Cross-Traffic Management

Abdul-Azeez Olanlokun
dept. electronics engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
Abdul-azeez.olanlokun@stud.hshl.de

Izuchukwu George Enekwa
dept. electronics engineering)
Hochschule Hamm-Lippstadt
Lippstadt, Germany
izuchukwu-george.enekwa@stud.hshl.de

Patrick Nonki
dept. electronics engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
Patrick.nonki@stud.hshl.de

Obi Nnamdi Elijah
dept. electronics engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
Nnamdi-elijah.obi@stud.hshl.de

***Abstract*—This paper focuses on the issue facing cross-traffic Management and autonomous vehicles driving through this intersection. Our aim is to find the best practice that suits this type of system through previous research and findings on managing how vehicles can go through the intersection without colliding and causing havoc that can lead to loss of lives and damage to the vehicles. We have done a series of researches by extensively reviewing different articles and research papers explaining and expanding the areas of autonomous vehicles and cross-traffic systems. We have combined the knowledge gained from these researches and the knowledge already gained from the previous semesters' studies to come to a final and best solution to tackle this problem. The best possible solution we were able to arrive at to manage this problem of cross-traffic system is the scheduling system of FIFO algorithm and the queuing system of the MM1 with one server, which manage which vehicles have the highest priority in the system. Our system works with the V2I (Vehicle to Infrastructure)model, as this is what we believe, through our research, is best suited for tackling the cross-traffic system issue. According to our study, the implementation of the system is best designed using the HW/SW code design approach. So we were able to implement/simulate our design in both software(FreeRtos) and hardware(VHDL), showing the complete system implementation, and the result was able to show how our system was able to tackle the problem in cross-traffic systems. The Autonomous Vehicles can cross through the intersection without colliding with one another.**

***Index Terms*—Crossing, Autonomous Vehicles, Traffic, V2I**

I. Introduction

There have been advances in technology when it comes to the automobiles/vehicles sectors. Different factors such as wireless networks, Iots, sensor networks, data sharing and cloud computing have spearheaded these advancements. These have led to the improvements in cross-traffic management systems where autonomous vehicles can communicate with one another and with infrastructures. These systems are termed V2I and V2V, Vehicle to Infrastructure and Vehicle to Vehicle communication, respectively. These systems are gradually taking away the old ways of the traffic light system, and they are more regulated autonomously. Different studies and research have been done to ensure these systems are sustainable and able to tackle issues facing cross-traffic systems. Traffic management is a complex system, especially at intersections, and needs proper management control to overcome these complexities. One of the good sides to using autonomous vehicles and intelligent infrastructure systems is the ability to communicate with one another to avert any collision. Human drivers are the most culprits of collisions at intersections due to many human factors and negligence.

In this study, we propose to manage issues facing cross-traffic management systems where autonomous vehicles can cross the intersection without collision. This paper is organized as follows; In The "Problem Space" section, we discussed the problem we are to solve and the current issue governing this problem. In the "Background and definition section, we described the history of autonomous vehicles and intersection management systems extensively. We also defined the terms of our system. In the "approach section", we have described how we can solve and tackle these stated problems. In the "Implementation and simulation section", we were able to describe the results of this paper. Finally, the "conclusion" Summarizes our paper.

II. Problem Space

The issues facing vehicles crossing the intersection have been around since the first industrial revolution. Knowing the factors that contribute to these issues and how to tackle them in the best ways possible is essential. We know negligence in intersection crossing could lead to catastrophes beyond control. Before the advancement in technology, human drivers, due to many factors, have mainly contributed to these intersection crossing issues. But with the recent advent of autonomous driving, these issues have reduced as technology has brought in many improvements in intersection management and vehicle-to-vehicle communication. We hope to improve further to minimise these issues.

III. Background and Definition

In this section, we will be looking at different topics whose knowledge would aid our project development:

A. Trajectory Modeling:

An intersection is a well-regulated zone where vehicles, regardless of signal presence, normally follow specific patterns whilst passing, e.g., travel paths. The planned paths of cars coming from different directions with varied intentions follow "pre-defined" pathways, depending on the rules of intersection passage. Assuming a trip trajectory is followed, it is simple to discover whether or not trajectories are in conflict with (or cross) one other. Non-conflicting trajectories can produce a so-called safe pattern, within which cars can safely pass through the junction. Trajectory planning has been widely employed in air traffic management and coordination, automobile avoidance systems, and robotic motion planning.

B. Cooperative intersection Management (CIM)

Intersections are classified into two types: signalized and non-signalized. Traffic lights are used at signalized junctions to direct traffic through the intersection. Non-signalized junctions generally feature yield or stop signs, or no signs at all. Cooperative intersection Management (CIM) allows automobiles to connect with infrastructure for traffic perception, junction passing negotiation, in-vehicle signage, and other purposes at signalized intersections. Accurate traffic perception of infrastructure allows for more intelligent traffic signal phase setup, whereas negotiation allows cars to play an active role in intersection management, resulting in more flexible and cooperative intersection control.

In general, CIM may be classed as being either centralized or distributed based on the presence or absence of a central control unit. Approaching the CIM challenge requires the use of cross-disciplinary methods. Centralized CIM is based on a coordination unit that gathers data and instructs cars on when and how-to pass-through intersections. The coordination unit is often an intersection manager, a traffic light, or something similar. The coordinating unit makes central decisions based on information from road infrastructure and cellular communications. There are no central units in distributed CIM. Vehicles communicate with one another, forming a VANET. Each vehicle makes decisions locally based on observations of the surroundings via the VANET. Each vehicle makes local decisions based on observations of the surroundings via the VANET. Vehicles may also interact with one another and with road infrastructure for the most efficient intersection passage. In certain circumstances, even when no established coordination units are present, a temporary unit, such as a vehicle leader chosen through agreement among cars, temporarily directs intersection traffic. The system can be termed centralized in this situation since only centralized procedures are engaged.

C. Cooperative Resource Reservation:

Tiles can be used to model intersections. Vehicles must reserve the tiles on their intended path for certain time periods in cooperative resource reservation. Vehicles can travel through the junction according to the reservation after the tiles and time slots are given. This ensures that space tiles are distributed in a non-conflict manner, i.e., one tile is not assigned to more than one vehicle at the same time slot. Furthermore, the allocation can take into account the traffic condition and use additional optimization targets to optimize junction performance, such as maximizing traffic flow or minimizing energy use. Cooperative resource reserve can be classed as centralized, where intersection infrastructure is responsible for distributing time slots and space tiles, or decentralized, where vehicles negotiate the resource reservation. For cooperative resource reservation, agents who dwell on vehicles and infrastructure also execute resource reservation and grant requests in accordance with the preceding statement. Two types of agents are proposed for centralized cooperative resource reservation: vehicle agents (VA) who reside in vehicles and intersection reservation agents (RA) who reside in infrastructure. When approaching an intersection, a vehicle seeks tile reservation through its VA by providing associated information like as arrival time, velocity, driving intentions, and vehicle dynamics, among other things. The RA replicates passage timetables, such as the tiles and time slots required to pass the intersection. During the simulation, the RA determines if the required tiles have already been reserved by another vehicle at the appropriate time slots in order to find conflicts. If no conflicts are detected, the reservation will be allowed. Otherwise, the reservation request will be refused, and the vehicle will have to file a new one. Once a tile is given, the vehicle must be able to decide if it can follow the reservation and drive through. If not, the car should cancel the reservation and seek a new one.

[5] investigated cooperative resource reserve using a simple intersection. Aside from usual assumptions, it was thought that once a reservation was granted, the vehicle would maintain the same speed until the junction was passed. The simulation was performed in contrast to two other intersection management strategies, overpass and traffic light. It was demonstrated that the cooperative resource reservation approach performed similarly to the overpass method and was twice or thrice better than the traffic light method. Following this exploratory analysis, the authors presented improved approaches and a communication protocol in [6]. Aside from resource reservation, the protocol could simulate various junction control rules such as overpasses, stop signs, and traffic lights. The RA approved all requests and verified them as they were to imitate the overpass. Only vehicles that stopped at the junction received confirmation when modeling the stop sign, while all other vehicles were refused and required to halt. In the case of traffic lights, RA evaluated the traffic light state to the reservation requests and allotted tiles solely to travel routes with green light status. The study looked at more realistic circumstances that allowed for left and right turns. Instead of assuming that cars maintained the same pace across the junction, the work permitted vehicles to accelerate within the intersection.

A first-come, first-served light policy (FCFS-light) was

adopted. The policy was designed to resemble a crossroads controlled by both a traffic signal and a cooperative resource reserve system. When distributing space tiles, the RA considered both tile occupation and traffic light status. Only tiles on green-lighted routes could be booked and assigned. For further discuss on the subject matter please refer to [4] for a comprehensive discussion over the topic. Cooperative resource reserve has been the subject of simulation and prototype. A mixed reality testing system for realistic demonstration was created in [7]. A genuine car at a realistic intersection and a simulation platform depicting the intersection were used in the system. For crossing the intersection, a number of virtual cars were produced, with one of the virtual vehicles serving as a proxy vehicle conforming to the dynamics of the actual vehicle, thereby linking the virtual and physical worlds. Despite the discovery of a wide range of defects, such as GPS, noisy sensor data, and communication instabilities, the reservation-based CIM outperformed traditional systems such as traffic lights and stop signs. In [2], mini vehicles were used for demonstrating the cooperative resource reservation scheme, where V2X communications was implemented with 802.11g Wi-Fi. A traffic simulator, ISR-TrafSim,1 was developed for testing the reservation-based CIM in [1]. Two types of intersections, roundabouts and crossroads intersections, were considered and different interaction procedures were applied.

D. FIRST COME FIRST SERVE(FCFS)

FCFS is a preferred choice for reservation-based control because of its fairness approach. It is simply based on a first come, first served basis, as the name says. The junction manager has complete vehicle information in the centralized model utilized in this work. It is aware of the next available tile slot for each vehicle, as well as the set entry timings and destination for all cars wishing to enter the junction. It is worth to consider that in a real crossroads with non-conflicting streams, cars do not always observe FCFS.

E. QUEUING THEORY

This is a model-based technique to estimating queuing behaviors based on a set of restrictions and assumptions. It is assumed that there are inputs for arrival distribution and service time distribution for a number of servers. The assessment of effectiveness or operational characteristic that gauges the performance of a queuing system is a direct outcome of queuing theory. If your queuing system distribution conforms to the assumptions of theory, you can simply apply the queuing theory formulae to anticipate the long-term behavior (also known as the steady state condition) of the queuing system in terms of its performance. For example, you may calculate the number of servers required or estimate the service time required to match your demand. The cost of queuing can also be estimated and experimentation with certain ideas is a necessity to improving the queuing system.

IV. V2X

V2X is majorly implemented in four primary modes of operation. They are Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), Vehicle-to-Pedestrian (V2P), and Vehicle-to-Network (V2N) communication [9]. These four modes can be utilized in tandem for safety, autonomous vehicle control augmentation through the utilization of data from adjacent sensors, and accident prevention. These four modalities are explained in more detail below:

A. Vehicle-to-Vehicle (V2V):

V2V allow cars in close proximity to build a mesh network and share data, allowing better decisions to be made through information sharing among existing nodes. This is accomplished by registering with a network operator and obtaining authorization. V2V apps operate by sending messages including V2V application information such as traffic dynamics, vehicle position, vehicle characteristics, and so on. For improved communication, message payloads are maintained flexible. Furthermore, 3GPP communications are mostly broadcast. As a result, assuring one-to-many data transfer with minimal latency, which is required for V2V [10].

B. Vehicle-to-Infrastructure (V2I):

V2I application data is sent via a Remote Switching Unit (RSU) or a locally accessible application server. RSUs are roadside stationary devices that serve as a transceiver. RSUs or accessible application servers receive the broadcast message and forward it to one or more UEs that support the V2I application. V2I can advise us about available parking spaces, traffic congestion, road conditions, and so forth. Because of the high cost and extended deployment time, its application or installation is more difficult.[10]

C. Vehicle-to-Pedestrian (V2P):

V2P transmission takes place between a vehicle and Vulnerable Road Users (VRUs) such as pedestrians and bicycles. UEs worn by drivers and pedestrians will be able to receive and transmit messages and alarms [8]. Vehicles may connect with VRUs even when they are in Non-Line of Sight (NLOS) or in low light conditions such as a dark night, severe rain, foggy weather, and so on. Because of antenna and battery capacity differences, pedestrian UEs have poorer sensitivity than vehicle UEs. As a result, V2P application-enabled UEs cannot send continuous messages like V2V-enabled UEs.

D. Vehicle-to-Network (V2N):

V2N communication takes place between a vehicle and a V2X application server. A terminal that supports V2N applications can interact with an application server that supports V2N applications, and the parties communicate via Evolved Packet Switching (EPS). V2X services are required for a variety of applications and operational settings. They help mobile operators communicate RSU functions over their network and reduce the time-to-market, cost and complexity of building and operating a network specifically designed for V2I, as communication between vehicles and the server can take place over 4G or even 5G networks. It does not have to be as accurate as V2V, but it must be reliable.

V. Approach used

The approach we decided to use for our system is the M/M/1 with one server controlling the queuing system. This method is described as follows: When a car comes to an intersection point, it sends the direction it wants to go; this direction is defined according to where it comes from and where it wants to go ("NE" if it comes from the North and goes to the East). A table slot is now created where all the cars coming from all directions are supposed to cross. If the vehicle arriving requests a slot that is not accessible, then it has to wait until the slot is free. After the slot has been freed, the car can enter the grid, pass through until its final destination, and exit the system. This action is repeated in parallel with the movement of another car that is also requesting a slot inside the system. Once a vehicle has finished crossing the system, it returns a message meaning it successfully reached its target.

VI. System Diagrams

Here we give the graphical views of our systems from the use case, state machine and sequence diagrams.

A. Use Case Diagram

This is where a more visualised details of our systems design is provided, with its main actors, roles and classes in a Unified modelling language. We have the following use cases for our diagram, the vehicle is able to give movement information and feedback to the cross traffic system. The vehicle communicates with infrastructure and gets directions, these are also stored in a cloud server. (Diagram located in appendix)

B. State Machine

For the state we have 4 initial states which we identified as our first bit (direction). These First bits are entrances into the intersection. The Final state of the bit shows the exit state of our system. As seen, if the slot is free, and the second bit is true, the vehicle will be permitted to drive, else it waits and checks again. (Diagram located in appendix)

C. Sequence Diagram

This defines the sequential flow of the system. The 3 block sequential states as seen, the vehicles, infrastructure systems and cloud server. For the first sequence, which is where the Vehicle sends a message to the Infrastructure system, requesting for route availability. The infrastructure systems then respond with the current status. If a slot is available then drive, else it waits. These functions were also set in a loop to always check again. Also, it is based on Vehicle with the shortest time interval. (Diagram located in appendix)

VII. Systems Implementation and Simulation

This section is related to the software and hardware part of our robot, referring to the implementation i.e., software coding in the freeRtos programming and hardware in VHDL, which consists of all the requirements stated before. For better understanding of our code, we decided to separate the degree of complexity and move step by step i.e.:

- The queue management, which illustrates how the queue is managed.
- Movement.

A. Creation of task

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <Arduino_FreeRTOS.h>

void setup() {
  char dir[1]; //direction
  char tab[8][2] = {"NS", "SE", "EN", "NW", "NS", "SE", "EN", "NW"}; // Repeat states to be able to have a complete loop
  xTaskCreate(Queue, "Queue", 128, NULL, 1, NULL); //
  xTaskCreate(Movement, "Movement", 128, NULL, 2, NULL); //The task with the higher priority
}
```

Fig. 1. xTaskCreate

For this part of our coding, we technically created variables on top of the program. For the section with "xTaskCreate" statement, it is a statement to create tasks in freeRtos. Here we created tasks mapped from our state diagram, which is based on the Queue and Movement. These functions illustrates the direction and priority assignment functions.

B. Driving Directions

```
void Movement(void *pvParameters)
{
  while(1){
    printf("Enter the direction (Ex: Enter NE if you come from North and going to East)");
    scanf("%s", dir);
    if (scanf(" %c", &dir) != 1)
    {
      printf("%s\n", "You didnt enter a valid value");
      return -1;
    }
  }
}
```

Fig. 2. Driving Directions

This part illustrates the drive direction of the vehicle. Here the vehicle simply identifies drive direction by sending or selecting a movement pattern or a direction message based on the request of an infrastructure manager or cloud manager asking to identify direction.. The user must enter a value in "dir". If the Value the user is entering is not a character, we print that the value is not valid.

C. Queuing Architecture

```
void Queue(void *pvParameters)
{
  while(1)
  {
    for (int i = 0; i <8 ; ++i) {
      if (dir[0] == tab[i][0]){       //check if the first bit of the direction and the first bit of the slot are the same
        for (int j = i; j <8 ; ++j) {
        do{                           // Check the next slot and replace it with an 'x' if free, and then move and restore the old value of the slot
            char c;
            strcpy(c,tab[j]);   //Put the value of the slot in c
            strcpy(tab[j], "x"); //replace the slot value by x
            strcpy(tab[j],c); //restore the slot value
          }while (dir[1] == tab[0][j]); //Exit when the last bit of the direction is the same as the last bit of the slot
        }
      }
    }
  }
}
```

Fig. 3. Queue Architecture

This part of the code simply describes and handles what goes on inside the queue. First we initialised a line of code that checks if the first bit of direction and first bit of the slot are the same, if the same, we proceed to the next loop. For movement, the vehicle requests that the next slot is checked

for availability, if available, it will then be reserved with an "x" placed on it. After movement happens, it restores the old slot value to "C". Which basically means, it returns a message that the slot is free again, or successful drive.

D. Hardware Implementation(VHDL)

During the hardware simulation of our system, we used VHDL Hardware Description Language, which is a hardware description language (HDL) capable of modelling the behaviour and structure of digital systems at multiple levels of abstraction, from that of the system down to that of design documentation and verification. IEEE Std 1076 has been standardised as IEEE Std 1076 since 1987. The latest version is IEEE Std 1076-2019. An IEEE-standardised HDL based on VHDL called VHDL-AMS (officially IEEE 1076.1) has been developed to model analog and mixed-signal systems.

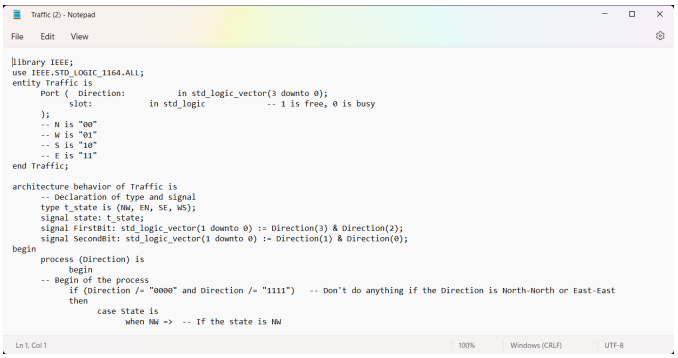

Fig. 4. VHDL CODE

Firstly, we intend to exit a state if two directions are the same, for example "don't do anything if the Direction is North-North or East-East". If the second bit of the direction is different from the second bit of the state and if the next slot is free then we can proceed to the next state. Otherwise, the state will remain the same. Also, if the second bit of the direction is the same as the second bit of the state we exit the state.

This process was used to check all directions for a safe movement.

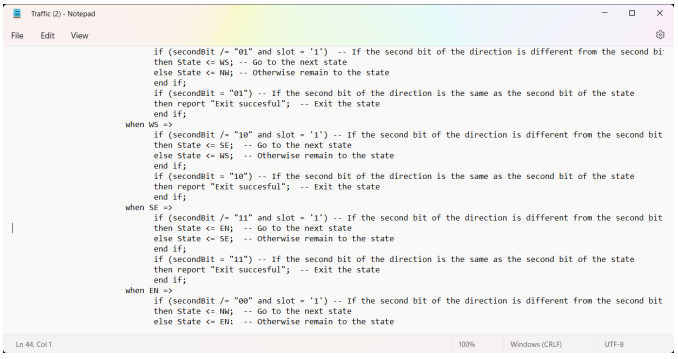

Fig. 5. VHDL Direction checks

The final result of our simulation is illustrated in the fig 7 below

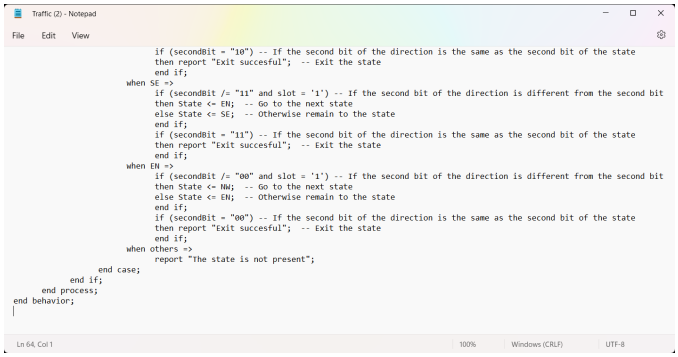

Fig. 6. VHDL Direction checks

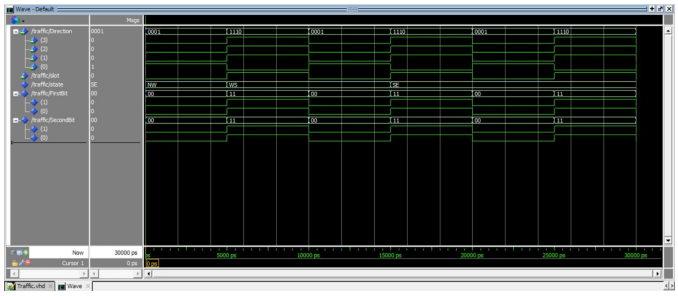

Fig. 7. Simulation Wave of our traffic system

VIII. Conclusion

The cross-traffic management implemented in this paper reduces traffic congestion and allows more fluidity on the roads. To be able to analyse our system, we decided to use the forward engineering method. First, we decided to model our system by identifying all its elements and implementing their interaction using the SysML methodology (Use case diagram, state diagram, sequence diagram, activity diagram...). Then we decided which approach to use for our system and finally agreed on the M/M/1 method for one server as the central controller. We then designed our system both in Hardware and Software. For the hardware part, we decided to describe our system using the VHDL language, and then we generated a test bench to test the behaviour of our system. In the software part, the most suitable way was to work with the freeRTOS implementation as we have the queuing task and the direction task working simultaneously. As this research is limited to V2I communication, a combination with V2V communication could be an improvement as it would help manage the queue between cars outside the queuing system. Thus, designs with sensors and cloud computing will be integrated much later to improve and strengthen our study. Nevertheless, the results we got from our research would be implemented to have a perceptible and efficient development.

IX. AFFIDAVIT

We, Abdul-Azeez Olanlokun,Enekwa Izuchukwu George, Patrick Nonki, Elijah Nnamdi Obi thus certify that we wrote the current paper and work entirely by ourselves, using only

the mentioned sources and aids. Other references to the statement and scope are noted by complete information of the publications concerned; words or sections of sentences cited literally are marked as such. The paper and work in a similar or identical format have never been submitted to an examination board or published. This paper had not yet been utilized in another test or as a course performance, even in part.

REFERENCES

[1] L. C. Bento, R. Parafita, and U. Nunes, "Intelligent traffic management at intersections supported by V2V and V2I communications," in Proc. 15th Int. IEEE Conf. Intell. Transp. Syst., 2012, pp. 1495–1502.

[2] C.-L. Fok et al., "A platform for evaluating autonomous intersection management policies," in Proc. IEEE/ACM 3rd Int. Conf. Cyber-Phys. Syst., 2012, pp. 87–96.

[3] L. C. Bento, R. Parafita, S. Santos, and U. Nunes, "Intelligent traffic management at intersections: Legacy mode for vehicles not equipped with V2V and V2I communications," in Proc. Int. IEEE ITSC, 2013, pp. 726–731.

[4] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," J. Artif. Intell. Res., vol. 31, no. 1, pp. 591–656, Jan. 2008.

[5] K. Dresner and P. Stone, "Multiagent traffic management: A reservationbased intersection control mechanism," in Proc. 3rd Int. Joint Conf. Auton. Agents Multiagent Syst., 2004, pp. 530–537.

[6] K. Dresner and P. Stone, "Multiagent traffic management: An improved intersection control mechanism," in Proc. 4th Int. Joint Conf. Auton. Agents Multiagent Syst., 2005, pp. 471–477

[7] M. Quinlan, J. Zhu, N. Stiurca, and P. Stone, "Bringing simulation to life: A mixed reality autonomous intersection," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2010, pp. 6083–6088.

[8] Cooperative Intersection Management: A Survey Lei Chen, and Cristofer Englund

[9] 3GPP TS 22.185, "Service requirements for V2X services", Release 15, 2018.

X. APPENDIX

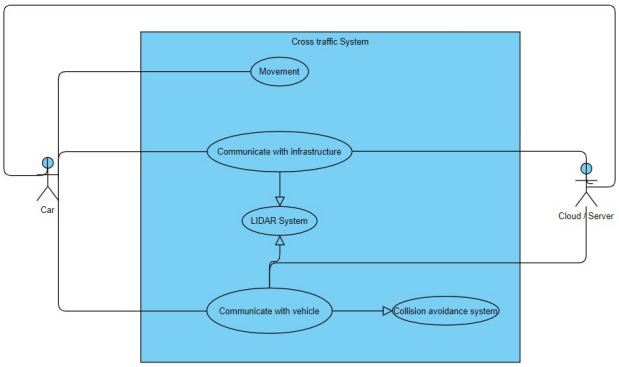

Fig. 8. Use Case

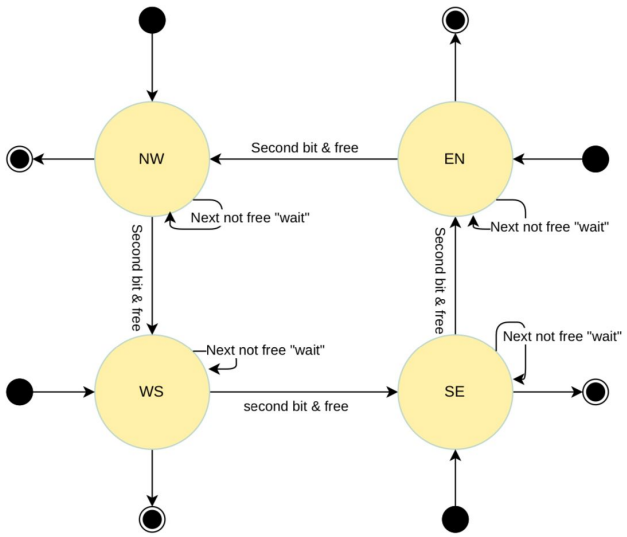

Fig. 9. State Machine

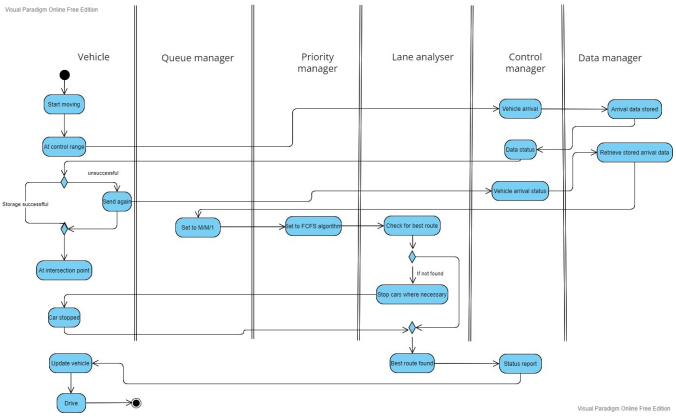

Fig. 10. Sequence Diagram

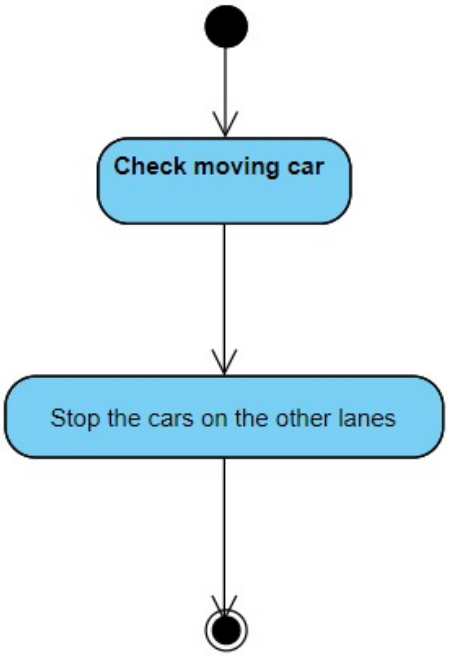

Fig. 11. Control Manager

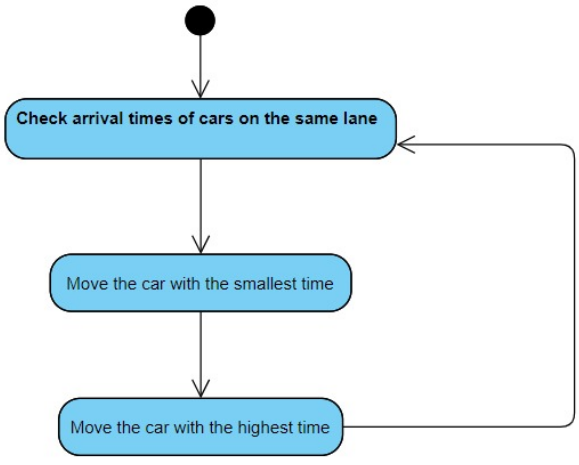

Fig. 12. Drive Direction

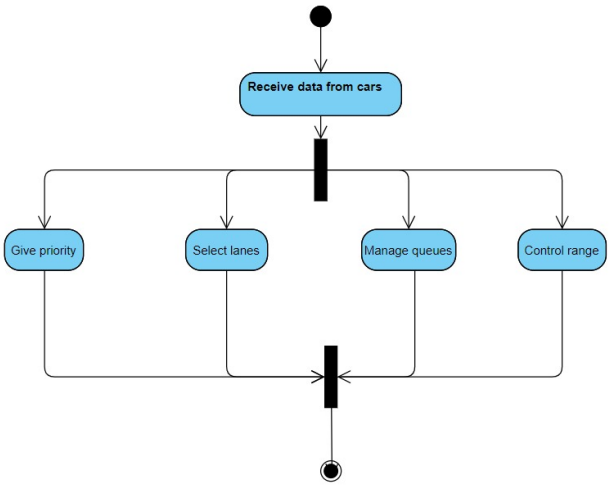

Fig. 13. Intersection Manager

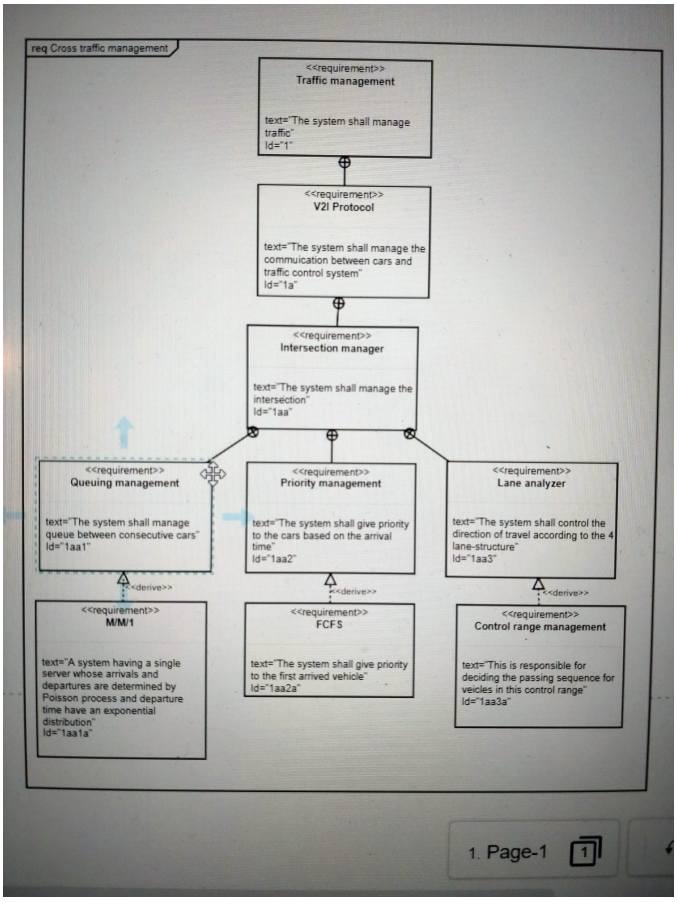

Fig. 14. Queuing Management System

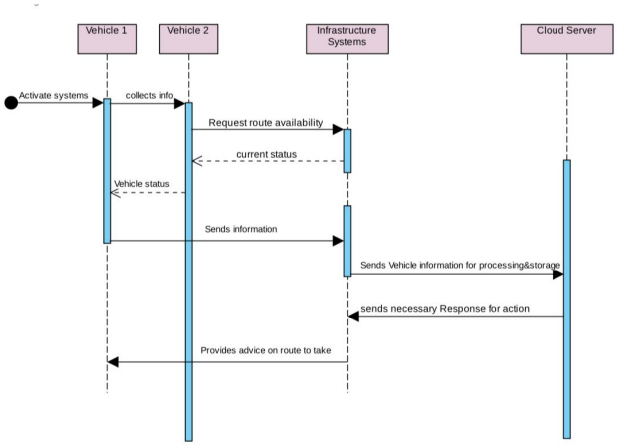

Fig. 15. Sequence Diagram for Car Movements