and 4.14 show the confidence interval 95% of passing cars during the simulation.

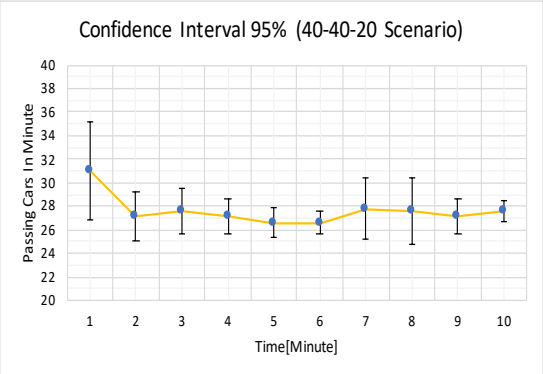

Figure 4.12: Confidence Interval 95%, Passing Cars, Unregulated Intersection, (Non-Homogeneous) Scenario.

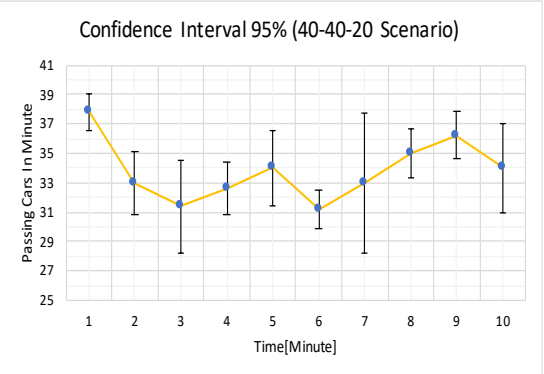

Figure 4.13: Confidence Interval 95%, Passing Cars, Traffic Lights, (Non-Homogeneous) Scenario.

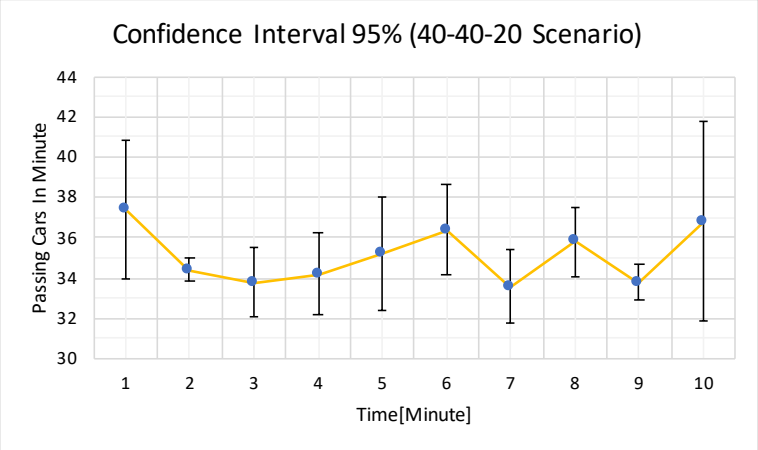

Figure 4.14: Confidence Interval 95%, Passing Cars, Scheduler,(Non-Homogeneous) Scenario.

Figures 4.15, 4.16 and 4.17 show the confidence interval 95% stop and go per car during the simulation.

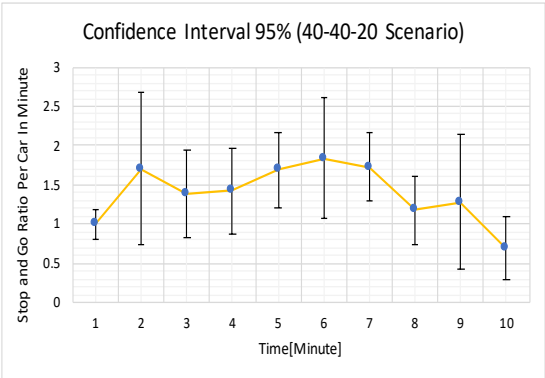

Figure 4.15: Confidence Interval 95%, Stop and Go per Car, Unregulated Intersection.

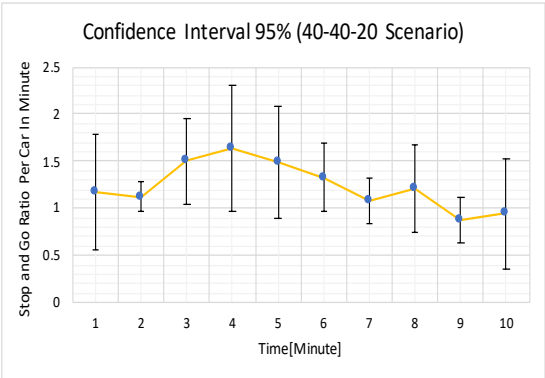

Figure 4.16: Confidence Interval 95%, Stop and Go per Car, Traffic Lights Scenario.

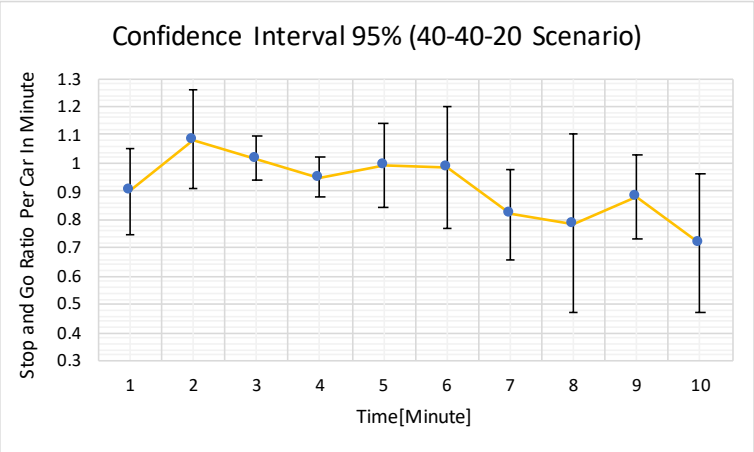

Figure 4.17: Confidence Interval 95%, Stop and Go per Car, Scheduler Scenario.

Figure 4.18 shows the average number of passing cars for three scenarios, as it is shown, the performance of the scheduler is significantly higher than the unregulated scenario, and is still better than the Traffic Lights scenario. the results show the advantage of using the scheduler instead of two other methods.

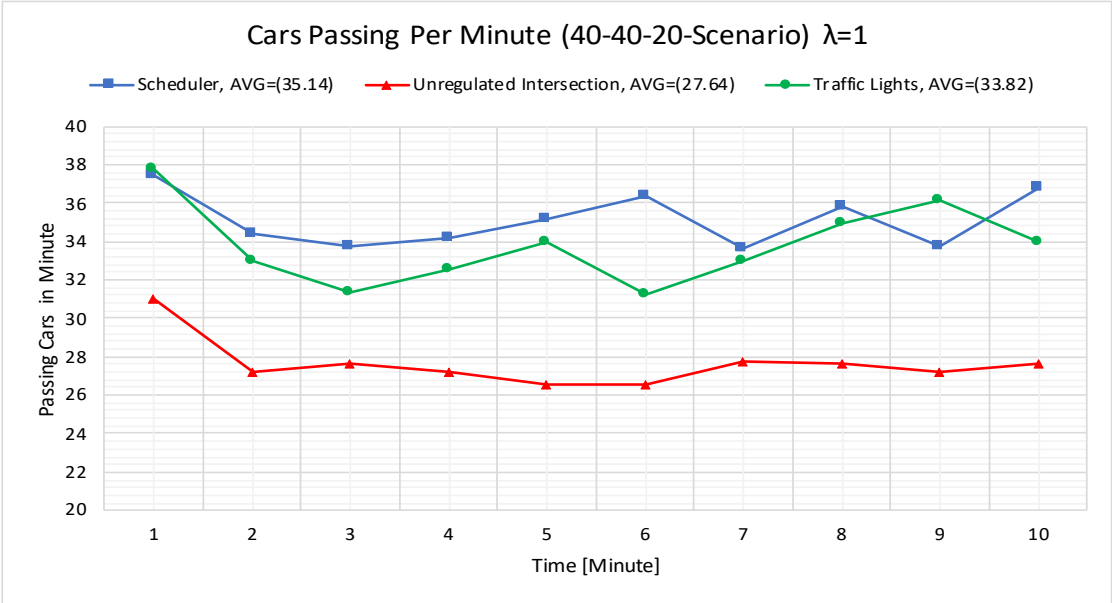

Figure 4.18: Comparison of Number of Passing Cars per Minute of Three Scenarios.

As figure 4.19 shows, the average ratio of stop and goes per car for the scheduler scenario is significantly less than two other methods, causes respectively 29%, and

26% less stop and go ratio per vehicle, which directly leads to less fuel consumption, less air pollution, and less vehicle depreciation.

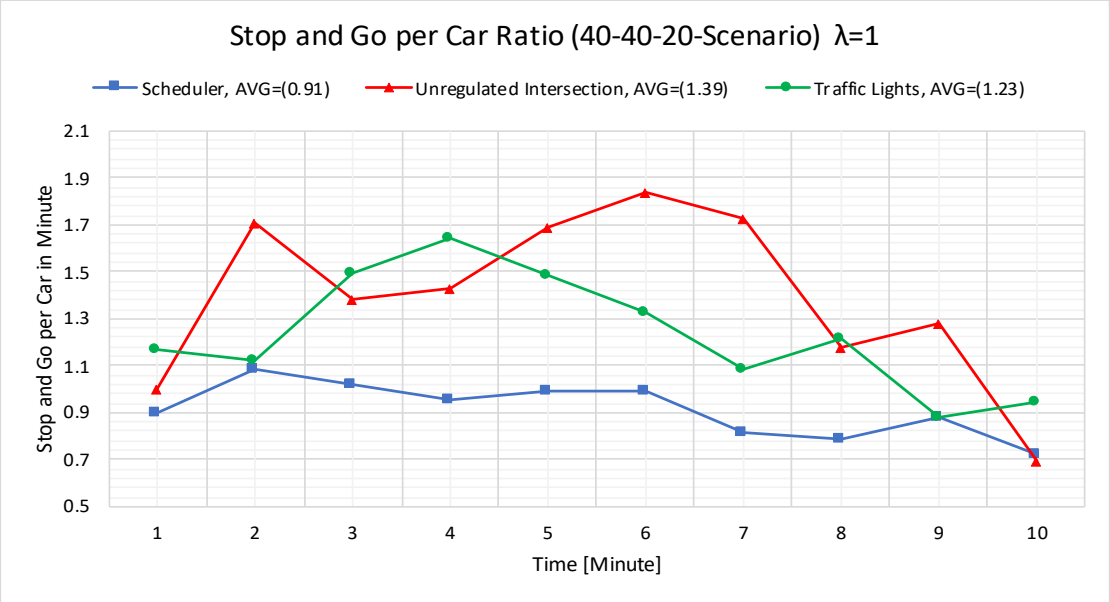

Figure 4.19: Comparison of Stop and Go Per Car Ratio of Three Scenarios.

Results Summery

To summarise the statistics, four tables will be presented; each of them shows the improvement caused by the scheduler scenario in different criteria. Figures 4.20 and 4.21 show the result summaries for fixed generation rate homogeneous scenario.

Homogeneous Scenario		
Scenario	Avg Cars Passing	Improvements
Scheduler	37.96	
Unregulated	26.88	41.2%
Traffic Lights	28.8	31.8%

Figure 4.20: Results Summery; Fixed Rate, Homogeneous (Cars Passing In Minute).

Homogeneous Scenario		
Scenario	Avg Stop and Go per Car	Improvements
Scheduler	1.556	
Unregulated	1.851	16.0%
Traffic Lights	2.21	29.5%

Figure 4.21: Results Summery; Fixed Rate, Homogeneous (Stop and Go per Car In Minute).

Figures 4.22 and 4.23 show the result summaries for fixed generation rate 40-40-20 scenario.

40-40-20 Scenario		
Scenario	Avg Cars Passing	Improvements
Scheduler	35.14	
Unregulated	27.64	27.1%
Traffic Lights	33.82	3.9%

Figure 4.22: Results Summery; Fixed Rate, 40-40-20 (Cars Passing In Minute).

40-40-20 Scenario		
Scenario	Avg Stop and Go per Car	Improvements
Scheduler	0.91	
Unregulated	1.39	34.5%
Traffic Lights	1.23	26.0%

Figure 4.23: Results Summery; Fixed Rate, 40-40-20 (Stop and Go per Car In Minute).

4.4.2 Variable Generation Rate

The key difference between these simulations was the variability of the vehicle generation rate to study how the different scenarios act when the rate of incoming the vehicles into the intersection changes. The assumptions of the scenario are listed below:

- Simulation time: 33 minutes (1980 [s])
- Transient period: 3 minutes
- Number of runs (different seeds): 10
- Lambdas: 3.5 , 3 , 2.5 , 2 , 1.5 , 1
- Data collecting period: Every 1 minute
- Number of samples for each simulation : 10
- Frequency of changing Lambdas: Every 5 minutes

Homogeneous Scenario

For this scenario, the vehicles distributed equally to all directions, and the car generation rate changed every 5 minutes. As figure 4.24 shows, the scheduler starts to outperform after $\lambda = 2$; the difference becomes greater every time the car generation rate increases. Therefore, in high-density traffic situations, the scheduler can perform significantly better than two other scenarios.

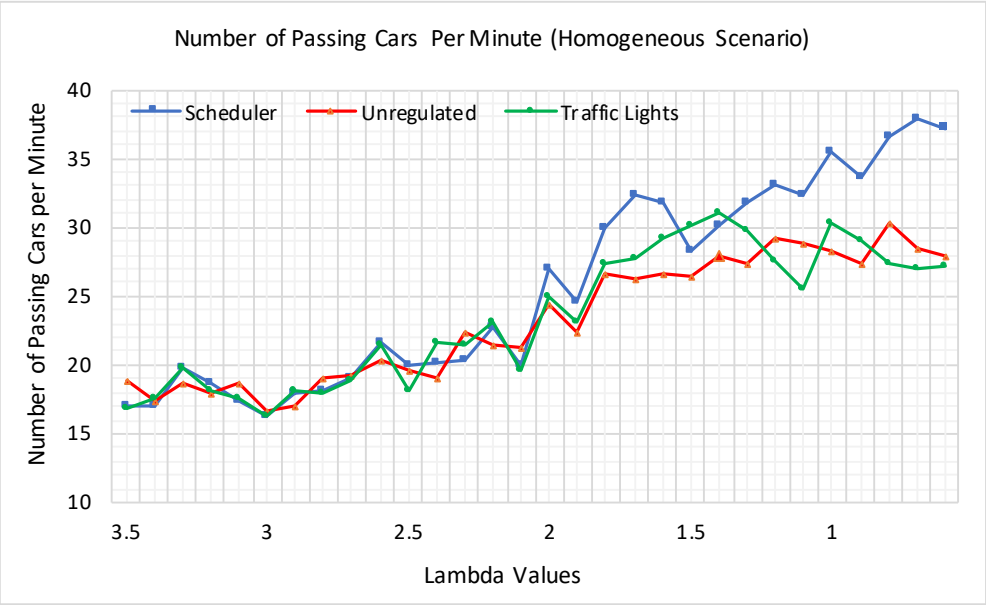

Figure 4.24: Comparison of Cars Passing (Homogeneous and Variable λs).

For the stop and go ratio per car, as figure 4.25 shows, the key value is $\lambda = 2$, which, after that, the scheduler starts to perform better than other methods. Therefore the method performs better in high dense situations. It is because of the scheduler's behavior, which starts to work more frequently whenever there are more vehicles at the intersection (the conditions for triggering the scheduler satisfy with higher probability).

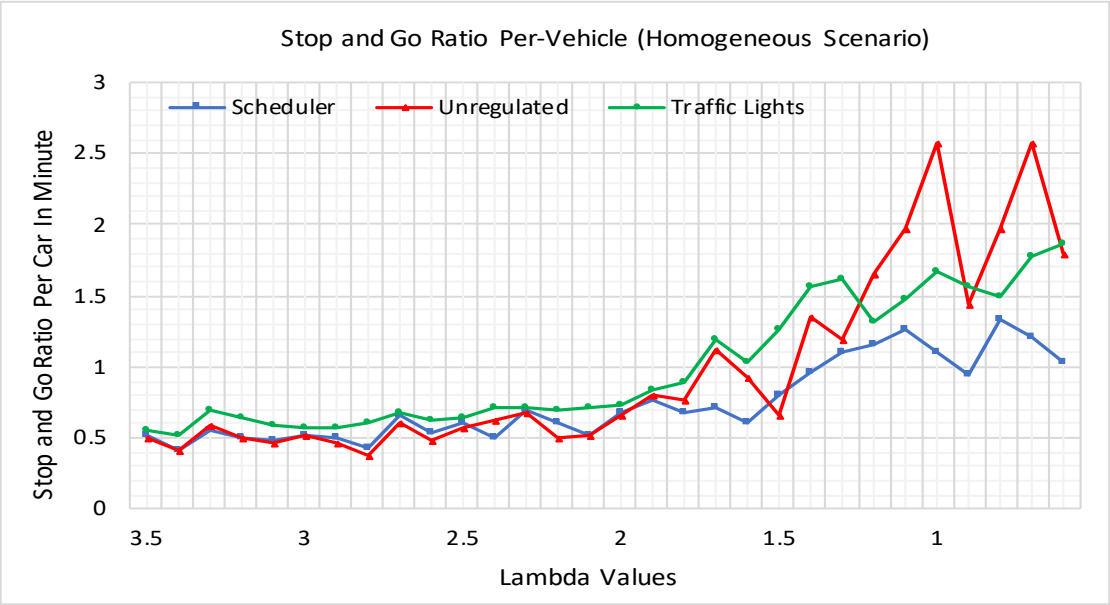

Figure 4.25: Comparison of Stop and Go Per Car Ratios (Homogeneous and Variable λs).

Figures 4.26 and 4.27 show the averages of passing cars and stop and go ratio for variable lambdas; for each lambda, the average value was computed. In these graphs, the advantage of using the scheduler is highlighted. According to the figures, the scheduler performs better than two other methods.

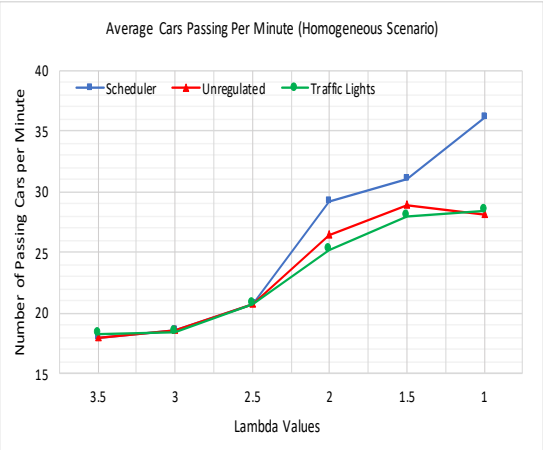

Figure 4.26: Passing Cars Averages, (Homogeneous Scenario).

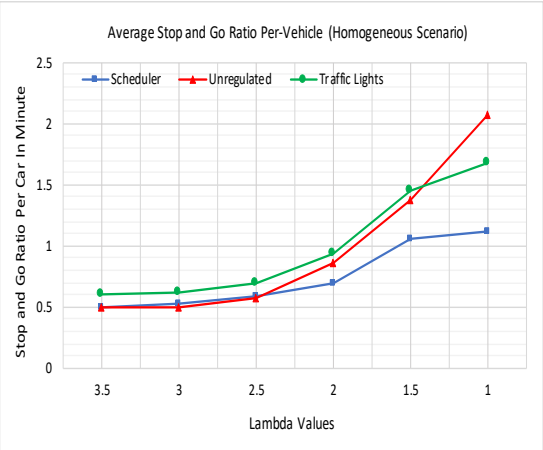

Figure 4.27: Stop and Go per Car Averages, (Homogeneous Scenario).

Therefore, if the incoming traffic rate to the intersection will be more than $\frac{1}{2}\frac{[Vehicle]}{[Second]}$ it is expected that using scheduler can have a significant effect on improving the fuel efficiency by decreasing stop and go ratio per car, also increasing the number of cars passing per minute thus increasing the time efficiency.

Non-Homogeneous Scenario

The conditions for these series of simulations were the same as 4.4.2. The only difference was the distribution of the vehicles, which was 40% from west 40% from the east, and the 20% remaining distributed other two directions equally. Figure 4.28 shows passing cars for variable traffic generation rate.

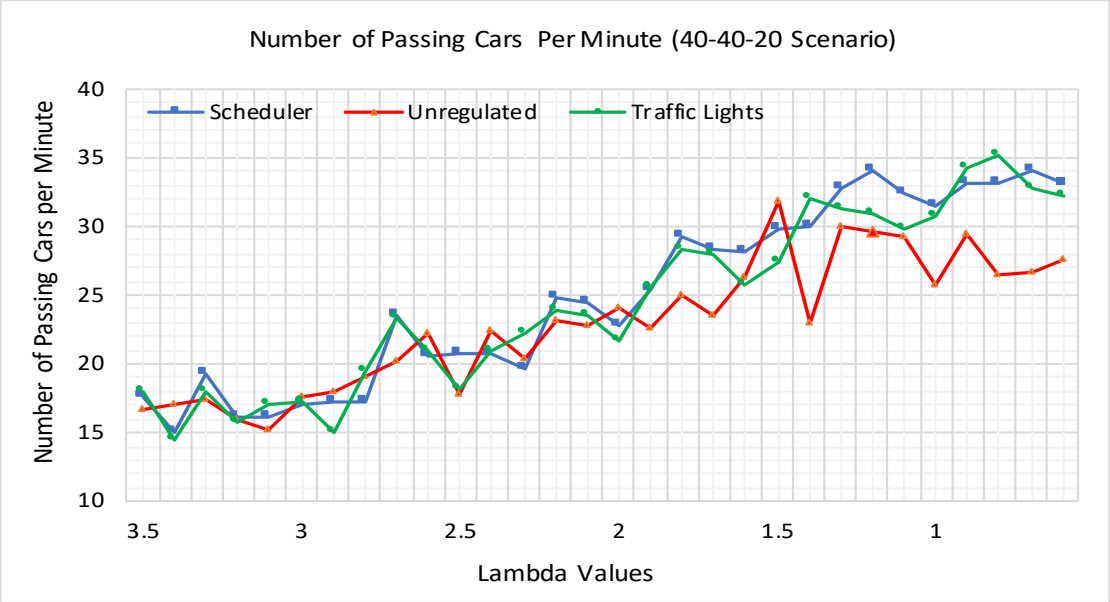

Figure 4.28: Comparison of Passing Cars (Non-Homogeneous and Variable λs).

Figure 4.29 shows the stop and go per car for three scenarios with different generation rates.

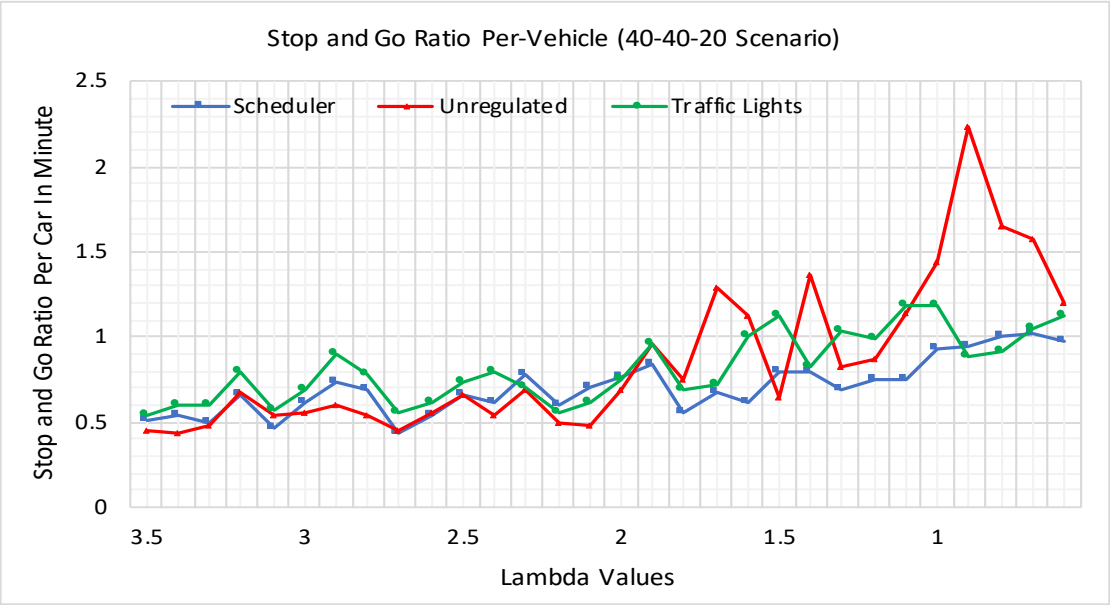

Figure 4.29: Comparison of Stop and Go Per Car Ratios (Non-Homogeneous and Variable λs).

Figures 4.30 and 4.31 show the average cars passing per minute and the average stop and go ratio per car per minute for different λ rates. Obviously, for the non-homogeneous scenario, the algorithm performs slightly better than two other scenarios after rate $\lambda = 2$ in the number of passing cars. Furthermore, in the stop and go ratio per vehicle, the scheduler algorithm outperforms after $\lambda = 2$.

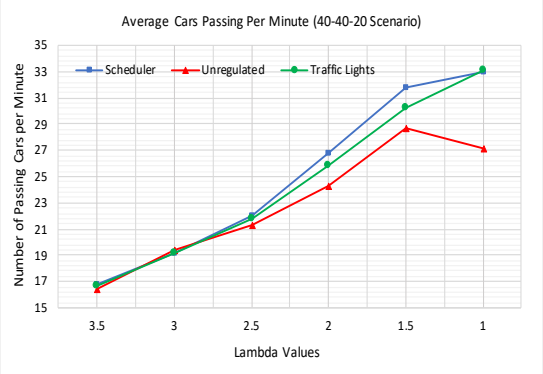

Figure 4.30: Passing Cars Averages (40-40-20 Scenario).

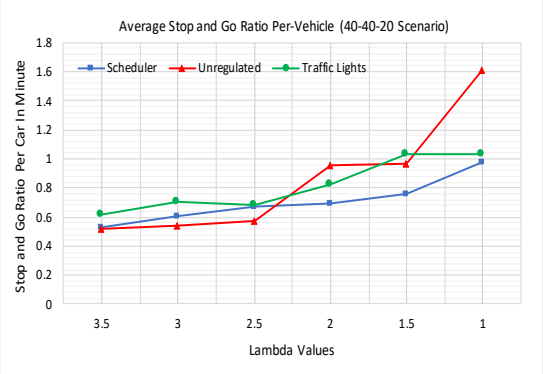

Figure 4.31: Stop and Go Averages (40-40-20 Scenario).

Chapter 5

Conclusion

In this thesis, it was shown that using a new paradigm V2V-based Virtual Traffic Light System that uses an optimized scheduler, rather than using other traditional methods like using traffic light or unregulated intersection, can significantly affect time efficiency and reduce fuel consumption. The proposed algorithm is relatively simple to implement and fast. The project needs very few components to run in the real-world, and more importantly, it is road-side infrastructure-free. Therefore, it is cost-efficient too. The algorithm performs better in higher densities of the vehicles. According to the evidences that has been shown, using the scheduler can improve in parallel two important parameters: (Number of passing cars, and stop and go per car); therefore, it has a double effect on efficiency.

Bibliography

[1] European Commission. *Self-driving cars in the EU: from science fiction to reality*. URL: https://www.europarl.europa.eu/news/en/headlines/economy/20190110STO23102/self-driving-cars-in-the-eu-from-science-fiction-to-reality (cit. on p. 2).

[2] Counter Print. *New Entrants to Autonomous Vehicle Solutions Will Drive the Market in the Coming Decade*. URL: https://www.counterpointresearch.com/new-entrants-autonomous-vehicle-solutions-will-drive-market-coming-decade/ (cit. on p. 3).

[3] Severino.A Fabio.A Pau.G. «A review on IEEE 802.11p intelligent Systems». In: *Sensor and Actuar Networks* 11 (2020) (cit. on p. 5).

[4] DOJANI.KARIM MIAO.LUSHENG YSKANDAR.HAMAM. «A Survay of IEEE 802.11p Mac Protocol». In: *IEEE Cyber Journals* 8 (2011) (cit. on p. 6).

[5] SNIP. *SAE J2735 DSRC Message List*. URL: https://www.use-snip.com/kb/knowledge-base/sae-j2735-dsrc-message-list/ (cit. on p. 7).

[6] Car2Car Communication consortium. *about C-ITS*. URL: https://www.car-2-car.org/about-c-its/ (cit. on p. 10).

[7] Andreas Festag. «Cooperative intelligent transport systems standards in Europe». In: *Communications Magazine, IEEE* 52 (Dec. 2014), pp. 166–172. DOI: 10.1109/MCOM.2014.6979970 (cit. on p. 11).

[8] *What is Omnet++*. URL: https://omnetpp.org/intro/ (cit. on p. 15).

[9] SUMO. *SUMO at a Glance*. URL: https://sumo.dlr.de/docs/SUMO_at_a_Glance.html (cit. on p. 15).

[10] VEINS. *VEINS TUTORIAL*. URL: https://veins.car2x.org/tutorial/ (cit. on p. 16).

[11] Wikipedia. *Bohem's Spiral Model*. URL: https://en.wikipedia.org/wiki/Spiral_model/ (cit. on p. 16).

[12] CASETTI.CLAUDIO ETTORE RAPELLI.MARCO. «A Distributed V2V-Based Virtual Traffic Light System». In: *IEEE* 7 (2020), pp. 122–128 (cit. on p. 20).

[13] F.V WEBSTER. *Traffic Signal Settings*. 1958 (cit. on p. 44).