MULTI AGENT-BASED CONTROL ARCHITECTURE IN INTELLIGENT TRANSPORTATION SYSTEM WITH INFRASTRUCTURE-BASED SENSING

Ismail Habib Muhammad
MULTI AGENT-BASED CONTROL ARCHITECTURE IN INTELLIGENT TRANSPORTATION SYSTEM WITH INFRASTRUCTURE-BASED SENSING

MASTER THESIS

submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

of master program

MEDIA AND KNOWLEDGE ENGINEERING

by

Ismail Habib Muhammad

TU Delft
Department of Media and Knowledge Engineering
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology
Delft, the Netherlands

TNO Science and Industry
Stieltjesweg 1
Delft, the Netherlands
MULTI AGENT-BASED CONTROL ARCHITECTURE IN INTELLIGENT TRANSPORTATION SYSTEM WITH INFRASTRUCTURE-BASED SENSING

Author: Ismail Habib Muhammad
Student id: 1333607
Email: ismailhabib@yahoo.com

ABSTRACT
Traffic congestion has negative impact in our life. In 2001, the European Commission estimated costs of congestion at around 120 billion Euro. The research of Intelligent Transportation System (ITS) becomes interesting since infrastructure expansion in not always feasible. In order to find out the impact of automated vehicle and intelligent infrastructure to traffic performance, we develop a model of hierarchical vehicle controller and intelligent infrastructure and implement them in a simulated environment using Multi Agent System concept. Several traffic scenarios are deployed to observe the effect of the designed vehicle controller and the model of intelligent infrastructure in traffic.

Keyword: ITS, traffic simulation, hierarchical control, intelligent infrastructure

Chair: Prof. Catholijn Jonker, Faculty EEMCS, Delft University of Technology
University Supervisor: Prof. Catholijn Jonker, Faculty EEMCS, Delft University of Technology
Company Supervisor: Dr. Zoltan Papp, TNO Science and Industry, Stieltjesweg 1 Delft, The Netherlands
Bismillahirrahmanirrahim. Thank to Allah SWT, the Almighty God who made all things possible.

The preparation of this thesis document would not have been possible without the support of those around me. I would like to take this opportunity with great pleasure to express my gratitude to them. By having their supports, I had the possibility to complete this thesis.

The thesis topic originally came from TNO. I really appreciate TNO for giving me the chance to conduct my thesis there; it is an honor and great experience working there. I would like to thank for the facilities TNO has provided, including the permission of using their knowledge and tools. Especially, to Zoltan who has become my supervisor in TNO. His experience and expertise are no doubt high quality.

I am deeply indebted to my supervisor, Prof. Dr. Catholijn M. Jonker. In between her busy hours a as a professor of Man-Machine Interaction Group in Delft University of Technology, she regularly spent her time to look over my work. She kept stimulating suggestions and encouragement that have helped me getting through my thesis. She even spent some time in assisting me writing and improving the English style and grammar of the thesis document.

Some friends have also supported me in this tough time. I want to thank them for all their help. I am obliged Baqir for being my partner in many projects and going through difficult times together, Puspa for helping me with report’s stuffs, Anggara who is very generous and helpful, Saputra and Pita for their support and help in finding an apartment, also to my ex-neighbors Mas Zaki and Mbak Selly, and to all my classmates, Indonesian friends, badminton friends, office mates, and people from MMI group.

Even though we are separated in distance, my parents have also given me huge supports. They always encourage and motivate me. They have reminded me that besides the effort I do, do not forget to pray. I am really grateful for all things they have done for me. I also would like to thank my brother who is the closest family I have here in Europe.

It is hard to get through anything without the support of those closest to you. At last but not least, I would like to give my special thanks to my wife Linda who has provided her vital encouragement and support. Things could have been more difficult without her.
# TABLE OF CONTENT

Abstract .......................................................................................................................................................... 5

Acknowledgement........................................................................................................................................ 7

Table of Content ......................................................................................................................................... 9

List of figures ........................................................................................................................................... 13

List of Tables .............................................................................................................................................. 17

List of Equations ....................................................................................................................................... 19

1 Introduction ............................................................................................................................................ 21

1.1 Intelligent Transportation System ................................................................................................. 22

1.2 Objectives and Methods ................................................................................................................... 22

2 Intelligent Infrastructure ......................................................................................................................... 25

2.1 Tisnet Overview ................................................................................................................................. 26

2.2 Process In TISNET ............................................................................................................................. 27

2.3 Modelling TISNET ............................................................................................................................. 28

3 Control Architectures for ITS .................................................................................................................... 33

3.1 Multi-Agent System .......................................................................................................................... 33

3.2 Subsumption Architecture ............................................................................................................... 35

3.3 Belief-Desire-Intention (BDI) ........................................................................................................... 37

3.3.1 A General Model of BDI Agent .................................................................................................... 37

3.4 ITS Control Architecture ................................................................................................................... 39

3.4.1 Physical Layer ............................................................................................................................... 40

3.4.2 Regulation Layer ............................................................................................................................ 40

3.4.3 Coordination Layer ....................................................................................................................... 41

3.4.4 Link Layer ..................................................................................................................................... 41

3.4.5 Network Layer ............................................................................................................................. 41

3.5 Discussion ......................................................................................................................................... 42

4 Controller Implementation ...................................................................................................................... 45

4.1 Motion Primitives ............................................................................................................................... 45

4.1.1 Reference Vehicle .......................................................................................................................... 46
4.1.2 Safe and Adjusting Distance .................................................................................................. 46
4.1.3 Dealing with Imperfect Information .................................................................................. 48
4.1.4 The Longitudinal Controller .............................................................................................. 52
4.1.5 The Lateral Controller ....................................................................................................... 55
4.2 BDI Agent Implementation .................................................................................................. 59
  4.2.1 Flow of Reasoning ............................................................................................................. 61
4.3 Control Functionalities ........................................................................................................ 63
  4.3.1 Scenarios .......................................................................................................................... 63
  4.3.2 Input/Output Parameters in the Control Functionalities layer ........................................... 77
4.4 Cooperation ........................................................................................................................ 78
  4.4.1 Scenarios .......................................................................................................................... 78
  4.4.2 Input/Output Parameters in the Cooperation Layer ............................................................ 85
5 Implementation ..................................................................................................................... 87
  5.1 Multi Agent Real-time Simulator (MARS) ......................................................................... 87
    5.1.1 The Multi-Agent Concept in MARS .............................................................................. 88
    5.1.2 MARS Runtime Framework .......................................................................................... 89
  5.2 Implementing the Models .................................................................................................... 90
  5.3 Defining Scenarios in MARS .............................................................................................. 92
  5.4 The Packages ...................................................................................................................... 92
6 Experiments and Results ....................................................................................................... 95
  6.1 Scenarios .............................................................................................................................. 95
    6.1.1 Sensor ............................................................................................................................ 96
    6.1.2 Controller ...................................................................................................................... 97
    6.1.3 Percentage of Instrumented Vehicle ............................................................................ 97
  6.2 Results ................................................................................................................................ 98
7 Conclusions and Recommendations ..................................................................................... 101
  7.1 Conclusions .......................................................................................................................... 101
  7.2 Recommendations .............................................................................................................. 102
8 References ............................................................................................................................ 105
List of Figures

Figure 1-1 An illustration of traffic congestion ................................................................. 21
Figure 2-1 Congestion Pricing in Singapore ................................................................. 25
Figure 2-2 Overview of TISNET .................................................................................. 26
Figure 2-3 Road Side Units and Their Range ................................................................. 27
Figure 2-4 Process in TISNET ..................................................................................... 27
Figure 2-5 TISNET Model Process ............................................................................... 29
Figure 2-6 Illustration of delay in TISNET ................................................................. 30
Figure 2-7 Filtering TISNET information ..................................................................... 31
Figure 3-1 General Model of an Agent ....................................................................... 33
Figure 3-2 A traditional decomposition of a mobile robot control system into functional modules .... 35
Figure 3-3 A decomposition of a mobile robot control system based on task achieving behaviors ........ 35
Figure 3-4 An example for intelligent vehicle decomposition ....................................... 36
Figure 3-5 Constantini’s General Agent Model .......................................................... 37
Figure 3-6 PATH AHS Framework .............................................................................. 40
Figure 3-7 The proposed ITS discussed in section 3.5 ............................................... 42
Figure 4-1 Controller Architecture ........................................................................... 45
Figure 4-2 Safe and Adjusting Distance .................................................................. 47
Figure 4-3 A vehicle with uncertainty of N meter in position (axis y) ......................... 48
Figure 4-4 Illustration of uncertainty in speed ......................................................... 49
Figure 4-5 Longitudinal Controller Mode .................................................................. 52
Figure 4-6 Condition in lane change ...................................................................... 56
Figure 4-7 Lateral Controller Mode ........................................................................... 56
Figure 4-8 An example situation where Self goes into cancel mode .......................... 58
Figure 4-9 The M Agent ......................................................................................... 60
Figure 4-10 Flow of reasoning .................................................................................. 61
Figure 4-11 Beliefs’ interpretation ............................................................................ 61
Figure 4-12 An example of Find Situations ............................................................... 62
Figure 4-13 An example of opportunity checks .......................................................................................... 63
Figure 4-14 Normal Overtaking – phase 1 ............................................................................................... 64
Figure 4-15 Normal Overtaking – phase 2 ............................................................................................... 65
Figure 4-16 Overtaking with Cooperation – phase 1 ............................................................................. 66
Figure 4-17 Overtaking with Cooperation phase 2 ................................................................................ 66
Figure 4-18 Overtaking with Cooperation – phase 3 ............................................................................. 67
Figure 4-19 Overtaking with Cooperation – phase 4 ............................................................................. 68
Figure 4-20 Accepting Request for Space – phase 1 .............................................................................. 69
Figure 4-21 Accepting Request for Space – phase 2 .............................................................................. 69
Figure 4-22 Normal Merging – phase 1 ................................................................................................ 70
Figure 4-23 Normal Merging – phase 2 ................................................................................................ 71
Figure 4-24 Merging with Cooperation – phase 1 ................................................................................ 72
Figure 4-25 Merging with Cooperation – phase 2 ................................................................................ 72
Figure 4-26 Merging with Cooperation – phase 3 ................................................................................ 73
Figure 4-27 Merging with Cooperation – phase 4 ................................................................................ 74
Figure 4-28 Merging with Cooperation – phase 5 ................................................................................ 75
Figure 4-29 Anticipating a Merging – phase 1 ...................................................................................... 76
Figure 4-30 Anticipating a Merging – phase 2 ...................................................................................... 77
Figure 4-31 Overtaking with Cooperation – phase 1 ............................................................................ 79
Figure 4-32 Overtaking with Cooperation – phase 2 ............................................................................ 80
Figure 4-33 Overtaking with Cooperation – phase 3 ............................................................................ 80
Figure 4-34 Overtaking with Cooperation – phase 1 ............................................................................ 81
Figure 4-35 Overtaking with Cooperation – phase 2 ............................................................................ 81
Figure 4-36 Merging with Cooperation – phase 1 ................................................................................ 82
Figure 4-37 Merging with Cooperation – phase 2 ................................................................................ 83
Figure 4-38 Merging with Cooperation – phase 3 ................................................................................ 83
Figure 4-39 Merging with Cooperation – phase 4 ................................................................................ 83
Figure 4-40 Anticipating a Merging – phase 1 ...................................................................................... 84
Figure 4-41 Anticipating a Merging – phase 2 ...................................................................................... 84
Figure 5-1 The concept of MARS .................................................................................................................. 88
Figure 5-2 The MARS runtime principle of operation [14] ............................................................................. 89
Figure 5-3 Flow of execution in ComposableEntity [11]................................................................................... 91
Figure 6-1 The initial condition of the scenario.................................................................................................... 95
Figure 6-2 The end condition of the scenario...................................................................................................... 95
Figure 6-3 Time to finish scenario vs % of instrumented vehicle in timid profile ....................................... 98
Figure 6-4 Time to finish scenario vs % of instrumented vehicle in aggressive profile ............................... 99
Table 2-1 Information collected by the environment scanner ................................................................. 31
Table 2-2 Selecting the appropriate time stamped information ............................................................ 31
Table 4-1 Situations where Motion Primitives will switch from Normal Mode ...................................... 53
Table 4-2 Situations where Motion Primitives will switch to Normal Mode .......................................... 53
Table 4-3 Situations where Motion Primitives will switch from Adjusting Mode ................................. 53
Table 4-4 Situations where Motion Primitives will switch to Adjusting Mode .................................... 53
Table 4-5 Situations where Motion Primitives will switch from Lock Mode ........................................ 54
Table 4-6 Situations where Motion Primitives will switch to Lock Mode ............................................ 54
Table 4-7 Situations where Motion Primitives will switch from Forced Adjusting Mode .................... 54
Table 4-8 Situations where Motion Primitives will switch to Forced Adjusting Mode ....................... 55
Table 4-9 Situations where Motion Primitives will switch from Forced Lock Mode ............................ 55
Table 4-10 Situations where Motion Primitives will switch to Forced Lock Mode .............................. 55
Table 4-11 Situations where Lateral Controller will switch from Off Mode ........................................ 57
Table 4-12 Situations where Lateral Controller will switch from Off Mode ........................................ 57
Table 4-13 Situations where Lateral Controller will switch from On Mode ........................................ 57
Table 4-14 Situations where Lateral Controller will switch to Off Mode ............................................ 57
Table 4-15 Situation where Lateral Controller will switch from Cancel Mode .................................... 58
Table 4-16 Situation where Lateral Controller will switch to Cancel Mode ....................................... 58
Table 4-17 Input parameters from the Physical layer ............................................................................ 58
Table 4-18 Output parameters to the Physical layer ............................................................................. 58
Table 4-19 Input parameters from the Control Functionalities layer .................................................... 59
Table 4-20 Output parameters to the Control Functionalities layer ..................................................... 59
Table 4-21 Normal Overtaking – rules 1 ................................................................................................. 64
Table 4-22 Normal Overtaking – rules 2 ................................................................................................. 64
Table 4-23 Overtaking with Cooperation – rules 1 ............................................................................... 65
Table 4-24 Overtaking with Cooperation – rules 2 ............................................................................... 66
Table 4-25 Overtaking with Cooperation – rules 3 ............................................................................... 67
LIST OF EQUATIONS

Equation 4-1 Safe distance formula........................................................................................................................................... 47
Equation 4-2 Adjusting distance formula ........................................................................................................................................... 47
Equation 4-3 Adjusting time formula ............................................................................................................................................... 48
Equation 4-4 Safe distance, adjusting distance, and adjusting time formula................................................................................... 49
Equation 4-5 Distance between two vehicles................................................................................................................................ 50
Equation 4-6 Distance formula with uncertainty .................................................................................................................................. 50
Equation 4-7 Vehicle predicted position for the current time ........................................................................................................... 51
Equation 4-8 Uncertainty values for speed and position ....................................................................................................................... 51
Equation 5-1 System state and the output in discrete state space model............................................................................................. 91
1 INTRODUCTION

The ability to move about our neighborhood, city, and between cities has been taken for granted for years. Transportation mobility affects the ability to do our jobs, quality of life, and the economic productivity. Increasingly, however, our mobility is jeopardized by congestion [1]. Traffic congestion increase stress and aggression in driving behavior [1] and might as well have negative impact on safety. Economically, in 2001 the European Commission estimated the total costs of road congestion at around 120 billion Euro [2]. For millions of people, congestion is a part of daily life.

Figure 1-1 An illustration of traffic congestion

Traffic congestion is a condition where a road network increasingly used and characterized by slower speeds, longer trip times, and increased queuing. When traffic demand is great enough that the interaction between vehicles slows the speed of the traffic stream, congestion is incurred. As demand approaches the capacity of a road, extreme congestion sets in and creates traffic jam. Traffic demands vary significantly depending of season of the year, the day of the week, and even the time of the day [1]. The capacity is also often mistaken as constant, while it actually depends on the weather, work zones, or traffic incidents [1].

The most obvious way to overcome traffic congestion is by expanding the road infrastructure to increase the capacity of traffic. However, infrastructure expansion is not always feasible, given the space limitation, cost of the infrastructure, and public opposition to new construction. Therefore, another approach is required.
1.1 INTELLIGENT TRANSPORTATION SYSTEM

The Intelligent Transportation Systems (ITS) program is a worldwide effort to add information and communications technology to transport infrastructure and vehicles. It aims to manage factors such as vehicles, loads, and routes to improve safety and reduce vehicle wear, transportation times and fuel consumption. The advances in sensing technologies, computer hardware and software, communication technologies, and control theory also increase the feasibility of ITS development.

One of the most exciting part of ITS is automation. The idea is to improve highway traffic flow and safety by applying intelligent and automation in driving. Autonomous control of driving tasks was considered to substantially improve the traffic flow [2].

There is a concept of group of vehicles arranged in a relatively small-fixed distance called “platoon”. Platooning is possible when full automation applied in every vehicle. It allows smooth merging, lane changing and splitting maneuvers in a way that is advantageous to the highway performance [2]. In order to perform a safe platooning, coordination between vehicles is a necessity. Coordination is possible with communication, which is why a communication technology is also required.

Since platooning enables vehicles to operate much closer together than is possible under manual driving conditions, each lane can carry at least twice as much as it can today. This should make it possible to greatly reduce highway congestion. Also, at close spacing aerodynamic drag is significantly reduced which can lead to major reductions in fuel consumption and exhaust emissions. The high-performance vehicle control system also increases the safety of highway travel, reduces driving stress and tedium, and provides a very smooth ride [3].

The control components of ITS are Intelligent Infrastructure and Intelligent Vehicle (IV). IV is considered as a new technology for obtaining a more efficient driver-vehicle operation. It is meant to improve safety, operational efficiency, and convenience while driving. An IV system senses the environment around the vehicle by using sensors and strives to achieve more efficient vehicle operation by assisting the driver or by taking full control of the vehicle. The IV application areas could be divided into three categories depending on the level of support to the provider [2]:

1. Advisory systems. These systems are designed to increase the awareness of drivers about the prevailing traffic situation using route navigation or potential danger warning. No control function offered by this system.
2. Partial control systems. These systems take partial control of vehicle maneuvers such as steering and braking. The driver is still in charge of the vehicle even if assistance is provided. An example of partial control system is adaptive cruise control.
3. Fully autonomous systems. These systems remove human driver intervention from the control and therefore take complete control of vehicle operations.

1.2 OBJECTIVES AND METHODS

The main research question of this thesis is to evaluate the impact of intelligent vehicle and intelligent infrastructure in traffic performance. The evaluation is conducted in a simulated environment system. This leads to two other objectives:
1. To create an evaluation and test environment for Intelligent Transportation System (ITS) including parameterized infrastructure-based sensor model.
2. To present a particular vehicle control architecture beneficial to the infrastructure-based sensor.

For the intelligent infrastructure, a system developed by TNO is used. For vehicle control, a literature review is conducted and a design of control architecture will be proposed.
The notion of an intelligent infrastructure plays an important part in Intelligent Transportation System (ITS). Technology for intelligent infrastructure already exists, is evolving, and being used. Inductive loop detection, video vehicle detection, radio communication, barcode stickers, license plate recognition, and infrared communication are important technologies that are currently applied in the form of electronic toll collection, congestion pricing and automatic traffic rules as explained below.

1. Electronic toll collection
   Electronic toll collection allows vehicles to drive through toll gates at a normal speed, thus, reduce the potential congestion at the toll gates. Technologies that have been used for electronic toll collection are radio communication, barcode stickers, license plate recognition, and infrared communication, either separately or in combination.

2. Congestion pricing
   The technology for congestion pricing is basically similar to the electronic toll collection. A fee is charged automatically when a vehicle enters a congested city center. This system has been implemented in Singapore, Stockholm, and London.

3. Automatic traffic rules
   Cameras or other vehicle-monitoring devices can be used for automatic road enforcement such as: dynamic speed advice, ramp metering, red light cameras, and bus lane.

This chapter will discuss an Intelligent Infrastructure developed by TNO called TISNET. The first two sections explain about TISNET and the last section discusses the proposed model of TISNET to be implemented in simulation.
2.1 TISNET OVERVIEW

TISNET (Traffic Infrastructure Sensor Network) is a traffic measurement technology developed by TNO Science and Industry. The idea of TISNET is to make information retrieval possible by applying object tracking with the help of Wireless Sensor Network (WSN) technology. While existing intelligent infrastructure still only provides advises for drivers, TISNET has the potential to become the technology to support full-automation in traffic.

There are three components in the TISNET system: Sensors, Road Side Units (RSUs), and RSU Communication Link. Motes in TISNET are distributed in the infrastructure (road). Each mote is equipped with a magneto-resistive sensor that is used to measure the changes in the Earth magnetic field caused by the presence of vehicle [4]. They collect data in form of binary detection signals from the road using its sensor and send them to the associated Road Side Unit (RSU) using wireless communication in one-way fashion. Each RSU is responsible for covering a predefined region of the road, i.e. it receives the detection signal from those units [4]. The RSUs are also responsible to process the information from sensors and broadcast the processing results via wireless link. While processing the information, a RSU is able to communicate with another RSU by using RSU Communication Link.

The architecture of TISNET described has a number of advantageous implementation features [4]:

1. The motes are extremely simple, consequently low cost, power efficient implementation is feasible.
2. The motes use one-way communication link to the associated RSU. It allows simple wireless communication design which further decreases complexity and costs.

3. The one-way communication simplifies the communication protocol design because the data interpretation is resilient to message loss to great extent no attempt is made to avoid message collision and there is no recovery mechanism used.

4. The event driven data interpretation scheme is economical both with respect to power consumption and message loss; consequently it enables higher detection density and accuracy at the same power consumption level.

![Figure 2-3 Road Side Units and Their Range](image)

2.2 PROCESS IN TISNET

![Figure 2-4 Process in TISNET](image)
This sub-chapter will explain the general process occurred in TISNET in order to generate information required by vehicle in form of Real-Time Map (RTM). RTM is a collection of vehicle states which contain information such as vehicle’s position and speed.

The whole process could be divided into five phases where the first two are occurred in motes and the rest are conducted by RSUs. The first phase of the process is Sensor Reading. A magneto-resistive sensor in a mote captures the changes in the Earth magnetic field caused by the presence of vehicle [4]. Using Object Detection, this raw data is transformed into a stream of binary detection signals. The changes in this signal are the detection events. This is the information that being sent to Road Side Units using wireless communication.

RSU collects the information from motes that exist within its range. This process is called Data Acquisition. Since the RSU has the knowledge about the placement of sensors, it can use the information gathered from motes and estimate the object motion states (position, speed, orientation) by applying State Estimation. Therefore, the object states are known to the RSU. The next phase of the process is Map Generation. By combining the collection of object states from its own and other RSUs, the RSU could generate a map of environment of a certain size. Finally, this map is made available to instrumented vehicles by broadcasting them wirelessly.

2.3 MODELLING TISNET

This section explains the simulation model of TISNET that is used in the demonstrator of ITS, as described in chapter 7. The simulation implements a parameterized TISNET model that does not simulate the whole process as explained in the previous sections, but instead it will take a different approach to generate a result similar to the real TISNET. The term “Real TISNET” and “Simulated TISNET” are used to separate between them.

In the real TISNET every RSU is acquiring the information to create situation awareness. For the real TISNET the acquisition of information is the main task. In simulated TISNET model, gathering information is easy since the environment is a simulation. Therefore, the focus in implementing the simulated TISNET is on the transformation of the simulation environment information into an appropriate model of the situation.

Some important characteristics of TISNET that need to be implemented in the simulation are: range, delay and jitter, message loss, and uncertainty.

1. Range

In TISNET RSU, there are two types of range: communication range and sensing range. The communication range defines the area where an RSU could send its message to. The sensing range is the area where an RSU is responsible to collect data from. For simplicity reason in this model the range is defined as a rectangular area.

2. Delay and Jitter

Data transmission requires time, which causes delay in processing. Jitter exists because of the deviation in transmission time. There are three kinds of delay and jitter in term of source and destination, which are:
a. Sensor to RSU transmission  
b. RSU to RSU transmission  
c. RSU to Vehicle transmission  

Sensor to RSU and RSU to Vehicle transmission use wireless communication while RSU to RSU transmission are conducted using wired RSU Communication Link. Therefore, it is expected to have much higher value of delay and jitter at Sensor to RSU and RSU to Vehicle transmission compared to RSU to RSU transmission. For this reason, RSU to RSU transmission delay and jitter will be considered as insignificant and would not be mentioned in further discussion.

3. Message Loss  
In data transmission, especially in wireless one, there is no guarantee that the transmission will always be successful and TISNET is not an exception.

4. Uncertainty  
The motion states cannot be reconstructed without error since it is originated from partial observation.

Using the information expressed in terms of the concepts above the TISNET simulation creates a map of the immediate environment of each RSU. We proposed a model of TISNET with the characteristics as stated above as follows. The simulated TISNET model is divided into two entities: An environment scanner and a representation for Road Side Unit (RSU). The environment scanner takes information directly from the simulated environment and sends it to RSUs for further processing. More details of the process are described below.

The processes that occur in the simulated TISNET model are:

1. Environment scanning  
2. Applying Sensor to RSU delay and jitter  
3. Filtering  
4. Uncertainty addition  
5. Applying RSU to vehicle delay and jitter  
6. Information broadcasting

![TISNET Model Process](image)

The first process in the TISNET model is **Environment Scanning**. The task of environment scanning is to capture the whole information about the environment, attach a time stamp and put it into memory storage. As the result, a collection of information about the environment in the present and in the past is available. This information includes time stamp and states of object (position, speed, etc).
The next step is to transmit the information to RSUs. The information does not contain the whole environment but limited to a certain area of interest as defined as **Range** in each RSUs. The information transmitted to RSUs might not be the most recent but it is the most appropriate with respect to the **Delay and Jitter** information. To fully understand these, a situation of traffic environment with real TISNET is illustrated below.

A traffic equipped with TISNET has 3 instrumented vehicles cruising on it. The TISNET consist of one Road Side Unit and three sensors (motes). The reddish part of road shows the range of Road Side Unit. Current time is 00:03 and the system started at 00:00. The total time required to transmit information from sensors to RSU is one second. Therefore the RSU at 00:03 is receiving the information with timestamp 00:02 from the sensors which have been sent one second earlier.

---

**Figure 2-6 Illustration of delay in TISNET**
In the simulated TISNET we want to appropriately reflect this delay. In a situation as described above, after three seconds since the system starts (timestamp 00:03), the environment scanner would have collected information as follow.

In the simulated TISNET we want to appropriately reflect this delay. In a situation as described above, after three seconds since the system starts (timestamp 00:03), the environment scanner would have collected information as follow.

<table>
<thead>
<tr>
<th>Time</th>
<th>Vehicle 1</th>
<th>Vehicle 2</th>
<th>Vehicle 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:00</td>
<td>In range</td>
<td>Not in range</td>
<td>Not in range</td>
</tr>
<tr>
<td>00:01</td>
<td>In range</td>
<td>In range</td>
<td>Not in range</td>
</tr>
<tr>
<td>00:02</td>
<td>Not in range</td>
<td>In range</td>
<td>In range</td>
</tr>
<tr>
<td>00:03</td>
<td>Not in range</td>
<td>Not in range</td>
<td>In range</td>
</tr>
</tbody>
</table>

Since current time is 00:03 and the total of delay and jitter is 1 second, the simulated TISNET will provide the RSU with the information with time stamp 00:02. In case the exact information is not available due to the discrete nature of the environment scanner (i.e., the current time is 00:03 and the total delay and jitter is 1,5 second and the scanner captures information every one second), the closest older information available will be chosen, i.e., the information with time stamp 00:01.

<table>
<thead>
<tr>
<th>Time</th>
<th>Vehicle 1</th>
<th>Vehicle 2</th>
<th>Vehicle 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:00</td>
<td>In range</td>
<td>Not in range</td>
<td>Not in range</td>
</tr>
<tr>
<td>00:01</td>
<td>In range</td>
<td>In range</td>
<td>Not in range</td>
</tr>
<tr>
<td>00:02</td>
<td>Not in range</td>
<td>In range</td>
<td>In range</td>
</tr>
<tr>
<td>00:03</td>
<td>Not in range</td>
<td>Not in range</td>
<td>In range</td>
</tr>
</tbody>
</table>

Before transmitting information from environment scanner to RSUs, a filtering should also be applied. The range of RSUs is used to filter the information about the environment. Any objects that are outside of the range of the RSU will be filtered out.

The next process is conducted separately for each RSU. So far the information received by RSUs is in perfect condition. To make it closer to the real TISNET, we need to introduce uncertainty. The process uncertainty addition is responsible for this task. Even though the uncertainty addition in the simulated

---

MULTI AGENT-BASED CONTROL ARCHITECTURE IN ITS WITH INFRASTRUCTURE-BASED SENSING
TISNET model is designed to be easily customizable, in this report we limit the form of uncertainty as a random noise addition to the states of vehicle. So, neither will existing objects disappear nor will non-existent objects appear in the information.

The next step of delay and jitter addition is similar to the first delay and jitter. Only in this case it emulates the delay and jitter that would occur in the transmission of information from RSU to Vehicle. The rest works exactly the same as the transmission process from environment scanner to RSU without any filtering process.

The last step is **broadcasting** the information held by RSUs to the vehicles in range. In the simulated TISNET model, the broadcasting is implemented by giving the vehicles access to the information stored in RSUs. A special procedure simulates message loss by determining whether or not a message will be lost. The procedure makes a weighed random decision. The weight parameter of the simulated TISNET models the percentage of message loss in the real TISNET. According to the decision, the message will be received by the vehicle or nothing will happen. In case of a message loss, the vehicle will not receive any information at that moment.
3 CONTROL ARCHITECTURES FOR ITS

This chapter discusses the existing literatures about controls. In ITS, the environment is distributed, therefore a Multi-Agent System is an obvious choice for it, the first section of this chapter gives general idea about Multi-Agent System. The next section introduces the subsumption architecture which is generally designed for a mobile robot, but might also be useful in ITS. In section 3.3, a reasoning system is introduced in form of Belief Desire Intention. Section 3.4 discusses one example of existing ITS Control Architecture from University of Berkeley. The last section of this chapter discusses the usability of the possible architecture to be used in this project.

3.1 MULTI-AGENT SYSTEM

A distributed paradigm is the most suited approach to implements an ITS. As it is naturally distributed (spatially) and consist of various components, which may range from several to thousands number of components (scalability issue), it is also relatively complex. Not to mention that by distributing the implementation accordingly, a certain level of reliability could be achieved more easily. Multi agent systems are emerging as a new way to design and analyze complex distributed systems [2].

An agent is an entity that can perceive its environment through sensors and act upon its environment through actuators in such a way that the performance measures are possibly met [4]. Agent orientation is emerging as a powerful new paradigm for computing. It offers a higher-level abstraction than the earlier paradigm of object orientation [5]. Most of the time a single agent would not be sufficient, that is why a system composed by two or more agents (Multi-agent System) having their own role is required.

The study of multi agent systems (MAS) focuses on systems in which many intelligent agents interact with each other. The agents are considered to be autonomous entities, such as robots. Their interactions can be either cooperative or selfish. That is, the agents can share a common goal (e.g. an ant colony), or they can pursue their own interests [6].

The characteristics of MASs are [6]:

1. Each agent has incomplete information or capabilities for solving the problem and, thus, has a limited viewpoint;
2. There is no system global control;
3. Data are decentralized; and
4. Computation is asynchronous.

A general representation of an agent and its interaction with an environment is as follow:

![Figure 3-1 General Model of an Agent](image)

Agents are acting upon an environment in which they are situated. Based on the way that an agent makes decisions on the actions on the environment, the agent can be classified as follows [4]:
1. Reactive agent. An agent that based its actions on the current perceptions of the current environment. It doesn’t take history into account or plan for the future [6].
2. Deliberate agent. An agent that choose its action according to the sequence of environment states. It is able to predict the outcome of some actions based on the experience of the previous actions.
3. Goal-based agent. An agent that search for available actions and predicts the effect of its actions and finally chooses an action that optimizes its goals (state of the environment).
4. Utility-based agent. Instead of using a goal, utility-based agent uses a utility function. The utility function maps a sequence of states onto numbers that define the preference of the state.

An intelligent vehicle might be a composite of both reactive and goal-based agent. It should be reactive because it needs to react accordingly to the current situation, for example in an accident avoidance situation. Intelligent vehicle is also a goal-based agent. It has to have some goals and preferences and make plans in order to achieve it.

The environment that might arise in the field of agent technology can be categorized along several dimensions [4].

1. Fully observable vs partially observable

An observable environment is an environment where it is possible for the agent’s sensors to determine all the information that are relevant to make a choice for its action. On the other hand, an environment is said to be partially observable if the sensor of agent is not capable in retrieving all information needed due to noises, sensor inaccuracy, or due to the fact that some parts of environment is inaccessible at some states. The typical environment for ITS is partially observable.

2. Deterministic vs stochastic

Deterministic environment is where the next state of environment is completely determined by the current state and the action executed; otherwise, it is stochastic. No uncertainty occurred in a fully-observable, deterministic environment. An ITS environment might be classified as deterministic unless failures in control are also take into account.

3. Static vs Dynamic

If the environment can change in the situation where the agent does nothing to change it then we can say that the environment is dynamic. Traffic environment in this case is obviously dynamic.

4. Discrete vs Continuous

The discrete/continuous distinction can be applied to the state of the environment, to the way time is handled, and the percepts and actions of the agent. For example: chess game is a discrete environment because it has a finite number of states, percepts, and actions. On the other hand, taxi driving is a continuous environment since it has continuous state and continuous time-problem. However, in a simulated system everything is in discrete.
3.2 SUBSUMPTION ARCHITECTURE

While Multi-agent System concept is a good start for an implementation of distributed control, it says nothing about the architecture of the whole system. This chapter will discuss a relatively generic layered control system for a mobile robot as proposed by Brooks [7].

Brooks explains that layers of control system are built to let the robot operate at increasing level of competence. Layers are made up of asynchronous modules which communicate over low bandwidth channels where each module is an instance of a fairly simple computational machine. Higher layer levers can subsume the roles of lower levels by suppressing their outputs. Lower levels continue to function as higher levels are added.

![Figure 3-2 A traditional decomposition of a mobile robot control system into functional modules](image)

![Figure 3-3 A decomposition of a mobile robot control system based on task achieving behaviors](image)

The key idea of levels of competence is that we can build layers of a control system corresponding to each level of competence and simply add new layer to an existing set to move to the next higher level of overall competence.
It is started by building a complete robot control system which achieves level 0 competences (the lowest layer or zeroth layer). This layer will never be altered. Another control layer called the first layer is built on top of the zeroth layer. It is able to examine data from the level 0 system and is also permitted to inject data into the internal interfaces of level 0 suppressing the normal data flow. This layer, with the aid of the zeroth, achieves level 1 competence. The zeroth layer continues to run unaware of the layer above it. This architecture is called a subsumption architecture.

This architecture naturally lends itself to solving the problems for mobile robots such as [7]:

1. Multiple Goals. Individual layers can be working on individual goals. The suppression mechanism then mediates the actions that are taken. The results of pursuing all of them to some level of conclusion can be used for the ultimate decision.
2. Multiple Sensors. There is no sensor fusion problem. Not all sensors need to feed into a central representation, only those which perception processing identifies as extremely reliable might be eligible to enter such a central representation. At the same time the sensor values may still be used by the robot.
3. Robustness. Multiple sensors clearly add to the robustness of a system when their results can be used intelligently. The other thing is that subsumption architecture itself is robust. Lower levels which have been well debugged continue to run when higher levels are added. Since a higher level can only suppress the outputs of lower levels by actively interfering with replacement data, in the cases that it cannot produce results in a timely fashion the lower levels will still produce results which are sensible (at a lower level of competence).
4. Extendibility. An obvious way to handle extendibility is to make each layer run on its own processor. This is practical as there are in general fairly low bandwidth requirements on communication channels between layers.

![Diagram of vehicle decomposition](image)

**Figure 3-4 An example for intelligent vehicle decomposition**

An example for system decomposition for intelligent vehicle as we propose is shown at Figure 3-4. The zeroth layer provides low intelligence. The nature of multiple goals is also shown in those layers. The navigation layers’ goal is to arrive at a certain position in the world, which is also the ultimate goal of a vehicle. Overtake layer might have a goal to always cruise the vehicle on its expected speed, that is why it always want to overtake a lower vehicle.
Another reason why this architecture is suitable for intelligent vehicle is that because it allows high level of intelligence while keeping the safety in real-time possible. While a higher layer works in a low frequency, lower layer works in much higher frequency. The outcomes from higher layer processing is not always necessary, because if there is no output from a higher layer then the vehicle can use the results from lower layer.

3.3 BELIEF-DESIRE-INTENTION (BDI)

In an intelligent system, including Intelligent Transportation System, an agent is expected to have a rational behavior. Beliefs, intentions, and commitments play a crucial role in determining how rational agents will act. Shoham defines an agent to be “an entity whose state is viewed as consisting of mental components such as beliefs, capabilities, choices, and commitments”.

The characteristic of intelligent vehicle naturally fit with the purpose and design of the BDI model as stated by Wooldridge [8]. Those characteristics are:

1. Situated. They are embedded in their environment.
2. Goal directed. They have goals that they try to achieve.
3. Reactive. They react to changes in their environment.
4. Social. They can communicate with other agents.

3.3.1 A GENERAL MODEL OF BDI AGENT

This section explains a general model of BDI agent as proposed by Constantini [10] and to discusses the potential usage for the implementation of the controller of the vehicle.

The objective of the general model by Constantini is to create a general agent model that is flexible and adaptive. The architecture of the proposed model is a layered representation with a base layer, consisting of basic agent features including control, and a higher layer of a meta-control with the task of tuning, supervising, and modifying the base layer.

![Constantini’s General Agent Model](Figure 3-5 Constantini’s General Agent Model)
3.3.1.1 BASE LAYER

Constantini mentions several important aspects that are fundamental of the agent-oriented paradigm. An agent model contains some of these aspects. These aspects are [10]:

1. A set of beliefs, possibly divided into various modules/theories, encompassing (some of) the following activities: reasoning, planning, proactivity, reaction, constraint solving, goal identification, preferences, history, and communication management.
2. A set of desires which are goals that have been adopted or goals that is under consideration and intentions which are plans in execution and plans under analysis.
3. A set of constraints, including temporal constraints that either induce or verify a partial order among actions in intentions and goals.
4. A set of mechanisms for interacting with the environment including: a sensing mechanism/device, an actuating mechanism/device, a mechanism/device supporting communication with other agents.
5. A set of mechanisms for managing beliefs including: a learning mechanism, a belief revision mechanism.
6. A control component for combining, exploiting, and supervising the above components, based on control information aimed at improving the control mechanism effectiveness.

The beliefs define what the agent knows and assumes to be true. Beliefs consist of sets of facts obtained in various ways, typically from an observation, rules and/or procedural knowledge that describes factual knowledge and/or expresses preferences or biases about which module to use and when.

The desires and intentions define what the agent is doing or is intending to do during its operation. The execution of intentions influences the beliefs directly and indirectly:

1. Direct influence: the execution of an intention modifies the beliefs using the mechanisms for managing beliefs (belief revision).
2. Indirect influence: the execution of an intention modifies the world. Since the world is changed, the observations of the agent will result in different beliefs.

Concerning rules or sets of rules or other form of “procedural” knowledge, if they fail to meet the agent’s expectation and the agent has learned something new; existing rules can be deactivated or even removed, and new rules can be incorporated. The evaluation whether the available knowledge is able to meet the expectation of the agent is typically a task of the Meta-control layer.

The control component is responsible for deploying the other components of the agent model with the help of control information. The control information may be provided during the design of the agent, e.g., what preconditions are necessary for different control functionalities. Other control information can be provided by the control itself, e.g., whether an execution is successful or not, which state of the execution is in.

3.3.1.2 META-CONTROL

Beyond the basic control there is a meta-control, where non-trivial supervising and tuning tasks (including agent reconfiguration) can be performed, based on suitable control and meta-control information. There are some potential higher-level modifications that can be made by the meta-control [10]:
1. The control component can be chosen from several alternatives. It means that the control component presently at work can be replaced by another one based on specific needs at certain stages of the agent life.
2. The components defining mechanisms for managing beliefs can be tuned or replaced by others.
3. Intentions can be filtered according to a posteriori preferences based on how the plans are going.
4. High-level aspects of the agent behavior can be monitored, e.g., how many goals are pursued, or how often a certain goal or action succeeds.
5. In the beliefs, non-trivial modifications can also be made, for instance for replacing/adding/dropping knowledge modules that were already available to the agent, but had not been chosen in its first instantiations.

The general agent model defined above has the potential to be used by the controller part of ITS implementation. The base layer defines the basic agent features and obviously something that everyone would look at when developing an agent. The meta-control layer offers an enhancement for the base layer in terms of supervising and tuning of the base layer. This is useful to implement a feature such as ability to learn. However, this is not the focus of our current research. Thus, we consider implementing the agent following the proposed base layer while the addition of meta-control layer is a possible choice for future development.

### 3.4 ITS CONTROL ARCHITECTURE

While the Subsumption Architecture is quite general, this chapter will discuss a more specific control framework that have been developed for ITS, to be more specified: a well-known architecture called PATH Automated Highway System that was proposed by the University of Berkeley. There are also several other framework such as Dolphin framework and Auto21 CDS framework which is pretty much similar or inspired by PATH framework.

The Automated Highway System (AHS) framework developed by PATH coordinates both roadside-vehicle and inter-vehicle activities. This framework assumes that the traffic is organized as “platoons” consisting of closely spaced vehicles, and also suggests that by platooning, the traffic flow on the highway increases [2]. To manage the control problem of a fully automated highway system with platoons is a cumbersome task. Hence a hierarchical, hybrid control approach has been proposed and implemented in this architecture. This control hierarchy breaks the total control of automated highway system into five self-organized functional layers [2]:

1. Physical Layer
2. Regulation Layer
3. Coordination Layer
4. Link Layer
5. Network Layer

This framework assumes that a network is made of many interconnected highways.
### 3.4.1 Physical Layer

The physical layer of each vehicle is the bottom layer of the hierarchy and involves actual vehicle dynamics. This layer receives information from the regulation layer and also sense information about the speed, acceleration, and engine state of the vehicle to the regulation layer.

### 3.4.2 Regulation Layer

The regulation layer within each vehicle executes the tasks specified by the coordination layer by converting them to throttle, steering and braking inputs for the actuators on vehicle. This controller uses five feedback laws to execute the lateral and longitudinal maneuvers and also notifies the coordination layer in case of any failures or unsafe outcomes of the maneuver. The five feedback laws user for achieving lateral and longitudinal control tasks are:

1. The follower law maintains the required constant spacing between the vehicles in a platoon.
2. The tracking law maintains the speed close to that of the speed specified by the link layer.
3. The merge law causes the leader of the platoon to join a platoon in front of it.
4. The split law reduces the speed of the vehicles in the platoon to maintain a safe headway between the split vehicles.
5. The change lane law allows a vehicle in a platoon to maintain a safe headway between the split vehicles.
6. The change lane law allows a vehicle in a platoon to change to the adjacent lane.
The follower and tracking feedback laws are used to carry out safe and efficient longitudinal control tasks in platoons. These laws regulate the platoon velocity to a desired value, and also maintain a safe distance from the preceding platoon. The merge, split, and change lane laws are used for achieving lateral control maneuvers. The lateral control laws keep the vehicles in their assigned lane or allow them to change to an adjacent lane.

### 3.4.3 COORDINATION LAYER

From this layer onwards, controllers are embedded in each vehicle. The controller in each vehicle receives the commands from the link layer, and organizes the traffic in platoons and coordinates the maneuvers of platoons with their neighbors. The basic maneuvers associated with this layer are lane change, split, and merge. This layer also assumes that the vehicles can exchange information with other platoons using protocols and each platoon can be engaged with only one maneuver at a time.

In a merge maneuver, two platoons join to make a single platoon, a split maneuver is the opposite, and the lane-change maneuver allows a free vehicle to change lanes. The split and join maneuvers are carried out by the platoon leaders. In order to do the maneuver safely, this controller uses protocols (message exchange) with its neighbor platoons. Once the controller has found that the maneuver can be carried safely (coordination achieved), then the controller sends this maneuvering information to the next layer down in the hierarchy to perform or execute this task.

### 3.4.4 LINK LAYER

Link layer operates at the roadside and has a controller located per each link. Each link controller receives commands from the Network layer and based on these commands, the controller monitors and coordinates the activities of each section on a link. A link controller calculates the maximum platoon size and optimum velocity for each section and also set the path (which lane to follow) for each vehicle to reach its destination as quickly as possible. While selecting the paths, the controller also tries to minimize the impact of unexpected incidents on the traffic flow. These controls are based on the macroscopic level of information.

### 3.4.5 NETWORK LAYER

The top layer in this hierarchy is called the network layer. This layer is responsible for controlling the traffic flow on the entire network to obtain a maximized traffic output. The task of the controller is to assign a route for each vehicle that enters the highway ensuring that each potential route on the highways is utilized properly. This route should help the vehicle to reach its destination as quickly as possible and in case of congestion; the network layer controller is responsible for rerouting the vehicles along with assuring maximum throughput on the network.
3.5 DISCUSSION

In chapter 3, several concepts related to control architectures in ITS have been discussed. These concepts include: Multi-Agents System, Subsumption Architecture, Belief-Desire-Intention model, and an ITS Control Architecture PATH. This section will discuss the usability of the concepts as explained above for the design of controller in ITS.

![Diagram showing the proposed ITS discussed in section 3.5]

The first concept, Multi-Agents System (MAS) paradigm shares some characteristics that are suitable for ITS environments. Using MAS paradigm, we could see the vehicles in ITS as agents in MAS. The same thing could be said about the TISNET infrastructure as explained in chapter 2. Agents have a limited view, just like vehicles or infrastructure sensors that are capable to extract information only from a certain range. Every sensor and vehicle in ITS control their own behavior, just like how agents work in MAS.
Several other MAS characteristics like decentralized information and asynchronous computation seem to match perfectly with what we expect to have in an ITS. Therefore, MAS is an obvious choice for implementing an ITS.

With MAS defining the general interaction between agents, Subsumption Architecture is a potential architecture for the agents, or to be more specific, vehicles. The Subsumption Architecture has several characteristics that would be advantageous as a framework for control in ITS. It is robust with respect to failures, capable of handling multiple goals, and allows for both reactive and strategic behavior. However, it is not without any drawbacks. Designing action selection through a distributed system of inhibition and suppression becomes increasingly difficult with the number of layers. We use four layers in the architecture, but would like to keep the possibility of extending the number of layers for possible future works. We conclude that the Subsumption Architecture is too complex to be used as a framework for vehicle control.

Another possibility for vehicle control is the ITS Control Architecture PATH. PATH architecture uses a hierarchical control with multiple layers. While this is similar to Subsumption Architecture, the hierarchical architecture in PATH framework does not use the inhibition and suppression methods, but rather using communication between adjacent layers. The advantage offered by such a multi-layered system is that the structure is flexible. It is possible to add or modify a layer without directly affecting the others since they are not tightly coupled. Similar to the Subsumption Architecture, it is also robust, failure in one component will not causing failure in the whole system. Note that in the PATH framework, the controller is distributed between the vehicle system and infrastructure, while in this report we want to separate them and exclusively put the controller in the vehicle system while the infrastructure provides sensors. Therefore we propose to use a hierarchical control for vehicles similar to the one from PATH architecture, only without the incorporation of infrastructure control.

In a real-time situation like traffic, the control system should be able to be both reactive and strategic. In hierarchical control architecture, this could be addressed by providing a layer for each level of abstraction. Ideally, a lower layer is simple and reactive; enough to keep the controller working while keeping safety conditions intact. A Final State Machine might be enough to represent these layers. Higher layers are more complicated. In these layers, we do not want hard-coded behavior. This is where the BDI model may become useful. Using a BDI model in one or more layers in hierarchical control architecture could provide more insight in understanding the behavior of control. Finally, using Final State Machine for lower layers and BDI for higher layers could give the control the ability to be both reactive and strategic.

In summary, the concept of MAS is used to describe the ITS environments. For vehicles, hierarchical controls with multiple layers are the most suitable one. Lower layers in vehicle control use Final State Machine and higher layers use BDI model.
4 CONTROLLER IMPLEMENTATION

The Vehicle Controller architecture consists of several hierarchical layers. The layers are autonomous. Each layer is able to communicate to the adjacent layers (layers that are either directly above or below itself). Layers may have access to several external components such as: driving parameters, sensory data, and communication data.

As shown in Figure 4-1, the lowest layer is the Physical layer. This layer works as an interpreter to translate the instructions coming from the upper layer into mechanical changes for the car. The Motion Primitives, Control Functionalities, and Cooperation layer will be explained in more detail in the following sections.

4.1 MOTION PRIMITIVES

The Motion Primitives layer is responsible for executing maneuvers of vehicle such as increase/decrease speed and lane change. Other than that, it also has to maintain the safety of the vehicle during the execution of maneuvers. Motion Primitives layer consists of longitudinal controller and lateral controller which do not dependent on each other. Longitudinal controller defines the vehicle behavior for longitudinal direction while lateral controller is responsible of doing the changing lane maneuver.
In order to understand the longitudinal and lateral controller that will be explained later, terms like “reference vehicle”, “safe distance”, and “adjusting distance” will be defined first.

### 4.1.1 Reference Vehicle

To avoid confusion between the vehicles we are about to discuss, we will call the vehicle that the intelligent controller is placed in as “Self”.

Motion Primitives layer is responsible for the safety of Self as well as executing maneuvers of Self. Typically, there is one other vehicle which is the most important to keep track of. We call this vehicle “reference vehicle”. Motion Primitives always choose the closest vehicle in front of Self as its initial reference vehicle. The reference vehicle that is chosen has to be in the same lane as Self. A vehicle that occupies two lanes at a time while one of the lanes is the same lane where Self is cruising on is considered to be in the same lane and therefore might be selected as a initial reference vehicle by the Motion Primitives.

In case the Motion Primitives layer received a request from a higher layer to pick another vehicle as reference vehicle, it will choose between that vehicle and its own initial reference vehicle. Motion Primitives will select the requested vehicle unless Self is already within the adjusting or safe distance of the initial reference vehicle. In the last case, it will select the initial reference vehicle as the reference vehicle. More explanation about safe and adjusting distance is available in the following part of this section.

We call the selected reference vehicle originated from initial reference vehicle as “Free RV” and the one given reference vehicle from another layer as “Requested RV”.

It is also possible that Self does not have any vehicle as a RV in case it does not find any vehicle in front of it and there is no requested vehicle from other layer.

### 4.1.2 Safe and Adjusting Distance

Safe distance is a dynamic value representing a minimum allowed distance to the reference vehicle. The value of safe distance depends on the minimum distance and time headway that should be decided by the user. The minimum distance basically could be any positive number, while time headway is the time required for a vehicle to stop completely. The value of time headway as a rule of thumb is somewhere between 0.7 and 1 second [11]. In the implementation, we use the value of 1 second. By keeping the safe distance, a vehicle will have enough time to stop in case the reference vehicle makes a sudden stop. This is very important when the reference vehicle is the vehicle right in front of Self vehicle (see Reference Vehicle).
The safe distance (Sd) is calculated using the formula [11]:

\[
S_d = V_r * Th + Md
\]

*Vr* = speed of reference vehicle  
*Th* = time headway  
*Md* = minimum distance

On the other hand, adjusting distance depends on the decelerating capacity of vehicle. The preferred value of deceleration usually varies between 2 and 4 m/s^2. This value has been chosen to avoid sharp breaking. The formula for Adjusting distance (Ad) is [11]:

\[
A_d = V_i * T_{ad} + S_d
\]

*Vi* = speed of adjusting vehicle  
*T_{ad}* = adjusting time  
*Sd* = safe distance

The adjusting time (T_{ad}) itself can be calculated as follows [11]:
4.1.3 DEALING WITH IMPERFECT INFORMATION

In the explanation of safety and adjusting distance above, we assume that some variables such as speed and position are already obtained and trustable. In reality, these variables are a result of observation process using sensors. The problem is there is no perfect sensor. As explained in chapter 2, the sensor which we use (TISNET) is not an exception. TISNET is not a perfect sensor and will not generate perfect information of the environment. This section will discuss the effect of imperfect information on safety and how to deal with it.

As mentioned in chapter 2, there are three possible forms of imperfect information as a result of TISNET’s observation: uncertainty, delay and jitter, and message loss.

4.1.3.1 UNCERTAINTY

As explained in chapter 2, uncertainty in TISNET’s observation is in the form of speed and position displacement. In terms of speed, information about a vehicle cruising with speed $V$ m/s with $N$ m/s of uncertainty means that the vehicle is actually cruising anywhere between $V-N$ and $V+N$ m/s. In terms of position, a vehicle located at $(X, Y)$ with $N$ meter uncertainty for both x and y axes could actually be anywhere inside the radius of $N$ meter.

It is obvious that with uncertainty in speed and position, the safety condition may not be achieved if we calculate distance between vehicle, the safety distance, and the adjusting distance as it is. The risk of

\[
T_{ad} = - \frac{(V_r - V_i)}{D}
\]

$V_r =$ speed of reference vehicle

$V_i =$ speed of adjusting vehicle

$D =$ preferred deceleration value

Equation 4-3 Adjusting time formula
unsafe driving is increases with uncertainty. The safest way to adjust is to use the worst possible value as the basis for the calculation.

Let us illustrate the situation. Self make an observation by receiving information from the TISNET sensor and found two vehicles, one of them is Self and the other one is another vehicle who became reference vehicle for Self. The speed of Self and reference vehicle are \( V_i \) m/s and \( V_r \) m/s and they are located at \((x_s, y_s)\) and \((x_r, y_r)\). The uncertainty for this information is \( N \) meter for position and \( M \) m/s for speed. Note that the speed of Self is not affected by uncertainty since its value is known to the controller; it does not rely on information from TISNET.

The worst possible situation possible here is when the speed of \( V_r \) is actually \( V_r - M \) m/s. Therefore to achieve safety, the equation for the safety distance \((Sd)\), the adjusting distance \((Ad)\), and the adjusting time \((T_{ad})\) become:

\[
S_d = (V_r - M) \times Th + Md
\]

\[
V_r = \text{speed of reference vehicle}
\]
\[
M = \text{speed uncertainty}
\]
\[
Th = \text{time headway}
\]
\[
Md = \text{minimum distance}
\]

\[
Ad = V_i \times T_{ad} + S_d
\]

\[
V_i = \text{speed of adjusting vehicle}
\]
\[
T_{ad} = \text{adjusting time}
\]
\[
S_d = \text{safe distance}
\]

\[
T_{ad} = - \frac{(V_r - V_i - M)}{D}
\]

\[
V_r = \text{speed of reference vehicle}
\]
\[
M = \text{speed uncertainty}
\]
\[
V_i = \text{speed of adjusting vehicle}
\]
\[
D = \text{preferred deceleration value}
\]
Besides the modifications of the formula for the safety distance, adjusting distance, and time headway, it is also necessary to modify the calculation of distance between Self and reference vehicle. Normally, the distance between those two vehicles \( d \) in axis \( y \) could simply be calculated as:

\[
\begin{align*}
\text{Equation 4-5 Distance between two vehicles} \\
d &= y_r - y_i \\
y_r &= \text{position of reference vehicle in axis } y \\
y_i &= \text{position of Self in axis } y
\end{align*}
\]

By taking uncertainty into account, the distance between them under worst condition is estimated by:

\[
\begin{align*}
\text{Equation 4-6 Distance formula with uncertainty} \\
d &= (y_r - N_y) - (y_i + N_y) = y_r - y_i - 2N_y \\
y_r &= \text{position of reference vehicle in axis } y \\
y_i &= \text{position of Self in axis } y \\
N_y &= \text{position uncertainty in axis } y
\end{align*}
\]

It is important to know that if the uncertainty value is too high the safety condition can no longer hold, e.g. the uncertainty is position is higher than the distance between two vehicles so that it is not possible to determine which vehicle is in front.

\[4.1.3.2 \text{ DELAY AND JITTER}\]

The presence of delay and jitter in transmission means that the information received is not the most up-to-date. This is critical in a real time situation such as traffic where safety is most important.

To cope with delay and jitter, the first thing to do is to predict the possible situation at the current time. For Self this is an easy task since the information of its speed is available. For another vehicle, we take the simple approach that if no information to the contrary is available; we assume that vehicles do not change speed. Therefore, the new position for every vehicle is:
As there is a possibility that any vehicle could change its speed during the delay and jitter, we should indicate this situation by increasing the uncertainty values with the worst possible situation. For reference vehicle, the worst possible situation is that it reduces its speed drastically with the maximum deceleration possible. Therefore, the new uncertainty values for speed and position are:

\[
N_t = N_0 + \text{dec} \times t
\]

- \( N_t \) = current uncertainty value for speed
- \( N_0 \) = initial uncertainty value for speed
- \( \text{dec} \) = the highest possible deceleration possible
- \( t = \text{delay} + \text{jitter} \)

\[
M_t = M_0 + \text{dec} \times t^2
\]

- \( M_t \) = current uncertainty value for position
- \( M_0 \) = initial uncertainty value for position
- \( \text{dec} \) = the highest possible deceleration possible
- \( t = \text{delay} + \text{jitter} \)

Once the new uncertainty values are calculated, the safety distance and adjusting distance are calculated using the formula described in Uncertainty section.
4.1.3.3 MESSAGE LOSS

Message loss occurs when the information that is sent by TISNET is not received by the vehicle. In this situation, vehicles do not have any information about the environment at that moment. In order to cope with message loss, a vehicle has to use the last information received before. The rest of the process is similar to the way the vehicle handles delay and jitter.

4.1.4 THE LONGITUDINAL CONTROLLER

As mentioned before, the longitudinal controller deals only with longitudinal direction (speed, acceleration, deceleration). It is structured as a Final State Machine. The longitudinal controller changes its state according to its observation and information from other layers.

![Diagram of Longitudinal Controller Mode]

In general, there are two modes of longitudinal control: FREE and FORCED, see Figure 4-5. Free mode is achieved when the Motion Primitives layer controls the behavior of the vehicle without any influence from other layers. On the other hand, Forced mode takes place when the Motion Primitives layer accepts a requested vehicle as the reference vehicle (i.e., “Requested RV”). Free mode consists of three states and forced mode consists of two states as shown in Figure 4-5.

4.1.4.1 NORMAL MODE

Normal Mode is the initial state for all vehicles. This state is achieved when there is no reference vehicle or the reference vehicle’s distance is larger than the adjusting distance. In Normal Mode, the vehicle
cruises on its expected speed. If the expected speed is not yet achieved then it will accelerate until the vehicle speed is equal to expected speed.

The Motion Primitives will switch from Normal Mode to another state in the following situations:

<table>
<thead>
<tr>
<th>Switch to</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJUSTING mode</td>
<td>Self gets within the adjusting distance between it and Free RV and its current speed is higher than Free RV’s speed. Self gets within the safety distance.</td>
</tr>
<tr>
<td>FORCED ADJUSTING mode</td>
<td>Self gets within the adjusting distance of Requested RV and its current speed is higher than Requested RV’s speed.</td>
</tr>
</tbody>
</table>

The Motion Primitives will switch to Normal Mode in the following situations:

<table>
<thead>
<tr>
<th>Switch from</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJUSTING mode</td>
<td>The Free RV speeds up and exceeds the expected speed of Self.</td>
</tr>
<tr>
<td>LOCK mode</td>
<td>The Free RV speeds up and exceeds the expected speed of Self.</td>
</tr>
<tr>
<td>FORCED ADJUSTING mode</td>
<td>The Requested RV speeds up and exceeds the expected speed of Self.</td>
</tr>
<tr>
<td>FORCED LOCK mode</td>
<td>The Requested RV speeds up and exceeds the expected speed of Self.</td>
</tr>
</tbody>
</table>

4.1.4.2 ADJUSTING MODE

Adjusting Mode is typically a transition between Normal Mode and Lock mode. Adjusting Mode can only be activated when the Motion Primitives is using a Free RV. In Adjusting Mode, the vehicle will decelerate until its current speed is equal to that of the reference vehicle and the distance between them is equal or more than the safety distance. Once this situation is achieved, the Motion Primitives will switch its state into Lock Mode.

The Motion Primitives will switch from Adjusting Mode in the following situations:

<table>
<thead>
<tr>
<th>Switch to</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORMAL mode</td>
<td>Self enters adjusting distance between it and Free RV and its current speed is higher than Free RV speed or Self vehicle enters safety distance.</td>
</tr>
<tr>
<td>LOCK mode</td>
<td>The speed of Self is equal to the speed of Free RV and the distance is equal or more than the safety distance.</td>
</tr>
</tbody>
</table>

The Motion Primitives will switch to Adjusting Mode in the following situations:

<table>
<thead>
<tr>
<th>Switch from</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORMAL mode</td>
<td>Self enters the adjusting distance between it and Free RV and its current speed is higher than Free RV speed or Self vehicle enters safety distance.</td>
</tr>
</tbody>
</table>
### 4.1.4.3 LOCK MODE

As the name suggests, lock mode is a state where the vehicle locks its speed, e.g. no acceleration or deceleration. Adjusting Mode can only be activated when The Motion Primitives is using a Free RV.

The Motion Primitives will switch from Lock Mode in the following situation:

<table>
<thead>
<tr>
<th>Switch to</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORMAL mode</td>
<td>The Requested RV speeds up and exceeds the expected speed of Self.</td>
</tr>
<tr>
<td>ADJUSTING mode</td>
<td>Free RV slows down which causes the distance between it and Self vehicle to become less than the safety distance.</td>
</tr>
<tr>
<td>FORCED ADJUSTING mode</td>
<td>Self enters the adjusting distance between it and Requested RV and its current speed is higher than Requested RV speed or Self vehicle enters the safety distance.</td>
</tr>
</tbody>
</table>

The Motion Primitives will switch to Lock Mode in the following situation:

<table>
<thead>
<tr>
<th>Switch from</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJUSTING mode</td>
<td>The speed of Self is equal to the speed of Free RV and the distance is equal or more than the safety distance.</td>
</tr>
</tbody>
</table>

### 4.1.4.4 FORCED ADJUSTING MODE

Forced adjusting mode is similar to adjusting mode with the only difference that it uses Requested RV instead of Free RV.

The Motion Primitives will switch from Forced Adjusting Mode in the following situations:

<table>
<thead>
<tr>
<th>Switch to</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORMAL mode</td>
<td>Self enters adjusting distance between it and Free RV and its current speed is higher than Free RV speed or Self vehicle enters safety distance</td>
</tr>
<tr>
<td>LOCK mode</td>
<td>The speed of Self vehicle is equal to the speed of Free RV and the</td>
</tr>
</tbody>
</table>
distance is equal or more than the safety distance

**FORCED LOCK mode**
The speed of Self is equal to the speed of Requested RV and the distance is equal or more than the safety distance.

The Motion Primitives will switch to Forced Adjusting Mode in the following situations:

<table>
<thead>
<tr>
<th>Switch from</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORMAL mode</td>
<td>Self vehicle enters adjusting distance between it and Free RV and its current speed is higher than Free RV speed or Self vehicle enters safety distance.</td>
</tr>
<tr>
<td>LOCK mode</td>
<td>Free RV breaks which causes the distance between it and Self vehicle to become less than the safety distance.</td>
</tr>
<tr>
<td>FORCED LOCK mode</td>
<td>Self vehicle enters adjusting distance between it and Free RV and its current speed is higher than Free RV speed or Self vehicle enters safety distance.</td>
</tr>
</tbody>
</table>

### 4.1.4.5 FORCED LOCK MODE

Forced lock mode is similar to lock mode with the only difference that it uses Requested RV instead of Free RV.

The Motion Primitives will switch from Forced Lock Mode in the following situation:

<table>
<thead>
<tr>
<th>Switch to</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORMAL mode</td>
<td>The Requested RV speeds up and exceeds the expected speed of Self.</td>
</tr>
<tr>
<td>ADJUSTING mode</td>
<td>Self vehicle enters adjusting distance between it and Free RV and its current speed is higher than Free RV speed or Self vehicle enters the safety distance.</td>
</tr>
<tr>
<td>FORCED ADJUSTING mode</td>
<td>Self vehicle enters adjusting distance between it and Free RV and its current speed is higher than Free RV speed or Self vehicle enters safety distance.</td>
</tr>
</tbody>
</table>

The Motion Primitives will switch to Forced Lock Mode in the following situation:

<table>
<thead>
<tr>
<th>Switch from</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORCED LOCK mode</td>
<td>The speed of Self is equal to the speed of Requested RV and the distance is equal or more than the safety distance.</td>
</tr>
</tbody>
</table>

### 4.1.5 THE LATERAL CONTROLLER

The Lateral Controller deals with vehicle maneuvers in lateral direction also known as lane changing. Like the Longitudinal Controller, the Lateral Controller is also structured as a Final State Machine. Without any request from higher layers, the Lateral Controller will be in OFF mode, which means no lane switch will
be executed. A request can only be executed by the Lateral Controller if it follows the safety conditions discussed below.

In order to execute a lane switch safely, there are safety conditions that have to be kept for the areas in front and back of the vehicle. Both conditions refer to the lane where Self intends to move to.

For the front area, one of these conditions must be present:

1. There are no vehicles detected within the front area.
2. The distance from the nearest car in the front area is bigger than the safety distance and its speed is higher than the speed of Self.
3. The distance from the nearest car in the front area is bigger than the adjusting distance.

As for the back area, the conditions are as follows:

1. There are no vehicles detected within the back area.
2. The distance from the nearest car in the back area is bigger than the safety distance and its speed is lower than the speed of Self.
3. The distance from the nearest car in the back area is bigger than the adjusting distance.

The Lateral Controller has three states: OFF, ON, and CANCEL mode.

---

Figure 4-6 Condition in lane change

Figure 4-7 Lateral Controller Mode
### 4.1.5.1 OFF MODE

In Off Mode, the Lateral Controller is practically doing nothing. This happens when there is no request from a higher layer accepted by the Lateral Control.

The Lateral Controller will switch from Off Mode in the following situation:

<table>
<thead>
<tr>
<th>Switch to</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON Mode</td>
<td>The Motion Primitives layer receives a request to make a lane change and the safety condition holds.</td>
</tr>
</tbody>
</table>

The Lateral Controller will switch to Off Mode in the following situations:

<table>
<thead>
<tr>
<th>Switch from</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON Mode</td>
<td>Self has finished executing a lane change and is centered inside a lane.</td>
</tr>
<tr>
<td>CANCEL Mode</td>
<td>Self has returned to the nearest lane and is centered inside it.</td>
</tr>
</tbody>
</table>

### 4.1.5.2 ON MODE

When the Motion Primitives layer receives a request from a higher layer to do a lane change, the Lateral Controller will switch to On Mode. When the lane change is completed, it will return to Off Mode.

The Lateral Controller will switch from On Mode in the following situations:

<table>
<thead>
<tr>
<th>Switch to</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF Mode</td>
<td>Self has finished executing a lane change and is centered inside a lane.</td>
</tr>
<tr>
<td>CANCEL Mode</td>
<td>During the execution of a lane change, a safety condition fails.</td>
</tr>
</tbody>
</table>

The Lateral Controller will switch to Off Mode in the following situation:

<table>
<thead>
<tr>
<th>Switch from</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON Mode</td>
<td>The Motion Primitives layer receives request to make a lane change and the safety condition holds.</td>
</tr>
</tbody>
</table>

### 4.1.5.3 CANCEL MODE

When Self is trying to change lane, there might be a situation in which some safety conditions fail. In this situation, the Lateral Controller will switch to the Cancel mode. In Cancel mode, Self is forced to go to the nearest lane even if it is not the lane Self originally decided to go to.
Figure 4-8 An example situation where Self goes into cancel mode

The Lateral Controller will switch from Cancel Mode in the following situation:

Table 4-15 Situation where Lateral Controller will switch from Cancel Mode

<table>
<thead>
<tr>
<th>Switch to</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF Mode</td>
<td>Self went to the nearest lane and is centered inside it.</td>
</tr>
</tbody>
</table>

The Lateral Controller will switch to Cancel Mode in the following situation:

Table 4-16 Situation where Lateral Controller will switch to Cancel Mode

<table>
<thead>
<tr>
<th>Switch from</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON Mode</td>
<td>During the execution of a lane change, a safety condition fails.</td>
</tr>
</tbody>
</table>

4.1.5.4 INPUT/OUTPUT PARAMETERS IN MOTION PRIMITIVES LAYER

Input parameters from the Physical layer:

Table 4-17 Input parameters from the Physical layer

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>Speed of the vehicle in two axes: x and y</td>
</tr>
<tr>
<td>Acceleration</td>
<td>Acceleration of the vehicle in two axes: x and y</td>
</tr>
</tbody>
</table>

Output parameters to the Physical layer:

Table 4-18 Output parameters to the Physical layer

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acceleration</td>
<td>Expected acceleration of the vehicle in two axes: x and y</td>
</tr>
</tbody>
</table>
Input parameters from the Control Functionalities layer:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change speed</td>
<td>Change the expected speed</td>
</tr>
<tr>
<td>Change lane (left/right)</td>
<td>Change the current cruising lane</td>
</tr>
<tr>
<td>Adjust distance and reference vehicle</td>
<td>Decrease speed until an expected distance to reference vehicle is achieved</td>
</tr>
</tbody>
</table>

Output parameters to the Control Functionalities layer:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Longitudinal state</td>
<td>The current longitudinal state</td>
</tr>
<tr>
<td>Lateral state</td>
<td>The current lateral state</td>
</tr>
</tbody>
</table>

4.2 BDI AGENT IMPLEMENTATION

The Control Functionalities and Cooperation layers are implemented using BDI agents. We are using a generic agent model [10] as explained in the previous chapter as a starting point. Note that the Control Functionalities and Cooperation layers each only represent a layer within the whole controller while the generic agent model describes a complete system with two layers. This does not refrain us from using the generic agent model as a guide; however, we use only one layer from the generic agent model which is the base layer.

To avoid confusion, we will use the term “M” agent as the proposed BDI agent for Control Functionalities and Cooperation layers.

According to [10], an agent could have some components from a set of beliefs, a set of desires, a set of constraints, a set of mechanism for interacting with the environment, a set of mechanism for managing beliefs, and control components. The identified components for M are:

1. A set of beliefs
   Constantinii [10] describe beliefs in a very broad way. “The set of beliefs expresses what the agent knows, can do and is able to remember”. Using the definition of beliefs, we define several types of belief in M as follows:
   a. Beliefs from observations
      The M agent is capable of making observation using sensors (see “a set of mechanisms for interacting with the environment”). The observation results are interpreted and stored by M as “facts”. Note that even though they are called “facts”; the facts actually subjective to M. The interpretation process is a transformation of observation values into facts, which is done by execution of the rules to transform observation results into facts (see “beliefs in the form of rules”).
   b. Beliefs from communication
      The M agent is capable of communicating with other layers within the controller or other agents in the world. The results of communication are interpreted and stored by M as facts.
   c. Beliefs-by-design
Some of the beliefs in M agent are facts that are determined during design.

4. Beliefs in the form of rules
   This is the part of beliefs that describes the reasoning procedures within M. There are many types of rules in M: inference of beliefs, rules for identifying situations, rules for identifying solutions for situations, and rules defining the relation of intentions (see “a set of desires and intentions”) and constraints (see “control information”). Other than that, there are also rules to transform observation results into facts, rules to select the most preferable situation to solve, and rules to select the most preferable solution.

   For the next part of this report, we use the term “Beliefs” to refer to the facts, and “Rules” to refer to beliefs that are in the form of rules unless indicated differently.

   ![M Agent Diagram](image)

   **Figure 4-9 The M Agent**

2. A set of desires and intentions
   Desires form the motivational states of M. Desires can be in the form of goals or preferences. Intentions represent the deliberative state of the agent: what the agent has chosen to do. Intentions are the results of a selection process of solutions defined by rules in the set of beliefs.

3. A set of constraints
   In M, constraints act as verification criteria for intentions. Intentions are only allowed to execute when the related opportunity exists that is described in the rules defining the relation of intention and constraints. We will use the term “opportunity” from now on to describe the prerequisite for executing an intention.

4. A set of mechanisms for interacting with the environment
   M has three types of mechanisms to interact with the environment:
   a. Sensing mechanisms: as defined in the architecture of the controller, every layer has access to the sensory system, which is the TISNET. As M is a layer within the controller, it also is capable of using the information from TISNET.
   b. Actuating mechanisms: since M is only part of the controller system and is located in a higher layer in the hierarchical structure, it does not have direct access to the controller’s actuator. However, it can influence the actuator indirectly by sending a message to the lower layer.
   c. Communication: the M agent is able to send and receive messages to/from other layers which are adjacent to it within the controller. It can also communicate to other vehicles using a wireless communication; however, this is only true for the Coordination layer.
5. A set of mechanisms for managing beliefs
To manage beliefs, M is able to make an addition, update, or removal of certain beliefs. Future work would be to extend this mechanism to allow M to learn and modify its behavior e.g., by modifying the rules in beliefs.

6. A control component
No specific control component is defined for M as it is only a part of higher layer in the controller.

### 4.2.1 FLOW OF REASONING

The flow of reasoning inside M is a closed loop with eight different processes. Each process will be described within this section.

![Figure 4-10 Flow of reasoning](image)

In the **Initialization**, M starts by including several Beliefs and Desires. These beliefs and desires are those that are not generated by the system, but instead determined during design time.

**Update Observations** is a process to observe the environment and interprets the observation results into beliefs. The process consists of two steps. The first step is to observe the environment and quantify it. Once the numbers are stored, M transforms the numbers into Beliefs, which is the second part of the Update Observations.

![Figure 4-11 Beliefs’ interpretation](image)
The reasoning starts with **find situations**. Situations are generated due to certain combinations of Beliefs and Desires. These combinations are defined as IF-THEN rules.

![Diagram of Internal States and Rules Database](image)

**Figure 4-12: An example of Find Situations**

In order to solve a situation we have to **select a situation to solve**. M decides the order of priority of situations when multiple situations detected and picks the one with top priority to be processed further in order to get a solution.

In order to resolve a situation, an action is necessary. An action to solve a situation is called solution. To **find solutions**, one or more solutions are selected from a database of situation-solutions.

From several possible solutions, M decides to **select an intention** according to its preferences. This also works similar to the selection of a situation.

In order to execute an intention, the required opportunities have to exist, which is done by the component **check opportunities**. Some intentions do not require any opportunities and therefore are always executable.

The component **Execute actions** may have various effects depending on the type of intentions and the availability of the opportunities. The execution could result in message sending or modification of beliefs.
4.3 CONTROL FUNCTIONALITIES

The Control Functionalities layer controls decision-making related processes without involving external components (e.g. no cooperation with other vehicles). As mentioned in the previous section, The Control Functionalities layer is implemented as a BDI agent. This section shows how the generic agent M is instantiated for the Control Functionalities layer.

4.3.1 SCENARIOS

In this section we will discuss several traffic scenarios and identify the rules to define the behavior of the Control Functionalities layer. The scenarios are as follows:

1. Overtaking
   a. Normal Overtaking
   b. Overtaking with Cooperation
   c. Accepting Request for Space
2. Merging
   a. Normal Merging
   b. Merging with Cooperation
   c. Anticipating a Merging

For every scenario, Self acts as the main actor and is illustrated in orange. Other vehicles are colored in gray.

4.3.1.1 OVERTAKING

Overtaking scenarios involves at least two vehicles cruising at different speeds on the same lane. A faster vehicle is positioned directly behind a slower one.

In the first two scenarios (Normal Overtaking and Overtaking with Cooperation), Self is the faster vehicle. It is trying to overtake the slower vehicle by executing lane change. For the other scenario (Accepting
Request for Space), Self is a vehicle on adjacent to the faster vehicle. The role of Self in that scenario is to help the faster vehicle to do the lane change and overtake the slower vehicle.

### 4.3.1.1.1 NORMAL OVERTAKING

The Normal Overtaking scenario involves two vehicles cruising on the same lane. The first vehicle is Self, positioned directly behind the other vehicle. Self believes that the other vehicle is slower than Self’s expected speed.

1. First we define the motivation for the Control Functionalities layer as a desire to “Cruise at expected speed”. The Control Functionalities layer also has a belief that “Slower vehicle in front will make cruising speed lower than expected”. The Control Functionalities layer’s observation results in a belief that there is “Slower vehicle in front”. We created several rules to deal with this situation:

   **Table 4-21 Normal Overtaking – rules 1**

<table>
<thead>
<tr>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the Control Functionalities layer desires to “Cruise at expected speed” and it believes that there is “Slower vehicle in front” and “Slower vehicle in front will make cruising speed lower than expected” then the Control Functionalities layer is having a situation.</td>
</tr>
</tbody>
</table>

To solve this kind of situation, the Control Functionalities layer can create an intention “Overtake”.

![Figure 4-14 Normal Overtaking – phase 1](image)

**Desire:**
- Cruise on expected speed
**Belief:**
- Slower vehicle in front
- Slower vehicle in front will make cruising speed lower than expected
**Effect:**
- Situation -> Intention: Overtake

2. Now, the Control Functionalities layer has an intention to “Overtake”. Before it acts upon the intention, it has to check whether there the required opportunities to execute it are met in the current situation. This is expressed by the following rule:

   **Table 4-22 Normal Overtaking – rules 2**
Rule
To execute intention “Overtake”, the opportunity “Space available for overtaking” should exist.

Suppose that the result of the Control Functionalities layer’s observation is that the opportunity “Space available for overtaking” exists. In that case the intention can be executed.

4.3.1.1.2 OVERTAKING WITH COOPERATION

Similar to the Normal Overtaking scenario, this scenario involves two vehicles, a slower vehicle in front and Self as the faster vehicle in rear on the same lane. However, unlike the previous scenario, the adjacent lane is not free since it is occupied by some other vehicles.

1. The first part of the scenario is exactly the same as the previous scenario. We define the motivation for the Control Functionalities layer as a desire to “Cruise at expected speed”. The Control Functionalities layer also has a belief that “Slower vehicle in front will make cruising speed lower than expected”. The Control Functionalities layer’s observation results in a belief that there is “Slower vehicle in front”. We created several rules to deal with this situation:

Table 4-23 Overtaking with Cooperation – rules 1

Rules
If the Control Functionalities layer desires to “Cruise at expected speed” and it believes that there is “Slower vehicle in front” and “Slower vehicle in front will make cruising speed lower than expected” then the Control Functionalities layer is having a situation.

To solve this kind of situation, the Control Functionalities layer can create an intention “Overtake”.

Figure 4-15 Normal Overtaking – phase 2
Desire:
Cruise on expected speed
Belief:
Slower vehicle in front
Slower vehicle in front will make cruising speed lower than expected
Effect:
Situation -> Intention: Overtake

Figure 4-16 Overtaking with Cooperation – phase 1

2. Now, the Control Functionalities layer has an intention to “Overtake”. Before it acts upon the intention, it has to check whether the required opportunities to execute it are met in the current situation. As illustrated above, the opportunities do not exist. This means that the Control Functionalities layer needs to find another way to solve its situation, which is by developing a new desire “Want space for overtaking” and a belief “No space for overtaking”. This is expressed by extending the rule from previous scenario:

Table 4-24 Overtaking with Cooperation – rules 2

<table>
<thead>
<tr>
<th>Rule</th>
</tr>
</thead>
</table>
| To execute intention “Overtake”, the opportunity “Space available for overtaking” should exist. Otherwise create a desire “Want space for overtaking” and a belief “No space for overtaking”.

| Intention: |
| Overtake |
| Opportunity: |
| Space available for overtaking (not exist) |
| Effect: |
| Create a desire: Want space for overtaking |
| Create a belief: No space for overtaking |

Figure 4-17 Overtaking with Cooperation phase 2
As a solution for the Control Functionalities layer’s desire to create space, other rules are required:

### Table 4-25 Overtaking with Cooperation – rules 3

<table>
<thead>
<tr>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>If Self desires to “Create a space for overtaking” and it believes that there is “No space for overtaking” and “Space can be created through the help of the Cooperation Layer” then Self is having a situation.</td>
</tr>
<tr>
<td>To solve this kind of situation, Self can create an intention “Request for space to the Cooperation Layer”.</td>
</tr>
</tbody>
</table>

We do not define any required opportunities for “Request for space to the Cooperation Layer”. It means that the execution of “Request for space” can always be done.

3. After the execution of “Request for space to the Cooperation Layer”, the Control Functionalities layer’s task is done and it is the Cooperation layer’s task to process the request. We assume that at a time a space is created (see Cooperation Layer for details). Then we have a similar situation with point (1), although the opportunity is now available. The same rules applied for this situation.
4.3.1.1.3 ACCEPTING REQUEST FOR SPACE

This is exactly the same situation as the previous scenario but from a different point of view. Two vehicles, a slower vehicle in front and the faster vehicle behind it on the same lane still exist. In this scenario, Self acts as a vehicle on the adjacent lane that has the opportunity to help the faster vehicle to do an overtaking.

1. The scenario starts with a message from the Cooperation layer that it has agreed to give another vehicle space for overtaking. The Control Functionalities layer has a desire to “Improve the traffic throughput” and it believes that by “Helping other vehicles will improve the traffic throughput”. A way to solve this situation is by creating an intention “Adjust distance”.

<table>
<thead>
<tr>
<th>Table 4-26 Accepting Request for Space – rules 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rules</strong></td>
</tr>
<tr>
<td>If the Control Functionalities layer desires to “Improve the traffic throughput” and it believes that “The Cooperation layer want to help another vehicle to make an overtaking” and “Helping other vehicles will improve the traffic throughput” then the Control Functionalities layer is having a situation.</td>
</tr>
<tr>
<td>To solve this kind of situation, the Control Functionalities layer can create an intention “Adjust distance”.</td>
</tr>
</tbody>
</table>
Desire:
Improve the traffic throughput
Belief:
The Cooperation layer want to help another vehicle to make an overtaking
Helping other vehicles will improve the traffic throughput
Effect:
Situation -> Intention: Adjust distance

Figure 4-20 Accepting Request for Space – phase 1

2. There is no opportunity defined for “Adjust distance” so the execution is always possible.

Intention:
Adjust distance
Opportunity:
(Nothing)
Effect:
Execute adjust distance

Figure 4-21 Accepting Request for Space – phase 2

4.3.1.2 MERGING

Merging scenarios involve a lane blocking in one lane that forces every vehicles cruising on the blocked lane to change lane before they reach the blocking point.

4.3.1.2.1 NORMAL MERGING

This Normal Merging scenario is very similar to normal overtaking. Instead of the existence of a slower vehicle, in this Normal Merging scenario, a non-moving obstacle appears in the same lane as Self. Self is
forced to do a merging before it reaches the obstacle. In this scenario, a space for a lane change is available so Self can execute the lane change immediately.

1. Similar to the Overtaking scenario, the motivation of the Control Functionalities is to “Cruise on expected speed”. The observation from Control Functionalities layer is that there is “Lane blocking in front”. The Control Functionalities layer believes that “Merging is necessary for vehicle in blocked lane to keep cruising at expected speed”. We created several rules for this situation:

<table>
<thead>
<tr>
<th>Table 4-27 Normal Merging – rules 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rules</strong></td>
</tr>
<tr>
<td>If the Control Functionalities layer desires to “Cruise” and it believes that there is “Lane blocking in front” and “Merging is necessary for vehicle in blocked lane to keep cruising at expected speed” then the Control Functionalities layer is having a situation.</td>
</tr>
<tr>
<td><strong>To solve this kind of situation, the Control Functionalities layer can create an intention “Merging”.</strong></td>
</tr>
</tbody>
</table>

![Normal Merging – phase 1](image)

Desire:
- Cruise on expected speed
Belief:
- Lane blocking in front
Merging is necessary for vehicle in blocked lane to keep cruising on expected speed
Effect:
- Situation -> Intention: Merging

2. Now, the Control Functionalities layer has an intention to “Merging”. Before it acts upon the intention, it has to check whether the required opportunities to execute it are met in the current situation. This is expressed by the following rule:

<table>
<thead>
<tr>
<th>Table 4-28 Normal Merging – rules 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rule</strong></td>
</tr>
<tr>
<td>To execute intention “Merging”, the opportunity “Space available for merging” should exist.</td>
</tr>
</tbody>
</table>

Suppose that the result of the Control Functionalities layer’s observation is that the opportunity “Space available for merging” exists. In that case the intention can be executed.
4.3.1.2.2 MERGING WITH COOPERATION

In merging with cooperation, immediate lane change is not possible due to the occupation of another lane by several vehicles which separated closely to each other. A merging vehicle will find it difficult to execute a lane change without the cooperation of others. The Cooperation layer is involved in this scenario.

1. The initial situation is similar to previous scenario, with the exception that the adjacent lane is occupied. The motivation of the Control Functionalities is to “Cruise at expected speed”. The observation from Control Functionalities layer is that there is “Lane blocking in front”. The Control Functionalities layer believes that “Merging is necessary for vehicle in blocked lane to keep cruising at expected speed”. We use the same rules as previous scenario:

   
   Table 4-29 Merging with Cooperation – rules 1

<table>
<thead>
<tr>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the Control Functionalities layer desires to “Cruise at expected speed” and it believes that there is “Lane blocking in front” and “Merging is necessary for vehicle in blocked lane to keep cruising at expected speed” then the Control Functionalities layer is having a situation.</td>
</tr>
<tr>
<td>To solve this kind of situation, the Control Functionalities layer can create an intention “Merging”.</td>
</tr>
</tbody>
</table>

Figure 4-23 Normal Merging – phase 2
2. The Control Functionalities layer has an intention “Merging”. In this scenario, the opportunity for “Merging” is “Space available for merging”, which does not exist. This means that the Control Functionalities layer need to find another way to solve its situation, which is by developing a new desire “Want space for merging” and a belief “No space for merging”. This is expressed by extending the rule from previous scenario:

<table>
<thead>
<tr>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>To execute intention “Merging”, the opportunity “Space available for merging” should exist. Otherwise create a desire “Want space for merging” and a belief “No space for merging”.</td>
</tr>
</tbody>
</table>

Figure 4-25 Merging with Cooperation – phase 2
As a solution for the Control Functionalities layer’s desire to create space, other rules are required:

**Table 4-31 Merging with Cooperation – rules 3**

<table>
<thead>
<tr>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the Control Functionalities layer desires to “Create a space for merging” and it believes that there is “No space for merging”, “Space can be created if merging partner exist”, and “Merging partner can be found through the help of the Cooperation Layer” then the Control Functionalities is having a situation.</td>
</tr>
<tr>
<td>To solve this kind of situation, the Control Functionalities can create an intention “Request for partner to the Cooperation Layer”.</td>
</tr>
</tbody>
</table>

We do not define any required opportunities for “Request for space to the Cooperation Layer”. It means that the execution of “Request for partner to the Cooperation Layer” can always be done.

**Figure 4-26 Merging with Cooperation – phase 3**
3. We assume that a time will come when the Cooperation layer manages to find a partner and will inform the Control Functionalities layer. The algorithm on how the partner is selected is discussed on the Cooperation layer section. To act upon this situation, we created several rules:

Table 4-32 Merging with Cooperation – rules 4

<table>
<thead>
<tr>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the Control Functionalities desires to “Create a space for merging” and it believes that there is “No space for merging”, “Space can be created if merging partner exist”, “Merging partner is found”, and “To create a space for merging, distance adjustment is necessary” then the Control Functionalities is having a situation.</td>
</tr>
<tr>
<td>To solve this kind of situation, the Control Functionalities can create an intention “Adjust distance”.</td>
</tr>
</tbody>
</table>

The intention “Adjust distance” does not require any opportunities and therefore can be executed.

Desire:
Create space for merging

Belief:
No space for merging
Space can be created if merging partner exist
Merging partner is found
To create a space for merging, distance adjustment is necessary

Effect:
Situation -> Intention: adjust distance
Intention:
Adjust distance
Opportunity:
(Nothing)
Effect:
Execute adjust distance

Figure 4-27 Merging with Cooperation – phase 4
4. After the execution of “Adjust distance”, we assume that the space for merging is now available. Therefore the Control Functionalities layer can execute a merging.

In the Merging with Cooperation scenario, in case the partner comes late and Self does not have enough time to adjust distance before merging then it is possible Self is forced to slow down (or even stop) and wait until it gets the chance to make a merge.

4.3.1.2.3 ANTICIPATING A Merging

When Self detects a lane block in another lane, it could help to improve the traffic by cooperating with other vehicles. The vehicle will do exactly the same thing as the merging vehicle (obviously, without lane change in the end).

1. The motivation for the Control Functionalities layer is to “Improve the traffic throughput”. The Control Functionalities layer believes that “Other lane is blocked” based on its observation. It is also believe that “Helping other vehicles to merge by partnering will improve the traffic throughput”. We create rules based on this situation:

   Table 4-33 Anticipating a Merging – rules 1

<table>
<thead>
<tr>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the Control Functionalities layer desires to “Improve the traffic throughput” and it believes that “Other lane is blocked”, “Helping other vehicles to merge by partnering will improve the traffic throughput” , and “Merging partner can be found through the help of the Cooperation Layer” then the Control Functionalities layer is having a situation.</td>
</tr>
<tr>
<td>To solve this kind of situation, the Control Functionalities layer can create an intention “Request for partner to the Cooperation Layer”.</td>
</tr>
</tbody>
</table>
Desire:
Improve the traffic throughput
Belief:
Other lane is blocked
Helping other vehicles to merge by partnering will improve the traffic throughput
Merging partner can be found through the help of the Cooperation Layer
Effect:
Situation -> Intention: Request for partner to the Cooperation Layer
Intention:
Request for partner
Opportunity:
(Nothing)
Effect:
Request for partner to the Cooperation Layer

Figure 4-29 Anticipating a Merging – phase 1

2. The scenario continues with the Cooperation layer sending a message about the suitable partner. The following rules apply for this situation:

Table 4-34 Anticipating a Merging – rules 2

<table>
<thead>
<tr>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the Control Functionalities desires to “Improve the traffic throughput” and it believes that “Other lane is blocked”, “Helping other vehicles to merge by partnering will improve the traffic throughput”, “Merging partner is found”, and “To create a space for other vehicles to merge, distance adjustment is necessary” then the Control Functionalities is having a situation.</td>
</tr>
<tr>
<td>To solve this kind of situation, the Control Functionalities can create an intention “Adjust distance”.</td>
</tr>
</tbody>
</table>
Desire:
Improve the traffic throughput
Belief:
Other lane is blocked
Helping other vehicles to merge by partnering will improve the traffic throughput
Merging partner is found
To create a space for other vehicles to merge, distance adjustment is necessary
Effect:
Situation -> Intention: Adjust distance
   Intention:
   Adjust distance
   Opportunity: (Nothing)
   Effect:
   Execute adjust distance

Figure 4-30 Anticipating a Merging – phase 2

4.3.2 INPUT/OUTPUT PARAMETERS IN THE CONTROL FUNCTIONALITIES LAYER

Input parameters from the Motion Primitives layer.

Table 4-35 Input parameters from the motion primitives layer

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Longitudinal state</td>
<td>The current longitudinal state</td>
</tr>
<tr>
<td>Lateral state</td>
<td>The current lateral state</td>
</tr>
</tbody>
</table>

Input parameters from the Control Functionalities layer.

Table 4-36 Input parameters from the Control Functionalities layer

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change speed</td>
<td>Change the expected speed</td>
</tr>
<tr>
<td>Change lane (left/right)</td>
<td>Change the current cruising lane</td>
</tr>
<tr>
<td>Adjust distance</td>
<td>Decrease speed until an expected distance is achieved. Reference vehicle is also passed along.</td>
</tr>
</tbody>
</table>
Input parameters from the Cooperation layer.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merging partner is found</td>
<td>A notification from cooperation layer that a partner for merging has been found. A reference vehicle is passed along</td>
</tr>
<tr>
<td>Agreed to help lane change</td>
<td>A notification from cooperation layer that the vehicle decided to help another vehicle to make a lane change. A reference vehicle is passed along</td>
</tr>
</tbody>
</table>

Output parameters to the Cooperation layer.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ask for space</td>
<td>Control functionalities layer would like to make a lane change but no space is available</td>
</tr>
<tr>
<td>Lane blocking</td>
<td>An obstacle on the current lane detected</td>
</tr>
<tr>
<td>Lane blocking on another lane</td>
<td>An obstacle on another lane detected</td>
</tr>
</tbody>
</table>

4.4 COOPERATION

The Cooperation layer controls coordination with external vehicles, including communication-related activities. The Cooperation layer lets the vehicle to interact with another in order make maneuvers that are not possible without cooperation. An example of cooperation application in traffic is a merging situation. With cooperation, vehicles are able to position themselves in such a way that the merging becomes smoother than a merging without cooperation.

4.4.1 SCENARIOS

The same scenarios from Control Functionalities layer are used. However, only those involved with Cooperation layer are presented in this part. The scenarios are as follows:

1. Overtaking
   a. Overtaking with Cooperation
   b. Accepting Request for Space
2. Merging
   a. Merging with Cooperation
   b. Anticipating a Merging

For merging, we use an algorithm to find a partner which is inspired by Proactive Traffic Merging Strategies [12]. The basic idea of the strategy is to reposition every vehicle in a sequence with the order of their (prediction of) arrival time at the merging point. Every vehicle should state their prediction of arrival time at the merging point when they realize the possible merging situation. Based on information from other vehicles, a vehicle will select a vehicle as its partner and reposition itself right behind the selected partner.
4.4.1.1 OVERTAKING

In the first scenario (Overtaking with Cooperation), Self is the faster vehicle. It is trying to overtake the slower vehicle by executing lane change. For the other scenario (Accepting Request for Space), Self is a vehicle on adjacent to the faster vehicle. The role of Self in that scenario is to help the faster vehicle to do the lane change and overtake the slower vehicle.

4.4.1.1.1 OVERTAKING WITH COOPERATION

This scenario involves two vehicles, a slower vehicle in front and Self as the faster vehicle in rear on the same lane. The adjacent lane is not free since it is occupied by some other vehicles.

1. The first part of the scenario does not involve the Cooperation layer at all.

![Figure 4-31 Overtaking with Cooperation – phase 1](image)

2. It is assumed that the Cooperation layer receives a request from the Control Functionalities layer. We define the motivation for the Cooperation layer as a desire to “Respond on requests”. The rule is as follow:

<table>
<thead>
<tr>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the Cooperation layer desires to “Respond on requests” and it believes that there is a “Request for space from CF layer” then the Cooperation layer is having a situation.</td>
</tr>
<tr>
<td>To solve this kind of situation, the Cooperation layer can create an intention “Broadcast a request for space”.</td>
</tr>
</tbody>
</table>

No opportunity is required to execute the intention; therefore the execution is always possible.
3. The role of the Cooperation layer is ended.

### 4.4.1.1.2 ACCEPTING REQUEST FOR SPACE

This is exactly the same situation as the previous scenario but from a different point of view. Two vehicles, a slower vehicle in front and the faster vehicle behind it on the same lane still exist. In this scenario, Self acts as a vehicle on the adjacent lane that has the opportunity to help the faster vehicle to do an overtaking.

1. The scenario starts with a request for overtaking from another vehicle. The Cooperation layer has a desire to “Improve the traffic throughput” and it believes that “Helping the requesting vehicle will improve traffic throughput”. This solution for this situation is by creating an intention “Request to help another vehicle to make an overtaking to the Control Functionalities layer”.

<table>
<thead>
<tr>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the Cooperation layer desires to “Improve the traffic throughput” and it believes that it is “Possible to help the requesting vehicle” and “Helping the requesting vehicle will improve traffic throughput” then the Cooperation layer is having a situation.</td>
</tr>
<tr>
<td>To solve this kind of situation, the Cooperation layer can create an intention “Request to help another vehicle to make an overtaking to the Control Functionalities layer”</td>
</tr>
</tbody>
</table>

No opportunity is required to execute the intention.
4.4.1.2 MERGING

Merging scenarios involve a lane blocking in one lane to force every vehicles cruising on the lane to change lane before they reach the blocking point.
4.4.1.2.1 MERGING WITH COOPERATION

In merging with cooperation, immediate lane change is not possible due to the occupation of another lane by several vehicles which separated closely to each other. A merging vehicle will find it difficult to execute a lane change without the cooperation of others.

1. Similar to the overtaking scenario, the Cooperation layer has a desire to “Respond on requests”. The Control Functionalities layer sends a request to help it with the merging. We define a rule on this situation:

Table 4-41 Merging with Cooperation – rules 1

<table>
<thead>
<tr>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the Cooperation layer desires to “Respond on requests” and it believes that there is “Request from the Control Functionalities layer to help with merging” then the Cooperation layer is having a situation.</td>
</tr>
<tr>
<td>To solve this kind of situation, the Cooperation layer can create an intention “Find a partner and inform the Control Functionalities layer”.</td>
</tr>
</tbody>
</table>

No opportunity is required to execute the intention.

Figure 4-36 Merging with Cooperation – phase 1
2. The Cooperation layer does not have any role in this part of scenario.

Figure 4-37 Merging with Cooperation – phase 2

3. The Cooperation layer does not have any role in this part of scenario.

Figure 4-38 Merging with Cooperation – phase 3

4. The Cooperation layer does not have any role in this part of scenario.

Figure 4-39 Merging with Cooperation – phase 4

4.4.1.2.2 ANTICIPATING A MERGING

The Cooperation layer does exactly the same thing when detecting a lane blocking whether Self is on the blocking lane or not.
1. The motivation for the Cooperation layer is to “Respond on requests”. The Cooperation layer receives message from the Control Functionalities layer that a partner is required. We create rules based on this situation:

Table 4-42 Anticipating a Merging – rules 1

<table>
<thead>
<tr>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the Cooperation layer desires to “Respond on requests” and it believes that “Request from the Control Functionalities layer to find partner” then the Cooperation layer is having a situation.</td>
</tr>
<tr>
<td>To solve this kind of situation, the Cooperation layer can create an intention “Find a partner and inform the Control Functionalities layer”.</td>
</tr>
</tbody>
</table>

No opportunity is required to execute the intention.

Figure 4-40 Anticipating a Merging – phase 1

2. The Cooperation layer does not have any role in this part of scenario.

Figure 4-41 Anticipating a Merging – phase 2
4.4.2 INPUT/OUTPUT PARAMETERS IN THE COOPERATION LAYER

Output parameters to the Cooperation layer.

Table 4-43 Output parameters to the Cooperation layer

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merging partner is found</td>
<td>A notification from cooperation layer that a partner for merging has been found. A reference vehicle is passed along</td>
</tr>
<tr>
<td>Agreed to help lane change</td>
<td>A notification from cooperation layer that the vehicle decided to help another vehicle to make a lane change. A reference vehicle is passed along</td>
</tr>
</tbody>
</table>

Input parameters from the Cooperation layer.

Table 4-44 Input parameters from the Cooperation layer

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ask for space</td>
<td>Control functionalities layer would like to make a lane change but no space is available</td>
</tr>
<tr>
<td>Lane blocking</td>
<td>An obstacle on the current lane detected</td>
</tr>
<tr>
<td>Lane blocking on another lane</td>
<td>An obstacle on another lane detected</td>
</tr>
</tbody>
</table>
5 IMPLEMENTATION

This chapter discusses the technical implementation of the TISNET model, the vehicle controller model, and the simulation scenarios.

We use Multi Agent Real-time Simulator (MARS) as the framework and engine for multi agent simulation. The TISNET model and the vehicle controller models are built as components in the MARS. The visualization is carried out by the VisServer program, which is connected to MARS via UDP/IP communication link. The VisServer uses the visualization output stream from MARS containing the object update information and convert it into a 3D representation of the simulation world. The VisServer also provides several visual models such as cars and roads, which can be instantiated to build visual representation of the simulation scenario.

In the first two sections in this chapter we discuss the MARS as well as the implementation of the TISNET model and the vehicle controller model in MARS. The last two chapters discuss the implementation of traffic scenario and the overview of packages developed in this work.

5.1 MULTI AGENT REAL-TIME SIMULATOR (MARS)

One way to test an ITS is by using a simulation. MARS (Multi Agent Real-Time Simulation) is an engine developed by TNO capable of simulating mixed continuous and discrete event dynamical systems such as traffic, robots, or whatever machine or object environment of interaction. Simulation in MARS can be executed under temporal constraints – which makes the application of simulators based on MARS possible in hardware-in-the-loop (HIL) configurations: virtual (i.e. simulated) and real components can be mixed in an experiment. The virtual components and real components are connected and influencing each other.

Multi-agent approach of MARS is a paradigm that decomposes the “big system” into interacting “intelligent” subsystem called agents with well-defined capabilities/responsibilities. In a traffic system problem domain, some components could be considered as agents: car, driver, traffic controller, etc depending on the modelers themselves.
5.1.1 THE MULTI-AGENT CONCEPT IN MARS

This section describes the main concept of MARS as explained in [14] and [15].

MARS was designed to provide guaranteed real-time performance and implemented on a distributed computing platform to assure scalability. In MARS, the simulation problem has to be decomposed using a multi-agent framework. The formal representation of the simulation experiment is called model. The model describes the collection of autonomous entities, which interact with the surrounding World.

The World is a set of objects and serves as a formal representation of the environment relevant to the entities. Objects are static components: they cannot change their attributes themselves, but entities can operate on them.

Entities may have:

1. Sensors to collect information about the actual state of the World (i.e. surrounding/relevant objects, their attributes).
2. Actuators to create/destroy objects and to modify object attributes in the World.

Entities can also be (and typically are) represented in the World via bound objects. A bound object has attributes, value of which is determined by the states of the associated entity (e.g. an object representing a vehicle in the World has position attribute, which is updated automatically as the corresponding state of the entity evolves in time).

Figure 5-1 The concept of MARS
The important consequence of this decomposition is that the behavioral model of the entities is completely self-contained: “the “only” interface between an entity and its environment is via its own sensor and actuators. To keep the framework generic, sensors and actuators are handled in an abstract way: sensors and actuators have no dynamics and have no data processing features. Instead they merely represent particular relationship between the entity they assigned to and the World. These abstract sensors/actuators can be interpreted as queries/action queries on the “world database”. Consequently real sensors/actuators are modeled as part of the entity’s internal dynamic (i.e. nonlinearities, bandwidth limitation, noise characteristics, etc., can be described in this way).

5.1.2 MARS RUNTIME FRAMEWORK

MARS Runtime Framework could be described as three parts: the basic scheme, partitioning, and communication and scheduling. However, the communication and scheduling part is only take part under real-time Hardware In-Loop (HIL) execution. The description is as follow [13].

5.1.2.1 THE BASIC SCHEME

For MARS a dedicated runtime environment was developed which, by implementing a mobile entity execution scheme, gives scalable real-time performance under certain circumstances. Running the simulation described with a model introduced above means the execution of M pieces of entities on N computing nodes (M>N>=1) connected via a communication infrastructure (RTE). Entities are connected to the RTE via the sensor/actuator API.

![Figure 5-2 The MARS runtime principle of operation [14]](image)

The attributes of the world objects from a distributed common memory, which serves as an information exchange place for the entities. RTEs implement this distributed common memory scheme on a message-parsing framework thus hiding the memory allocation details from the experiment developers/users [14]. The “engine” of the entity simulation is an integrator (numerical solver). Each simulation node incorporates an integrator. This local integrator invokes the entity’s code (i.e. the algorithm of the entity’s behavior, the state update rule) in timely manner (synchronized with other nodes’ progress in time). An entity merely sees the abstract sensor/actuator interface as the only way to communicate with its environment. This runtime architecture isolates the entity code from the configuration details (i.e. the number of nodes involved).
5.1.2.2 PARTITIONING

The implementation of the message parsing common memory architecture is the key issue for assuring good and scalable situation performance. Various data models, caching mechanisms, access control algorithms, etc. have been developed for different DVE applications. Application specific characteristics should be exploited in order to push the performance limits. The entity interactions in typical MARS experiments are constrained spatially: entities interact more frequently and in a more “detailed way” if they are relatively close to each other.

In the execution scheme proposed the association between the entities and the simulation nodes is not static. Instead, the simulation nodes are assigned to disjunctive regions of the World (i.e. in case of M simulation nodes the world is divided into M non-overlapping regions). A simulation node is responsible for the simulation of the behavior of all those entities, which are located in the region the simulation node is assigned to. Entities in a subset are relatively close to each other, consequently they may be in intensive interaction, but these interactions are local to the simulation node thus no inter-node communication is involved. The entity assignment scheme involves entity migration and communication around the borders, but the total communication load is below that of the static allocation scheme under certain conditions [14].

5.2 IMPLEMENTING THE MODELS

As mentioned in the previous section, a dynamical component in MARS is called an entity. We consider the TISNET model and the vehicle controller model as such. The components in MARS are developed in JAVA language. As MARS has already provided several basic components to develop entities, it is not necessary to develop entities from scratch. In the implementation of the TISNET model and the vehicle controller model, we use an abstract component called “ComposableEntity”.

The ComposableEntity structured similar to the multi-layer architecture we proposed in chapter 4. It consists of one or more components called subsystem, which is comparable to layer in the proposed architecture of the vehicle controller. In a subsystem, there are three functions that are possible to be implemented to define the behavior of the subsystem, they are called: sample, output, and step. The flow of execution in the ComposableEntity is as follow:
The execution starts from the sample function in the lowest subsystem. It proceeds to sample procedures in other subsystems and eventually reach the sample procedure in the highest subsystem. Typically the sample procedures are used to collect information from the environment. Once all the sample functions are finished, the execution continues with the output function in the highest subsystem and proceeds to the output function in other subsystems until it reaches the output in the lowest subsystem. The step functions sequence of execution is the same as the sequence of execution of the sample functions.

MARS basically simulates the evolution of a discrete state-space model for each entity:

\[
\begin{align*}
x(k+1) &= F(x(k), u(k)) \\
y(k) &= G(x(k), u(k))
\end{align*}
\]

- \( k \) is the (discrete) time
- \( x \) is the system state
- \( y \) is the output of the system
- \( u \) is the input
The sample functions collect input from the sensors, the output functions calculate the value of G, and the step does the state update (F).

5.3 DEFINING SCENARIOS IN MARS

To define a scenario in MARS, there are two components required: an XML file to define a scenario including the components in a scenario and a Java class containing the user defined (extension) components. In this section we give some ideas about the contents of the XML and Java file.

When the scenario uses only standard MARS provided components, the XML is enough to define the complete scenario. The dedicated sensor and control models used in the work are implemented as Java extensions to the MARS standard library. The components inside both file are described in these following tables.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>General simulation parameters</td>
<td>Including: Java class to use, time step for the simulation, type of visualization to use</td>
</tr>
<tr>
<td>Static objects</td>
<td>Defines the infrastructures in the scenario. Some important parameters to be set: name, position, orientation, and visual representation</td>
</tr>
<tr>
<td>Entities</td>
<td>Defines the entities in the scenario. Some important parameters to be set: Name, position, orientation, visual representation, behavior model (class)</td>
</tr>
<tr>
<td>Visualization</td>
<td>Defines the model path, lighting, and camera position and orientation</td>
</tr>
</tbody>
</table>

Table 5-2 Components in the Java file

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entities’ parameters</td>
<td>Some parameters cannot be defined in the XML file, e.g. speed, acceleration for vehicle and range for TISNET</td>
</tr>
<tr>
<td>Abstract components</td>
<td>Some components does not have any representation in the simulation, e.g. uncertainty control, list for RSUs and vehicles</td>
</tr>
</tbody>
</table>

5.4 THE PACKAGES

The Java packages that are products of the implementation are organized as follow:

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>experiment</td>
<td>Contains the scenario definition as explained in chapter 5.3</td>
</tr>
<tr>
<td>highway</td>
<td>Contains the definition of highway (internal representation for the vehicle controller)</td>
</tr>
<tr>
<td>Directory</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sensor.tisnet</td>
<td>Contains the TISNET model’s implementation</td>
</tr>
<tr>
<td>sensor.tisnet.plugin</td>
<td>Contains the plugins for TISNET model’s implementation</td>
</tr>
<tr>
<td>smartdriving</td>
<td>Contains the vehicle controller’s implementation</td>
</tr>
<tr>
<td>smartdriving.bdi</td>
<td>Contains the BDI agent’s implementation</td>
</tr>
<tr>
<td>smartdriving.motionprimitives</td>
<td>Contains the Motion Primitives layer’s implementation</td>
</tr>
<tr>
<td>smartdriving.controlfunctionalities</td>
<td>Contains the Control Functionalities layer’s implementation</td>
</tr>
<tr>
<td>smartdriving.cooperation</td>
<td>Contains the Cooperation layer’s implementation</td>
</tr>
<tr>
<td>smartdriving.representation</td>
<td>Contains the structure for vehicle’s internal representation</td>
</tr>
<tr>
<td>logger</td>
<td>Contains the logger for experiment</td>
</tr>
</tbody>
</table>
6 EXPERIMENTS AND RESULTS

The TISNET sensor model and controller model are implemented under MARS framework using Java language and Eclipse IDE. To visually observe the simulation, VisServer visualization environment from TNO is used. Furthermore, to collect results, an observer is built in the simulation. This observer has access to simulation information such as time stamp and vehicles motion states (position, speed, etc).

6.1 SCENARIOS

All scenarios are conducted in a 2-lanes straight highway. Twenty vehicles are distributed on both lanes (10 vehicles each) and all of them have the same expected speed which is 20 m/s. Some scenarios use only equipped vehicles while the others contain both instrumented and non-instrumented vehicles. Equipped vehicles mean that the vehicles have the ability to retrieve information from the TISNET sensor and capable of transmitting/receiving information with other equipped vehicles. Non-instrumented vehicles do not have access to TISNET and could not communicate with other vehicles. Non-instrumented vehicles only capable of observing their surrounding within a limited range (100 meter). An obstacle is put on the right lane 1500 meter away from the nearest placement point. Every vehicle is moving forward and in order to pass the obstacle they have to be in the left lane. It means that the vehicles that cruise on the right lane should make a lane change maneuver to the left lane. Scenario is considered to be finished once the last vehicle passes the obstacle.

Initial condition: all vehicles are at least 1500 meters away from the obstacle (note: not all vehicles are shown in the picture)

End condition: all vehicles pass the obstacle and are in the left lane (note: not all vehicles are shown in the picture)
Both equipped and unequipped vehicles will use the merging algorithm as described in chapter 4. However, equipped vehicles will use “Merging with Cooperation” strategy and unequipped vehicles will use the “Normal Merging” strategy.

The most important thing to look from the scenarios is the highway throughput which is the time required to complete a scenario. In order to collect this information, an observer is built in the simulation. Scenarios are varies by sensor, controller, and percentage of instrumented vehicle.

6.1.1 SENSOR

The modeled TISNET sensor has several parameters which are: range, uncertainty, delay, and message loss. Range determines the size of map broadcasted by a Road Side Unit (see chapter 2). Bigger value of range means the vehicles will have the opportunity to detect an event –in this case, a road blocking– earlier and therefore may react faster. Hypothetically, with bigger value of range, the vehicles may execute maneuver earlier and the scenario should be finished faster than with smaller value of range. Four different ranges are chosen to be included in the scenarios.

<table>
<thead>
<tr>
<th>Name</th>
<th>Range distance (meter)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short range</td>
<td>100</td>
</tr>
<tr>
<td>Middle range</td>
<td>200</td>
</tr>
<tr>
<td>Long range</td>
<td>400</td>
</tr>
<tr>
<td>Very long range</td>
<td>1000</td>
</tr>
</tbody>
</table>

The other parameters come from the quality of sensor. A high-quality sensor is expected to have low value of uncertainty, delay, and message loss while a lesser one will have higher value for those parameters. With higher quality sensor, the highway performance should be better and the scenario should be finished earlier. The quality of sensor will be categorized into 3 levels as we proposed below.

<table>
<thead>
<tr>
<th>Quality</th>
<th>Uncertainty (meter)</th>
<th>Delay (second)</th>
<th>Message loss (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 1</td>
<td>2</td>
<td>1</td>
<td>10%</td>
</tr>
<tr>
<td>Level 2</td>
<td>1</td>
<td>0.5</td>
<td>2%</td>
</tr>
<tr>
<td>Level 3</td>
<td>0.5</td>
<td>0.1</td>
<td>0.1%</td>
</tr>
</tbody>
</table>

These parameters are actually independent of each other; we decide to put them together in order to avoid an unacceptably high number of scenarios. For non-instrumented vehicles, their sensor is assumed to have 2 meters uncertainty but without delay and message loss.
### 6.1.2 CONTROLLER

Every vehicle in all scenarios will use the same controller as described in chapter 5. However, there are some differences in the scenarios regarding the driving profile. There will be three types of driving profile used: aggressive, normal, and timid. Every profile has a set of value which will be supplied as parameters for the controller. These parameters are:

1. Preferred acceleration and deceleration values. Timid driver accelerate and decelerate slower than normal and aggressive driver.
2. Minimum distance. Minimum distance is the smallest distance between vehicles that is allowed (see chapter 5). An aggressive driver allows smaller minimum distance while a timid driver will have the biggest minimum distance among the 3 profiles. Minimum distance will affect the distance between vehicle as well as the behavior in changing lane maneuver.

<table>
<thead>
<tr>
<th>Driving Profile</th>
<th>Preferred Acceleration (m/s²)</th>
<th>Preferred Deceleration (m/s²)</th>
<th>Minimum Distance (meter)</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timid</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Aggressive</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

The scenarios with aggressive driver are expected to perform better than the other timid driver.

### 6.1.3 PERCENTAGE OF INSTRUMENTED VEHICLE

Scenarios will also be differentiated by the percentage of instrumented vehicle used. The expectation is that the highway will perform better with more instrumented vehicle. The instrumented of equipped vehicle that will be used in the scenarios are: 0%, 25%, 65%, and 100%.
6.2 RESULTS

The results presented in two forms: table and graph. Table 6-4 shows the complete results of all experiments. Figure 6-3 shows average time to finish scenario for the Timid profile scenario. Figure 6-4 shows the same graph for the Aggressive profile scenario.

<table>
<thead>
<tr>
<th>Sensor range</th>
<th>Sensor quality</th>
<th>% of Instrumented Vehicle</th>
<th>Time to complete scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short range</td>
<td>Level 1</td>
<td>0</td>
<td>149.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>145.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>65</td>
<td>144.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>141.00</td>
</tr>
<tr>
<td></td>
<td>Level 2</td>
<td>0</td>
<td>149.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>143.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>65</td>
<td>143.20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>146.40</td>
</tr>
<tr>
<td></td>
<td>Level 3</td>
<td>0</td>
<td>149.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>141.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>65</td>
<td>151.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>151.20</td>
</tr>
<tr>
<td>Medium range</td>
<td>Level 1</td>
<td>0</td>
<td>149.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>149.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>65</td>
<td>144.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>150.40</td>
</tr>
<tr>
<td></td>
<td>Level 2</td>
<td>0</td>
<td>149.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>142.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>65</td>
<td>151.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>154.40</td>
</tr>
<tr>
<td></td>
<td>Level 3</td>
<td>0</td>
<td>149.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>141.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>65</td>
<td>144.20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>144.20</td>
</tr>
<tr>
<td>Long range</td>
<td>Level 1</td>
<td>0</td>
<td>149.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>146.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>65</td>
<td>150.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>124.80</td>
</tr>
<tr>
<td></td>
<td>Level 2</td>
<td>0</td>
<td>149.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>147.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>65</td>
<td>141.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>124.00</td>
</tr>
<tr>
<td></td>
<td>Level 3</td>
<td>0</td>
<td>149.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>149.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>65</td>
<td>141.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>123.60</td>
</tr>
<tr>
<td>Very Long range</td>
<td>Level 1</td>
<td>0</td>
<td>149.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>142.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>65</td>
<td>131.20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>131.20</td>
</tr>
<tr>
<td></td>
<td>Level 2</td>
<td>0</td>
<td>149.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>144.20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>65</td>
<td>136.20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>125.60</td>
</tr>
<tr>
<td></td>
<td>Level 3</td>
<td>0</td>
<td>149.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>149.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>65</td>
<td>134.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>125.20</td>
</tr>
</tbody>
</table>

Figure 6-3 Time to finish scenario vs % of instrumented vehicle in timid profile
We made several analyses regarding the experiment results:

First, it is important to notice that the results vary heavily due to the nature of random uncertainty value. An experiment may have a significant difference in result when executed more than once. Consult section 2.2, 2.3, and 4.1.3 for more details. Another remark is that scenarios with 0% instrumented vehicle in a profile only executed once because the difference in range and quality of sensor does not affect them.

Regarding the driving profile, it is obvious that vehicles with more aggressive profile are capable of finishing the scenario faster since they accelerate faster and maintain shorter distance with another compared to vehicles in timid profile scenarios. Every scenario shows significant difference in result between the two profiles.

We assumed that vehicles with access to better sensor quality are capable of finishing the scenario faster. However, the differences are not shown very clearly from the results. In some cases, vehicles with worse sensor perform better than those with better sensor. This might also happen due to the nature of random uncertainty value.

Most scenarios show that higher percentage of instrumented vehicle gives the traffic better throughput.

The experiment results shows that range of view plays a very important role in cooperative driving. When the range is long enough to allow the vehicles reposition themselves and merge before they reach the merging point, but if the range is short, the vehicles may not have sufficient time to reposition themselves. This explains why the Short and the Middle range scenarios hardly show any improvement with the introduction of instrumented vehicle and cooperative driving.

Figure 6-4 Time to finish scenario vs % of instrumented vehicle in aggressive profile

We made several analyses regarding the experiment results:

First, it is important to notice that the results vary heavily due to the nature of random uncertainty value. An experiment may have a significant difference in result when executed more than once. Consult section 2.2, 2.3, and 4.1.3 for more details. Another remark is that scenarios with 0% instrumented vehicle in a profile only executed once because the difference in range and quality of sensor does not affect them.

Regarding the driving profile, it is obvious that vehicles with more aggressive profile are capable of finishing the scenario faster since they accelerate faster and maintain shorter distance with another compared to vehicles in timid profile scenarios. Every scenario shows significant difference in result between the two profiles.

We assumed that vehicles with access to better sensor quality are capable of finishing the scenario faster. However, the differences are not shown very clearly from the results. In some cases, vehicles with worse sensor perform better than those with better sensor. This might also happen due to the nature of random uncertainty value.

Most scenarios show that higher percentage of instrumented vehicle gives the traffic better throughput.

The experiment results shows that range of view plays a very important role in cooperative driving. When the range is long enough to allow the vehicles reposition themselves and merge before they reach the merging point, but if the range is short, the vehicles may not have sufficient time to reposition themselves. This explains why the Short and the Middle range scenarios hardly show any improvement with the introduction of instrumented vehicle and cooperative driving.

Figure 6-4 Time to finish scenario vs % of instrumented vehicle in aggressive profile

We made several analyses regarding the experiment results:

First, it is important to notice that the results vary heavily due to the nature of random uncertainty value. An experiment may have a significant difference in result when executed more than once. Consult section 2.2, 2.3, and 4.1.3 for more details. Another remark is that scenarios with 0% instrumented vehicle in a profile only executed once because the difference in range and quality of sensor does not affect them.

Regarding the driving profile, it is obvious that vehicles with more aggressive profile are capable of finishing the scenario faster since they accelerate faster and maintain shorter distance with another compared to vehicles in timid profile scenarios. Every scenario shows significant difference in result between the two profiles.

We assumed that vehicles with access to better sensor quality are capable of finishing the scenario faster. However, the differences are not shown very clearly from the results. In some cases, vehicles with worse sensor perform better than those with better sensor. This might also happen due to the nature of random uncertainty value.

Most scenarios show that higher percentage of instrumented vehicle gives the traffic better throughput.

The experiment results shows that range of view plays a very important role in cooperative driving. When the range is long enough to allow the vehicles reposition themselves and merge before they reach the merging point, but if the range is short, the vehicles may not have sufficient time to reposition themselves. This explains why the Short and the Middle range scenarios hardly show any improvement with the introduction of instrumented vehicle and cooperative driving.

Figure 6-4 Time to finish scenario vs % of instrumented vehicle in aggressive profile

We made several analyses regarding the experiment results:

First, it is important to notice that the results vary heavily due to the nature of random uncertainty value. An experiment may have a significant difference in result when executed more than once. Consult section 2.2, 2.3, and 4.1.3 for more details. Another remark is that scenarios with 0% instrumented vehicle in a profile only executed once because the difference in range and quality of sensor does not affect them.

Regarding the driving profile, it is obvious that vehicles with more aggressive profile are capable of finishing the scenario faster since they accelerate faster and maintain shorter distance with another compared to vehicles in timid profile scenarios. Every scenario shows significant difference in result between the two profiles.

We assumed that vehicles with access to better sensor quality are capable of finishing the scenario faster. However, the differences are not shown very clearly from the results. In some cases, vehicles with worse sensor perform better than those with better sensor. This might also happen due to the nature of random uncertainty value.

Most scenarios show that higher percentage of instrumented vehicle gives the traffic better throughput.

The experiment results shows that range of view plays a very important role in cooperative driving. When the range is long enough to allow the vehicles reposition themselves and merge before they reach the merging point, but if the range is short, the vehicles may not have sufficient time to reposition themselves. This explains why the Short and the Middle range scenarios hardly show any improvement with the introduction of instrumented vehicle and cooperative driving.
Another nice thing we learn from the experiment results are that it is not necessary to have fully instrumented vehicle in traffic to have improvement in throughput. Especially in the Very Long range scenarios, traffic with 25% and 65% instrumented vehicles typically shows improvement compared to those without instrumented vehicles. Obviously, with fully instrumented vehicles in traffic we have the better throughput.
7 CONCLUSIONS AND RECOMMENDATIONS

In this chapter an overview of the objectives stated in the introduction are addressed. Some directions for the future works are also presented.

7.1 CONCLUSIONS

In this work we present a model and implementation in simulation for the parameterized TISNET and vehicle controller for Intelligent Transportation System (ITS). We will describe the conclusions based on the objectives we defined in the Introduction chapter.

The main objective of this thesis is:

To evaluate the impact of intelligent vehicle and intelligent infrastructure in traffic performance

We have conducted some experiments using the developed test environment and the TISNET model and the proposed vehicle controller as mentioned in chapter 6.

The application of the TISNET and the proposed vehicle controller has some potential in improving the traffic throughput. In the experiments, this potential mostly is shown by the scenarios with big range where the vehicles have enough time to execute the maneuvers as part of merging strategy. The improvement is quite significant given that the strategy employed is not yet sophisticated.

The experiment results do not show clearly that the quality of sensor influences the traffic throughput by significant amount. This is possibly because the experiment does not involve a wide range of sensor quality.

From the experiment we can also see that it is possible to mix the instrumented vehicle with non-instrumented vehicle and experience improvement in traffic. However, it is still too early to give recommendation about the required percentage of instrumented that will give significant improvement.

The other objectives of this thesis are:

1. To create an evaluation and test environment for Intelligent Transportation System (ITS) including parameterized infrastructure-based sensor model.
2. To present a particular vehicle control architecture beneficial to the infrastructure-based sensor. We will discuss the objectives in turn.

To create an evaluation and test environment for Intelligent Transportation System (ITS) including parameterized infrastructure-based sensor model.

We have developed an evaluation and test environment based on MARS (Multi Agent Real-time Simulator) framework. Using the framework, we set-ups several traffic scenarios and incorporating the appropriate models in it. The scenario is scalable to the point where computer resources allow it. A simulation observer is also developed to extract the results of a scenario as explained in chapter 6.

We use TNO’s TISNET as the infrastructure-based sensor given its potential to extract vehicle states in real time situation. The modeling of TISNET is based on a possible architecture of TISNET as explained in chapter 2. We modeled TISNET according to the characteristics we mentioned in chapter 2.
To present a particular vehicle control architecture beneficial to the infrastructure-based sensor.

We proposed a vehicle control based on hierarchical architecture in chapter 4. The decision was based on the advantages offered by such multi-layered system. The proposed architecture offers robustness and flexibility. The architecture is robust in the sense that a failure in one component (layer) will not cause failure in the whole system. This feature is beneficial especially to allow iterative development cycle. The flexibility allows us to make additions or modifications in one layer without directly affecting other functionalities (in other layer).

We also proposed a structure for layers based on BDI agent. This allows us to define the behavior of vehicle in an insightful way.

The vehicle control architecture also took care of the characteristic of infrastructure-based sensor, in this matter, TISNET, and designed to keep the safety of the vehicle in reasonable situation.

### 7.2 RECOMMENDATIONS

The current work can be further developed in many directions:

1. Experiments
2. Sensor
3. Control

#### Experiments

The number of experiments conducted in this work is limited due to the sheer amount of possible parameters to be supplied to the experiments. With the current experiments, it is difficult to provide an accurate recommendation for the development of TISNET.

In order to execute a large number of experiments, a script for executing many scenarios is needed because at the moment, MARS framework does not provide such feature. We might also want to execute one experiment several times in order to minimize the effect of random value as explained in the experiment result.

#### Sensor

In this work we solely use TISNET as a sensor for vehicles. Incorporating another sensor into the vehicle controller might have the potential to further improve the performance of vehicles. Since the weakness of TISNET is that it heavily relies on communication technology, a complementary sensor that might be able to give improvement should be something deployed locally inside the vehicle.

#### Control

The vehicle control architecture proposed here left so many development possibilities:

- Introduce a vehicle model to be able to predict other vehicles behavior and therefore better knowledge is available for the vehicle control
- Develop better strategies to solve a traffic situation, especially in merging scenario as explained in chapter 4. The current merging scenario is inspired by [12] with several assumptions which are not exactly suitable for the distributed real-time traffic problem.
- A more realistic vehicle model. The current model of vehicle uses a kinematic model, which is a simplification of a real vehicle model. With a more realistic model, more accurate results and many other examinations are made possible, e.g. fuel consumption.


[7]. Agent Oriented Modelling, Software Versus the World. Yu, E.


[15]. High Fidelity Real Time Simulator with Mixed Real and Virtual Sensors. Papp, Z. and Dorrepaal, M.


[21]. Towards Agents for Policy Making. Dignum, Frank, Dignum, Virginia and Jonker, Catholijn M.


[33]. How to Build Complete Creatures Rather than Isolated Cognitive Simulators. Brooks, Rodney A.


[37]. MARS Technology: What does it offer? TNO.

