UNIVERSITY OF NAIROBI


SCHOOL OF COMPUTING AND INFORMATICS

# MULTI-AGENT BASED TRAFFIC LIGHT CONTROL SYSTEM

BY: KELLY OMONDI AKUKU

REG.NO: P58/64346/2011

**Supervisor**

**Mr: ERICK AYIENGA**

November 2014

**A Project Report Submitted in Partial Fulfillment of the Requirements of the Master of Science in Computer Science.**

**Declaration**

This project as presented in this report is my original work and has not been presented for any other University Award.

Signed:……. Date:……..

Kelly Omondi Akuku

P58/64346/2011

The report has been submitted as partial fulfilment of requirements for the Masters of Science in Computer Science of the University of Nairobi with my approval as the University supervisor.

Signed:…….. Date:…….

Mr. Erick Ayienga

School of computing and informatics

University of Nairobi

## Abstract

With development taking the centre stage, life has become fast-paced journey that time is one of the most important factor. Transport sector has not been left behind on this sphere of development.

Transport technologies have been of great help to humanity, but have come with their own challenges. Traffic jam, congestion, accidents and pollution are the economic losses due to them. Looking at the challenge of road congestions, the mangement of the intersections traffic, contribute greatly to this phenomena.

The intersections in Nairobi are operated through pre-timed electronic controlled traffic lights. These pre-timed traffic lights operating from one intersection, execute their duties at that particular intersection; however, their coordination to the corresponding traffic lights in an adjacent intersection is very inefficient, thereby leading to the deployment of traffic police officers to control the traffic at the intersections. This leads to wastage of manpower energy as well as treating a dynamic situation as static.

To solve this unpredictable traffic congestion situation in Nairobi, this project details the use of multi-agent based framework to model the situation as dynamic in nature. We design and implement the framework, which serves to monitor the changing traffic condition and manages the traffic in accordance to density and wait time. This framework seeks to compare traffic density of lanes, compare wait times and release each lane accordingly.

We use JADE development framework and Prometheus methodology in modelling the agent based solution as it offers an elaborate and stage-by-stage requirements for the system design and code guidelines to ensure system specifications are fully met.

This framework presents aspects of observe-think-and-act model, thereby ensuring coordination of traffic during the whole process of movement. This system if fully implemented, seeks to benefit the road users that includes me and you, thereby increasing the uptake of agent technology in addressing complex and dynamic issues.

***Key Words**; Agents, Multi-Agent Systems, Traffic Congestion, Intersections.*

## Dedication

I dedicate this work to my loving and generous uncle Azariah Ochieng' Akuku, my loving and caring aunt Elidah Akoth Akuku and to my late dad and mum Joshua Ouma Akuku and Isabella Adhiambo respectively. They all inspired me to always yearn for excellence in every pursuit.

Not forgetting my grandmother, Caren Mbiro for her unceasing prayers that has seen me through all the joyous and turbulent times.

## Acknowledgements

I will always remain thankful to the Lord Almighty for seeing me through up to this level. During the period of my research work, I was blessed and came across some wonderful people who enabled me to deeply understand my weaknesses and strengths.

To begin with, I would like to sincerely thank my supervisor, Mr. Erick Ayienga for his ideas, insight and guidance throughout the entire period of my research. This research would have not been complete without his guidance.

Secondly, I would like to thank the University of Nairobi, School of Computing fraternity for providing me with a conducive environment to pursue and complete my research interests.

Finally yet importantly, I owe my special and unreserved thanks to my beautiful wife, Janet Achieng for her wonderful support during the entire period of my studies.

In addition, to you all that made it possible for me to reach this last stage of my endeavours, thank you all.

## Table of contents

**Table of figures**

**Definition of Important Terms**

i. **Agents**: Agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators. Examples include humans, robots, or software agents. They are also capable of interacting with other agents, not simply by exchanging data, but by engaging in analogues of the kind of social activity that we all engage in every day of our lives: cooperation, coordination, negotiation, and the like (Wooldridge, 2002).

ii. **Multiagents:** Multi agent systems involve a team of agents working together socially to accomplish a task (Karsiti and Ahmed, 2009).

iii. **Traffic congestion:** Traffic congestion is a condition on road networks that occurs as use increases, and is characterized by slower speeds, longer trip times, and increased vehicular queuing (Srikanth and Anitha, 2012).

iv. **Communication and Cooperation**- The capacity to interact and communicate with other agents (in multi-agent systems), to exchange information, receive instructions and give responses and cooperate to fulfil their own goals.

v. **Java Agent Development Framework (JADE)**: Is an open source platfrom for peer-to-peer agent based applications (Bellifemine et al., 2014).

vi. **Foundation for Intelligent Physical Agents (FIPA):** is an international non-profit Computer Society standards organization that promotes agent-based technology and the interoperability of its standards with other technologies (FIPA, 2005).

vii. **Number of vehicles handled**: The quantity of vehicles that have passed through the intersection at a particular time.

# CHAPTER ONE

## 1. INTRODUCTION

In recent times, the transport industry is facing challenges in regard to time spent waiting on the roads as opposed to movement. This could be attributable to situations like unfavourable weather conditions, rush hour, road works, accidents or incidents, vehicles loading on the highway, or a vehicle that fails to start at a highway etc, leading to pilings of vehicles on the roads. Over the years, population growth and economic prosperity in urban centres has resulted into increased acquisition of automobiles in urban centres. Hence increase on the number of vehicles on the roads and continual expansion of urban centres. The *United Nations Population Foundation* published in its technical report (UNFPA, 2007) that for the first time, more than half the world's population lives in urban areas and the balance of people continue shifting to the cities. Consequently, drivers and passengers spend a large percentage of their day stuck in traffic. The technical report of the *Texas Transportation Institute* (Schrank et al., 2012) showed that in 2011 traffic congestion represented a $121 billion annual drain on the U.S. economy, with 5.5 billion hours of extra time and 2.9 billion gallons of fuel spent on traffic. According to a recent study by IBM Corporation, a multinational technology and consulting company, Kenya loses close to Sh.50 million a day due to traffic congestion in the city of Nairobi and its environs (Sunday, 2012). A huge amount of time that could be spent on productive activities is wasted on the roads, resulting into huge costs on fuel among other challenges.

For efficient management of the traffic at intersections according to the changing traffic conditions, this project proposes a multi-agent approach for traffic light control in two intersections. It aims to introduce agents at the intersections to manage traffic and replace the pre-timing of the lights. The aim is to reduce each vehicle's waiting time at each of the intersections. This is to be achieved by an agent's observe-think-act process. i.e., the agent continuously observes the prevailing traffic condition by collecting traffic data, and the data is then used for reasoning with the traffic-light-control rules by the agent's inference engine to determine how a signal will be changed on each traffic light at each intersection, so that the traffic can be controlled efficiently and effectively.

Several cities have come up with strategies aimed at reducing traffic congestions. One way to more effectively handle challenges in other cities has been to make the traffic control instruments more intelligent, and have them deal with the complexities of managing the traffic control instrument to the situation at hand. In addition, several studies have been done on intelligence control of transportation traffic over the years.

It is therefore expedient to come up with better-automated and intelligent methods that are responsive to the changing conditions that in effect, make wise use of the available infrastructure in Nairobi to provide efficient flow of traffic. This document is organised as follows. In the next section, problem statement is stated, followed by the proposed solution, objectives, purpose the study seeks to serve, scope and the assumptions. Chapter two explore various literatures highlighting the history of traffic congestion and traffic light control systems, including agent-based approaches. The third chapter illustrates a methodology adopted in investigating and comparing the agent-coordinated Traffic Lights and the current Pre-Timed Traffic Lights technique. The resources and period are highlighted in chapter four. Finally, the references and appendices.

### 1.1. Problem Statement

The main objective of traffic lights at the intersections is to manage traffic efficiently. This is to ensure minimum queue length at the roads joining it. This can be achieved by ensuring efficient coordination of the traffic lights at the intersections with respect to density of vehicles on the roads joining it. The traffic lights that are currently used in Kenyan urban centres use **pre-timed time intervals**. These time intervals are mostly scheduled to have equal intervals during morning, mid-day and evening hours, with the assumption that these are peak hours. In between the peak hours, they show different intervals with the assumption that these are off-peak hours. This does not solve the problem effectively since such changes are still pre-timed. There is also poor interaction between the existing traffic control systems of various intersections to the adjacent ones, thereby resulting into traffic snarl-up. The police officers are used in such circumstances, to coordinate the movement of traffic, hence usually leading to biasness from the part of the officers resulting to massive time wastage to motorists waiting on the roads not released. This pre-timing therefore, presents traffic snarl-up in Nairobi as a static problem, not a dynamic in nature of which is a false impression.

### 1.2. Proposed Solution

The aforementioned problem therefore presents a scenario, which could easily be solved by an **agent-coordinated traffic light system**. A traffic light is a device emitting light in different colours, i.e. green, red and yellow, to issue different instructions. Green illustrates move forward, red means stop and yellow shows caution. The project seeks to present a system of agents that coordinate, cooperate and share information through a wireless link based on the average waiting time and queue lengths on the roads to the intersections. The agents are to control the traffic lights and determine for how long to set the lights to achieve minimum waiting time and reduced queue lengths based on the

prevailing traffic conditions. Evaluation of the proposed system will be done through comparison of the data found from the **pre-timed traffic light** controlled intersections and the **multi-agent based traffic light** controlled intersections by use of simulations.

A and B are separate agents interlinked by a wireless network.

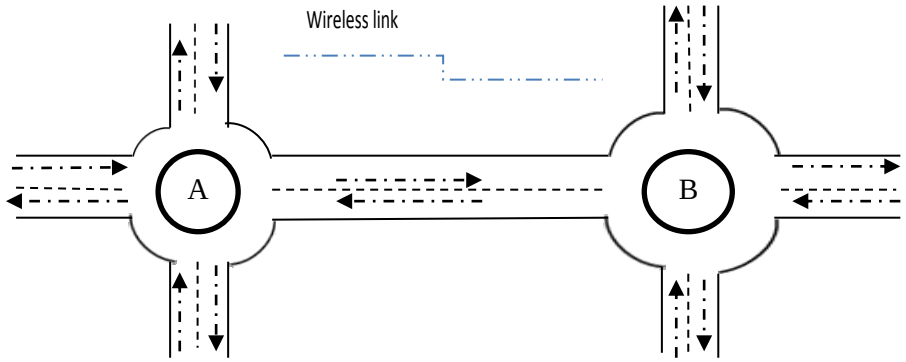


**Figure 1: Agents A and B as shown in the Intersections**

Agents A and B in Figure 1 above, monitors each intersections in which they are located.

They monitor these intersections by visualising the environment, recording the various observations, reproduces these observations and act by interacting with the other agent based on the same data. See Figure 2 below.

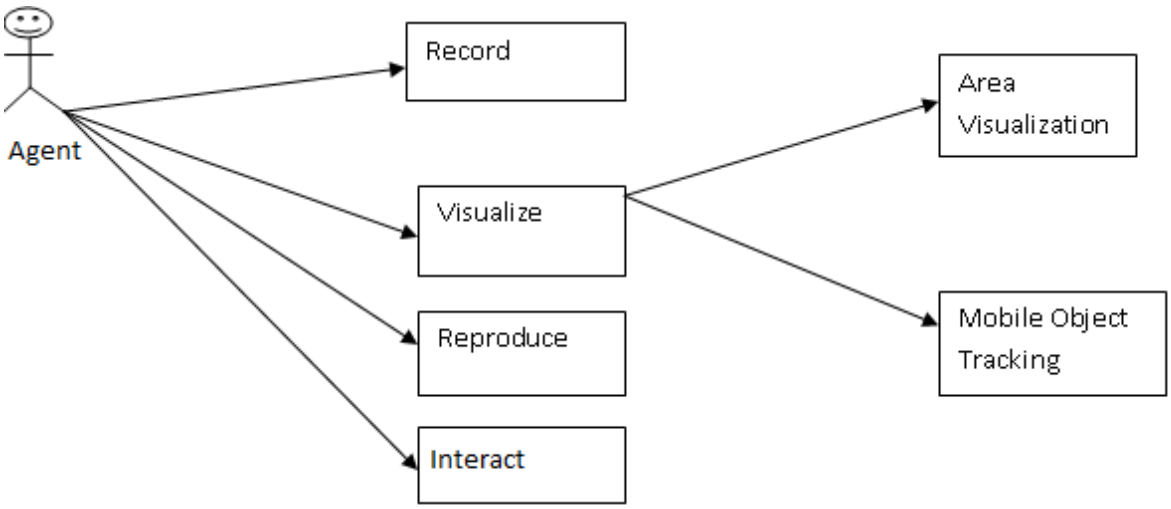


**Figure 2: Agent behaviour in their particular environment**

## 1.3. Objectives

This research aims to achieve the following objectives:

i. Research on the use of multi-agent systems in controlling traffic lights and road congestions.

ii. Design a model that shows the interaction of the agents to control traffic lights hence directing the oncoming vehicles on how to use the intersection.

iii. Evaluate the performance of agent-coordinated traffic lights, and the pre-timed coordinated traffic lights technique at the intersections.

## 1.4. Purpose of the Study

The purpose of the study is to assess the effect of using multi-agent systems in managing traffic lights at the intersections, thereby ensuring efficient flow of road traffic and effective use of existing infrastructure.

## 1.5. Scope of the Study

This study focusses on urban traffic (Nairobi, Kenya) to examine and provide a potential solution under conflicting use of the limited road network by the road users. In addition, to narrow the focus of the research, throughout the study, a homogeneous vehicle composition is assumed. All vehicles in the traffic are assumed to exhibit the same flow pattern.

Due to the limited time, facilities and the existing road network, none of the traffic simulation models is tested on real and physical environment.

As a control approach, the project concentrates on two intersections as the problem domain, i.e. Uhuru Highway/Haille Selassie Intersection and Uhuru Highway/Kenyatta Avenue Intersection. The project uses the traffic flow data as per these two intersections to model the traffic flow pattern in Nairobi, Kenya into a multi-agent system and carry out a software simulation of the same.

Related studies on usage of multi-agent concept in control of traffic flow are also reviewed.

A system model is developed to demonstrate the traffic flow at the intersections as guided by agents using **JADE version 4.3.1** software platform. JADE (Java Agent Development Framework) is a software Framework fully implemented in Java language. It simplifies

the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications and through a set of graphical tools, that supports the debugging and deployment phases (Bellifemine et al., 2014).

### 1.6. Project Deliverables

The overall deliverable in this research project includes:

i. A prototype of the Multi -agent Based Traffic Light Control System in a Java environment;

ii. A detailed report of the research project.

### 1.7. Assumptions

The overall assumptions in this research project include:

i. The agents communicate through a flawless wireless link;

ii. There is constant power supply to enable the operation of these agents at the control points in a real environment;

iii. Only vehicles use the roads, thereby congestion caused is based on the number of vehicles using the road;

iv. All motorists are to adhere to the traffic light signals.

# CHAPTER TWO

## 2. LITERATURE REVIEW

There is a lot of work related to multi-agent traffic control systems that are inspired by works from Artificial Intelligence and Distributed Systems & Networks. This section seeks to review related literature; approaches and tools previously used and elaborate the possible paths this research endeavours to take. Starting with the general introduction, review of the development of traffic flow control methods, various classifications of traffic controllers, gap analysis and the contribution this work seeks to introduce and finally, chapter summary.

### 2.0 General Introduction

Automobile development has prompted the construction of infrastructure necessary to provide more easily drivable roads; thereby resulting into the road network system. The development of road networks has resulted into intersections (nodes) of the road, i.e. where particular roads (branches) connect. It is from these nodes that a distributed kind of system is realized and therefore particular branches can be easily controlled from the intersection (nodes). These kinds of road networks are commonly found in major urban areas. This has opened up the economy and the job market as people now have a means to travel to work, hence the travelling salesman. This also has allowed mass distribution of manufactured items to consumers. Chung and Huang (2007), observes that with the growing number of vehicles, the traffic congestion and transportation delay on urban arterials are increasingly worldwide. Therefore, it is of practical importance to develop, verify and validate simple, yet powerful models that help in design and improve the safety and efficiency of transportation. Several researchers have therefore focussed their attention on use of emerging technologies to ensure efficient and coordinated movement of traffic on the roads.

### 2.1 Development of Traffic Flow Control Methods

According to Britannica encyclopaedia (Jovanis, 2012), traffic congestion was a feature of city life at least as early as Roman times. A basic cause, then as now, was poor city planning, with roads laid out in such a way as to bring traffic from all quarters to a central crossing point. In the 1$^{st}$ century BC Julius Caesar banned wheeled traffic from Rome during the daytime, a measure gradually extended to cities in the provinces. Late in the 1$^{st}$ century AD, the emperor Hadrian was forced to limit the total number of carts entering Rome. From then, there has been need to improve safety and efficiency of transportation. Traffic control has been an ever evolving and complex issue in the world over. However, the efficiency of transportation has improved since the introduction of traffic lights in major urban areas. According to Lämmer and Helbing (2008), one grand challenge in this connection is the

optimization of traffic lights in urban road networks, especially the coordination of vehicle flows and traffic lights. A typical goal is still to minimize travel times, to find optimal cycle times and to study the corresponding spatio-temporal patterns of traffic flows. Panait and Luke (2004) states that in recent years, there has been increased interest in decentralized approaches to solving complex real-world problems. Many such approaches fall into the area of distributed systems, where a number of entities work together to cooperatively solve problems. The ever-increasing capacity and flexibility of emerging technologies has made it possible to create cooperative agent systems and reduce infrastructural investment, operational costs and accidents while striving to make transportation efficient.

### 2.1.1. Agents in Traffic Flow Control

According to Wooldridge (2000), an agent is an encapsulated computational (or physical, even human) system, that is situated in some environment, and that is capable of flexible, autonomous behaviour in order to meet its design objective. These agents can be organised to form a multi-agent system. Multi-agent system is therefore a decentralized computational (software) system, often distributed (or at least open to distribution across hardware platforms) whose behaviour is defined and implemented by means of complex, peer-to- peer interaction among autonomous, rational and deliberative unit agents (Wooldridge, 2000). Improving traffic flow is very important, as the urban traffic has shown that how congestion is managed, can either slow or accelerate the economic output of a population. However, Bakker et al., (2010) observes that improving traffic control is difficult because the traffic system, when it is modelled with some degree of realism, is a complex, nonlinear system with large state and action spaces, in which substandard control actions can easily lead to congestions that spread quickly and that are hard to dissolve. This therefore called for the intelligent approaches towards management of traffic in major areas, since traffic control is fundamentally a problem of sequential decision-making, and at the same time, is a complex task that is challenging to the straightforward approaches such as the pre-timed interval traffic lights.

### 2.1.2. Agent Technology

Agent technology has been the subject of extensive discussion and investigation within the scientific community for several years, but it is perhaps only recently that it has seen any significant degree of exploitation in commercial applications (Bellifemine et al., 2007). According to Pěchouček et al. (2010), agent-based computing is a subfield of computer science and artificial intelligence that studies the concepts of autonomy of individual computational processes (running either software applications or hardware robots), and inter-action between such heterogeneous autonomous processes. As the society continues to face

complex problems, there continue to be thoughts of how solutions can be found. Therefore, agent technology is an approach that is widely being studied to offer solutions in the face of complexities.

Intelligent agents possess the ability to decompose and solve problems in a collaborative manner. They observe their environment, reason about their own and other agent's actions, interact with other agents, and execute their actions concurrently with other agents. Interactions may communicate facts or beliefs via an agent communication language and may depend on ontology to reach a common understanding of a situation.

Agent-based computing can also support peer-to-peer knowledge and data sharing in social networks, by allowing intelligent agents to negotiate various data confidentiality policies, and thus support wider exploitation of knowledge and experience. Adoption of agent-based techniques in these domains not only facilitates control of distributed systems, but also provides scalability, robustness and reliability, which cannot be achieved by centralized control (Pěchouček et al., 2010).

#### 2.1.2.1. Characteristics of Agents

Agents normally have a set of features. The main features of agents include the following (Pěchouček et al., 2010) and (Gensereth and Ketchpel, 1994) cited in (Li et al., 2006):

- ✓ *Autonomy*: the agent is accountable for execution of its own actions and is not controlled from outside.
- ✓ *Reactivity*: the agent is able to react quickly to the events in the environment and to the requests from other agents; it is able to reconsider her activity according to the change of the environment in timely fashion.
- ✓ *Intentionality*: the agent is able to maintain her long-term intention encoded by the agent's designer and is capable to consider both the long-term intentions and immediate reactive inputs when selecting the next action.
- ✓ *Social capability*: the agent is able to interact, collaborate and form teams. It is also able to perform different levels of reasoning about the other agents.
- ✓ *Proactiveness*: Agents do not only act in response to their environment, but also exhibit goal-directed behaviour by taking the initiative.

#### 2.1.2.2. Multi Agents Systems (MAS)

Although a single agent can act and perform a set of tasks; the increasing interconnection and networking of computers is making such situations rare. A distributed computing model can be set to have a set of agents interact and share information. Multi Agent System is used to describe several agents, which interact with each other. According to Pěchouček et al., (2010), the concept of agents use is considered at the following design levels:

- ✓ *Organization-level*: related to the agent communities as a whole (organizational structure, trust, norms, obligations, self-organisation, etc.);
- ✓ *Interaction-level*: concern communication among agents (languages, interaction protocols, negotiations, resource allocation mechanisms);
- ✓ *Agent-level*: concern individual agents (agent architecture, reasoning, learning, local processing of social knowledge)

Agents and multi-agent systems will be one of the landmark technologies in computer science of the years to come. It will bring extra conceptual power, new methods and techniques that will essentially broaden the spectrum of our computer applications because this new paradigm shifts from the single intelligent entity model to the multi-intelligent entity one, which is in fact the true model of human intelligence acting.

#### 2.1.2.3. Characteristics of Multi Agent Systems (MAS)

The following are the major characteristics of MAS, (Frayret and Santa-Eulalia,, 2004):

- ✓ Each agent has limited capacities and information of problems resolution;
- ✓ Each one has a partial point of view;
- ✓ The MASs have no global control;
- ✓ All data are decentralized;
- ✓ All calculations are asynchronous.

#### 2.1.2.4. The role of Multi Agent Systems (MAS) in traffic light control

Vlassis (2007) identifies some of the benefits of using MAS technology in large software systems as:

- ✓ *Speed and efficiency*: due to the parallel and asynchronous computation, provided that the overhead of coordination does not offset the gain.

- ✓ *Robustness and reliability*: in the sense that the whole system cannot be seriously affected when one or more agents fail. The available agents can take their part.

- ✓ *Scalability and flexibility*: It is easy to add new agents to the system, hence can make it possible to manage a large area of problem.

- ✓ *Cost*: It is more cost effective compared to a whole centralised system since it is a subsystem of low cost units.

- ✓ *Development and reusability*: specialists can develop agents separately. The beauty of this is that you can replace or add any one component without affecting the rest of the system. Therefore, the overall system can be maintained and tested easily. It is also possible to reuse the functionally specific agents in other fitting engagements in the system.

### 2.2 Various Classifications of Traffic Lights Controllers

According to Katwijk (2008), Traffic controllers can be classified according to the method in which they allocate green time for each phase and can be roughly grouped as:

- ✓ *Fixed-time control*: A signal timing plan is selected according to a fixed schedule (e.g., time-of-day, day-of-week) from a set of predetermined plans, which were developed offline based on historical traffic data. The duration and order of all green phases remain fixed and are not adapted to fluctuations in traffic demand.

- ✓ *Actuated control*: In order to adapt the control scheme to fluctuations in traffic demand, traffic detectors are placed that indicate the presence or absence of vehicles. Using this information green phases are extended or terminated depending on the current traffic demands.

- ✓ *Adaptive control*: It continuously optimizes the signal plan according to the actual traffic load. Changes to the active signal plan parameters are automatically implemented in response to the current traffic demand as measured by a vehicle detection system

Pre-timed signals can provide fairly efficient operation during peak traffic periods, assuming signal-timing settings reflect current conditions. However, this is not always the case, as the vehicle pile-up differs as per various roads joining an intersection. In addition, during off-peak times, particularly at night, traffic on the major roadways are often stopping for no reason because of little or no traffic or pedestrians on the cross streets.

Actuated signal control differs from pre-timed in that it requires "actuation" by a vehicle or pedestrian in order for certain phases or traffic movements to be serviced. Actuation is achieved by vehicle detection devices and pedestrian push buttons. The most common method of detecting vehicles is to install inductive loop wires in the pavements at or near the stop signs. Video detection is also used at select locations in the globe. Actuated signals consist of three types: semi-actuated, fully-actuated and volume-density.

According to Shenoda (2006), research has delved deeply into these two methods with two main concerns:

**i.** To provide general improvements to the methods in order to enhance their overall performance

**ii.** To assess their application in different situations, i.e. to determine which of the two methods is best suited to a particular intersection or network of intersections and how the chosen method can best be applied

Shenoda (2006) continues to note that Adaptive traffic signal control is a relatively new method; research began in the 1970's, has only recently been increasing, and even now, implementation is very sparse in North America. However, if more concentration can be given to adaptive control, it can solve the two problems above. It has the potential to diminish the need for constant adjustments to enhance performance, which is the concern of (**I**), and can replace both methods, since the signal can be programmed to act as one of the two or as its own signal control paradigm, which is the concern of (**II**). Vehicle detection devices have also proved to be so expensive to install and maintain, hence the need for a cheaper and reliable technology, which at the same time improves performance at the intersections over some period.

It is from this concept of adaptive control that some researchers have looked into the idea of intelligent controls of intersections using agents. The adaptive control illuminate the fundamental concept of adjustments always required at the intersections every time. Therefore, the concept of intelligent control of traffic in relation to the varying demands at the intersections has been, and continues to be studied and cited widely;

Hirankitti et al. (2007) studied the concept of multi agent for intelligent traffic light control. According to their approach, their system consisted of agents and their world. The world consisted of cars, road networks, traffic lights. Each of the agents controlled all traffic lights at one road junction by an observe-think-act cycle. That is, each agent repeatedly observed the current traffic condition surrounding its junction. Then uses that

information to reason with condition-action rules to determine how the agent can efficiently control the traffic flows at its junction, or collaborate with neighbouring agents so that they can efficiently control the traffic flows, at their junctions, in such a way that would affect the traffic flows at its junction. Their research demonstrated that a rather complicated problem of traffic-light control on a large road network could be solved elegantly by the rule-based multi-agent approach.

Apart from Vehicle Actuated Signal Control that Hirankitti et al. (2007) cite in the study, they also mention the following:

- ✓ First, Mikami *et al* (1994) and Ying *et al* (2003) are cited by Hirankitti et al. (2007) to have introduced machine-learning methods, such as reinforcement learning and genetics algorithm, to learn traffic patterns of different time in a day and used them to control the traffic lights. However, he confirms that this seemed feasible when all the commuters behave normally which is not possible in real life.

- ✓ Secondly, Hirankitti et al. (2007) cites some works that are based on a multi-agent approach. For example, (Hirankitti et al., 2007)Xiong *et al* (2004) adopted case-based reasoning to control traffic lights. The agent observed traffic condition at a junction and used this information to match with candidate cases from its previous case-base; consequently, it applied the solution of the selected case to control the traffic lights. This approach is similar to a rule-based approach which Hirankitti et al. (2007) investigated in his work. An agent proposed by Enrique *et al* (2001), used some properties of the current states of all traffic patterns as the criterion to determine what will be the next traffic-light pattern.

However, their study sought to compare the performance of the traffic-light control by collaborative agents with that of the traffic-light control by individual agents without collaboration.

Tubaishat et al. (2007) proposed a decentralized traffic light control using wireless sensor network. They classified their system architecture into three layers; the wireless sensor network, the localized traffic flow model policy, and the higher-level coordination of the traffic lights agents. An intersection control agent in the vicinity controlled traffic lights. An intersection agent coordinates four traffic lights at a time. At each traffic light, there are three nodes with magnetometer sensors. These sensors multi-hop to the access point

in its location, lane number, and number of vehicles passed within *t* time. The sensor nodes are positioned at *d* distance from the traffic light to allow for enough time for the data to multi-hop to the intersection agent analysed and then send to the targeted traffic light.

They created a protocol by which the agents can communicate the bare minimum of information necessary to function appropriately. Their protocol consists of several message types for each agent, as well as some rules governing when the messages should be sent and what sorts of guarantees accompany them.

a) *Sensor Nodes to Intersection Control Agent* (*ICA)*: Sensor nodes count number of vehicles approaching an intersection. Every node monitors one lane. The message sent from the sensor nodes to the intersection control agent includes number of vehicles, time duration of the collected data, and lane number.

b) *Intersection Control Agent (ICA) to Sensor Nodes*: After receiving information from all the nodes monitoring a specific intersection, the intersection agent decides the best flow model (policy) for the vehicle flow.

c) *Greedy Intersection Control Agent (ICA) to Intersection Control Agent (ICA)*: Intersection control agent can exchange information with other intersection control agents in its vicinity to improve the flow of vehicles in a wider area. This is because the agent can select a better policy depending on more information collected. They called this situation greedy policy because each agent satisfies its intersection flow without paying attention of other intersections flow.

d) *Intersection Control Agent (ICA) to Intersection Control Agent (ICA) with Coordination*: This is the same as the previous one except that the agents coordinate among themselves to achieve even better flow. The intersection control depends not only on the analysis of one agent but also on the coordination of multiple agents.

Their main concern and contribution was the real-time adaptive control of the traffic lights. Their aim was to maximize the flow of vehicles and reduce the waiting time while maintaining fairness among the other traffic lights. They tested their model using Green Light District Simulator (GLD). GLD is a Java based traffic simulator that allows the design of arbitrary roads and intersections, monitor traffic flow statistics and test different traffic light controllers.

Ibrahim and Taha (2012) proposed a fuzzy based traffic simulation system. The proposed fuzzy engine was built and divided into four classes (Fuzzy_engine, Fuzzy_rules, Linq_var, Member_fun). In this model, once the user finished the first step of intersection design, the user then connects the design with the fuzzy engine. The fuzzy engine rules could be designed mannualy by the user or generated automatically for all selected combinations of selected variables and subsequently, the user filling of consequent fuzzy terms.

### 2.3 Traffic Simulation Platforms

Over the years, several platforms have been developed to help researchers simulate their traffic optimization designs. Among these platforms, include **TransModeler Traffic Simulation Software**, which is a powerful and versatile traffic simulation package applicable to a wide array of traffic planning and modeling tasks. TransModeler can simulate all kinds of road networks, from freeways to downtown areas, and can analyse wide area multimodal networks in great detail and with high fidelity (Caliper Corporation, 2014). However, there needs to a researcher to experience the flexibility of a simulator by developing the system himself/herself, hence the main reason not use this kind of simulator in this research.

### 2.4 The Gap

The review carried in this section shows considerable insufficiencies on the already existing techniques and technologies in use, the key ones that stand out include:

- ✓ Flexibility and reliability: - fixed time control and actuated control are not flexible and reliable under dynamic environments.
- ✓ Fault Tolerance and robustness: - most of the reviewed technologies (especially fixed time control) are not fault tolerant and cannot be scaled under dynamic environments.
- ✓ While Tubaishat et al. (2007) and Hirankitti et al. (2007) have done their researches, their models address only the roads with long queues, thereby exposing other motorists on roads with shorter queues to longer waiting time. They are mum on how such a problem can be addressed.
- ✓ While Ibrahim and Taha (2012) proposed a fuzzy logic based approach, it was left to the user to decide the rules and subsequent arrangement of the interfaces as befitting the user's design. Hence no affirmative approach affirmed.

The advantages of using multi agents in the area of traffic light management as earlier outlined, together with the key common deficiencies of the reviewed technologies and studies forms the basis and motivation of carrying out the study. Our system therefore, introduces agents in adaptive control, based on changing situations on the road. The agents influence traffic light controllers based on the load (queue length and wait time) on each road.

## 2.5 Chapter Summary

In this chapter, we have reviewed all the technologies relevant to the study, including but not limited to; traffic lights concept, agents and their role in traffic light control, work already done in the research area relevant to the study, and a gap analysis, which would later be used to come up with the conceptual model.

## CHAPTER THREE

### 3. METHODOLOGY

This chapter discusses the methodology to be used to complete the proposed project. A methodology is a process followed when solving a problem. In multi agent systems development, there are several set of methodologies followed. According to Giorgini and Henderson-Sellers (2005), they include Tropos, MAS-CommonKADS, PASSI, Prometheus, Gaia, ADELFE, MESSAGE, INGENIAS, RAP, and MaSE among others, which are still in development. All these methodologies provide valuable and unique approaches, perspectives and contributions to aid in the development of agent based applications. Therefore, this project is to be based on Prometheus, which is one of the Agent Development Methodologies. According to Padgham and Winikoff (2004), the methodology has been developed over the last several years in collaboration with Agent Oriented Software (OSA). Prometheus is detailed and complete in the sense of covering all activities required in developing intelligent agent systems. The main reason for using Prometheus in this project is that it differs from existing methodologies in the following ways. It:

- ✓ Supports the development of intelligent agents, which use goals, beliefs, plans, and events. By contrast, many other methodologies treat agents as "simple software processes that interact with each other to meet an overall system goal.
- ✓ Provides 'start -to- end support' (specification to detailed design and implementation) and a detailed process.
- ✓ Provides hierarchical structuring mechanisms, which allow design to be performed at multiple levels of abstraction. Such mechanisms are crucial to the practicality of the methodology on large designs.
- ✓ Uses an Iterative process rather than a linear 'waterfall model', hence can involve a mixture of activities from different phases with a changing focus.

Therefore, this methodology would ensure that the project objectives are achieved.

Padgham and Winikoff (2004) continues to describe the Prometheus methodology in detail as follows:

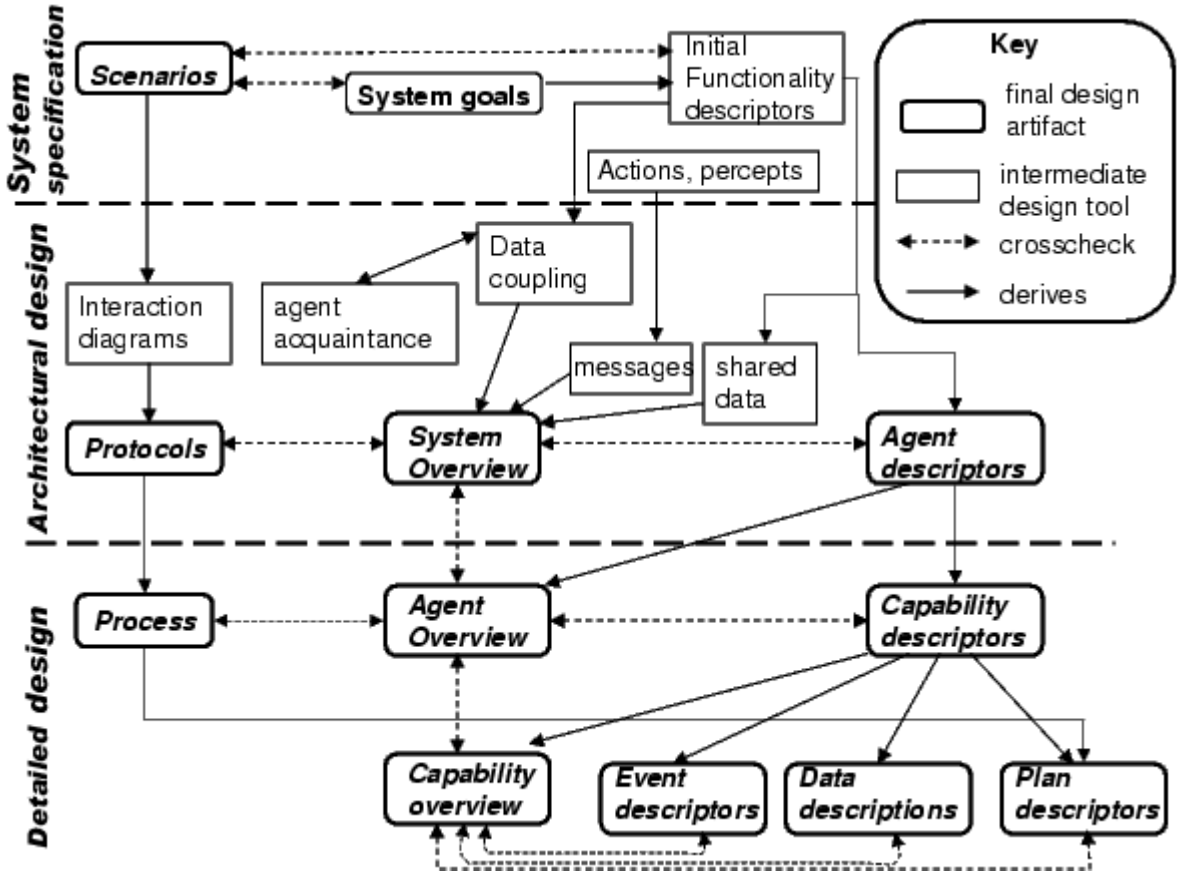


**Figure 3: The Prometheus Methodology (Padgham and Winikoff, 2004)**

The models and phases are described below:

It consists of three phases:

### 3.0 System Specification Phase

It focuses on identifying the basic functionalities of the system, along with inputs (percepts), outputs (actions). It builds the system's environment model, identifies the goals and describes key use case scenarios. Therefore, we will undertake the below steps to develop this model:

#### 3.0.1 Use Case Scenario

In this step, we will derive the functional description of the system composed of a series of use case diagram. We will give a brief overview explaining who is using the system and what they are trying to accomplish from the system. There will be a description of incoming percepts, messages sent and the actions taken.

### 3.0.2 System goals

In this step, System goals are identified mainly based upon the requirements specification of the system. Goals will be decomposed into sub-goals, where necessary. After that, system functionalities that achieve these goals will be defined using a detailed diagram. In so doing, each goal defines the objective of the system.

## 3.1 Architectural Design Phase

It uses the outputs from the previous phase to determine the agent types, design the overall system structure, and define the interaction between agents. Under this phase, we have the following steps:

#### 3.1.1. **Agent Descriptors**

The agents will be determined using the functionalities of the system. I.e. functionalities that are related to each other (e.g. using the same data) and possibly the interactions between two functionalities.

#### 3.1.2. **System Overview**

An illustration of the agent types and the communication links between them, data used, system's boundary and environment in terms of actions, percepts and external will be designed. I.e. This stage will provide a general scenario of how the system as a whole will function.

#### 3.1.3. **Protocols**

Interaction protocols will define the intended valid sequence of messages between agents. They will be developed, showing the alternatives at each point of interaction in a diagram.

### 3.2 Detailed Design Phase

This is the final phase of Prometheus. It looks at the internals of each agent, and how it will accomplish its tasks within the overall system. This stage emphasises on defining capabilities, internal events, plans and detailed data structure for each agent type defined in the previous step. Under this phase, we have the following three main steps:

#### 3.2.1. **Capabilities descriptors**

The information on which events to be generated, which events to be received, details involving interactions with other capabilities and references to data read and written by the capability will be defined under this step with an aid of a diagram. In addition, at a lower level of detail, there are other types of descriptors: individual plan descriptors, event descriptors, and data descriptors. These descriptors provide the details so that they can be used in the implementation phase on a need basis.

#### 3.2.2. **Agent overview**

This gives the top-level view of each agent's internals. We will use conventional diagram to show the capabilities of the agent, the flow of tasks between these capabilities and data internal to the agent.

#### 3.2.3. **Process**

An illustration of the links and interaction between the Architectural phase and the detailed design phase.

### 3.3 Code model

This is a model of the solution at the code level and will contain the source code of the target system.

### 3.4 Verification, Testing and Results Justification

In this phase, we shall discuss system agent's verification and validation testing. Finally, we shall generate data based on the test scenarios. The testing activity has been divided into two different steps:

a) The first test is devoted to verifying the pre-timed traffic lights behaviour with regards to the vehicular density on roads in Nairobi (Kenyatta Avenue Intersection and Haille Sellasie Avenue Intersection),

b) The second test, is devoted to verifying the agent controlled traffic lights behaviour with regards to the vehicular density on roads in Nairobi (Kenyatta Avenue Intersection and Haille Sellasie Avenue Intersection),

In all the cases, the results will be studied to verify the performances of each scenario.

### 3.5 Chapter Summary

In this chapter a description is provided of how we will use the Prometheus agent based methodology to model our proposed agent based solution and provide away through which we will do our testing.

## CHAPTER FOUR

### 4. SYSTEMS ANALYSIS, DESIGN AND IMPLEMENTATION

In this section, we describe how we have used the Prometheus Methodology from system specification analysis to the implementation our Multi Agent System. This has been achieved through the construction of seven models obtained by performing twelve sequential and iterative activities, which are elaborately discussed from the section below; the conceptual model for the proposed solution is as below:

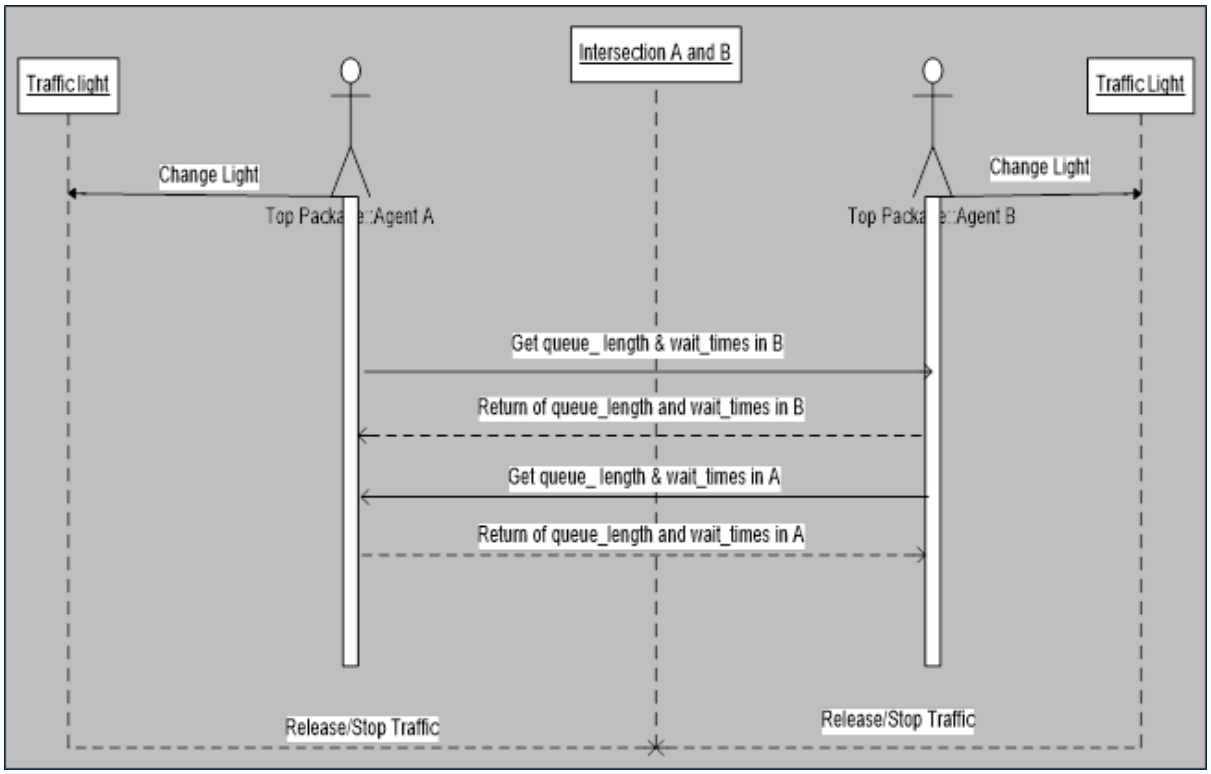


Figure 4: Conceptual Model of the System

The Model comprises of the following components:

- ✓ Agent A and B: Responsible for monitoring the queue length and waiting time of the roads joining the intersection; Share traffic information; change traffic lights.
- ✓ Intersections A and B: Consists of traffic lights
- ✓ Traffic lights: Emits Green/Red/Yellow lights to give direction to motorists

### 4.0 Systems Specification Model

This phase comprises of four different activities and produces a description of the functionalities required from the system and fragmentation of them according to the agent pattern. The different activities are outlined as below:

### 4.0.1. Use Case Description

In this case, we illustrate the functionality of the solution using conventional use-case diagram. Figure. 5 depicts the relationship between these agents.

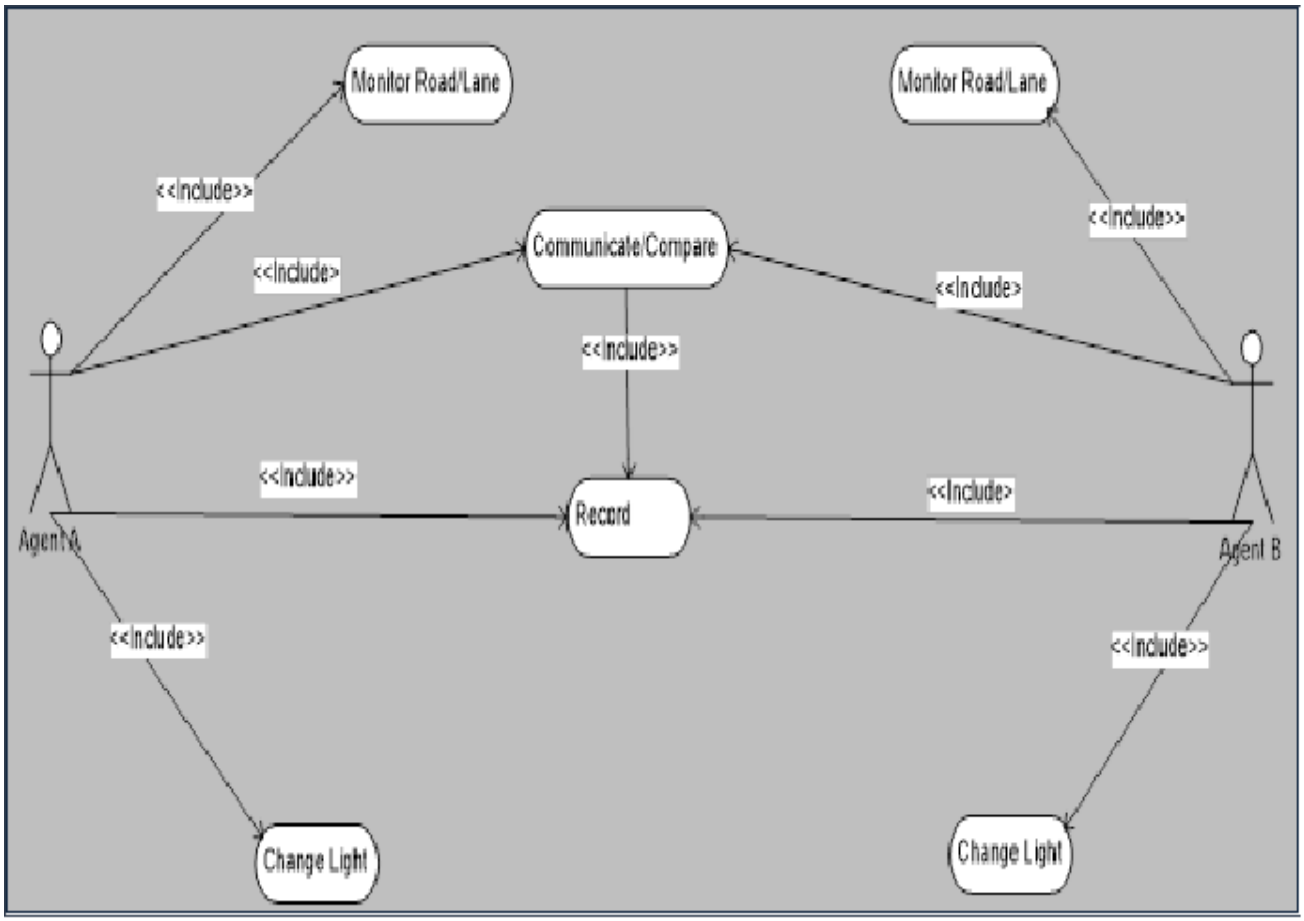


**Figure 5: Use Case Description Diagram**

### 4.0.2. System goals

System goals starts from the use-case diagram of the previous step. The goals of the system at the intersections is as illustrated in the figure. 6 below:

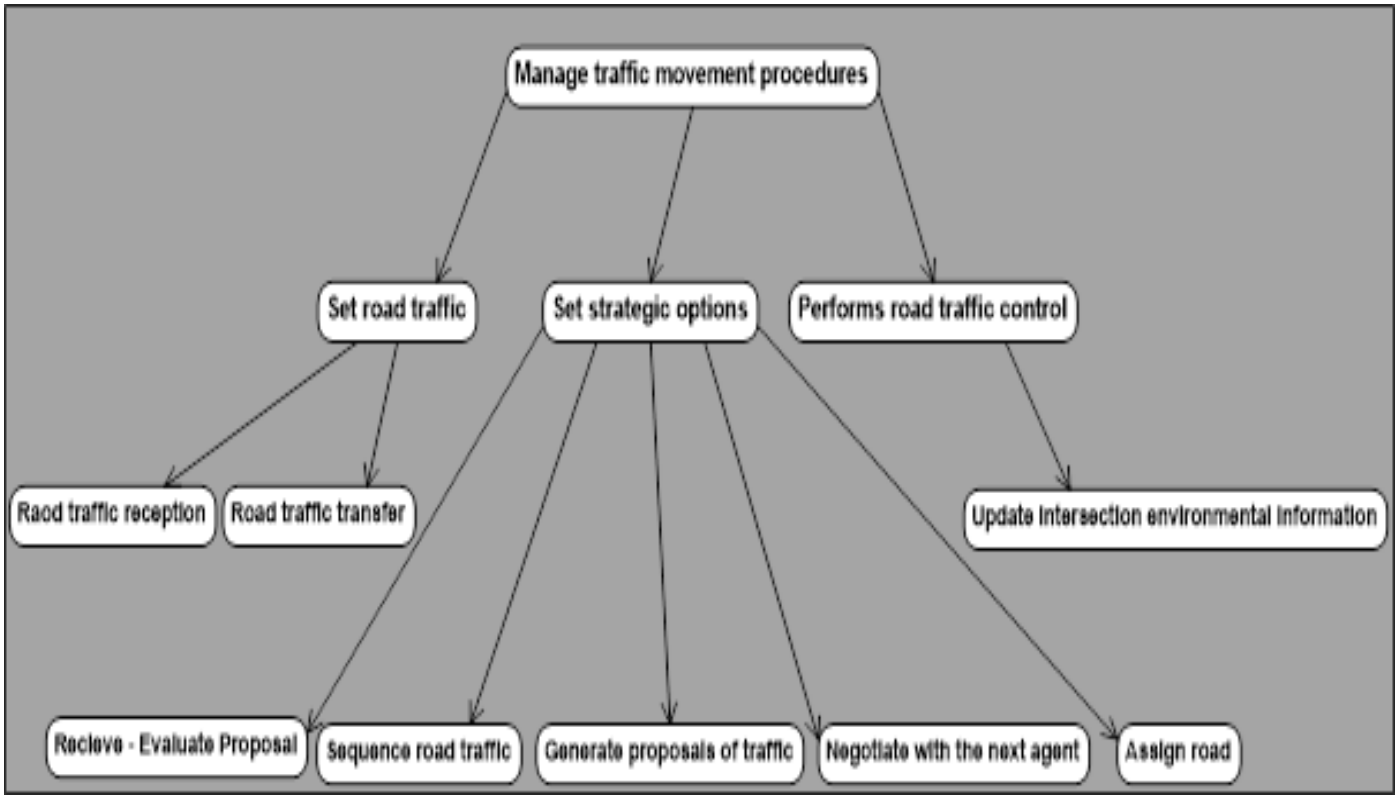


**Figure 6: System goals at the intersection**

## 4.1 Architectural Design Model

In this step, we are to determine the various functional areas of the road intersection and then decide the importance of the relationships between each function as shown in figure. 7 below:

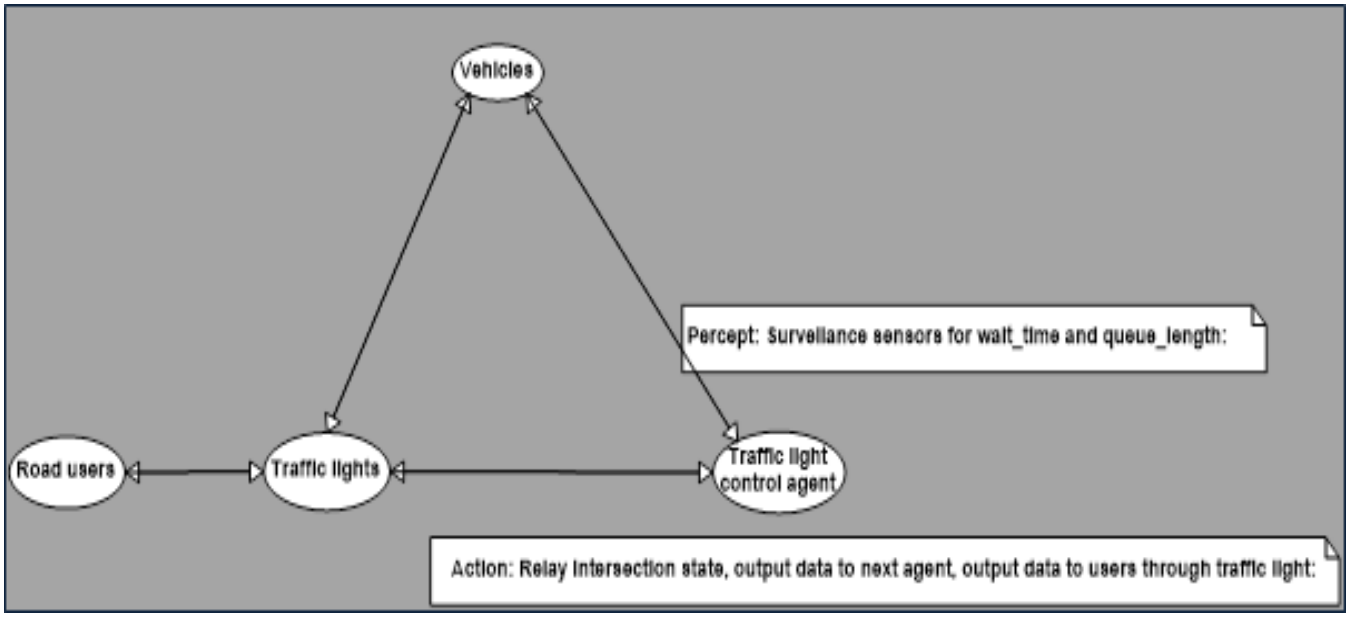


**Figure 7: System - User interaction**

### 4.1.0 Protocols

Figure. 8 below presents a communication protocol between the agents. It presents a planning for each agent to evaluate proposals and generate counter-proposals.

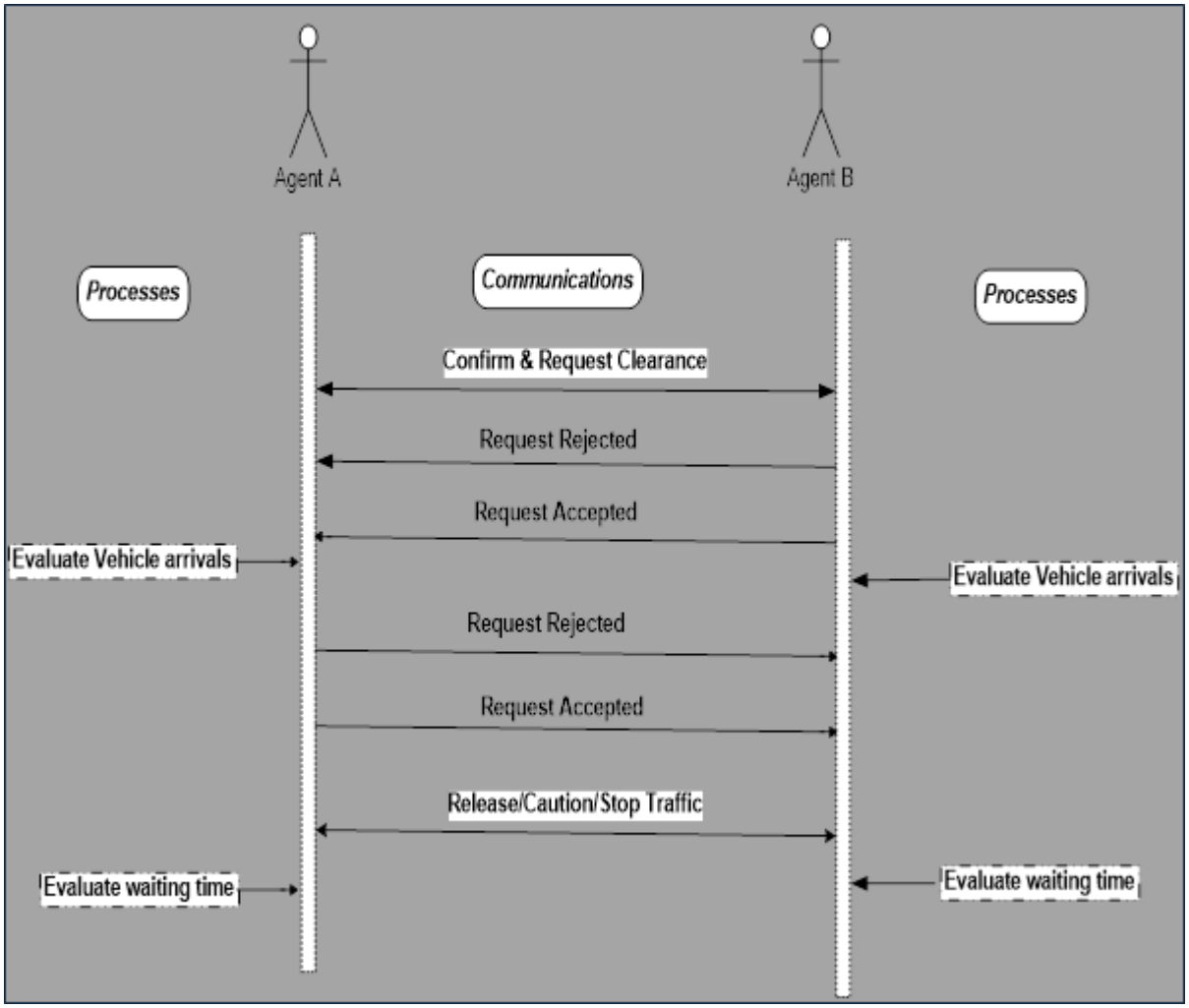


**Figure 8: Agent Communication**

#### 4.1.0.1. Protocol description

As specified by the FIPA architecture, an Agent Interaction Protocol is used for each communication. In our case, all of them are FIPA standard protocols (FIPA, 2005).

## 4.2 Detailed Design Phase

The detailed design phases focuses on the internal structure of each agent and the implementation of the functionalities contained. The figure below shows the events i.e. the requests/proposals it gets from the adjacent agent, evaluation of the request/proposal viz-a-viz monitory of its events. The arrows signify the incoming and outgoing nature of events. In this model, the agent has a capability for each of its main activities. Events are actions (affecting the environment in some way i.e. accept/reject), percepts (knowledge

coming from the environment i.e. queue_length and wait_time), and messages (to and from other agent i.e. proposal/request).

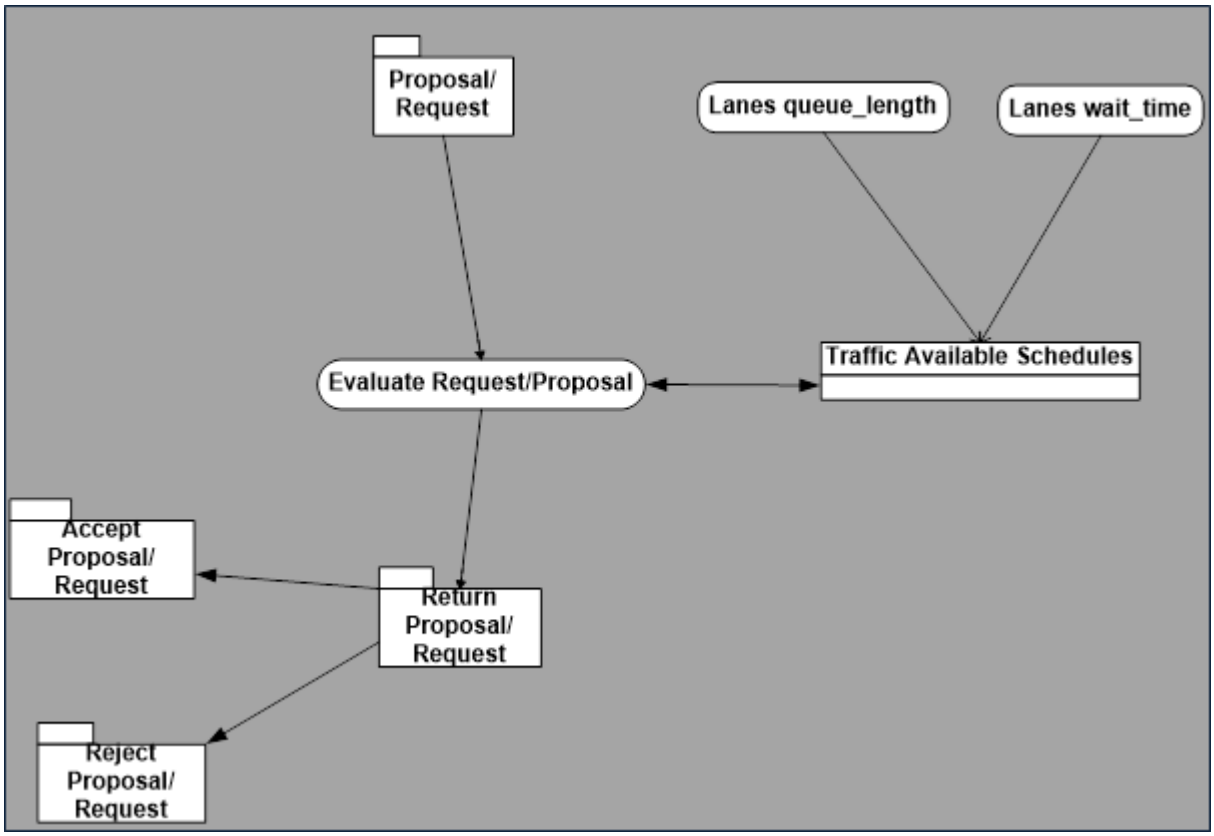


**Figure 9: Agent internal communication**

Agents processes like the ones illustrated on the above diagram can be implemented by means of plans. Plans contain a set of instructions in order to take decisions, generate/receive messages and new events. Furthermore, plans are triggered by events such as arriving messages or events generated by other plans. An agent overview then shows interaction between plans, shared data and events. Main capabilities of these agents are depicted together with data used or produced, agent inner events and communication messages. The processes, events and data have been grouped into the following capabilities:

- *Traffic monitoring*: This capability checks the state of traffic, i.e. queue length and wait time
- *Manage traffic environmental information*: This capability is associated with goal of maintaining up-to-date information on traffic movement. Plans for this capability capture information about stopped roads, arrival and exit rates of vehicles.

## 4.3 Implementation

The executable model consists of entities that have been developed in the detailed design phase (i.e. agents, capabilities, plans, data, events and message). For code generation, we have decided to use JADE platform because it is one of the most extended multi-agent platforms, provides FIPA standards infrastructure for inter-agent communications and for managing distributed agents software. JADE agents have been built based on behaviours rather than plan-based design.

### 4.3.1. Road Traffic model implementation under JADE platform

Figure 10. below shows an adaptation of the behaviour based model on JADE platform. Each of the agent capabilities have been defined as behaviours. We also describe the main algorithms used in design.

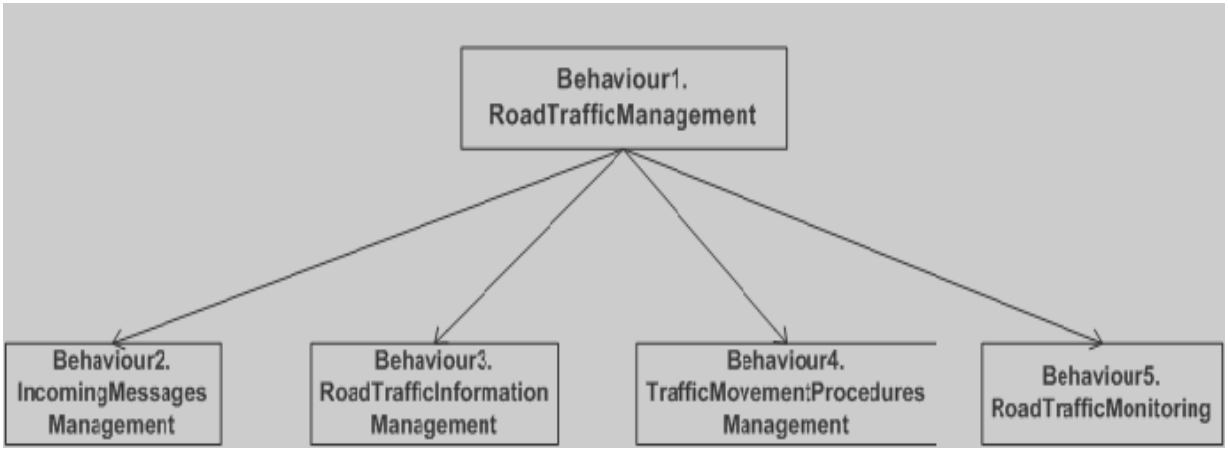


**Figure 10: Agent behaviours based on JADE**

### 4.3.2. Algorithm implementation of the traffic model illustrating the processes

For functionality, the main algorithms implemented in the solution includes:

- Monitor algorithm
- Compare algorithm
- Send message algorithm

### 4.4 Code model

See the appendix for sample code.

### 4.5 System verification, testing and results justification

- **System verification and testing** - In this case, the verification and testing have been carried out as follows:
    - The first test is the verification of pre-timed traffic lights in regard to:
        - The number of vehicles processed through the intersections at specific set times.
        - The wait time of vehicles at the intersections based on the intersections throughput.
    - The second test is the verification of the agent-based controlled traffic lights in regard to:
        - The number of vehicles processed through the intersection at specific set times.
        - The wait time of vehicles at the intersections based on the intersections throughput.
- **Results justification** - In this case, the figures below shows the performance of each system. The figures below illustrate the sample traffic network behaviour at the intersections in the two scenarios.
    - Figure 11 – 13 shows various vehicle movements as controlled by the traffic lights and witnessed on the lanes of the pre-timed control system.
    - Figure 14 – 16 shows various vehicle movements as controlled by the traffic lights and witnessed on the lanes of the agent-based control system.

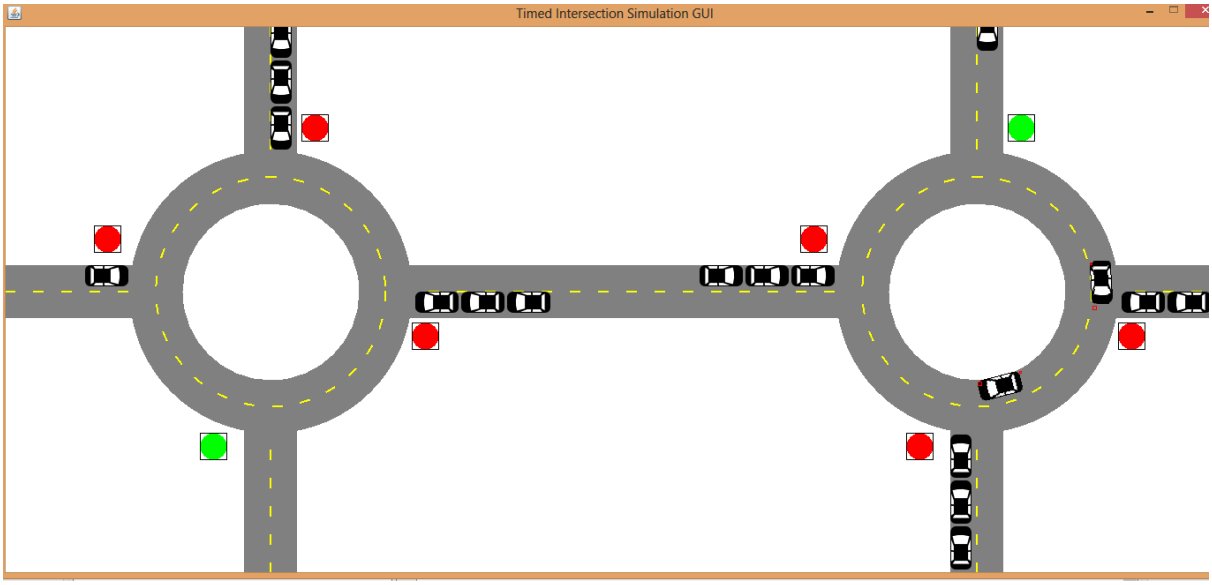


**Figure 11: Pre-timed controlled traffic lights intersection screenshot1.**

From Figure 11, two lanes allows the vehicles to move, yet the traffic is clear. The blocked lanes have traffic on them, yet cannot move.

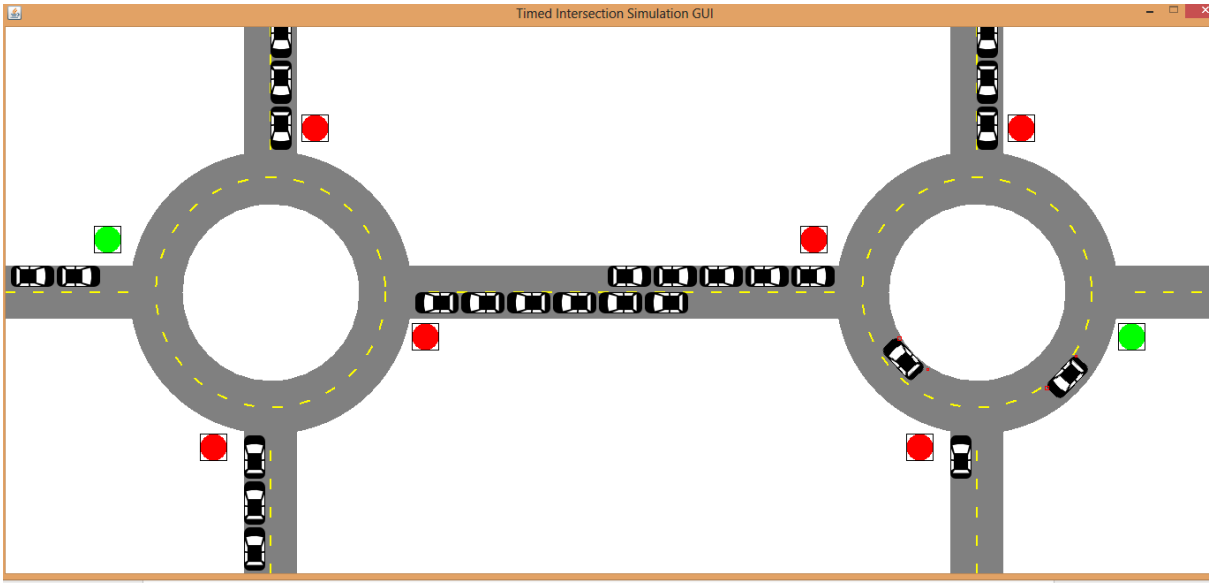


**Figure 12:Pre-timed controlled traffic lights intersection screenshot2.**

From Figure 12, one lane has allowed the vehicles to move, yet the traffic is clear. The traffic is continuously building on blocked lanes, yet cannot move.

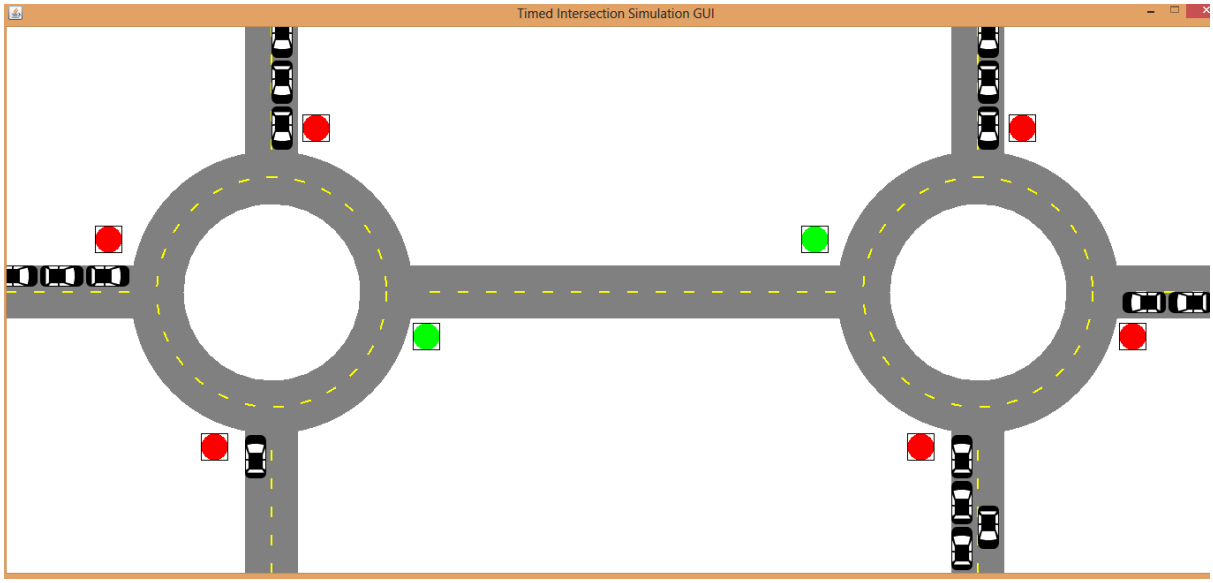


**Figure 13: Pre-timed controlled traffic lights intersection screenshot3.**

From Figure 13, the middle lanes traffics are clear, yet the system indicates that they still have the priority. The blocked lanes have traffic on them, yet cannot move.

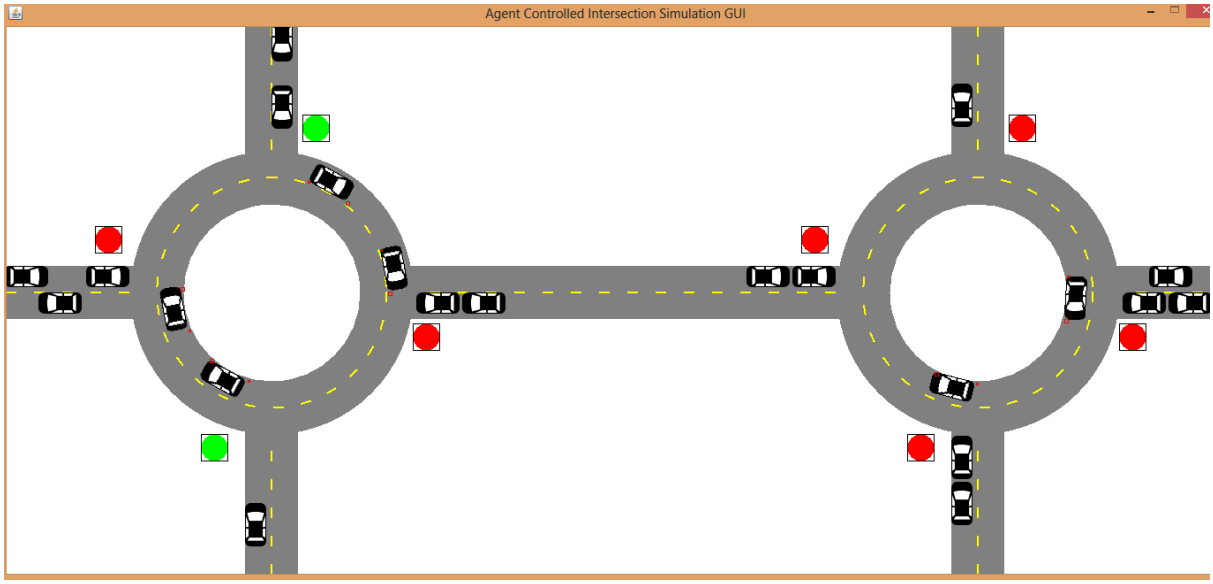


**Figure 14: Agent-Based traffic lights controlled intersection screenshot1**

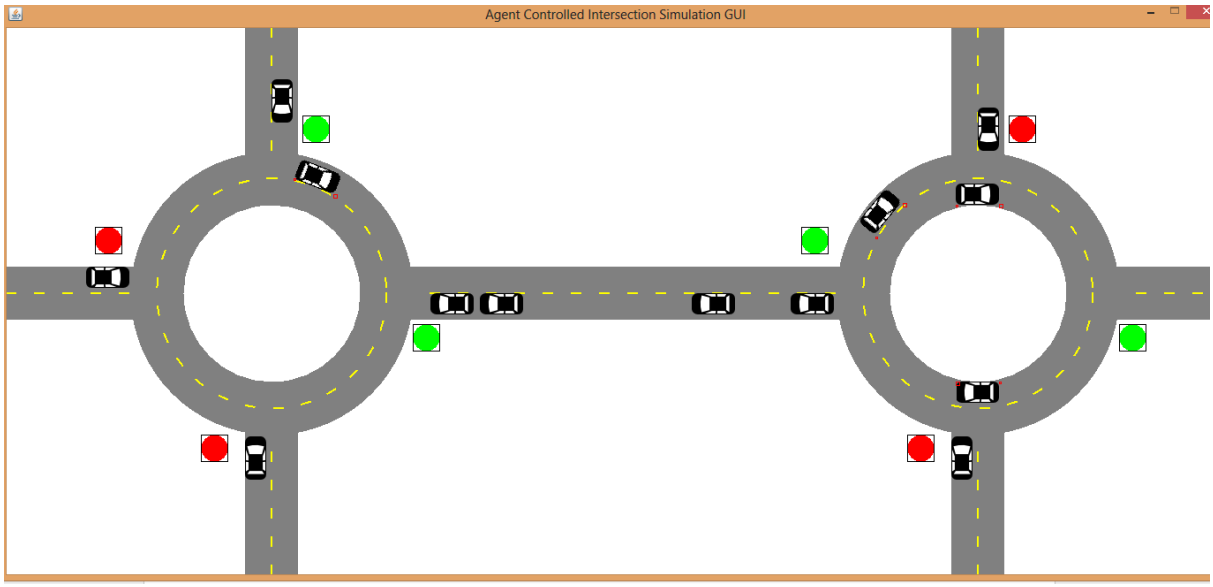


**Figure 15: Agent-Based traffic lights controlled intersection screenshot2**

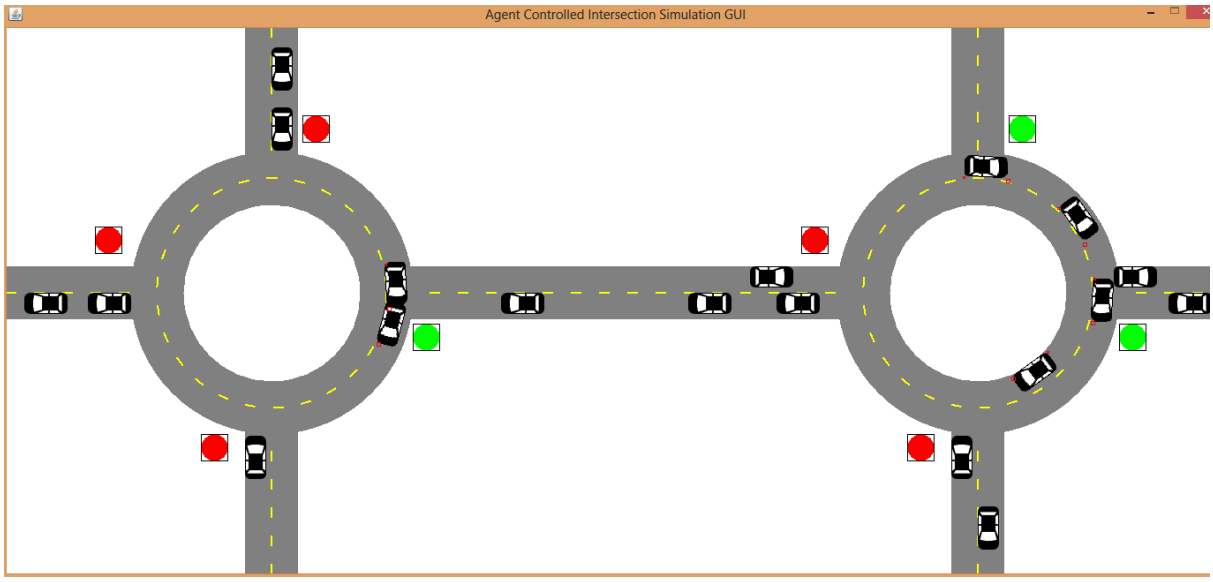


**Figure 16: Agent-Based traffic lights controlled intersection screenshot3**

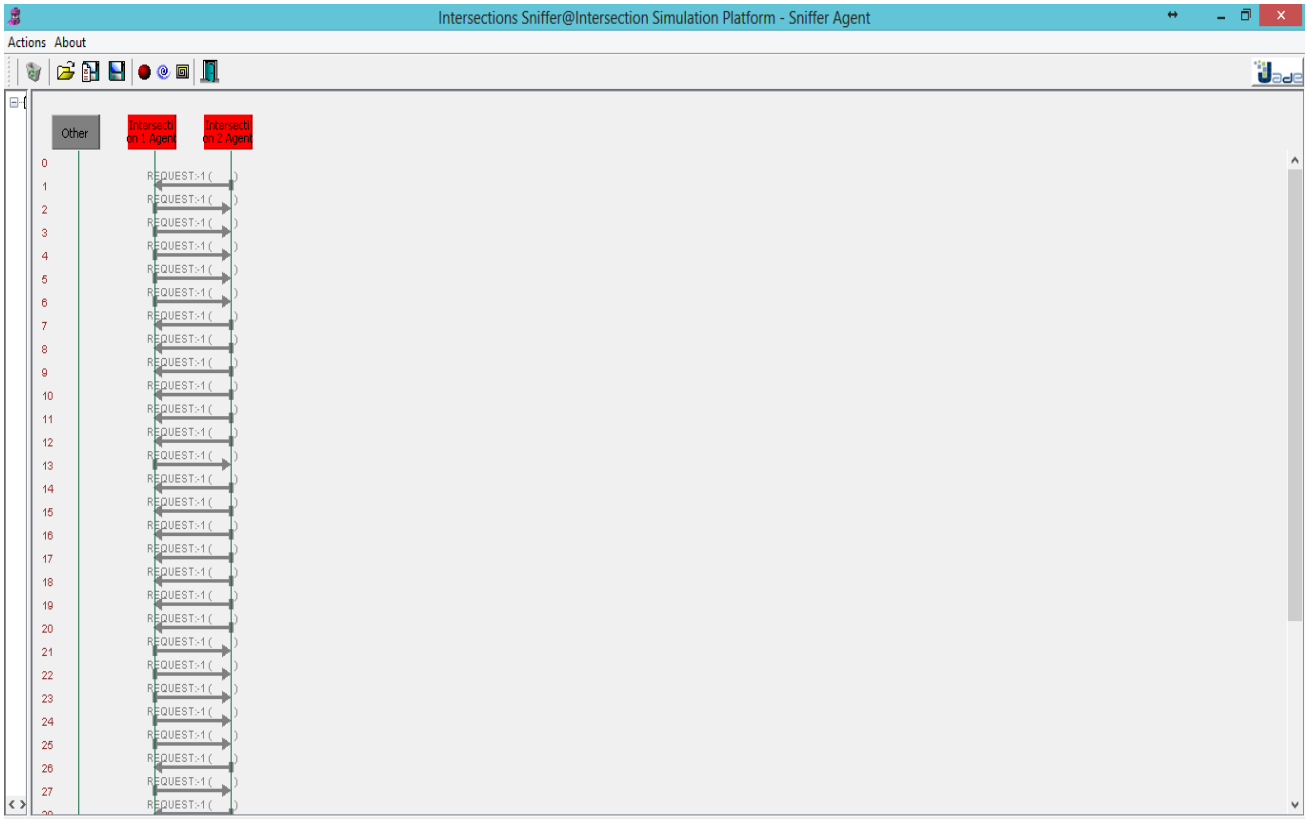


**Figure 17: Agents communication interface**

### 4.5.1 Test environment

The test environment consisted of the following:

- Software consisting of:
  - Java environment (Created by NetBeans Version 8.0)
  - JADE Version 4.3.1
  - Windows 8
- Hardware consisting of:
  - Laptop (all of the above software installed)

### 4.5.2 *Test case1 (Pre-timed controlled traffic lights testing)*

**Monitoring queue length (Red light and green lights are pre-timed)**

**Input**: Number of vehicles arriving in random number seeds for a period of 16 minutes.

**Test item**: The overall number of vehicles processed at the intersections by the pre-timed lights after every 2 minutes up to the $16^{th}$ minute.

**Output**: Graphical and table outputs of the performance of the system as shown in figures 18 to 25 and tables 1 and 2 below.

### 4.5.3 *Test case2 (Multi-Agent based traffic light control testing)*

**Monitoring queue length (Wait times are agent controlled)**

**Input**: Number of vehicles arriving in random number seeds for a period of 15 minutes.

**Test item**: The overall number of vehicles processed at the intersections by the agent controlled system of lights after every 2 minutes up to the $16^{th}$ minute.

**Output**: Graphical and table outputs of the performance of the system as shown in figure 18 to 25 and tables 1 and 2 below.

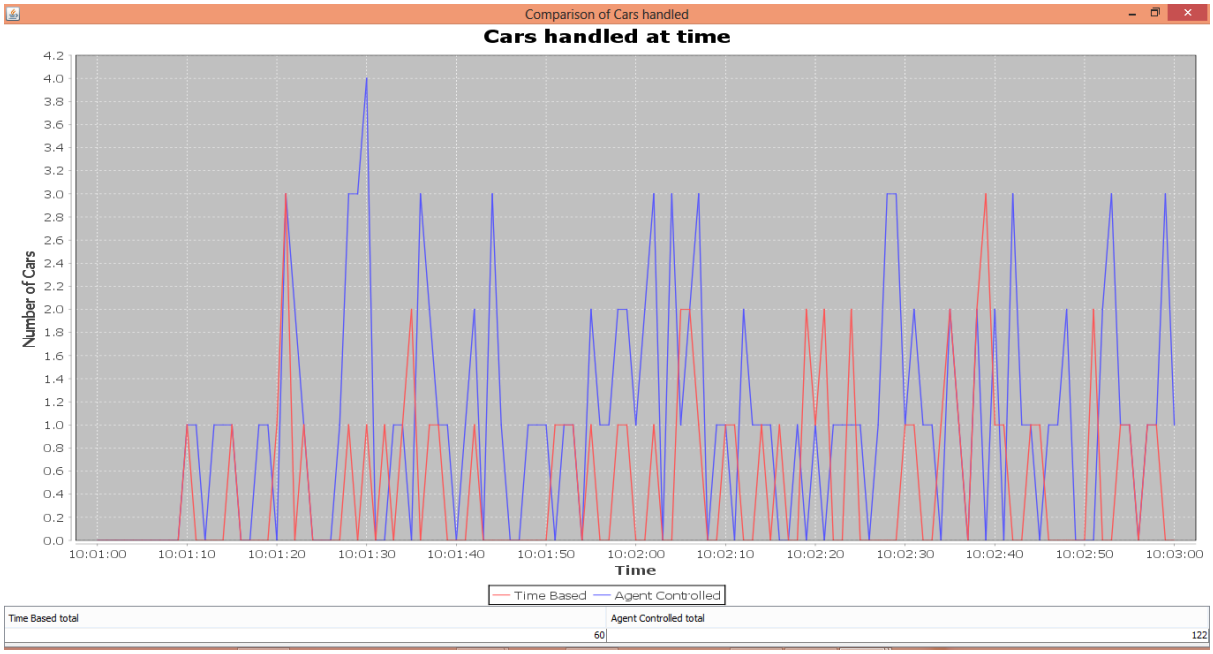


**Figure 18: Overall number of cars handled after 2 minutes of simulation**

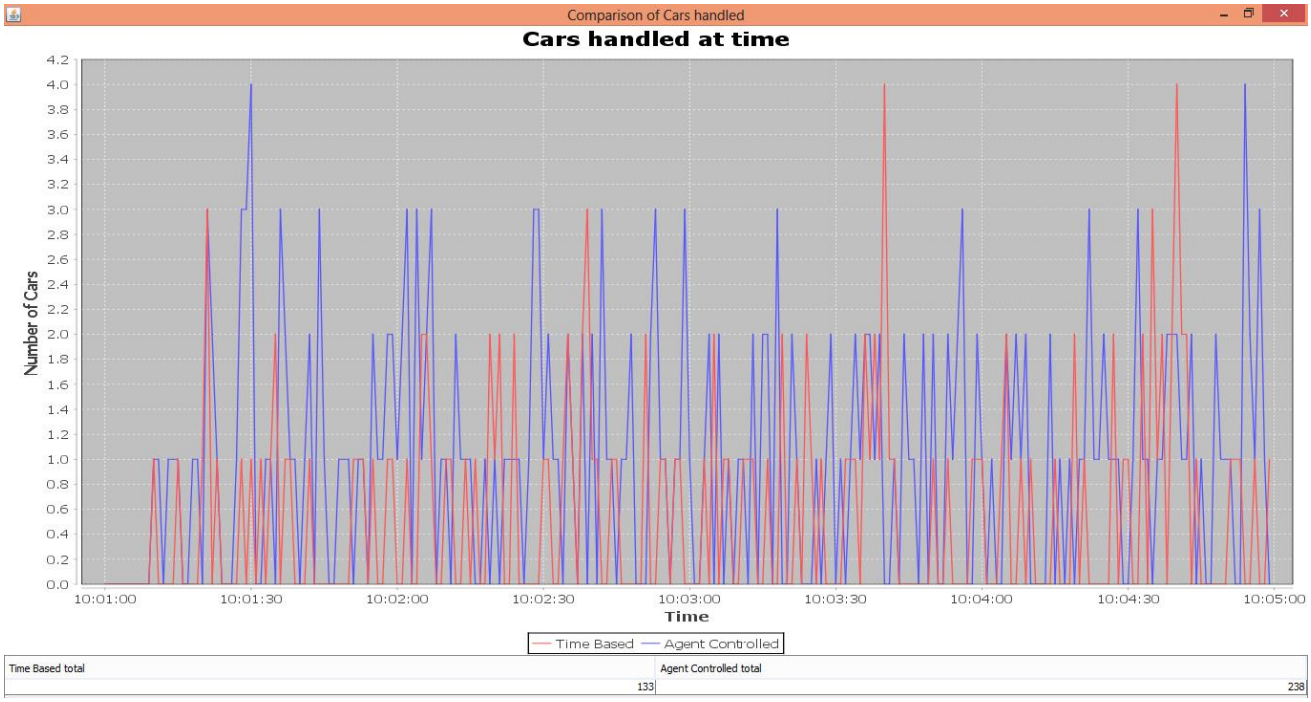


**Figure 19: Overall number of cars handled after 4 minutes of simulation**

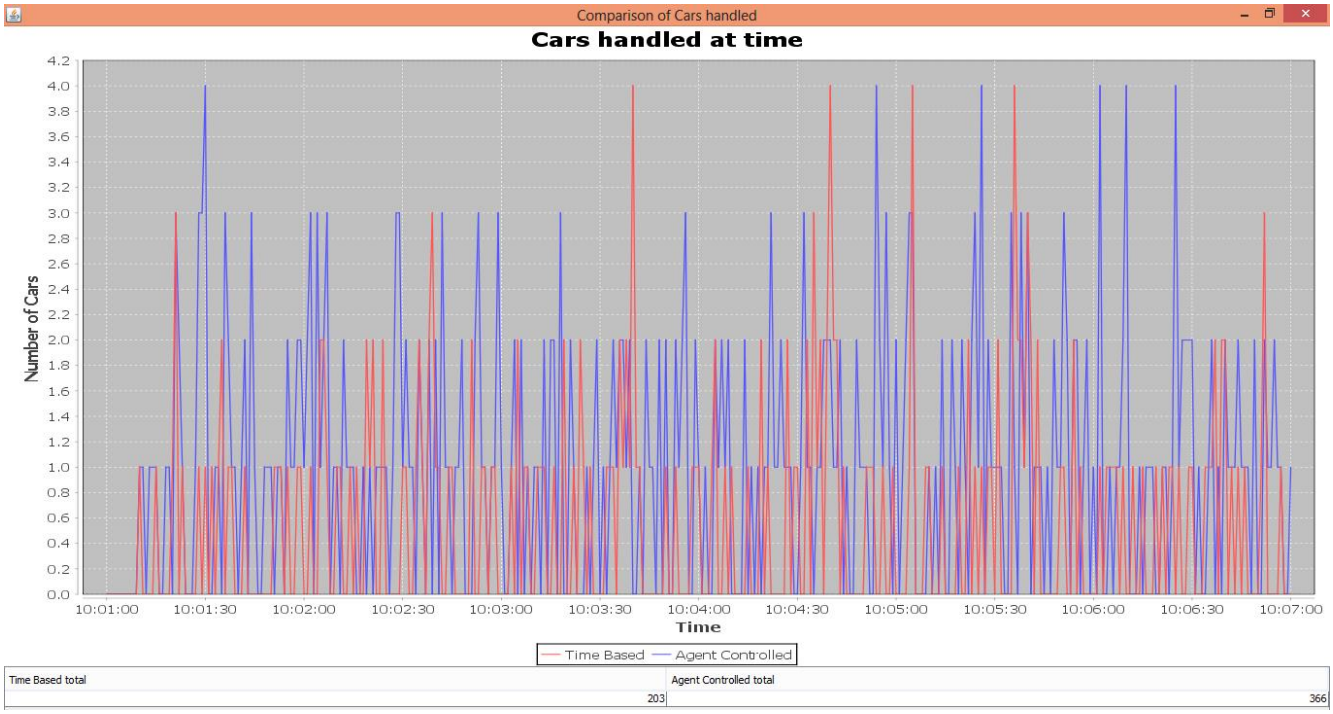


**Figure 20: Overall number of cars handled after 6 minutes of simulation**

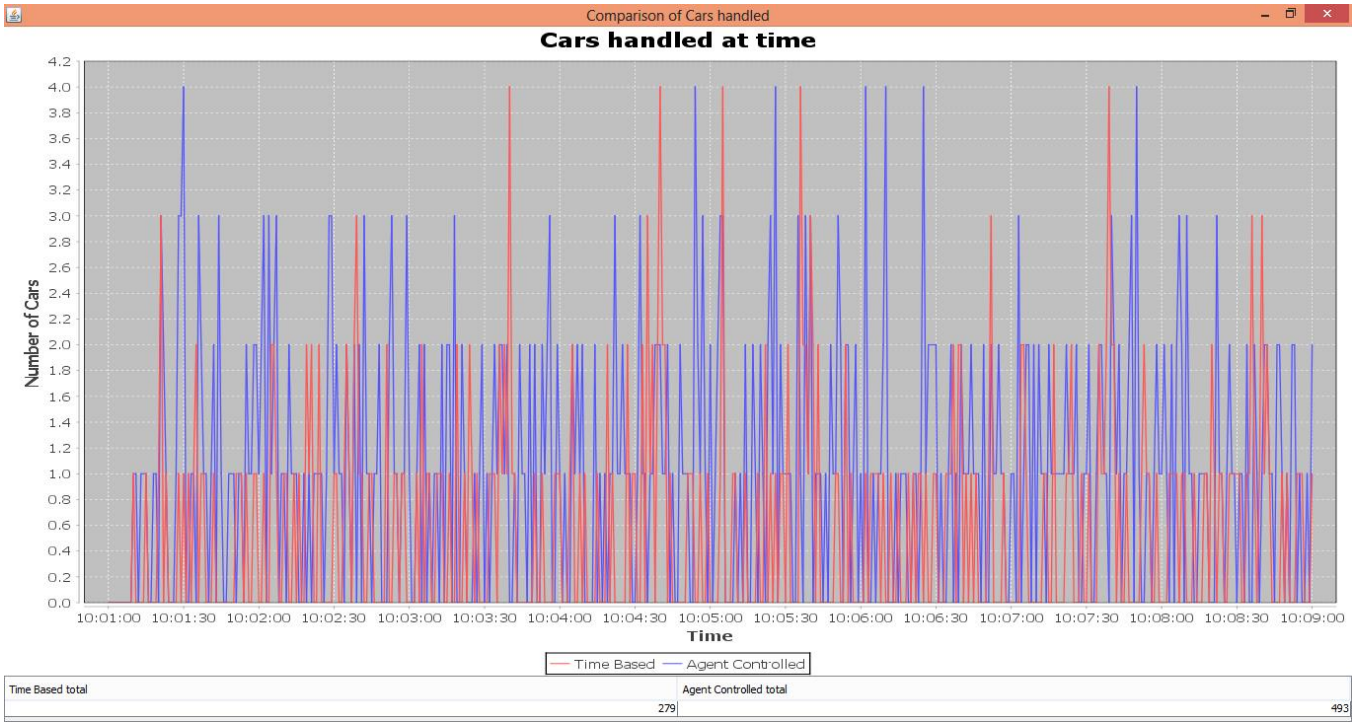


**Figure 21: Overall number of cars handled after 8 minutes of simulation**

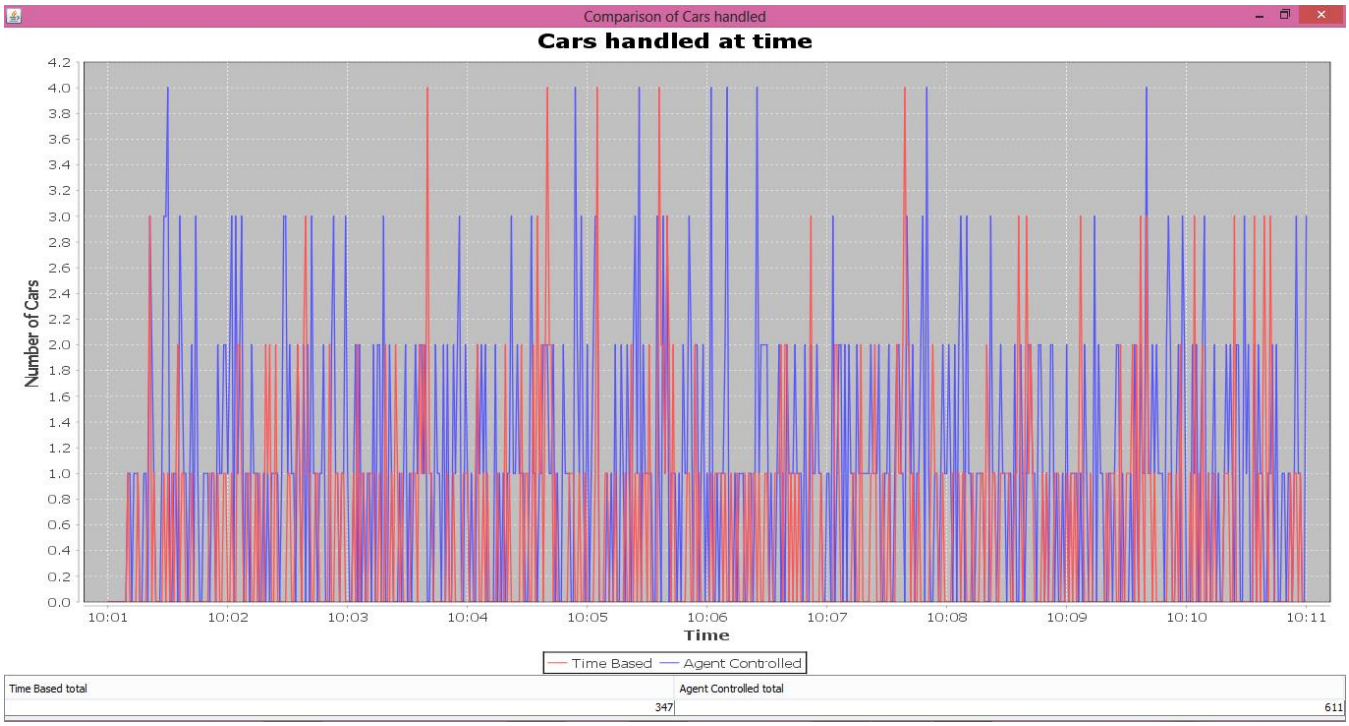


**Figure 22: Overall number of cars handled after 10 minutes of simulation**

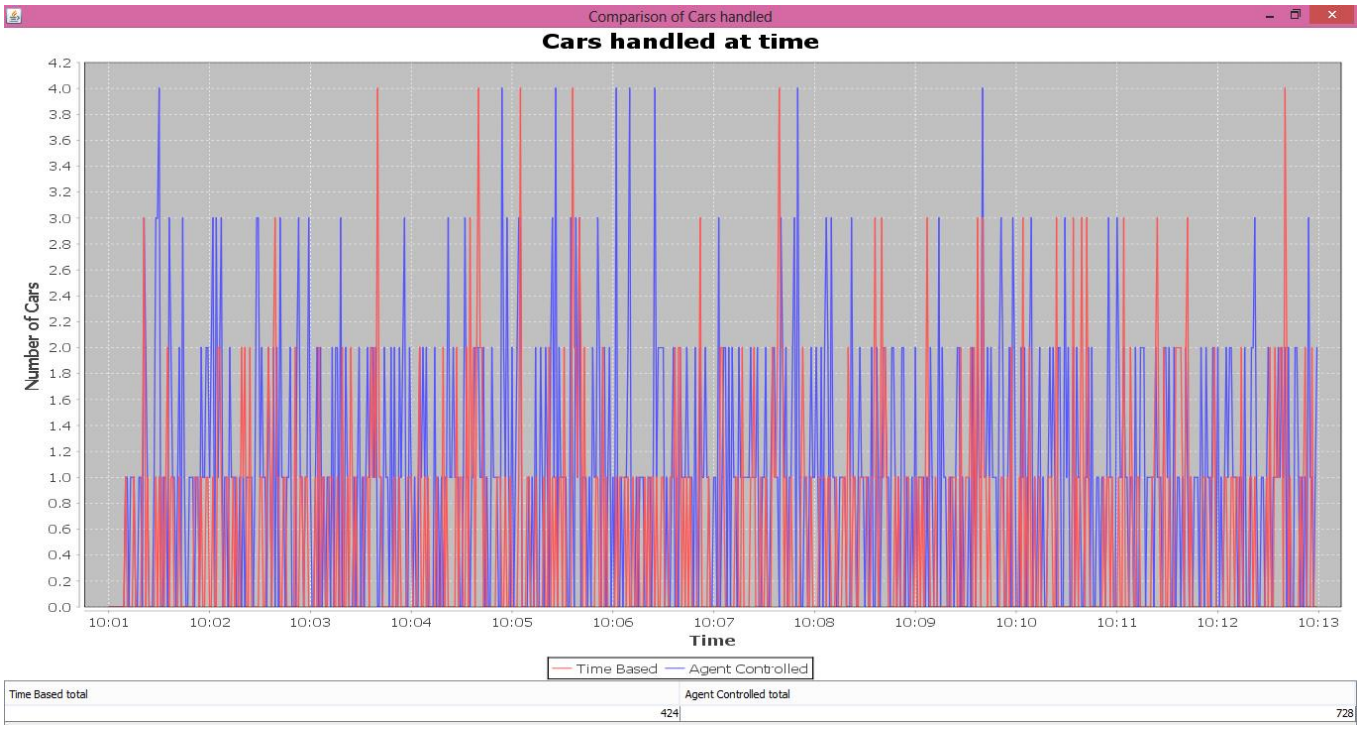


**Figure 23: Overall number of cars handled after 12 minutes of simulation**

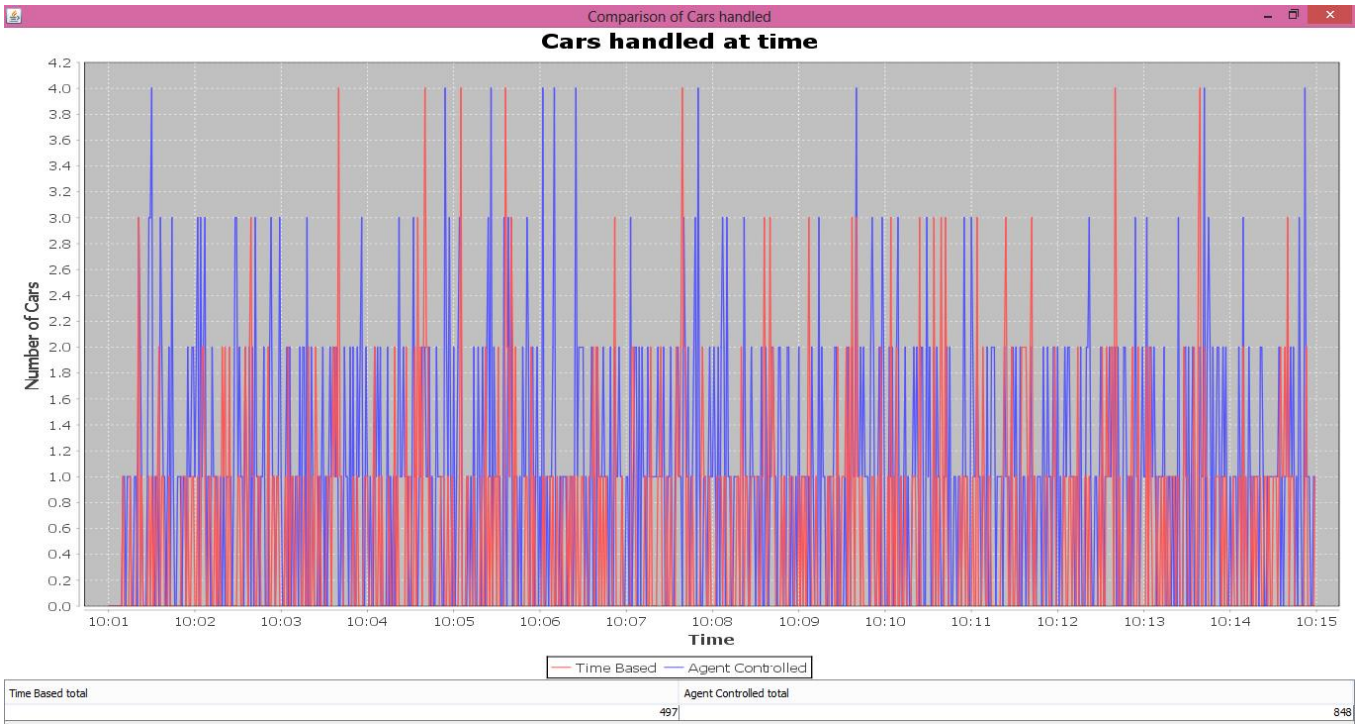


**Figure 24: Overall number of cars handled after 14 minutes of simulation**

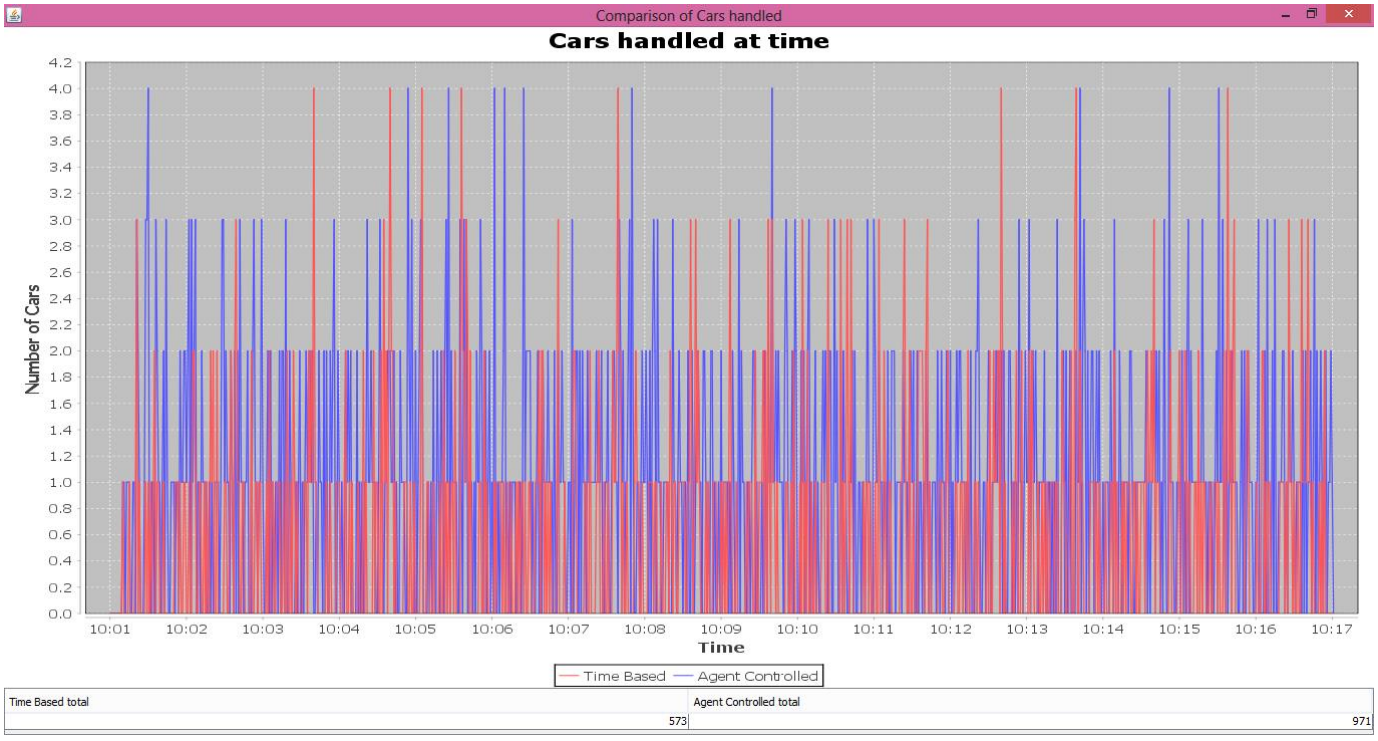


**Figure 25: Overall number of vehicles handled after 16 minutes of simulation**

| Time | Minutes Taken | Pre-timed System Vehicles Output | Agent-based System Vehicles Output |
|---|---|---|---|
| 10:01 | 0 | 0 | 0 |
| 10:03 | 2 | 60 | 122 |
| 10:05 | 4 | 133 | 238 |
| 10:07 | 6 | 203 | 366 |
| 10:09 | 8 | 279 | 493 |
| 10:11 | 10 | 347 | 611 |
| 10:13 | 12 | 424 | 728 |
| 10:15 | 14 | 497 | 843 |
| 10:17 | 16 | 573 | 971 |

**Table 1: Pre-timed and Agent-Based Systems Performances**

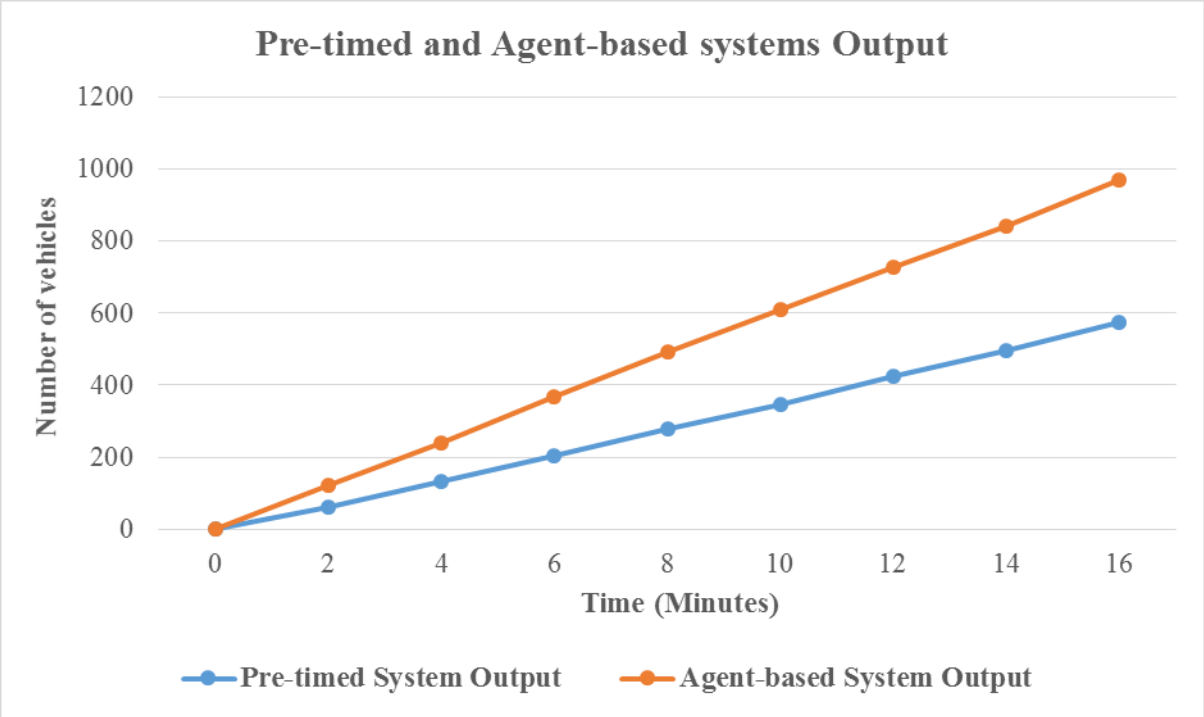


**Table 2: Linear output of the pre-timed and agent-based systems**

### 4.6 Chapter Summary

The Prometheus methodology offers a useful guideline from system requirements analysis, design and coding phase. Thus, we are able to implement and test the results.

## CHAPTER FIVE

### 5. DISCUSSION OF THE RESULTS

This chapter discusses the outcome of the tests carried out. An agent coordinated traffic light system and a pre-timed traffic light system have been developed to map out the movement of the vehicles in relation to the density on various lanes.

#### 5.0 Output summary

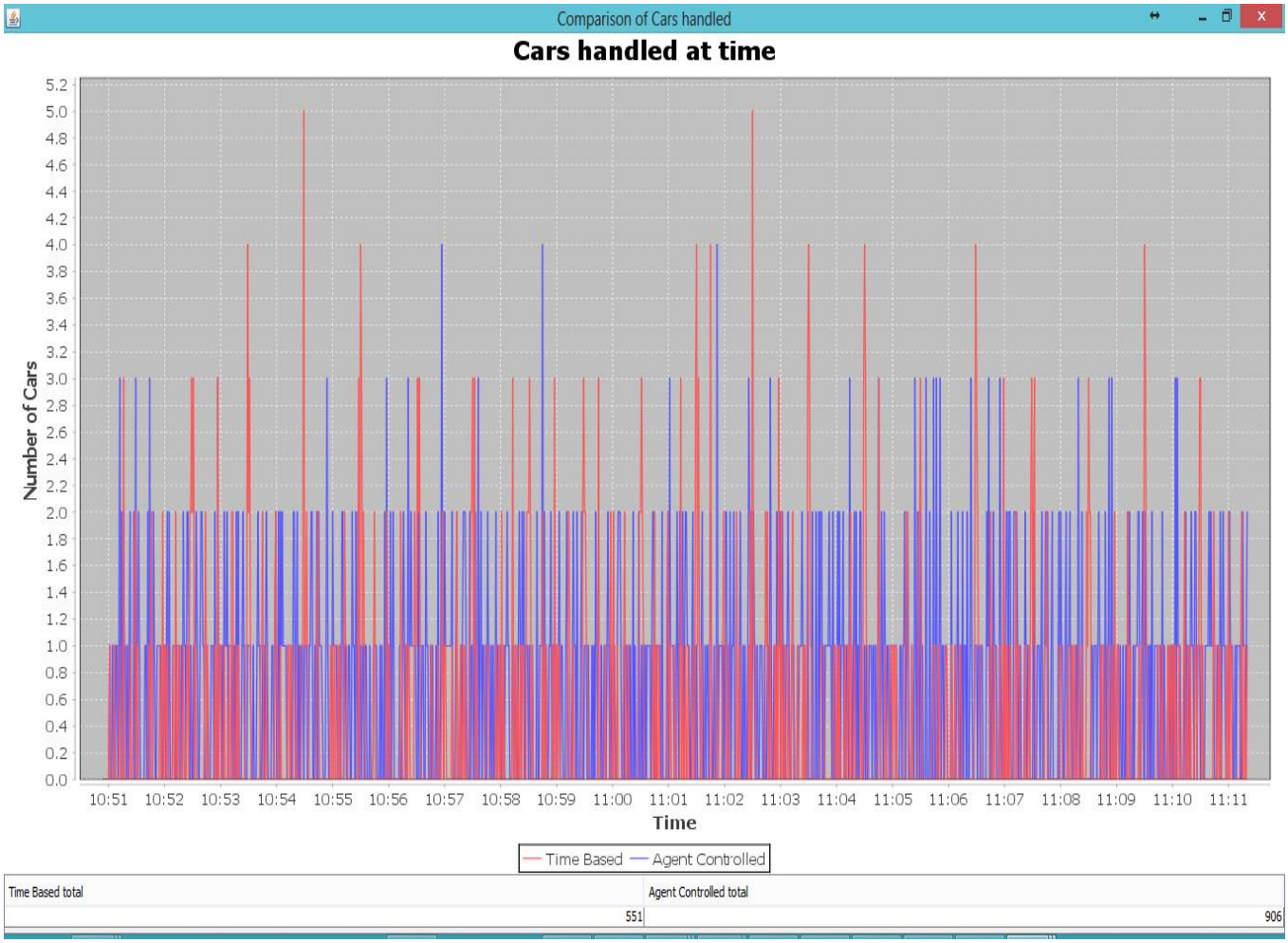


**Figure 26: Number of vehicles handled by the two systems between in 20 minutes**

From the two systems developed, the following observations were made:

i. Vehicles counter at pre-timed system shows smaller values;

ii. Vehicles counter at agent coordinated system shows larger values;

iii. A pre-timed traffic light system is experiencing more notable stops as depicted on the system;

iv. An agent-based system's intersections are continuously experiencing steady movement of vehicles in all lanes, with a few notable stops;

In agent-based control, whenever vehicles density increases in a certain lane, the agent is able to evaluate that lane in terms of wait time and vehicle density with other lanes in the same intersection, liaise with the next agent, and then it handles it based on that evaluation.

These results therefore, presents an averagely higher output on the intersections managed by the agent-based controlled traffic system. The performance of agent-based traffic light control intersection proved to be flowing more efficiently than the pre-timed traffic light intersection.

# CHAPTER SIX

## 6. CONCLUSION

### 6.0 Chapter overview

This chapter presents achievements of the project, challenges, limitations, recommendations and conclusions.

### 6.1 Achievements

In cases of intersection controls, the more complex the system becomes, the more numerous details can be achieved in the model developed. Although the model presented is a simple one, from this research, it is notable how agent technology and related applications are increasingly being applied in complex scenarios to offer solutions.

The project achieved the intended goals, met system requirement specifications and performed effectively as designed. The performance of the prototype was tested as per the output shown at the verification and testing state.

The following specific objectives were achieved:

- We were able to come up with the prototype of a pre-timed traffic light system and measure it's performance;
- We were able to design and implement a multi-agent based traffic light controlled model of a system to provide a platform for improving the performance of traffic lights at the intersections;
- We were able to simulate the effectiveness of an agent-based approach in managing road traffic snarl-ups;
- The results provided, showed the effectiveness of an agent-based approach in managing traffic as opposed to pre-timing in a dynamic situation.

### 6.2 Project challenges

Making the decision on the agent methodology that could best fit the project proved to be a great challenge, since most of them were difficult to understand in one area or the other. Setting up the project designs in line with the Prometheus methodology proved challenging as well. However, this was solved after studying most of the agent methodologies, and getting to know more about Prometheus methodology and its advantages compared to others.

Choosing the platform to use also became a great challenge, with the project requiring a high-level understanding of the functionality of each platform whose tutorials were not readily available and with limited tools. However, JADE platform proved flexible. Moreover, it has third party extensions, including Protégé Ontology environment plug in to allow exportation of Ontologies to be used with JADE agents.

### 6.3 Project limitations

The design is primarily focused on agents interacting between themselves and managing the traffic lights infrastructure, which does not cover the whole scope of Kenyan road models that include traffic police officers, breakdowns on the road leading to more congestions hence, does not reflect the broad picture.

Secondly, the project has assumed tests with the two systems running on a computer, hence does not present the comparative performance of the systems anchored on the real Nairobi environment.

Lastly, the solution implemented is so basic and has not been designed to incorporate mechanisms of fault tolerance in a large network.

### 6.4 Recommendations and future work

From this research, it is observed that it is not obvious to expect effective flow of vehicles at the intersections based on traffic lights only. There need to be effective management of those lights to ensure maximum output. We would like to recommend that more research be done in the field of intersection traffic management in Kenya, to enable the prioritisation of road users. E.g., giving priority to emergency road users like ambulances and allocating time space to pedestrians. The research can also be extended to a wider area, to include more than two intersections in multi-lane road network, to test the scalability of such a system.

I therefore have no doubt that this research has contributed positively towards more absorption and adoption of the agent technology.

It is my hope that when this kind of system is fully developed and implemented; the numerous problems of traffic congestion will be solved, hence furthering the positive uptake of agent technology.

**References**


Bakker, B., Whiteson, S., Kester, L. and Groen, F.C.A. (2010), "Traffic Light Control by Multiagent Reinforcement Learning Systems", *Interactive Collaborative Information Systems*, Springer-Verlag Berlin Heidelberg, p. 36.

Bellifemine, F., Caire, G. and Greenwood, D. (2007), *Developing Multi-Agent Systems with JADE*, John Wiley & Sons Ltd, available at: http://homepages.abdn.ac.uk/w.w.vasconcelos/pages/teaching/CS4027/abdn.only/jade_book.pdf.

Bellifemine, F., Caire, G., Rimassa, G., Poggi, A., Trucco, T., Cortese, E., Quarta, F., et al. (2014), "Java Agent Development Framework", *JADE*, available at: http://jade.tilab.com/.

Caliper Corporation. (2014), "TransModeler Traffic Simulation Software", available at: http://www.caliper.com/TransModeler/.

Chung, T.-H. and Huang, Y.-S. (2007), "Design of a Supervisor for Traffic Light Systems", *Springer Berlin Heidelberg*, pp. 704–711.

FIPA. (2005), "Foundation for Intelligent Physical Agents", available at: http://www.fipa.org/specifications/index.html.

Frayret, J.-M. and Santa-Eulalia,, L.A. (2004), *Basics of Multi-Agent Systems*, available at: http://www.forac.ulaval.ca/fileadmin/docs/EcoleEte/2004/EcoleEte1405200402.pdf.

Giorgini, P. and Henderson-Sellers, B. (2005), "Agent-Oriented Methodologies: An Introduction", Idea Group Inc, available at: http://www.troposproject.org/files/Henderson01.pdf.

Hirankitti, V., Krohkaew, J. and Hogger, C. (2007), "A Multi-Agent Approach for Intelligent Traffic-Light Control", *World Congress on Engineering*, Presented at the WCE 2007, London, U.K, Vol. 1, available at: http://www.iaeng.org/publication/WCE2007/WCE2007_pp116-121.pdf.

Ibrahim, L. and Taha, M.A. (2012), "Traffic Simulation System Based on Fuzzy Logic", *Computer Science 12 ( 2012 ) 356 – 360*, Presented at the Conference Organized by Missouri University of Science and Technology, Elsevier B.V. Selection and/or peer-review, Washington D.C, doi:10.1016/j.procs.2012.09.084.

Jovanis, P.P. (2012), "Traffic control", *Encyclopaedia Britannica*, available at: http://www.britannica.com/EBchecked/topic/601854/traffic-control.

Karsiti, M.N. and Ahmed, S. (Eds.). (2009), *Multiagent Systems*, I-Tech Education and Publishing, available at: http://www.intechopen.com/books/multiagent_systems.

Katwijk, R.T. van. (2008), *Multi-Agent Look-Ahead Traffic-Adaptive Control*, The Netherlands TRAIL Research School, The Netherlands, available at: http://www.dcsc.tudelft.nl/~bdeschutter/research/phd_theses/phd_vankatwijk_2008.pdf.

Lämmer, S. and Helbing, D. (2008), "Self-control of traffic lights and vehicle flows in urban road networks", *Journal of Statistical Mechanics: Theory and Experiment*, Vol. 2008, doi:10.1088/1742-5468/2008/04/P04019.

Li, J., Fuh, J.Y.H., Zhang, Y.F. and Nee, A.Y.C. (2006), *Multi-Agent Based Distributed Manufacturing*, available at: http://www.intechopen.com/books/manufacturing_the_future/multi-agent_based_distributed_manufacturing.

Padgham, L. and Winikoff, M. (2004), "Prometheus: A Methodology for Developing Intelligent Agents", RMIT University, available at: http://www.cs.rmit.edu.au/agents/Papers/aose02.pdf.

Panait, L. and Luke, S. (2004), "Cooperative Multi-Agent Learning: The State of the Art.", George Mason University, available at: http://cs.gmu.edu/~eclab/papers/panait05cooperative.pdf.

Pěchouček, M., Jakob, M., Novák, P., Rehák, M., Šišlák, D. and Vokřínek, J. (2010), "Agent Technology and its Applications", available at: http://www.crr.vutbr.cz/system/files/brozura_07_1003.pdf.

Schrank, D., Eisele, B. and Lomax, T. (2012), *TTI's 2012 URBAN MOBILITY REPORT Powered by INRIX Traffic Data*, Texas: Texas A&M Transportation Institute, available at: http://d2dtl5nnlpfr0r.cloudfront.net/tti.tamu.edu/documents/mobility-report-2012.pdf.

Shenoda, M. (2006), *Development of a Phase-by-Phase, Arrival-Based, Delay-Optimized Adaptive Traffic Signal Control Methodology with Metaheuristic Search*, The University of Texas at Austin, available at: http://repositories.lib.utexas.edu/bitstream/handle/2152/24348/shenodam79542.pdf?sequence=2.

Srikanth, G.U. and Anitha, S. (2012), "Traffic prediction for efficient decentralised routing using multi-agent systems.", *International Journal of Communication Engineering Applications*, Vol. 03 No. 01, available at: http://technicaljournals.org/NPDF/IJCEA-Y12-TJ-SA-C103.pdf.

Sunday, F. (2012), "Companies develop technologies to ease traffic congestion.", *Business Daily*, Nairobi, available at: http://www.businessdailyafrica.com/Companies-develop-technologies-to-ease-traffic-congestion-/-/1248928/1398180/-/fe6n79z/-/index.html.

Tubaishat, M., Shang, Y. and Shi, H. (2007), "Adaptive Traffic Light Control withWireless Sensor Networks", University of Missouri - Columbia, available at: http://www.academia.edu/2878397/Adaptive_traffic_light_control_with_wireless_sensor_networks.

UNFPA. (2007), *State of World Population 2007. Unleashing the Potential of Urban Growth*, New York: United Nations Population Fund, p. 108.

Vlassis, N. (2007), *A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence*, Morgan & Claypool, available at: http://www.uma.ac.ir/files/site1/a_akbari_994c8e8/a_concise_introduction_to_multiagent_systems_and_distributed_artificial_intelligence__synthesis_lectures_on_artificial_intelligence_and_machine_learni.pdf.

Wooldridge, M. (2000), *Reasoning about rational agents*, MIT Press, London.

Wooldridge, M. (2002), *An Introduction to Multi-Agent Systems.pdf*, JOHN WILEY& SONS, LTD, England, available at: http://coltech.vnu.edu.vn/httt/media/courses/AI++/Tai%20lieu/TLTK.pdf (accessed 25 August 2014).

**APPENDIX 1: Sample code 1 (Abstract Agent)**

```
package prj.manage.traffic.agents;
import jade.core.AID;
import jade.core.Agent;
import jade.domain.DFService;
import jade.domain.FIPAAgentManagement.DFAgentDescription;
import jade.domain.FIPAAgentManagement.SearchConstraints;
import jade.domain.FIPAAgentManagement.ServiceDescription;
import jade.domain.FIPAException;
import org.slf4j.LoggerFactory;
public class AbstractAgent extends Agent {

   protected final org.slf4j.Logger log = LoggerFactory.getLogger(getClass());

   @Override
   protected void takeDown() {
      log.debug(" takeDown -- closing agent");
      try {
         DFService.deregister(this);
      } catch (FIPAException ex) {
         log.error(" takeDown -- error ", ex);
      }
      try {
         super.takeDown(); //To change body of generated methods, choose Tools | Templates.
      } catch (Exception e) {
      }
   }

   @Override
   protected void setup() {
      super.setup(); //To change body of generated methods, choose Tools | Templates.
      log.debug(" setup -- initializing agent {} ", getLocalName());
      register();
   }
```

```
protected final void register() {
   log.debug(" register -- registering agent {} with DF ", getLocalName());
   DFAgentDescription dfd = new DFAgentDescription();
   dfd.setName(getAID());
   ServiceDescription sd = new ServiceDescription();
   sd.setType(getClass().getSimpleName());
   sd.setName(getLocalName());
   dfd.addServices(sd);
   try {
      DFService.register(this, dfd);
   } catch (FIPAException fe) {
      log.error(" register -- error ", fe);
   }
}

protected final AID getService(String service) {
   log.debug(" getServices -- searching DF for one {} ", service);
   DFAgentDescription dfd = new DFAgentDescription();
   ServiceDescription sd = new ServiceDescription();
   sd.setType(service);
   dfd.addServices(sd);
   try {
      DFAgentDescription[] result = DFService.search(this, dfd);
      if (result.length > 0) {
         return result[0].getName();
      }
   } catch (FIPAException fe) {
      log.error(" getService -- error ", fe);
   }
   return null;
}

protected final AID[] getServices(String service) {
   log.debug(" getServices -- searching DF for {} ", service);
   DFAgentDescription dfd = new DFAgentDescription();
```

```
        ServiceDescription sd = new ServiceDescription();
        sd.setType(service);
        dfd.addServices(sd);

        SearchConstraints ALL = new SearchConstraints();
        ALL.setMaxResults(new Long(-1));

        try {
            DFAgentDescription[] result = DFService.search(this, dfd, ALL);
            AID[] agents = new AID[result.length];
            for (int i = 0; i < result.length; i++) {
                agents[i] = result[i].getName();
            }
            return agents;

        } catch (FIPAException fe) {
            log.error(" searchDF -- error ", fe);
        }

        return null;
    }
    public static final String INTERSECTION_1_AGENT = "Intersection 1 Agent";
    public static final String INTERSECTION_2_AGENT = "Intersection 2 Agent";
    public static final String TRAFFIC_SNIFFER = "Intersections Sniffer";
    public static final String TRAFFIC_RMA = "Intersections RMA";
}
```

**APPENDIX 2: Sample code 2 (Intersection Agent)**

```
package prj.manage.traffic.agents;

import jade.core.AID;
import jade.core.behaviours.CyclicBehaviour;
import jade.core.behaviours.OneShotBehaviour;
import jade.core.behaviours.TickerBehaviour;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;
import jade.lang.acl.UnreadableException;
import java.io.IOException;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.joda.time.DateTime;
import prj.manage.traffic.GUIFrame;
import prj.manage.traffic.agents.onto.AgentActionWrapper;
import prj.manage.traffic.roundabout.RoundAbout;
import prj.manage.traffic.transport.Lane;
import prj.manage.traffic.transport.Road;
import prj.manage.traffic.transport.RoadMap;

/**
 *
 * @author
 */
public class IntersectionAgent extends AbstractAgent {

    private static GUIFrame guiFrame;
    private static int ROUND_ABOUT_CAPACITY = 12;
    private Map<Integer, DateTime> timesOpen = new HashMap<>();

    @Override
    protected void setup() {
        super.setup(); //To change body of generated methods, choose Tools |
Templates.
        if (isIntersection1Agent()) {
            initializeControlledGUI();
        } else if (isIntersection2Agent()) {
            while (guiFrame == null) {
                try {
                    Thread.sleep(100l);
```

```
            } catch (InterruptedException ex) {

Logger.getLogger(IntersectionAgent.class.getName()).log(Level.SEVERE, null, ex);
            }
          }
          guiFrame.getIntersectionUI2().getRoadMap().setIntersectionAgent2(this);
        }
        addBehaviour(new MonitorIntersection(2));
        addBehaviour(new ReceiveRequests());
        addBehaviour(new ActOnAllRoads(true));
    }

    public RoadMap getRoadMap() {
        return guiFrame.getIntersectionUI2().getRoadMap();
    }

    private boolean isIntersection1Agent() {
        return getLocalName().contains(INTERSECTION_1_AGENT);
    }

    private boolean isIntersection2Agent() {
        return getLocalName().contains(INTERSECTION_2_AGENT);
    }

    private void initializeControlledGUI() {
        if (guiFrame == null) {
            new Thread(new Runnable() {

                @Override
                public void run() {
                    guiFrame = new GUIFrame();

guiFrame.getIntersectionUI2().getRoadMap().setIntersectionAgent1(IntersectionAge
nt.this);
                    guiFrame.setVisible(true);
                }
            }).start();
        }
    }

    private class ActOnAllRoads extends OneShotBehaviour {

        private boolean open;

        public ActOnAllRoads(boolean open) {
            this.open = open;
        }
```

```
        @Override
        public void action() {
            while (guiFrame == null) {
                try {
                    Thread.sleep(100l);
                } catch (InterruptedException ex) {

Logger.getLogger(IntersectionAgent.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
            int[][] roadLane = myRoads();
            for (int[] is : roadLane) {
                if (open) {
                    addBehaviour(new OpenRoadBehaviour(is[0], is[1]));
                } else {
                    addBehaviour(new CloseRoadBehaviour(is[0], is[1]));
                }
            }
        }

    }

    private class MonitorIntersection extends TickerBehaviour {

        public MonitorIntersection(long seconds) {
            super(IntersectionAgent.this, seconds * 1000l);
        }

        @Override
        protected void onTick() {
            //count number of vehicles on roundabout, if excess close all roads
            boolean overloaded = false;
            RoundAbout ra =
getRoadMap().getRoundAboutByNumber(isIntersection1Agent() ? 1 : 2);
            if (ra.countCars() >= ROUND_ABOUT_CAPACITY / 2) {
                addBehaviour(new ActOnAllRoads(false));
                overloaded = true;
            }
            //make sure joining road is not clogged
            Road roadByNumber = getRoadMap().getRoadByNumber(4);
            Lane lane = null;
            if (isIntersection1Agent()) {
                lane = roadByNumber.getLane1();
            } else {
                lane = roadByNumber.getLane2();
            }
```

```
        if (!lane.isCanMove()) {
            if (lane.countCars() > 2 || overloaded) {
                requestOpenLane(4, isIntersection2Agent() ? 2 : 1);
            }
        }
        if (overloaded) {
            return;
        }
        List<Lane> myLanes = getMyLanes();
        long incomingTraffic = getIncomingTraffic();
        if (allLanesClosed()) {
            //all lanes closed, roundabout not overloaded
            //open lane with most cars
            for (Lane lane1 : myLanes) {
                if (lane1.countCars() > incomingTraffic / myLanes.size()) {
                    addBehaviour(new
OpenRoadBehaviour(lane1.getRoad().getNumber(), lane1.getNumber()));
                }
            }
            return;
        } else if (allLanesOpen()) {
            for (Lane lane1 : myLanes) {
                if (lane1.countCars() < incomingTraffic / myLanes.size()) {
                    addBehaviour(new
CloseRoadBehaviour(lane1.getRoad().getNumber(), lane1.getNumber()));
                }
            }
            return;
        }
        addBehaviour(new ActOnAllRoads(false));
        //pick road with the most cars and open it too
        for (Integer integer : timesOpen.keySet()) {
            DateTime dT = timesOpen.get(integer);
            if (dT.plusSeconds(5).isBeforeNow()) {
                roadByNumber = getRoadMap().getRoadByNumber(integer);
                for (int[] is : myRoads()) {
                    if (is[0] == integer) {
                        if (is[1] == 1) {
                            lane = roadByNumber.getLane1();
                        } else {
                            lane = roadByNumber.getLane2();
                        }
                        break;
                    }
                }
                if (lane != null) {
                    if (!lane.isCanMove()) {
```

```
                    if (lane.countCars() > 0) {
                        lane.setCanMove(true);
                    }
                } else {
                    timesOpen.put(integer, DateTime.now());
                }
            }

        }
    }
    myLanes = getMyLanes();
    for (Lane lane1 : myLanes) {
        if (!lane1.isCanMove()) {
            if (lane1.countCars() > 3) {
                lane1.setCanMove(true);
            }
        }
    }
}

private class ReceiveRequests extends CyclicBehaviour {

    private MessageTemplate messageTemplate;

    public ReceiveRequests() {
        messageTemplate =
MessageTemplate.and(MessageTemplate.MatchPerformative(ACLMessage.REQUE
ST),
                MessageTemplate.MatchSender(new AID(isIntersection2Agent() ?
INTERSECTION_1_AGENT : INTERSECTION_2_AGENT,
AID.ISLOCALNAME)));
    }

    @Override
    public void action() {
        ACLMessage receive = receive(messageTemplate);
        if (receive != null) {
            try {
                Serializable contentObject = receive.getContentObject();
                if (contentObject instanceof AgentActionWrapper) {
                    AgentActionWrapper aw = (AgentActionWrapper) contentObject;
                    if (aw.getAction() == AgentActionWrapper.OPEN_ROAD) {
                        addBehaviour(new OpenRoadBehaviour(aw.getValue1(),
aw.getValue2()));
                    }
                }
```

```
            } catch (UnreadableException ex) {

Logger.getLogger(IntersectionAgent.class.getName()).log(Level.SEVERE, null, ex);
            }
          }
          block();
        }

    }

    private class OpenRoadBehaviour extends OneShotBehaviour {

        public OpenRoadBehaviour(int road, int lane) {
          this.road = road;
          this.lane = lane;
        }

        @Override
        public void action() {
          log.info(" OpenRoadBehaviour.action -- opening road {} lane {} ", road,
lane);
          Road roadByNumber = getRoadMap().getRoadByNumber(road);
          if (roadByNumber != null) {
            if (lane == 1) {
              roadByNumber.getLane1().setCanMove(true);
            } else {
              roadByNumber.getLane2().setCanMove(true);
            }
            timesOpen.put(road, DateTime.now());
          }
        }
        private int road, lane;
    }

    private class CloseRoadBehaviour extends OneShotBehaviour {

        public CloseRoadBehaviour(int road, int lane) {
          this.road = road;
          this.lane = lane;
        }

        @Override
        public void action() {
          log.info(" CloseRoadBehaviour.action -- closing road {} lane {} ", road,
lane);
          Road roadByNumber = getRoadMap().getRoadByNumber(road);
          if (roadByNumber != null) {
```

```
            if (lane == 1) {
                roadByNumber.getLane1().setCanMove(false);
            } else {
                roadByNumber.getLane2().setCanMove(false);
            }
        }
        timesOpen.put(road, DateTime.now());
    }
    private int road, lane;
}

private void requestOpenLane(int road, int lane) {
    ACLMessage acl = new ACLMessage(ACLMessage.REQUEST);
    acl.addReceiver(new AID(isIntersection1Agent() ?
INTERSECTION_2_AGENT : INTERSECTION_1_AGENT,
AID.ISLOCALNAME));
    AgentActionWrapper aw = new AgentActionWrapper();
    aw.setValue1(road);
    aw.setValue2(lane);
    aw.setAction(AgentActionWrapper.OPEN_ROAD);
    try {
        acl.setContentObject(aw);
        send(acl);
    } catch (IOException ex) {
        Logger.getLogger(IntersectionAgent.class.getName()).log(Level.SEVERE,
null, ex);
    }
}

public long getIncomingTraffic() {
    long x = 0;
    for (Lane object : this.getMyLanes()) {
        x += object.countCars();
    }
    return x;
}

public boolean allLanesClosed() {
    List<Lane> myLanes = getMyLanes();
    for (Lane lane : myLanes) {
        if (lane.isCanMove()) {
            return false;
        }
    }
    return true;
}
```

```
public boolean allLanesOpen() {
    List<Lane> myLanes = getMyLanes();
    for (Lane lane : myLanes) {
        if (!lane.isCanMove()) {
            return false;
        }
    }
    return true;
}

public List<Lane> getMyLanes() {
    List<Lane> lanes = new ArrayList<>();
    for (int[] is : myRoads()) {
        Road roadByNumber = getRoadMap().getRoadByNumber(is[0]);
        Lane lane;
        if (is[1] == 1) {
            lane = roadByNumber.getLane1();
        } else {
            lane = roadByNumber.getLane2();
        }
        lanes.add(lane);
    }
    return lanes;
}

private int[][] myRoads() {
    int[][] roadLane = new int[4][2];
    if (isIntersection1Agent()) {
        roadLane[0][0] = 1;
        roadLane[0][1] = 2;
        roadLane[1][0] = 2;
        roadLane[1][1] = 1;
        roadLane[2][0] = 3;
        roadLane[2][1] = 1;
        roadLane[3][0] = 4;
        roadLane[3][1] = 2;
    } else {
        roadLane[0][0] = 4;
        roadLane[0][1] = 1;
        roadLane[1][0] = 5;
        roadLane[1][1] = 1;
        roadLane[2][0] = 6;
        roadLane[2][1] = 2;
        roadLane[3][0] = 7;
        roadLane[3][1] = 2;
    }
    return roadLane;}}
```

**APPENDIX 3: Pseudocode**

**If** current_lane(A) and state_queue_length (≤2) and wait_time (≤4) and state_destination_lane (≤2)
//**Action**: Liaise with agent2 (at intersection2) to stop corresponding lanes
**then** turn_light_pattern_red

**If** current_lane (A) and state_queue_length (≥2) and wait_time (≥4) and state_destination_lane (≤2)

//**Action:** liaise with agent2 (at intersection2) to release corresponding lanes

**then** turn_light_pattern (corresponding lanes)_green and other lanes Red

**If** current_lane (A) and state_queue_length (≥2) and wait_time (≥4) and state_destination_lane (≥2)

//**Action:** Liaise with agent2 (at intersection2) to release corresponding lanes

**then** turn_light_pattern_green

**If** current_lane (A)and state_queue_length (≥2) and wait_time(≤4) and state_destination_lane(≤2)

//**Action:** Liaise with agent2 (at intersection2) to stop corresponding lanes

**then** turn_light_pattern_red

**If** current_lane (A) and state_queue_length (≤2) and wait_time(≥4) and state_destination_lane(≤2)

//**Action:** Liaise with agent2 (at intersection2) to release corresponding lanes

**then** turn_light_pattern_green

**NB:** The above procedure is coordinated for all the lanes joining the intersection.

≥ means tending to maximum.

≤ means empty or tending to zero.