Title: MANETS/VANETS in AutomobileSystems: A Simulative Study into how Mobile Ad-hoc Networks can be used in Traffic Control Systems

By
Goodbye Mavata (UZ, 2010)

R970215V
A Thesis Research submitted in partial fulfillment of the requirements for the degree of Masters Computer Science

University of Zimbabwe

2010

Supervised by Mr. B. Nyambo
Supervisor
Dedication

Oh Lord Thou art my hiding place. Tower of refuge and strength. Whom have I in Heaven besides You; my heart and my strength though many times they fail, but the is one Truth that always prevail. You’re the strength of my heart and my portion forever!
Acknowledgements

I acknowledge my dependency upon thee, Oh, Lord God Jesus Christ.

So, I am very thankful to my supervisor Mr. B. Nyambo for his unparalleled support, mantle, and counsel through out the research. You have been quite resourceful. I need also acknowledge the Computer Science Department for their constructive criticisms and suggestions. I mention Dr. G. Hapanyengwi, Mr. Rupere and Mr. B. Mapako in this regard.

It would not be fair if I don’t mention some people like Mr. M. Mavata and Miss D. Charambeni for their ideas and contributions. I mention the people that come first in my life, last- my family S.Letha, Tinevimbo and Tinemishe Mavata saying you have always given me reason to go the ends of the earth in search for the hidden treasures of darkness and the riches of secret places.
# Table of Contents

Dedication ..................................................................................................................................................... 2

Acknowledgements ....................................................................................................................................... 3

Table of Figures ............................................................................................................................................. 6

List of Tables ................................................................................................................................................. 7

Abstract ......................................................................................................................................................... 8

Chapter One  Introduction ...................................................................................................................... 9

  Background to the problem/problem situation ....................................................................................... 9

  Statement of the Problem ......................................................................................................................... 13

    Research Questions ............................................................................................................................. 13

  Aims and Objectives ................................................................................................................................ 14

  Research Designs .................................................................................................................................... 15

  Significance of the Chosen Area of the Study ......................................................................................... 17

    Limitations ........................................................................................................................................... 17

    Assumptions ........................................................................................................................................ 18

    Delimitations ....................................................................................................................................... 18

    Definition of Terms ................................................................................................................................. 19

Chapter Two  Critical Literature Review ............................................................................................... 20

  2.0 Vanets ............................................................................................................................................... 20

  2.1 Traffic Signal Control ......................................................................................................................... 21

    2.1.1 Terminology ................................................................................................................................... 21

    2.1.2 Control Types ............................................................................................................................... 24

  2.2 Transit Priority Strategies .................................................................................................................. 30
# Table of Figures

- **Figure 2-1**: Example intersection. ...............................................................................................................................22
- **Figure 2-2**: Example signal phase diagram. ................................................................................................................23
- **Figure 2-3**: Example signal group diagram. ................................................................................................................23
- **Figure 2-4**: Relation between phase and signal group specifications. ........................................................................24
- **Figure 2-5**: Types of signal control logic. ....................................................................................................................25
- **Figure 2-6**: Green interval extension of an actuated phase. ......................................................................................26
- **Figure 2-7**: Example of a gap-reduction function. .....................................................................................................27
- **Figure 2-8**: Progressive traffic flow under signal coordination. ...............................................................................29
- **Figure 2-9**: Bi-directional progressive flow under signal coordination. .................................................................29
- **Figure 2-10**: Vehicle trajectory without signal priority. .........................................................................................31
- **Figure 2-11**: Vehicle trajectory with transit phase extension. ....................................................................................34
- **Figure 2-12**: Vehicle trajectory with early start to transit phase. ..........................................................................34
- **Figure 2-13**: Vehicle trajectory with insertion of extra transit phase. .................................................................35
- **Figure 2-14**: Time windows for priority calls. ........................................................................................................40
- **Figure 2-15**: Execution of green extension. ...............................................................................................................42
- **Figure 2-16**: Execution of shortening of current phase. ............................................................................................44
- **Figure 2-17**: Execution of insertion of extra phase. .................................................................................................45
- **Figure 2-18**: Execution of green restart. ..................................................................................................................47
- **Figure 4-1**: The User Interface ....................................................................................................................................59
- **Figure 4-2**: Weights added Vs Time ...........................................................................................................................60
- **Figure 4-3**: Node Weight growth Vs time ................................................................................................................63
- **Figure 4-4**: Standard deviation Vs time graph. ..........................................................................................................64
- **Figure 4-5**: Starvation Avoidance. ............................................................................................................................71
- **Figure 4-6**: Starvation avoidance graph. ...................................................................................................................72
- **Figure 4-7**: Pie charts showing how the green times were shared by the nodes. ....................................................73
List of Tables

Table 3 - 1: Junction Class definition................................................................. 52
Table 3 - 2: Node Class definition........................................................................... 54
Table 3 - 3: The Pseudo Code .............................................................................. 56
Table 4 - 1: mean and standard deviation for Node weights............................... 62
Table 4 - 2: Pearson correlation coefficient for Node 0........................................ 66
Table 4 - 3: Pearson correlation coefficient for Node 1........................................ 68
Table 4 - 4: Pearson correlation coefficient for Node 2........................................ 69
Table 4 - 5: Pearson correlation coefficient for Node 3........................................ 70
Abstract
This thesis describes the modeling and implementation of an advanced traffic signal control system within a simulation environment, thus creating a laboratory for the evaluation of advanced traffic control strategies at road intersections, including transit signal priority. The simulation is done as a C++ implementation code and NS2 simulation is proposed for further evaluation and testing for a microscopic traffic simulation for Intelligent Transportation System (ITS) design.

The control system is designed with a generic and flexible logic that allows it to simulate a wide range of traffic signal control types and strategies. The strategies include means of deadlock avoidance and means to avoid starvation to all the vehicles approaching an intersection requesting for a right of way.

The control system is also designed as a distributed control system in which vehicles leading in a road segment i.e. for vehicles approaching an intersection, the leader refers to the car leading the pack at or in the junction. This vehicle is made to participate in a leader election process. This is a bid or a contest to gain or win the right of way for the vehicles for which the leader is in the same road segment with.

Specialized features of advanced control strategies are implemented within the Control system framework which allows the implementation of transit signal priority and other specialized vehicles that might require prioritization within the simulation environment, allowing the simulation of both passive and active signal priority strategies.

The capabilities of the control system are illustrated through a case study in which a simulation is done for a three and four way intersection and the results of the simulation studied against the objectives of the prioritization strategies.

An evaluation of the currently implemented system is performed, and recommendations for improvement and further study are offered.
Chapter One  Introduction

Background to the problem/problem situation

Controlling traffic, especially at road junctions, is an area of major concern for metropolitan city planners and routing experts. This situation can be exasperated/aggravated if robots at robot controlled intersections stop working/malfunction and the police are not available to give hand insignia to control the traffic flow. For example, in Harare, the Municipal police and the traffic cops are always involved in continuous running battles with the taxi/commuter omnibus operators who have put road safety under siege by their unscrupulous driving methods in an effort to beat robots/queues hoping to make more trips in the hour. These drivers speed and can swerve through the lanes and pavements or overtake on the wrong side at any instant without giving adequate warning to other road users. Driving or commuting to work five days a week is a total nightmare especially when one gets locked in a traffic jam and many productive hours are lost. Once in a while the robots are not working and the traffic police are not available to give hand signals to the hooting and yelling swearing, and cursing road users. The situation can get worse if there is an accident along one way or a big lorry has had a breakdown at the narrower intersection. Long queues, slow winding queues of moving vehicles pileup and the situation can be very discouraging for one to use certain notorious roads. Driving through a city or CBD where the roads are narrower and there are inadequate parking spaces during pick hours is a catastrophe for time-savers and the impatient. Recent surveys in the United States of America (USA) show that traffic congestion costs are a staggering total amount in: Delay of 3.7 billion hours, Wasted Fuel of 2.3 billion gallons and an Annual Cost of $63 billion to the US economy alone [1], a situation which can not be allowed to continue especially for developing countries like Zimbabwe who import this commodity using scarce hard currencies only. It is from such experiences and observations that Intelligent Transportation Systems (ITS) can become handy in easing if not solving this traffic jungle madness [2]. ITS applies advanced technologies to surface transportation systems and are viewed widely as the solution to the transportation and traffic control problems that the 21st Century societies will face. Basically it
involves the use of traffic signals that are designed to manage vehicle conflicts at intersections by allocating time among the conflicting traffic streams which must share the use of the intersection. The signal controller allocates usage of the intersection using basic fixed-time methods or intelligent strategies that detect and respond to traffic conditions in real time. The signal controllers can also be used as a tool for managing traffic flow, say for example, either along a corridor or throughout a network, and thereby, provide a more efficient use of the urban street network.

Many systems have been tried, but it can be seen that these systems have not really worked as expected because of the error of one critical omission- the vehicles/cars on the road do not see each other. This is the diagnostics! Cars do not communicate with each other as do other devices such as computers and cell phones. The cars do not communicate with the road either, i.e. the road does not recognize the car nor the car recognize the road hence a series of accidents when they move out of the road. At the roads intersection, the robot and the car do not communicate at all but it is the driver who must decode the stop and go signals as colors. The driver who is subject to reaction time, illusion, emotion, visual impairment, poor judgment, erratic choices, imprecision less calculative ability and other inadequacies is still directly in control of the vehicles at all the time. The time has now come to give the new Intelligent Transportation Systems (ITS) that incorporate in-vehicle information technology (e.g. mobile computing and wireless communication) the opportunity to take over by designing Vehicle-to-Vehicle communication systems. This is the prognostics! ITS will be useful for a wide variety of applications that include incident detection, crash reporting, congestion warning and traveler information dissemination. In-vehicle sensors and communication devices would offer the potential for more detailed and accurate data collection (e.g. second-to-second position, speed, acceleration and deceleration), better information transfer dissemination (than the human drivers) as well as allowing for extended coverage to areas beyond the usual where roadside equipment has been placed. These new emerging technologies, collectively named the Mobile Ad-hoc NETworks (or generally referred to as MANETs) and the Vehicle-to-vehicle Ad-hoc NETworks (or generally referred to as VANETs) when incorporated will extend the existing
roadside infrastructures and will require the proper design, construction and installation of appropriate sensors, access points and centralized servers to complement relevant devices and display systems embedded in the vehicles. In the neighboring Republic of South Africa good use with achievement has been achieved with the GPS, branded the “Road Angel” which is able to warn drivers in advance of shortest route possible, the road currently in use, accident zones, attack zones and speed or mobile cameras along the highways, etc. Thus it is imperative to understand the impact of ITS applications that utilize these networks on the end-user, and more importantly to isolate fundamental performance limitations of VANETs in order to expand their applicability to support appropriate higher level application services.

A very important category is driver assistance and car safety. This includes many different things mostly based on sensor data from other cars. One could think of brake warning sent from preceding car, tailgate and collision warning, information about road condition and maintenance, detailed regional weather forecast, premonition of traffic jams, caution to an accident behind the next bend, detailed information about an accident for the rescue team and many other things. One could also think of local updates of the cars navigation systems or an assistant that helps to follow a friend’s car [2]. Another category is infotainment for passengers. For example internet access, chatting and interactive games between cars close to each other. The kids will love it. Next category is local information as next free parking space (perhaps with a reservation system), detailed information about fuel prices and services offered by the next service station or just tourist information about sights. A possible other category is car maintenance. For example online help from your car mechanic when your car breaks down or just simply service information. So far no inter-vehicle communication system for data exchange between vehicles and between roadside and vehicles has been put into operation. But there are several different research projects going on.

The advent of extremely fast methods of communication and computation in the past decade has created many new opportunities for controlling traffic on road networks. New control system such as SCOOT, a traffic-responsive system, was developed in the U.K. for optimizing
network traffic performance [4]. New control algorithms such as Optimized Policies for Adaptive Control (OPAC), an on-line traffic signal timing optimization algorithm, were developed in the U.S [4].
Statement of the Problem

In today’s world, traffic congestions waste hundreds of hours of our life. This has caused many researchers to try to resolve the problem with the idea of Intelligent Transportation System. For some applications like a travelling ambulance, it is important to reduce delay even by a second. Even in third world countries where the volumes of traffic are small as compared to first and second world, a painfully considerable amount of time is lost at traffic light controlled intersections, where a driver has got to obey a stop light even when there are no other cars entering the junction.

Research Questions

1. Can vehicles be made to be conscious of the presence of other vehicles in a road network or conversely can the road network be made to be conscious of the presence of the vehicles?
2. Can some intelligence be integrated into traffic regulation and control systems to implement a real time responsive system, and what devices can be used to effect this?
3. What algorithms and implementation strategies can be designed for a traffic control system domain or area (particularly traffic light control systems) so as to exhibit some form of intelligence?
4. To what level of response and data transfer rates can the system be allowed to give some propagation delay and malfunction without such a system endangering public safety?
5. What is the impact of such a system on already existing systems and infrastructure?
Aims and Objectives

The objective of this research is to design and implement an advanced traffic signal control system within a simulation environment, thus creating a laboratory for the evaluation of advanced traffic control strategies at road intersections, including transit signal priority. The simulation is done as a C++ implementation. We propose a framework for which further work can be done with a network simulator or a microscopic simulator so that a parallel and rigorous testing of the proposed architecture can be evaluated in a pseudo real situation. This is outlined in Appendix B.

We propose a means for vehicles to communicate i.e. vehicles to be aware of each other on a road network. Using vehicular ad-hoc networking, transit vehicles can pass on traffic control information which can be manipulated to offer adaptive signal priority strategies at a road intersection.

We go on to propose an algorithm to negotiate an intersection in the event of failure or no traffic lights. This assumes that vehicles will have computational power to perform lightweight arithmetic such as aggregating values and simple integer comparisons to find the largest values of an array. The strategy though aimed at each individual vehicle at the leading node to perform the calculations in a leader election process, a variation of it allows it to be centralized to a controller that will perform the calculations. Vehicles approaching the intersection will only aggregate their values and pass them to the controller. However, this approach is not infrastructure-less and is thus abandoned for the infrastructure-less approach in which embedded systems will be incorporated into vehicles giving them the computational power they need to perform calculations. This will mean that in the event that traffic lights fail or at an uncontrolled intersection, vehicles approaching this intersection will have the capacity to handle or negotiate the intersection with efficiency that can supersede that of pre-timed logic control strategies. This proposes that the traffic lights and related infrastructure are actually redundant and not portable as compared to this approach. If it were that all vehicles would have these wireless devices, and could communicate smoothly across vendor-divide and that these devices are rugged enough not to inconvenience by way of failure, then there would be no need of traffic lights on our roads. With scalability, the system could be made large enough
that use of road signs could actually be removed from off the road sides and all information and expected behavior could be housed solely in the car, then the whole system for a large city will be as primitive as rural road network but as intelligent and sophisticated as one made from the future.

We develop C++ code to implement the algorithm and finally simulate an intersection and analyze statistics from such a simulation to test algorithm.

**Research Designs**

In this paper, the researchers propose a completely infrastructure-less approach for minimizing waiting time at a road intersection, and controlling traffic lights to provide absolute priority for an emergency vehicle and bus/commuter public transport. I use the idea of vehicular ad-hoc networking to reduce the imposed travelling time. Then, I simulate my proposed protocol and compare it with a centrally controlled traffic light system.

ITS applications for transit, or Advanced Public Transportation Systems (APTS), have the same goals, namely improvements of efficiency without the need for major infrastructure enhancements. One such application is Bus Rapid Transit (BRT), a transit concept that uses buses to provide a high level of service usually associated with rail transit. The reason that rail transit can provide such a high level of service, however, is that it operates on a right-of-way that is fixed and exclusive. This is typically not the case for city buses, which instead operate on a shared right-of-way in an open and more chaotic system. In such an environment, buses face delays caused by interactions with other vehicles and by the presence of traffic signals at intersections. These two factors can have a significant negative impact on operations. One method of addressing these operational challenges is by the use of infrastructure solutions such as exclusive bus lanes. While often effective in reducing delays due to congestion, these solutions can be prohibitively expensive or, in many urban areas, infeasible due to inadequate street space. Another method is the use of control strategies, which use the existing traffic signal control system to give priority to transit vehicles.
This convergence of APTS and urban traffic control is known as transit signal priority. Transit signal priority strategies can be categorized into two basic types: passive and active. Passive priority strategies are those that use static signal settings to favor streets with transit routes. These rely on signal timing plans that are prepared off-line and are designed to impede transit vehicles as little as possible. Active priority measures are those which employ dynamic detection and response to transit vehicles, altering signal settings in real-time in order to reduce delay.

Implementing transit signal priority can offer many challenges. One major concern is how to implement transit priority within the existing signal control system. Another is determining what impacts the priority implementation will have on other traffic. Most fundamental, however, is the question of what benefits the priority implementation offers and whether these benefits outweigh the costs.

Because passive priority strategies require no equipment other than the existing traffic controller hardware, these strategies can be implemented and tested relatively easy in the field. Implementation of active priority strategies, however, requires a significant hardware investment, including specialized detectors for transit vehicles and, in some cases, more advanced signal controllers. Field testing of active strategies, therefore, is often too costly to justify, especially when the benefits may be uncertain. In these cases, simulation can be used to evaluate a proposed strategy before it is implemented determining whether field implementation will have beneficial results. Microscopic traffic simulation is an ideal tool for these evaluations, as it simulates vehicle movements at a detailed level, modeling interactions with other vehicles and response to traffic control devices.

For instance, Webster’s model is commonly used for designing timing plans of pre-timed control. However, this model is not sensitive to the design parameters of actuated control [2]. Therefore, the capacity estimation models for pre-timed control cannot be used to determine the capacities of actuated intersections in dynamic traffic assignment (DTA). The traffic controls are oversimplified in dynamic traffic assignment because the traffic controlled intersections are
usually treated as pre-timed. For example, DynaMIT uses the pre-determined approach capacities calibrated by the method of the Highway Capacity Manual (HCM) in traffic assignment. In order to capture the within-day dynamics of traffic, capacity estimation models for actuated or adaptive control should be developed and implemented in dynamic traffic assignment.

**Significance of the Chosen Area of the Study**

Mobile Ad-hoc Networks are one of the developing current technologies; a study in the area will give the researcher a leading edge in research and development work on technology and gadgets in which MANETS are going to be fundamental communication protocol and infrastructure. The researcher has no prior knowledge of these current technologies so a study in the area can find no better justification. Success in the research can translate in hours, days and years of business and social time that is lost every day in traffic jams and ingenuous robot control systems. If the success of Vehicular Ad-hoc Networks is going to serve a life, preventing car road accidents, a study in the area is a worthy cause. So far no inter-vehicle communication system for data exchange between vehicles and between roadside and vehicles has been put into operation in Zimbabwe. But there are several different research projects going on. This study seeks to initiate or increase knowledge in this area of networking devices and open up a channel of events by being prophetic about road safety use and precautions by future generations’ road users.

**Limitations**

This research is being carried out with full knowledge that the GPS, VANETs and MANETs, and NS-2 Simulation technologies are relatively new, or have not yet been introduced in Zimbabwe. Only wireless communication using infrared radiation, Bluetooth devices, microwaves and radio waves are available locally, presenting a major challenge in programming, connectivity and
networking between mobile users because of noise/interference. Consequently the researcher will have to strive to acquaint himself with the new programming techniques required in these areas. Also, the networking and software packages available for use in this study and that render visibility between devices still lag behind those used in similar studies already carried out, or currently being carried out in the Developed Nations such as the United States or the European countries. Relevant literature and textbooks available on this subject are hard to come by so that the researcher has to rely on internet sources. Consequently, it is possible for the results of this study to be adjudged as suffering an impediment due to circumstances beyond the control of the researcher.

**Assumptions**
We assume that all vehicles shall have wireless communication and computing capabilities that shall be in built from manufacture or added on. These wireless communication devices shall form basis of Vehicular Ad-Hoc Networking. They shall be capable of communicating their geo – spatial data so that the road segment in which a vehicle is in shall be implied without worrying of how it is achieved. So an association of a vehicle to a road segment shall be assumed for any vehicles that approach an intersection from or within that segment. We also assume that the physical proximity of the vehicles at or close to the intersection is such that very few data packets that are transmitted between the vehicles are lost and a close to 100% communication or packet delivery shall be assumed. The response time for these wireless devices, and the computational time shall be within tolerable ranges such that the control system proposed with will not endanger public safety.

**Delimitations**
Due to financial and time constraints and other limited resources, this study will be delimited to a C++ simulation and possibly a proposal for NS2 simulation.
Definition of Terms

MANETs  Mobile Ad-hoc Networks
VANETs  Vehicular Ad-hoc Networks
ITS     Intelligent Transportation Systems
NS-2    Network Simulator 2
GPS     Global Positioning System
dynamic traffic assignment
HCM     Highway Capacity Manual
ADTPS   Adaptive transit priority strategies
BRT     Bus Rapid Transit
SCOOT   Urban traffic control systems
OPAC    Optimized Policies for Adaptive Control
CBD     Central Business District
APTS    Advanced Public Transportation Systems
Chapter Two  Critical Literature Review

2.0 Vanets

A Vehicular Ad-Hoc Network, or VANET, is a form of Mobile ad-hoc network, to provide communications among nearby vehicles and between vehicles and nearby fixed equipment, usually described as roadside equipment.

The main goal of VANET is providing safety and comfort for passengers and other road users. To this end a special electronic device will be placed inside each vehicle which will provide Ad-Hoc Network connectivity for the passengers. This network tends to operate without any infrastructure or legacy client and server communication. Each vehicle equipped with VANET device will be a node in the Ad-Hoc network and can receive and relay others messages through the wireless network. Collision warning, road sign alarms and in-place traffic view will give the driver essential tools to decide the best path along the way.

There are also multimedia and internet connectivity facilities for passengers, all provided within the wireless coverage of each car. Automatic payment for parking lots and toll collection are other examples of possibilities inside VANET.

Most of the concerns of interest to MANETs are of interest in VANETs, but the details differ. Rather than moving at random, vehicles tend to move in an organized fashion. The interactions with roadside equipment can likewise be characterized fairly accurately. And finally, most vehicles are restricted in their range of motion, for example by being constrained to follow a paved highway.

In VANET, or Intelligent Vehicular Ad-Hoc Networking, defines an intelligent way of using Vehicular Networking. In VANET integrates on multiple ad-hoc networking technologies such as WiFi IEEE 802.11 b/g, WiMAX IEEE 802.16, Bluetooth, IRA, ZigBee for easy, accurate, effective and simple communication between vehicles on dynamic mobility. Effective measures such as media communication between vehicles can be enabled as well methods to track the automotive vehicles are also preferred.
In VANET helps in defining safety measures in vehicles, streaming communication between vehicles, infotainment and telematics.

Vehicular Ad-hoc Networks are expected to implement variety of wireless technologies such as Dedicated Short Range Communications (DSRC) which is a type of WiFi. Other candidate wireless technologies are Cellular, Satellite, and WiMAX. Vehicular Ad-hoc Networks can be viewed as component of the Intelligent Transportation Systems.

Vehicular Networks are an envision of the Intelligent Transportation Systems (ITS). Vehicles communicate with each other via Inter-Vehicle Communication (IVC) as well as with roadside base stations via Roadside-to-Vehicle Communication (RVC). The optimal goal is that vehicular networks will contribute to safer and more efficient roads in the future by providing timely information to drivers and concerned authorities.

2.1 Traffic Signal Control

Traffic signal controls are implemented for the purpose of reducing or eliminating conflicts at intersections. These conflicts exist because an intersection is an area shared among multiple traffic streams, and the role of the signal system is to manage the shared usage of the area. Signals accomplish this by controlling access to the intersection, allocating usage time among the various users. The logic for this allocation can vary from simple time-based methods to complex algorithms which calculate the allocation in real time based on traffic demand. This section gives an overview of traffic signal control concepts and defines terminology and basic control types and strategies.

2.1.1 Terminology

Because definitions of signal control terms can vary across different countries and different controller types, this section will establish a consistent terminology that will be followed throughout the thesis. There are essentially two distinct methods of specifying basic signal control logic. The method that is standard in the United States is based on “phases,” while the method standard in much of Europe is based on “signal groups.” [14][17]. In traffic signal operation, specified combinations of movements receive right-of-way simultaneously. A
“phase” is the portion of the signal timing cycle that is allocated to one of these sets of movements. Each phase is divided into “intervals,” which are durations in which all signal indications remain unchanged. In the U.S., a phase is typically made up of three intervals: green, yellow, and all red. A phase will progress through all its intervals before moving to the next phase in the cycle. These definitions are illustrated using the example intersection shown in Figure 2-1.

The intersection has three approaches and six possible movements, which are numbered as shown in the figure.

A potential phase diagram for this intersection is shown in Figure 2-2. In this example, the cycle is divided into three phases. Movements 1, 3, and 4 are active in phase 1; movements 1 and 2 are active in phase 2; and movements 5 and 6 are active in phase 3. Each phase represents a distinct time period within the cycle, and in operation the controller moves from one phase to another in the specified order. The timing for the signal is defined by specifying the phase “splits,” which are the percentages of the cycle length allocated to each phase. This split time is further divided among the intervals of each phase, resulting in a specified duration for every interval in every phase.
The alternate specification is based on the concept of a “signal group,” which is defined as a set of signals that must always show identical indications. A signal group controls one or more traffic streams that are always given right-of-way simultaneously. The timing for a signal group is specified by “periods,” which are the durations in which the indication of that signal group does not change.

As an example, the same control logic shown in Figure 2-2 can be expressed in terms of signal groups, as shown in Figure 2-3. Although there are six intersection movements, only four signal groups are needed to represent the logic, because movements that always obtain right-of-way simultaneously can be controlled by a single signal group. Therefore, while movements 1 and 2 must be controlled by two separate groups, movements 3 and 4 can be controlled by a single group because they are never active independently of each other. The same applies for movements 5 and 6, which can also be controlled by a single group.
The timing of each signal group is represented by a horizontal bar whose length represents the cycle length. Each bar is divided into different segments that represent the different periods for each signal group. In this example, each signal group has three periods: green, yellow, and red. In operation, these signal groups advance in time independently, each group changing indication when it reaches a new period. Although signal phases are not explicit in the signal group diagram, phasing can be inferred by reading the diagram vertically. The start of every green period corresponds to the start of a phase, and the time in which all signal groups remain in a single period corresponds to an interval. The correspondence between the two specifications for the above example is demonstrated in Figure 2-4.

**Figure 2-4: Relation between phase and signal group specifications.**

### 2.1.2 Control Types

There is a wide range of logic by which signal phasing and timings can be controlled. Logic types can be categorized along two axes [14]. The first is the type of control logic, specifically how the controller responds to local traffic conditions. This logic can be pre-timed, actuated, or adaptive. The second is the scope of the control strategy, i.e. over what area the strategy is applied. Possible strategies are isolated intersection control, arterial control, and network control. The diagram in Figure 2-5 shows the matrix of possible control types.
### Control Logic

**Pre-timed control** is the most basic type of control logic that can be implemented. In pre-timed control, the cycle length and the phase splits are set at fixed values, as are the timings of each interval within each phase. Historical flow data is typically used to determine appropriate values for these parameters. The key attribute of pre-timed control is that the logic is not demand-responsive, meaning that the signals operate without regard to fluctuations in traffic demand.

**Actuated control** uses demand-responsive logic to control signal timings, with phase durations set based on traffic demand as registered by detectors on the intersection approaches. The most common feature of actuated control is the ability to extend the length of the green interval for a particular phase. The interval might be extended, for example, when a vehicle is approaching a signal that is about to change to yellow, allowing that vehicle to pass through the intersection without stopping.

Figure 2-6 demonstrates how the green interval of a phase can be extended by vehicle actuation [28]. Three parameters are required: the minimum green time, the extension time, and the maximum green time. Regardless of demand, green is retained for at least the specified

---

**Figure 2-5: Types of signal control logic.**

<table>
<thead>
<tr>
<th>Control Logic</th>
<th>Isolated Intersection</th>
<th>Arterial Coordination</th>
<th>Network Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-timed</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Actuated</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Adaptive</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>
minimum duration. If a vehicle is detected and less than the extension time remains in the interval, the interval is extended from the time of actuation by the length of the extension time. This can occur repeatedly, as shown in the figure, with the end of the interval delayed by the extension time from the time of each actuation. The interval will be terminated either when no additional actuation occurs during the latest extension time or when the specified maximum interval length is reached. The extension time is often referred to as the “gap time,” because the interval will be extended if a vehicle has a time gap (headway) from the vehicle in front that is less than this value.

The extension time is usually set to be the travel time from the point of detection to the intersection, as this will extend the interval for just enough time for a detected vehicle to be able to cross the intersection. However, the extension time can also be set to vary as a function of the elapsed green time, usually reducing the extension time as the maximum time is neared. A variable extension length is often used when detectors are located a long distance from the intersection, because a long extension time is desirable at the start of the phase to ensure that vehicles can cross the intersection, while a shorter extension is desired near the end of the phase so that the phase is not extended unnecessarily [29]. A typical “gap-reduction” function is shown in Figure 2-7.

Figure 2-6: Green interval extension of an actuated phase.
Another common feature of actuated control is the ability to skip a phase if no demand for that phase is present. If there are no vehicles waiting for any movements of a certain phase (as determined by the detectors at the stop lines), the controller can skip over that phase and move directly to the next phase in the sequence.

Adaptive control, like actuated control, responds to traffic demand in real time, but its logic can change more parameters than just interval length. The most common adjustments made are to the cycle time and to the phase splits, which determine the allocation of the cycle time to the various phases. These strategies rely on traffic data collected for each approach upstream of the intersection, and this data is used by the controller to estimate conditions at the intersections and to respond to them in real-time. This logic is often optimization-based, allocating green time to maximize measures such as vehicle throughput or to minimize measures such as vehicle delays or stops. Adaptive logic can also be predictive, projecting future conditions based on detector inputs and historical trends and adjusting signal settings accordingly. Adaptive traffic control systems are becoming more widespread, both in application and in development. Urban traffic control systems such as SCOOT are implemented widely [9], and applications of systems such as OPAC and UTOPIA are also becoming more prevalent [24] [32].
Control Scope

**Isolated intersection control** is a control strategy in which the signals for one intersection are operated without consideration of any adjacent signals. In such a case, each intersection will have signal timings that are most appropriate for that single intersection. The local control logic can be pre-timed, actuated, or adaptive; but in the case of demand-responsive logic, the controller will only consider local conditions immediately upstream of the intersection.

**Arterial coordination** is a strategy in which the interaction between adjacent signals is considered. The goal of such strategies is most often to provide “progression” through multiple intersections, allowing vehicles to move through successive signals without encountering a red signal. Figure 2-8 shows an example of how this can be achieved [25]. In this time-space diagram, the horizontal axis represents distance along an arterial, and the vertical axis represents time. The vertical bands represent three signals along the arterial with their indications displayed over time. As shown in the diagram, the timing of the signals can be set such that a vehicle travelling at a certain constant speed can obtain green lights at each intersection. The green times at the signals create a “green band,” and vehicles whose trajectories fall within this band will be unimpeded by the signals. This result is achieved by setting each signal at a different “offset,” defined as the time difference between the start of the signal’s green interval and a system reference time. Setting the offset difference between adjacent intersections to equal the travel time between those intersections will establish progression. Arterial coordination can also be used to provide progression to both directions of traffic, as shown in Figure 2-9.
Figure 2-8: Progressive traffic flow under signal coordination.

Figure 2-9: Bi-directional progressive flow under signal coordination.

With pre-timed signals, arterial coordination is established by using the same cycle length for all signals and by defining an appropriate offset for the green interval at each signal. The offsets define the green band, and the common cycle length ensures that the signals remain synchronized and that the green band will be present in each cycle. Arterial coordination under
actuated control operates on a similar principle, with fixed cycle lengths and offsets for the coordinated green intervals. However, the actuated control logic allows added flexibility, as the non-coordinated phases (such as those for cross streets or for left turns from the arterial) can be skipped or extended based on demand. The coordinated phases, however, must always be green at a fixed time for a specified duration during each cycle in order to maintain the green band for progression. Under adaptive control logic, arterial coordination can be implemented by optimizing measures such as travel time or stops on a corridor-wide level rather than on a single intersection level. By using inputs and measures from the entire corridor, a more efficient control strategy can be realized. For example, an adaptive control strategy might anticipate demand at one intersection based on the signal operation at an upstream intersection, predicting the arrival of a platoon of vehicles that has been released by the upstream signal.

**Network control** has the broadest scope of the control strategies, as it considers the performance of a network as a whole in the implementation of signal control. Most often, network control is an extension of arterial coordination that considers progression for all traffic in all directions of travel. An example of a system where network control can be effective is a urban grid network, in which often no direction of travel may be dominant. In this environment, both pre-timed and actuated control can be easily used to provide limited progression in multiple directions. With adaptive control, however, consideration of network performance may exponentially increase the size of the optimization problem, and solving this in real time may be too computationally intensive. For this reason, adaptive network control algorithms and strategies are still very much under development [17].

2.2 **Transit Priority Strategies**
Transit signal priority strategies aim to reduce delay for transit vehicles at signalized intersections. The reason for this special consideration of transit vehicles is their high carriage capacity. Traffic signals are timed so as to minimize the total delay to all vehicles at an intersection, but minimizing this may not be optimal if the delay per passenger. The passenger load of the vehicles play a critical role under such consideration. A 30-second delay to an overloaded bus is clearly not equivalent to a same time delay to a single-occupancy vehicle. It
then follows from such school of thought and reasoning that a better metric to use to measure the impacts imposed on the users of the transportation network may be total person delay instead of total vehicle delay. Granting priority to transit vehicles, therefore, is more likely to minimize total delay per person and maximize total person throughput. By reducing the probability of a transit vehicle encountering a red signal, and, if this does occur, by reducing the wait time until the green signal, Signal priority strategies attempt to reduce delay.

![Figure 2-10: Vehicle trajectory without signal priority.](image)

### 2.2.1 Passive Priority Strategies

Passive priority is defined as the use of static signal settings to reduce delay for transit vehicles. Such strategies can be as simple as allocating more green time to the street with the transit route by increasing the split for the phase in which the transit vehicle has right of way. Because this reduces the percentage of the cycle during which the transit phase has a red signal, both the probability of a transit vehicle arriving during red and the average wait time if it does will be decreased [22].

Another common passive strategy is the use of a shorter cycle length, which can reduce delay by shortening the wait time until the next green phase. However, this comes at the expense of reduced capacity for the intersection overall due to the increase in lost time, the time in each cycle during which no vehicle movements occur. Lost time typically results from the all-red
safety clearance intervals between conflicting movements and from the vehicle startup delay at the beginning of each phase. Since lost time in each cycle is independent of the cycle length, a shorter cycle will mean that a higher percentage of the cycle is wasted as lost time. If an intersection is near saturation, this strategy may actually increase delays; but if excess capacity exists, this strategy can reduce delays to individual vehicles. Split phasing is a related strategy in which the green phase for the transit corridor occurs twice within the same cycle. The cycle length can remain unchanged if each of the two green phases is half the length of the original phase. As when the cycle length is reduced, lost time is increased, but the increase will be smaller in this case because only one additional phase transition is added per cycle. This strategy benefits transit vehicles by reducing the amount of time between green phases, thus reducing the wait for vehicles encountering a red signal. Signal coordination is another strategy that can be used to benefit transit vehicles. Arterial progression, for example, can be designed to favor transit vehicles by timing the green band at the average transit vehicle speed instead of the average automobile speed, which is typically faster. Although this strategy increases the travel time for automobiles, it helps ensure that transit vehicles can keep pace with the signal progression. However, progression for city transit vehicles may be difficult to maintain due to the presence of transit stops, which prevent those vehicles from moving at a constant rate through the network. Because the dwell time at transit stops is variable, static signal settings can not ensure proper progression. A general problem with passive priority strategies is that they typically make the intersection operate less efficiently overall, especially if transit frequency is not very high. This is because the signal settings will be sub-optimal when transit vehicles are not present, which will be the case the large majority of the time. For this reason, such strategies may not always be feasible, especially under highly saturated conditions. In 30 such cases, using a shorter cycle length or larger transit splits may lead to over-saturation of the intersection, leading to long queues and delays. Although passive priority strategies have definite limitations, in many cases they are the only viable options, especially when cost considerations require the use of existing traffic controller hardware. However, the amount of recent research into passive strategies is minimal and mostly general in nature[33], reflecting the limited value of such strategies.
2.2.2 Active Priority Strategies

Active strategies address these limitations of passive strategies by altering signal settings dynamically and only when necessary, making adjustments in real-time to the signal timing in order to minimize delay to an approaching transit vehicle. This is more infrastructure intensive than passive priority, requiring devices to detect transit vehicles upstream of the intersection and advanced controllers to employ strategies for granting priority.

There are three basic actions that a controller can perform in response to the detection of a transit vehicle: extension of the green interval in the current phase, ending another phase early to give an early green to the vehicle, and inserting an extra phase to allow the vehicle to pass before returning to the regular timing. The response used will depend on when in the cycle the vehicle is detected.

If the vehicle is approaching the intersection near the end of the green interval for its approach, the current interval can be extended until the vehicle has passed through the intersection, as shown in Figure 2-11. Without extension, the vehicle would have to wait for green in the next cycle, leading to significant delay.

If the transit vehicle is approaching a red signal, the two other actions can be used. In the case where the vehicle will normally get a green in the next phase, the current phase can be ended early to allow the vehicle to get an early green. This is possible if the vehicle will arrive at the signal near the end of the red period for its approach, as shown in Figure 2-12.
If other phases need to be served before the normal return to green, a short phase for the transit approach can be inserted, with the controller returning to normal operation once the vehicle has passed. Such a case is shown in Figure 2-13, where the controller breaks from its normal plan to serve the transit phase before returning to the regular signal timings.
A major concern with active priority strategies is the effect they have on other traffic. Under light traffic conditions, active priority may have little effect on other traffic because excess capacity within the cycle can be redistributed to the transit phase. However, active priority can have major negative effects in peak period operations, when intersections are operating near saturation with little or no time to spare from the non-transit movements. Both simulation studies and field tests have demonstrated the detrimental effect on cross-streets, especially those near saturation [20] [22]. Due to these effects, the system-wide value of active transit priority may only be worthwhile if transit has a high ridership in the corridor, causing the benefits per person to outweigh the costs. Facility design can also present problems in implementing active priority strategies. For example, near-side transit stops (i.e. those placed just upstream of an intersection) complicate active priority because the green phase may be extended unnecessarily while the vehicle is held at the stop. Even if the dwell time is taken into consideration, because this dwell time is variable, the extension required will always be uncertain. For this reason, far-side bus stops are preferable when active priority strategies are employed. Unlike for passive priority, much research is being devoted to the development and
analysis of active priority strategies. These strategies fall into three general categories: unconditional, conditional, and adaptive.

**Unconditional Strategies**
An unconditional strategy is one which gives priority status to every transit vehicle detected, meaning that the signal controller will attempt to initiate one of the priority actions described above upon detection of any transit vehicle. The disadvantage of this strategy is that priority may be granted unnecessarily, such as to a vehicle that is ahead of schedule. However, unconditional priority requires no further information other than the presence of the vehicle to be transmitted to the signal controller, which makes it the only option for transit systems with limited communication capabilities.

**Conditional Strategies**
Conditional strategies grant priority status based on certain criteria, in most cases properties of the specific transit vehicle. The most common criterion for conditional priority is the lateness of the vehicle relative to its schedule. However, further criteria such as vehicle headway (i.e. the time between successive vehicles) or passenger load are being considered for future applications.

The advantages of conditional strategies over unconditional strategies are demonstrated in research by Furth and Muller [20]. In a field test in Eindhoven, the Netherlands, comparisons were made between conditional priority, unconditional priority, and the base case of no transit priority. While the unconditional strategy had the most reduction in travel time for the transit vehicles, moving to a conditional strategy led to major improvements in service to other vehicles with only a small sacrifice in operating speed to the transit vehicles. Other benefits of
Development of conditional priority strategies in recent literature focus maximizing schedule adherence of buses while minimizing impacts on other traffic [33]. Strategies have also been developed for use in specific applications and under constraining external conditions. For example, a framework for integrating transit priority with arterial signal progression, developed by Vasudevan and Chang [36] considers the schedule delay of the transit vehicle, delay caused by interruption of arterial progression, and vehicle queue lengths in the determination of the control decision. Other strategies focus on incorporating bus priority into existing controller hardware and software, which can place significant restrictions on the priority implementation [5].

**Adaptive Strategies**

Adaptive transit priority strategies are those which use optimization-based control schemes to determine if and how to grant priority. In such schemes, the delay of the transit vehicle is considered along with the delay faced by all other vehicles. The controller then calculates the optimal solution for how to allocate time between the competing approaches. Because phases and timings are not fixed, adaptive strategies do not require predefinition of specific priority actions, such as phase extension or insertion, as the controller is constantly changing the allocation of green time based on demand. Transit priority strategies can easily be implemented within most existing adaptive systems by giving more weight to transit vehicles in the optimization routine. For example, weighting transit vehicles by 50 will mean that the controller will treat the bus as if it were 50 cars, thereby favoring that vehicle’s approach in the optimization [32].

However, implementing transit priority within existing adaptive control strategies has certain flaws [12]. A general problem is that adaptive control systems consider network-wide effects in their optimization, while providing transit priority is a local controller concern. This may lead to conflicting goals in the optimization and therefore sub-optimal results. Another problem is that most adaptive control systems use macroscopic models of traffic flow in their estimation and
optimization routines, and these models can not capture certain details of transit vehicle movements. For example, dwell time at transit stops and interactions between the transit vehicle and other vehicles will not be considered, so travel time for transit vehicles may be underestimated. Finally, constraints on the optimization may limit the opportunity for transit priority, especially during peak hours when transit priority is most essential. For example, many systems have constraints on allowable queue lengths for the intersection approaches. During peak demand these constraints may be limiting, such that no extra time can be taken from other approaches and given to the transit movement. This may essentially eliminate the possibility of transit priority under certain conditions, especially in peak conditions when priority is most needed.

Recent research has addressed these issues with the development of adaptive strategies that focus on transit priority at the intersection level [10][12].

Implementation of Transit Signal Priority
The model for the implementation is PRIBUSS, an advanced bus priority strategy developed and employed in Stockholm, Sweden. This section details the logic of PRIBUSS and how it is modeled within the framework of a generic controller.

2.3 PRIBUSS
PRIBUSS is an active signal priority strategy for buses that was developed for use in the city of Stockholm, Sweden. Its name is an acronym for “Prioritering av Bussar i Samordnade Signalsystem,” or “Prioritization of Buses in a Coordinated Signal System.” The strategy was developed by Gatu- och Fastighetskontoret (GFK), the administration in charge of traffic planning and operations in the city, in cooperation with Storstockholms Lokaltrafik (SL), Greater Stockholm’s local public transit agency. The objective of the system is to provide priority to public transit buses without significant disruption of signal operations, especially under coordinated control.
2.3.1 General Concepts

There are four functions that PRIBUSS employs in order to grant priority. Three of these functions are equivalent to the basic active priority actions described in Section 2.2.2. “Green Extension” extends the green period for enough time to allow the bus to pass through the intersection. “Phase Shortening” ends the current phase early in order to start the bus phase early. “Extra Phase Insertion” gives green to the bus out of sequence, adding an additional phase for the bus. The fourth action, “Restart Green,” is a variant of Green Extension in which a signal whose green period has just ended will repeat the green period for enough time for an approaching bus to pass. These actions will be described in more detail in the following section.

Implementation of the PRIBUSS strategy requires the detection of buses upstream of the intersection in order to start any priority action. When a bus crosses an upstream detector, this “check-in” detector indicates the presence of the bus to the signal controller. In order for the controller to know when the bus has passed the intersection and is no longer in need of priority, a “check-out” detector is also needed in or immediately downstream of the intersection. The controller keeps count of the number of buses between the check-in and check-out detectors by means of a counter. This counter is increased by 1 when the check-in detector is actuated and is decreased by 1 when the check-out detector is actuated. Priority is only called for when the counter has a value of 1 or more. When the value of this counter becomes 0, indicating that no more buses are immediately upstream of the intersection, any priority action underway will be stopped.

The type of priority action called for depends on when in the cycle the bus is detected. As an example, Figure 2-14 shows the valid times for each priority call superimposed on a signal group diagram.* Group 1 is the priority group, meaning that the bus has right-of-way when this group is in green. Group 2 is the group which displays green immediately following the priority group, and Group 3 is the group which displays green immediately preceding the priority group. If a bus is detected during the green period of Group 1, it can call for green extension. However, the priority action may or may not be needed, depending on when in the period the bus is detected. For example, a bus detected at the start of the green period will most likely not need any priority action to be taken, as adequate green time will exist for it to clear the intersection.
A bus that is detected near the end of the green period, however, will require green extension in order to make it through the intersection before the end of the period.

Figure 2-14: Time windows for priority calls.

A call for green restart can only occur once the green period has ended and before the starting red/yellow indication of the following group is displayed. If a bus is detected during this window, Group 1 can return to green to allow the bus to pass. Once Group 2 has started, the next available priority action is extra phase insertion, which can be called for until Group 3 becomes active. A bus detected during this time will be served by a green period inserted between the green periods of Groups 2 and 3. Once Group 3 has started, a detected bus will call for phase shortening, which will terminate the green period of Group 3 early to give an early start to Group 1. No priority call is needed once Group 3 has ended, because the green period of the priority group is the next in sequence. A large number of parameters is required to implement the PRI_BUSS strategy in a signal controller. However, a smaller set of parameters governs the key details of the implementation and is common to all priority actions:

- Guaranteed Time: Each signal group has a minimum duration during which any conflicting priority calls are inhibited. This ensures a minimum service time for non-priority phases.
• Detector Locations: Each priority group requires a bus detector upstream of the intersection to indicate the presence of the vehicle. The distance of the detector from the stop line is used to estimate the arrival time of a vehicle to the intersection once it has been detected.

• Allowed Window: The window within which priority calls can be registered is specified by start and end times in each cycle. These values will depend on multiple factors, including the estimated travel time of the bus from its detection point to the intersection and the time available from other signal groups for priority actions.

2.3.2 Priority Actions
This section provides detailed explanations of the four priority actions that can be implemented as part of the PRIBUSS strategy.

Green Extension
If a bus is detected during the green period for its approach and that period is about to end, priority can be given by means of green extension. This function will hold the green period for the approaching bus in order to give the bus enough time to pass through the intersection. Once the bus has passed, the period will terminate and the controller will return to the normal timing.

Figure 2-15 shows the method by which green extension is implemented. In this diagram, the vertical axis represents time. On the left side of the figure is a time-space diagram that shows the locations of the check-in and check-out detectors. On the right side of the figure is a signal group diagram that shows signal indications over time. The signal groups shown in the diagram are the two groups affected by the priority action: the priority group, and the following conflicting group that receives a delayed start.
Extension is only useful to buses which arrive at the intersection between times C and D. Time C is the normal ending time for the green period as specified in the signal timings. Buses arriving earlier than time C can cross the intersection during the normal green period and do not require additional extension time. Time D is the latest ending time allowed for the green period. This point is determined by the latest time the following signal group can start and still receive its guaranteed green time. Buses arriving after time D can not be given green extension because no more time can be taken from the following group.

Because the check-in detectors are typically located 100-200 meters upstream of the intersection, there is a delay between when the vehicle is detected and when it reaches the intersection. This delay is equal to the travel time between the check-in detector and the intersection. Because of this delay, a bus that crosses the check-in detector before time C may reach the intersection after time C, requiring green extension. Therefore, the time period in which a detected bus will receive priority must be shifted earlier to account for this delay. This is represented by the time period A-B in the figure. A bus that checks in during this time period
will reach the intersection in the C-D time period and will require priority. This assumes, however, that the assumed travel time between the detector and the intersection is correct. In operation, this travel time will vary due to traffic and other conditions, so a longer travel time is usually assumed in order to be conservative.

A further distinction is made between the time period in which priority can be initiated and the time period in which an ongoing priority action can be continued with the detection of another bus. At time C, the bus phase will end if a call for extension has not been initiated. Therefore, time period A-C is the window in which extension can be initiated. If extension is already ongoing, however, additional buses detected before time B will further extend the period.

Shortening of Current Phase

If a bus is detected during the red period for its approach and the bus is due to receive green in the next scheduled phase, the current phase can be shortened to allow the next phase to receive green earlier than normal. Figure 2-16 shows how this priority action is implemented. The signal groups shown are the two groups affected by the priority action: the priority group, and the preceding conflicting group that is terminated early.

The time window in which shortening can act is defined by period C-D in the diagram. Time C is the earliest time that the green period for the bus can start. This point is determined by the guaranteed time for the preceding signal group, which defines the earliest time that its green period can end. Time D is the normal start time for the bus green period, at which time shortening no longer applies.

The time window in which detection of a bus will call for shortening is defined by time period A-B. Time A, which is the earliest time a call for shortening can be made, is equal to the start time of the preceding phase. A bus detected before time A can receive an inserted phase, so shortening is not required. A bus detected after time A will call for priority; however, priority will not be granted until time C, when the guaranteed time for the preceding phase has been met. Time B, which is the latest time a call for shortening can be made, is equal to the normal end time of the preceding phase. At time B, the transition to the bus’s normal phase begins, so a bus detected after time B will receive green at the usual time.
**Figure 2-16: Execution of shortening of current phase.**

Insertion of Extra Phase

If a bus is detected during the red period for its approach and the bus is not due to receive green in the next scheduled phase, an extra phase for the bus approach can be inserted between the current phase and the next phase. Figure 2-17 shows how phase insertion is implemented. Three signal groups are affected by the phase insertion and are shown in the diagram: the priority group being inserted, the preceding group that is terminated early, and the following group which receives a delayed start. The time window in which the extra phase can be inserted is defined by period C-D in the diagram. Time C is the earliest time that the green period for the inserted bus phase can start. This point is determined by the guaranteed time for the preceding signal group, which defines the earliest time that its green period can end. Time D is the latest time that the green period for the inserted bus phase can end. This
point is determined by the latest time the following signal group can start and still receive its guaranteed green time.

The time window in which detection of a bus will call for an inserted extra phase is defined by time period A-B. Time A, which is the earliest time this call can be made, is equal to the start time of the preceding phase. A bus detected before time A can receive a green restart, so insertion is not required. A bus detected after time A will call for priority; however, priority will not be granted until time C, when the guaranteed time for the preceding phase has been met. Time B, which is the latest time a call for insertion can be made, is equal to the normal start time of the following phase. A bus detected after time B may call for phase shortening or insertion between two other phases, but it can not be served until after the active phase has received its guaranteed time.

Green Restart
If a bus is detected when the green period for its approach has just ended, the green period can be restarted to allow the bus to pass through the intersection without waiting for the next phase. Figure 2-18 shows how this priority action is implemented. As with green extension, the two affected groups shown are the priority group and the following conflicting group that receives a delayed start. The time window in which green restart can act is defined by period C-D in the diagram. Time C is the time at which the bus period can restart green, taking into account the red clearance time and the starting red/yellow indication.* Time D is the latest time that the restarted green period can end. This point is determined by the latest time the following signal group can start and still receive its guaranteed green time.

As with green extension, there is a distinction between the time period in which green restart can be initiated (A-B1) and the time period in which the restarted period can be continued (A-B2). Time A, which is the earliest time this call can be made, is equal to the normal end time of the green interval of the priority phase. A bus detected before time A can receive green

* Time C is the time at which the bus period can restart green, taking into account the red clearance time and the starting red/yellow indication.
extension, so green restart is not required. Time B1, which is the latest time green restart can be initiated, is equal to the normal start time of the red/yellow period of the following signal group. If green restart has not been initiated by time B1, the following period will begin and green restart will no longer be possible. If green restart is already ongoing, however, additional buses detected before time B2 (equal to time D less the travel time between the detector and the intersection) will extend the restarted period. Additional buses detected after time B2 will not extend the period because they will not be able to reach the intersection before the restarted phase must be terminated.
Chapter Three  Methodology

Introduction

In this chapter, the researchers give an outline of the research design and describe how a C++ simulation was used in the study. An algorithm and new protocol are designed to control the traffic flow at a traffic intersection where the robots are malfunctioning or non-existent.

3.0 An adaptive transit priority strategy approach

As outlined in the previous chapter, Adaptive transit priority strategies are those which use optimization-based control schemes to determine if and how to grant priority. In such schemes, the delay of the transit vehicle is considered along with the delay faced by all other vehicles. The controller then calculates the optimal solution for how to allocate time between the competing approaches. Because phases and timings are not fixed, adaptive strategies do not require predefinition of specific priority actions, such as phase extension or insertion, as the controller is constantly changing the allocation of green time based on demand. Transit priority strategies can easily be implemented within most existing adaptive systems by giving more weight to transit vehicles in the optimization routine. For example, weighting transit vehicles by 50 will mean that the controller will treat the bus as if it were 50 cars, thereby favoring that vehicle’s approach in the optimization [32]. We adopt this approach and weight each vehicle by the number of passengers it will carry. The rationale being that we aim to reduce delay per person in our adoption of advanced signal priority strategy. Thus a two sitter will have a weight of two whilst a mini-bus carrying eighteen passengers will have a weight of eighteen; consequently a bus with a full capacity of seventy five will have such a weight. However trickier situation arise when other special vehicles are involved whose prioritization can not be weighted in terms of the number of people it can carry. As one poet has mentioned, ‘... all animals are equal, but some are more equal than others.’ [Animal Farm] How much weight would one give an ambulance with a man fighting for his last breathe and needs medical attention, more so, a firemen’s van when there is a call of duty and what more for presidential
motorcade. The weighting for the vehicles are then a matter of government policy (Ministry of Transport) and legislative rather than a concept of science that the researchers have to deal with. Without lost of generality, where a specialized vehicle is involved, a prohibitively large value of weight shall be associated with the node element.

We, need to come up with an intelligent transportation system that will implement a control system which is going to be simulated in a C++ simulation and then propose a frame work for which this simulation can be done in an NS2 simulator. Appendix B, outlines from the researchers capability, how such simulation can be performed within a Network simulator such as NS2.

To understand what we aim to do, consider a manned intersection with robots that have failed. A policeman (Traffic Controller) stands in the middle of the intersection and directs traffic according to his gut-feeling and heuristics to control the traffic. His goal is to regulate the traffic at the intersection, to avoid collisions (accidents) and at the same time avoid congestion of the road network which may result in inconveniencing delays to commuting public and private vehicles. He also aims to offer prioritization to public transport system so as to reduce delay per person by a considerable amount. Safety of the public is unquestionably his greater concern so that Ambulance and other public safety vehicles like the firemen vehicles and worse still presidential motorcade will explicitly be given highest prioritization. Such a system is obviously as intelligent as the man in the middle controlling it. This is obviously very tedious and cumbersome for the officer who some times suffers a lot of fatigue, gets hungry on duty, a few exchanges of cursing words with the driving public biases his next decision of whom to give the right of way. His mistakes cascade into agonizing traffic jams, and lots of honking and cursing. What if that guy in the middle can be substituted? Somebody more rugged, but deaf and dumb could take up his job. Someone who will not feel sleepy or lapse in decision-making employed 24 hours a day and 7 days a week and only biased by the heuristics that govern his domain of expertise – regulating traffic. The policeman can only make decisions based on his short sight, or as far as he can see and can not see a car or type of car that joins the queue or network segment in the horizon. Comparatively, this substitute system can be made to have such long
sight and detail about all the network segments and can be up to date in real time even for stretches of more than a couple of kilometers should there be traffic queues at an intersection that would stretch to that much. Any vehicle that would join the network segment are noted instantly and their properties noted in real time.

This bring us to a couple of our research questions

- Can vehicles be made to be conscious of the presence of others in a road network?
- Can the road network be made to be conscious of the presence of the vehicles?

In answer to the above questions, we propose the use of VANET gadgets to be integrated into automobiles. These gadgets will transmit and receive messages between these mobile nodes. The messages can either be UDP or TCP/IP packets that will be exchanged and routed by other nodes till they reach their intended destination. UDP is going to be preferred since we assume that waiting for an ACK from the recipient will only result in unnecessary propagation delay in real time. If a mobile node sends a packet and is lost, the node may be considered to have been dropped without seriously affecting the operation of the system.

This means that if mobile nodes using UDP or TCP/IP can communicate these messages then utilities offered by the same protocols can be used to infer or ‘finger’ something about the nodes. For example a hop count between a sender and receiving node can be used to estimate the number of cars within that node.

Can then some intelligence be integrated into traffic regulation and control systems?

This means with some ingenuity some intelligence can be simulated for Traffic Control Systems. A real time responsive system can be designed and implemented for vehicular ad-hoc networks that can coordinate an intelligent transportation system. Considering the physical proximity of vehicles at an intersection, the packet loss due to signal weakness can be considered very minimal to be neglected. Also considering, the amount of calculation involved, the computations are simple arithmetic so that the response time for the system is within tolerable range, that a real implementation of the system will not endanger public healthy and safety.
We focus the research to traffic regulation at what could be a traffic light controlled intersection. The rationale being, most of commuting time is lost at these intersection, if not life. Traffic delays in straight lanes and road ways. Road intersections become the sole rate determining step in delay achieved to travelling public, and a research into how we can reduce delay per person finds no better justification.

Algorithm
We propose a variation of a leader election algorithm. When vehicles approach and intersection they communicate their weighted values. Vehicles in the same node aggregate their weights together to contest in leader election process. They will be contesting for the right of way. Its is a survival of the fittest and the heavily weighted node wins. However a starvation avoidance is considered to avoid bias and lop sidedness this ideology entails.

To implement this algorithm we propose a means of weighting vehicles. The vehicles are weighted according to carriage capacity. Thus a four-sitter will carry a weight of four and a eighteen sitter bus will have eighteen for a weight. The weighting of an ambulance or a presidential motorcade is more a ‘political’ issue than it is scientific, thus it remains in the better judgment of the authorities than it is a persuasion of a thoughtful calculation. In this research a large number or weight is used to symbolize an approaching special vehicle or person. Thus a general approach to see or look at a vehicle or vehicles approaching an intersection as simple addition and subtraction of weights (in the case of vehicles leaving an intersection) is taken without consequences or deviation from the goal of the strategy. If suppose, a firemen’s truck is weighted at 50, then 10 cars each with a carriage capacity of 5 resulting in an aggregated weight of 50 will be an equivalence. We may choose to say that if an ambulance approaches with a patient fighting for his last breathe then he gets absolute priority. This simply means giving it a weight exorbitantly large that no other road segment can aggregate its node weight to surpass it. Thus without loss of generality, all vehicles will be
treated as node elements with a value they contribute to the total node weight of the segment to which the vehicle is part of.

We now define road intersection in an object-oriented approach as a class object which can be instantiated. The class declaration is shown below for a type Junction, since a road intersection is a junction.

```
• class Junction
  • {
    •   public:
    •     friend void setWeight(Junction x, int y);
    •     friend int getGreatest(int noOfWings);
    •     static int nodeWeight[4];
    •     Node wing[4];
    •     void add(ofstream& of, int x);
    •     Junction (int);
    •     int getMax(int) const;
    •   }
```

*Table 3-1: Junction Class definition.*

We start by the constructor Junction(int), which defines a constructor that accepts one parameter of type int specifying the number of wings the intersection will have. A deeper consideration of this approach allows us to consider these wings as phases or signal groups. A reference to chapter 2.1.1 on Terminology may be necessary, to view the algorithm in this light. This means filters which are found on most intersections can be considered taken care of in the implementation of the algorithm. The full constructor definition can be found in appendix A from line number 116 to 128. Apart from initializing an instance of a junction object to the number of phases, signal groups or wings of an intersection, the same constructor, creates an intersection with no vehicles in all the nodes making sure that a clean object has been instantiated and ready to add to its nodes the weights of vehicles they add.
We define friend functions of the junction class as helper functions for the class

friend void setWeight(Junction x, int y);
friend int getGreatest(int noOfWings);

The full function definitions for the class can be found in appendix A but we give a brief explanation for the functions here as a black box. Like the name imply setWeight(), sets the weight of a given instance of a junction. All the nodes (as specified by the second parameter) of a junction (identified by the first parameter of type junction) will be set to their respective node weight whose values are looked up in the public variable of the class nodeWeight[]. This array is defined as a class variable as depicted by the key word static preceding the variable declaration so that the values held by the class instance are persistent throughout the program execution.

getGreatest() is a getter function returning the node with the greatest weight in the leader election process.

We nest a Node object inside the Junction class highlighting the hierarchical approach implied by this object-oriented approach. It makes a lot of sense to say that a junction has nodes i.e. wings or roads emanating or terminating in that intersection.

The member function add() defined for the junction is responsible for adding all the weights associated with node growth. As the function is called, random numbers are generated simulating the random nature in which vehicles adding to an intersection would come. The full function definition can be found in Appendix A from the lines of code 405 to 423. Each time this function is called, it writes to an output stream, the values of the nodes added so that statistical data is sampled for analysis.
We now outline the inner class or the nested class of the junction object: Node. The wings or the way of an intersection or junction are defined as a type Node Class. The definition of a Node Class is outlined below

```cpp
class Node {
public:
    int weight;
    int ID;
    int status;
    void request(ofstream& of, int x);
    void go(ofstream& of, int, Junction& j);
    void stop();
};
```

Table 3 - 2: Node Class definition.

Three integer variables are declared in the class: weight, ID and status. Intuitively, the weight refers to the aggregated weighted values of all vehicles within a particular node or wing; ID identifies the wing in which the vehicles are in. Lastly, the status is a Boolean keeping the track of whether that node has or should be granted the right of way or not. The value for this variable is checked with each call of the member function request() and go(). The constructor for the Junction class set these variables appropriately. That is the node ID are set correctly and all the nodes are blocked i.e. the status variable is set to zero /false and the node weights are initialized to zero. It’s a new junction and there are no vehicles as yet. Successive calls to the container class function Junction::add() modifies these variables. Having pre-empted something about the Node::request() function, we now add more flesh to the functions involvement in the definition of the node class. When a node gets a chance it passes its request to the junction object. Its right of way is not immediately granted but a leader election process is started. A call to getMax() is made and the node with the most dense weight is returned and a comparison is performed checking on whether the requesting node is actually the greatest.
More on the function can be learned by scrutinizing the function definition as it is given in the appendix A. The lines of code 150 to 170 outline the function completely. The requests that are made are then written to an output file for analysis of the algorithm in terms of how many requests were issued and how many times did a particular node got the right of way, denied or extended its green time.

The Node::go() is the engine of the algorithm. It is a simulation of how the node looses its weight given the right of way or green time. This function generates a random number stipulating the randomness of how vehicles approach and leave an intersection. Note that the decrease symbolizes a weight loss not the number of vehicles that leave, but without loss of generality, it may be taken to mean vehicles that leave, obviously, implying the assumption that all vehicles will have a single weighting of value 1. Starvation avoidance is handled by this function. Vehicles in the same road as the node with the greatest weight are simultaneously granted the right of way based on the simple principles of driving. When vehicles in the same road approach an intersection, then if one gets the right of way then the other one in the same road moves with it. These concepts can be extended to phases where group movements can be grouped in such way that vehicles in similar groups can be given the right of way simultaneously if the movements of one group will not affect the other vehicles. To avoid starvation for other road users, after each wing, phase, group movements other vehicles in the other road that may be infinitely blocked are given a short time to pass, which corresponds to the weight-loss that is subtracted from the vehicle nodes each time a member of the opposite wing is granted the right of way.
Pseudo code

While (true)
{
    for ( i =0; i< noOfWings; i++ )

    {
        roadIntersection.add(fout,noOfWings);
        setWeight(roadIntersection,noOfWings);// resets the weights for all the wings of the intersection
        roadIntersection.wing[i].request(freq,roadIntersection.getMax(noOfWings));
        roadIntersection.wing[i].go(fLeavers,noOfWings,roadIntersection);
        setWeight(roadIntersection,noOfWings);// resets the weights for all the wings of the intersection
        roadIntersection.wing[i].stop();

        delay(timeout);
    }
}
We now consider all the functions and member variables in the abstraction and encapsulation of their abstract class as we explain the pseudo code of the implementation of the algorithm.

The ellipsis extend the declarations and the header files that where used in the algorithm. The while loop directs the program flow. Looping forever to depict that from the moment a road intersection is made and commissioned it remains operational until further notice i.e. maintenance of the road or whatever the reasons may be. However, in the actual implementation, a control value is passed into the algorithm to allow sampling and testing of the algorithm.

A call to add() by the road intersection object adds weights to all the wings of an intersection. The random nature by which the function adds weight is a simulation of how vehicles ad-hoc approach an intersection. A call to this function is immediately followed by setWeight() helper function to reset the node weights following changes brought in by its predecessor. Then Node::request() is called by a specific node that has a turn to request for a right of way. The node may be granted or denied or if it still had the right of way then it might extend its green time. A call to go() will handle the movement of the vehicles at the intersection generating random numbers for the weights that a lot to simulate the negotiation of the intersection. Again a call to setWeight() is made to review state of the node after some movements. New node weights have to be set to associate correctly with each intersection. At the end of each phase a call to stop() is made so all the nodes are blocked to allow a non conflicting resume as the execution loops indefinitely. The function delay() is simply to give a time delay before resumption of the simulation, so that output information can be read from the standard output of the program.

The results and analysis of this research are studied in chapter 4. The full source code for this implementation is found in Appendix A.
Chapter Four  Presentation, Analysis and Interpretation

Introduction

In this section we present the results, and findings of this research. These statistics that are used are drawn from the simulation results outputted by the implementation. We start looking at the presentation by first explaining the user interface.

4.1 User Interface

The user interface, like for all C++ programs that are non graphical, is a DOS like looking output screen. It is text based allowing the interaction of the scientist and the program. That is he can set some environment parameters such as the number of wings the intersection can have and the number of sampling results he intents to use per simulation and the delay in seconds required for program to halt so he can take a look and follow up the program execution. A snapshot of the program execution is shown in the figure 4.1 below
4.2 Output Files
Three CSV files are outputted by a running instance of this code or simulation. Statistical data is sampled from these files that are created dynamically at run time: Weightloss.csv, WeightsAddedVsTime.csv and finally requests.csv. It is from these files that all the raw data is drawn. The file WeightLoss.csv is written by a every system call to Node::go(). Its is meant to depict the changes to the variations of weight loss with time or with every request that has been granted. A further analysis to the contents of file is given shortly in this section. As the vehicles approach an intersection node weights are added with time or every successive call to the function Junction::add(). To monitor and solicit data for analysis, a stream has been opened such that this information is written to an output file. Figure 4.2 show a sampled data set demonstrating the random nature in which nodes are added to an intersection. This manages to eliminate bias in the execution of the program. Different sets of data can be generated from different program execution but the outputted data is able to demonstrate the variation in the node weight that is added to a particular node.
Figure 4-2: Weights added Vs Time

A graph showing how the node weight grew versus the time is shown in figure 4.3 below. A sample of 20 values was taken. After the simulation was run for those 20 time units, a closer look at the average node weight for the nodes show that the weights of the nodes tend to converge towards a mean.

Figure 4.3 below shows the final values of the node weights at the end of 20 time units of the simulation. We now define the node values of the weights as variation of a random variable X.
Let $X$ be a random variable with mean value $\mu$:

$$E[X] = \mu.$$ 

Here the operator $E$ denotes the average or expected value of $X$. Then the standard deviation of $X$ is the quantity

$$\sigma = \sqrt{E[(X - \mu)^2]}.$$ 

That is, the standard deviation $\sigma$ (sigma) is the square root of the average value of $(X - \mu)^2$.

The table below is used to tabulate the mean and standard deviation from the mean for sampled values from the table of results drawn from figure 4.3.

<table>
<thead>
<tr>
<th>Node 0</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
<th>$\mu$</th>
<th>$\delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
<td>1</td>
<td>7</td>
<td>4.5</td>
<td>4.123106</td>
</tr>
<tr>
<td>7</td>
<td>12</td>
<td>0</td>
<td>9</td>
<td>7</td>
<td>5.09902</td>
</tr>
<tr>
<td>12</td>
<td>19</td>
<td>6</td>
<td>28</td>
<td>16.25</td>
<td>9.464847</td>
</tr>
<tr>
<td>32</td>
<td>33</td>
<td>9</td>
<td>33</td>
<td>26.75</td>
<td>11.84272</td>
</tr>
<tr>
<td>39</td>
<td>31</td>
<td>14</td>
<td>36</td>
<td>30</td>
<td>11.16542</td>
</tr>
<tr>
<td>42</td>
<td>32</td>
<td>15</td>
<td>27</td>
<td>29</td>
<td>11.22497</td>
</tr>
<tr>
<td>21</td>
<td>35</td>
<td>23</td>
<td>37</td>
<td>29</td>
<td>8.164966</td>
</tr>
<tr>
<td>28</td>
<td>46</td>
<td>39</td>
<td>33</td>
<td>36.5</td>
<td>7.767453</td>
</tr>
<tr>
<td>39</td>
<td>61</td>
<td>39</td>
<td>45</td>
<td>46</td>
<td>10.3923</td>
</tr>
<tr>
<td>64</td>
<td>63</td>
<td>56</td>
<td>63</td>
<td>61.5</td>
<td>3.696846</td>
</tr>
<tr>
<td>65</td>
<td>61</td>
<td>72</td>
<td>65</td>
<td>65.75</td>
<td>4.573474</td>
</tr>
<tr>
<td>78</td>
<td>74</td>
<td>86</td>
<td>85</td>
<td>80.75</td>
<td>5.737305</td>
</tr>
<tr>
<td>80</td>
<td>87</td>
<td>81</td>
<td>85</td>
<td>83.25</td>
<td>3.304038</td>
</tr>
</tbody>
</table>
The results of the simulation show that as the simulation runs the standard deviation of the nodes from the mean generally falls showing a tendency of the algorithm to balance or consider fairly the weights of the nodes from all sides. As the results of the table show, the standard deviation changes from 4.1 up to 11.8 by the time the simulation gets to the middle but falls back to 6 towards the end of the simulation. This shows that the standard deviation is not monotonically increasing as can be the expectation, if the algorithm is not implemented.

Figure 4.4 show the trend for the variation of the standard deviation of the node weight with time. A trend analysis using moving averages generally shows a decline in the standard deviation with time.
Figure 4-3 Node Weight growth Vs time
As a direct consequence of this, the graph showing the variation of node weighting with time suggest a strong positive correlation coefficient and a relation that is linear. We establish this fact using Pearson’s Product moment coefficients.

**Pearson's product-moment coefficients**

The most familiar measure of dependence between two quantities is the Pearson product-moment correlation coefficient, or "Pearson's correlation." It is obtained by dividing the covariance of the two variables by the product of their standard deviations.

The population correlation coefficient $\rho_{X,Y}$ between two random variables $X$ and $Y$ with expected values $\mu_X$ and $\mu_Y$ and standard deviations $\sigma_X$ and $\sigma_Y$ is defined as:
\[ \rho_{X,Y} = \text{corr}(X, Y) = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}, \]

where \( E \) is the expected value operator, \( \text{cov} \) means covariance, and \( \text{corr} \) a widely used alternative notation for Pearson's correlation.

We now calculate Pearson product-moment correlation coefficients for the four nodes depicted in the data set for node 0 right through to node 3.

The correlation coefficient for Node 0 is given in the table below.

<table>
<thead>
<tr>
<th>Node weight (X)</th>
<th>Time Unit (Y)</th>
<th>X-μ</th>
<th>Y-μ</th>
<th>(X-μ)(Y-μ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>-61.65</td>
<td>-9.5</td>
<td>585.675</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>-55.65</td>
<td>-8.5</td>
<td>473.025</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>-50.65</td>
<td>-7.5</td>
<td>379.875</td>
</tr>
<tr>
<td>32</td>
<td>4</td>
<td>-30.65</td>
<td>-6.5</td>
<td>199.225</td>
</tr>
<tr>
<td>39</td>
<td>5</td>
<td>-23.65</td>
<td>-5.5</td>
<td>130.075</td>
</tr>
<tr>
<td>42</td>
<td>6</td>
<td>-20.65</td>
<td>-4.5</td>
<td>92.925</td>
</tr>
<tr>
<td>21</td>
<td>7</td>
<td>-41.65</td>
<td>-3.5</td>
<td>145.775</td>
</tr>
<tr>
<td>28</td>
<td>8</td>
<td>-34.65</td>
<td>-2.5</td>
<td>86.625</td>
</tr>
<tr>
<td>39</td>
<td>9</td>
<td>-23.65</td>
<td>-1.5</td>
<td>35.475</td>
</tr>
<tr>
<td>64</td>
<td>10</td>
<td>1.35</td>
<td>-0.5</td>
<td>-0.675</td>
</tr>
<tr>
<td>65</td>
<td>11</td>
<td>2.35</td>
<td>0.5</td>
<td>1.175</td>
</tr>
<tr>
<td>78</td>
<td>12</td>
<td>15.35</td>
<td>1.5</td>
<td>23.025</td>
</tr>
<tr>
<td>80</td>
<td>13</td>
<td>17.35</td>
<td>2.5</td>
<td>43.375</td>
</tr>
<tr>
<td>94</td>
<td>14</td>
<td>31.35</td>
<td>3.5</td>
<td>109.725</td>
</tr>
<tr>
<td>76</td>
<td>15</td>
<td>13.35</td>
<td>4.5</td>
<td>60.075</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>99</td>
<td>16</td>
<td>36.35</td>
<td>5.5</td>
<td>199.925</td>
</tr>
<tr>
<td>107</td>
<td>17</td>
<td>44.35</td>
<td>6.5</td>
<td>288.275</td>
</tr>
<tr>
<td>122</td>
<td>18</td>
<td>59.35</td>
<td>7.5</td>
<td>445.125</td>
</tr>
<tr>
<td>122</td>
<td>19</td>
<td>59.35</td>
<td>8.5</td>
<td>504.475</td>
</tr>
<tr>
<td>125</td>
<td>20</td>
<td>62.35</td>
<td>9.5</td>
<td>592.325</td>
</tr>
</tbody>
</table>

.. | µx = 62.65 | µy = 10.5 |
.. | δx = 40.26789 | δy = 5.91608 |
.. | E((X-µ)(Y-µ)) = 219.775 |
.. | Pearson correlation coefficient |
.. | 219.775/40.26*5.91 = 0.92254 |

Table 4-2: Pearson correlation coefficient for Node 0.
<table>
<thead>
<tr>
<th>Node weight (X)</th>
<th>Time Unit (Y)</th>
<th>X-μ</th>
<th>Y-μ</th>
<th>(X-μ)(Y-μ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>1</td>
<td>-58.3</td>
<td>-9.5</td>
<td>553.85</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>-55.3</td>
<td>-8.5</td>
<td>470.05</td>
</tr>
<tr>
<td>19</td>
<td>3</td>
<td>-48.3</td>
<td>-7.5</td>
<td>362.25</td>
</tr>
<tr>
<td>33</td>
<td>4</td>
<td>-34.3</td>
<td>-6.5</td>
<td>222.95</td>
</tr>
<tr>
<td>31</td>
<td>5</td>
<td>-36.3</td>
<td>-5.5</td>
<td>199.65</td>
</tr>
<tr>
<td>32</td>
<td>6</td>
<td>-35.3</td>
<td>-4.5</td>
<td>158.85</td>
</tr>
<tr>
<td>35</td>
<td>7</td>
<td>-32.3</td>
<td>-3.5</td>
<td>113.05</td>
</tr>
<tr>
<td>46</td>
<td>8</td>
<td>-21.3</td>
<td>-2.5</td>
<td>53.25</td>
</tr>
<tr>
<td>61</td>
<td>9</td>
<td>-6.3</td>
<td>-1.5</td>
<td>9.45</td>
</tr>
<tr>
<td>63</td>
<td>10</td>
<td>-4.3</td>
<td>-0.5</td>
<td>2.15</td>
</tr>
<tr>
<td>61</td>
<td>11</td>
<td>-6.3</td>
<td>0.5</td>
<td>-3.15</td>
</tr>
<tr>
<td>74</td>
<td>12</td>
<td>6.7</td>
<td>1.5</td>
<td>10.05</td>
</tr>
<tr>
<td>87</td>
<td>13</td>
<td>19.7</td>
<td>2.5</td>
<td>49.25</td>
</tr>
<tr>
<td>96</td>
<td>14</td>
<td>28.7</td>
<td>3.5</td>
<td>100.45</td>
</tr>
<tr>
<td>92</td>
<td>15</td>
<td>24.7</td>
<td>4.5</td>
<td>111.15</td>
</tr>
<tr>
<td>92</td>
<td>16</td>
<td>24.7</td>
<td>5.5</td>
<td>135.85</td>
</tr>
<tr>
<td>115</td>
<td>17</td>
<td>47.7</td>
<td>6.5</td>
<td>310.05</td>
</tr>
<tr>
<td>125</td>
<td>18</td>
<td>57.7</td>
<td>7.5</td>
<td>432.75</td>
</tr>
<tr>
<td>127</td>
<td>19</td>
<td>59.7</td>
<td>8.5</td>
<td>507.45</td>
</tr>
<tr>
<td>136</td>
<td>20</td>
<td>68.7</td>
<td>9.5</td>
<td>652.65</td>
</tr>
</tbody>
</table>
\( \mu_x = 67.3 \quad \mu_y = 10.5 \)

\( \delta x = 40.0593 \quad \delta y = 5.91608 \)

\[
E((X-\mu)(Y-\mu)) = 222.6
\]

Pearson correlation coefficient

\[
\frac{222.6}{40.0593 \times 5.91} = 0.939264
\]

**Table 4 - 3: Pearson correlation coefficient for Node 1.**

<table>
<thead>
<tr>
<th>Node weight (X)</th>
<th>Time Unit (Y)</th>
<th>X-( \mu )</th>
<th>Y-( \mu )</th>
<th>(X-( \mu ))(Y-( \mu ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>-61.05</td>
<td>-9.5</td>
<td>579.975</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>-62.05</td>
<td>-8.5</td>
<td>527.425</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>-56.05</td>
<td>-7.5</td>
<td>420.375</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>-53.05</td>
<td>-6.5</td>
<td>344.825</td>
</tr>
<tr>
<td>14</td>
<td>5</td>
<td>-48.05</td>
<td>-5.5</td>
<td>264.275</td>
</tr>
<tr>
<td>15</td>
<td>6</td>
<td>-47.05</td>
<td>-4.5</td>
<td>211.725</td>
</tr>
<tr>
<td>23</td>
<td>7</td>
<td>-39.05</td>
<td>-3.5</td>
<td>136.675</td>
</tr>
<tr>
<td>39</td>
<td>8</td>
<td>-23.05</td>
<td>-2.5</td>
<td>57.625</td>
</tr>
<tr>
<td>39</td>
<td>9</td>
<td>-23.05</td>
<td>-1.5</td>
<td>34.575</td>
</tr>
<tr>
<td>56</td>
<td>10</td>
<td>-6.05</td>
<td>-0.5</td>
<td>3.025</td>
</tr>
<tr>
<td>72</td>
<td>11</td>
<td>9.95</td>
<td>0.5</td>
<td>4.975</td>
</tr>
<tr>
<td>86</td>
<td>12</td>
<td>23.95</td>
<td>1.5</td>
<td>35.925</td>
</tr>
<tr>
<td>81</td>
<td>13</td>
<td>18.95</td>
<td>2.5</td>
<td>47.375</td>
</tr>
<tr>
<td>92</td>
<td>14</td>
<td>29.95</td>
<td>3.5</td>
<td>104.825</td>
</tr>
<tr>
<td>Node weight (X)</td>
<td>Time Unit (Y)</td>
<td>X-µ</td>
<td>Y-µ</td>
<td>(X-µ)(Y-µ)</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------</td>
<td>------</td>
<td>------</td>
<td>------------</td>
</tr>
<tr>
<td>94</td>
<td>15</td>
<td>31.95</td>
<td>4.5</td>
<td>143.775</td>
</tr>
<tr>
<td>107</td>
<td>16</td>
<td>44.95</td>
<td>5.5</td>
<td>247.225</td>
</tr>
<tr>
<td>115</td>
<td>17</td>
<td>52.95</td>
<td>6.5</td>
<td>344.175</td>
</tr>
<tr>
<td>126</td>
<td>18</td>
<td>63.95</td>
<td>7.5</td>
<td>479.625</td>
</tr>
<tr>
<td>126</td>
<td>19</td>
<td>63.95</td>
<td>8.5</td>
<td>543.575</td>
</tr>
<tr>
<td>140</td>
<td>20</td>
<td>77.95</td>
<td>9.5</td>
<td>740.525</td>
</tr>
</tbody>
</table>

\[ \mu_x = 62.05 \quad \mu_y = 10.5 \]

\[ \delta x = 47.41139 \quad \delta y = 5.91608 \]

\[ E((X-\mu)(Y-\mu)) = 263.625 \]

Pearson correlation coefficient

\[ 263.625/(47.41139*5.91) = 0.939875 \]

Table 4-4: Pearson correlation coefficient for Node 2.
<table>
<thead>
<tr>
<th>Node</th>
<th>X</th>
<th>Y</th>
<th>X-μ</th>
<th>Y-μ</th>
<th>E((X-μ)(Y-μ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>6</td>
<td>-40.1</td>
<td>-4.5</td>
<td>180.45</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>7</td>
<td>-30.1</td>
<td>-3.5</td>
<td>105.35</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>8</td>
<td>-34.1</td>
<td>-2.5</td>
<td>85.25</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>9</td>
<td>-22.1</td>
<td>-1.5</td>
<td>33.15</td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>10</td>
<td>-4.1</td>
<td>-0.5</td>
<td>2.05</td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>11</td>
<td>-2.1</td>
<td>0.5</td>
<td>-1.05</td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>12</td>
<td>17.9</td>
<td>1.5</td>
<td>26.85</td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>13</td>
<td>17.9</td>
<td>2.5</td>
<td>44.75</td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>14</td>
<td>24.9</td>
<td>3.5</td>
<td>87.15</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>15</td>
<td>33.9</td>
<td>4.5</td>
<td>152.55</td>
<td></td>
</tr>
<tr>
<td>107</td>
<td>16</td>
<td>39.9</td>
<td>5.5</td>
<td>219.45</td>
<td></td>
</tr>
<tr>
<td>118</td>
<td>17</td>
<td>50.9</td>
<td>6.5</td>
<td>330.85</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>18</td>
<td>52.9</td>
<td>7.5</td>
<td>396.75</td>
<td></td>
</tr>
<tr>
<td>118</td>
<td>19</td>
<td>50.9</td>
<td>8.5</td>
<td>432.65</td>
<td></td>
</tr>
<tr>
<td>133</td>
<td>20</td>
<td>65.9</td>
<td>9.5</td>
<td>626.05</td>
<td></td>
</tr>
</tbody>
</table>

$\mu_x = 62.05$  \quad $\mu_y = 10.5$

$\delta x = 47.41139$  \quad $\delta y = 5.91608$  \quad $E((X-\mu)(Y-\mu)) = 223.65$

Pearson correlation coefficient

$223.65/(47.41139*5.91) = \mathbf{0.93353}$

Table 4 - 5: Pearson correlation coefficient for Node 3.
The analysis of Pearson correlation coefficients for all the nodes suggest that all the nodes are being treated fairly as all the nodes had comparable values of +0.9. Using student-T distribution or normal distribution at 5% at level of significance, the Pearson correlation coefficient is 0.93.

The similarity of this statistic for all the nodes is strong evidence of avoidance of starvation of an one node. Figure 4.6 below shows a snapshot of an output file generated by the simulation showing how starvation was being avoided for both traffic in the same as the node with right of way and for those in the other road that may be blocked indefinitely if suppose it a less busy road meeting with a main or busy road.

<table>
<thead>
<tr>
<th>Node</th>
<th>Before</th>
<th>Weight Lost</th>
<th>After</th>
<th>SAME ROAD</th>
<th>CARS LEFT</th>
<th>AVOIDED STARVATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Node: 0</td>
<td>1</td>
<td>0</td>
<td>Node:2</td>
<td>0 Node:1&amp;3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Node: 1</td>
<td>15</td>
<td>6</td>
<td>Node:3</td>
<td>4 Node:0&amp;2</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>Node: 2</td>
<td>1</td>
<td>0</td>
<td>Node:0</td>
<td>0 Node:1&amp;3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Node: 3</td>
<td>7</td>
<td>7</td>
<td>Node:1</td>
<td>0 Node:0&amp;2</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Node: 0</td>
<td>18</td>
<td>11</td>
<td>Node:2</td>
<td>6 Node:1&amp;3</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>Node: 1</td>
<td>12</td>
<td>12</td>
<td>Node:3</td>
<td>0 Node:0&amp;2</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>Node: 2</td>
<td>17</td>
<td>17</td>
<td>Node:0</td>
<td>3 Node:1&amp;3</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>Node: 3</td>
<td>9</td>
<td>9</td>
<td>Node:1</td>
<td>0 Node:0&amp;2</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>Node: 0</td>
<td>18</td>
<td>12</td>
<td>Node:2</td>
<td>8 Node:1&amp;3</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>Node: 1</td>
<td>19</td>
<td>19</td>
<td>Node:3</td>
<td>0 Node:0&amp;2</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>Node: 2</td>
<td>8</td>
<td>8</td>
<td>Node:0</td>
<td>0 Node:1&amp;3</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>Node: 3</td>
<td>23</td>
<td>23</td>
<td>Node:1</td>
<td>0 Node:0&amp;2</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>Node: 0</td>
<td>41</td>
<td>32</td>
<td>Node:2</td>
<td>9 Node:1&amp;3</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>Node: 1</td>
<td>28</td>
<td>28</td>
<td>Node:3</td>
<td>0 Node:0&amp;2</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>Node: 2</td>
<td>9</td>
<td>9</td>
<td>Node:0</td>
<td>0 Node:1&amp;3</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>Node: 3</td>
<td>53</td>
<td>20</td>
<td>Node:1</td>
<td>2 Node:0&amp;2</td>
<td>7</td>
</tr>
<tr>
<td>17</td>
<td>Node: 0</td>
<td>59</td>
<td>39</td>
<td>Node:2</td>
<td>8 Node:1&amp;3</td>
<td>7</td>
</tr>
<tr>
<td>18</td>
<td>Node: 1</td>
<td>43</td>
<td>31</td>
<td>Node:3</td>
<td>1 Node:0&amp;2</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>Node: 2</td>
<td>14</td>
<td>14</td>
<td>Node:0</td>
<td>0 Node:1&amp;3</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>Node: 3</td>
<td>36</td>
<td>36</td>
<td>Node:1</td>
<td>0 Node:0&amp;2</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>Node: 0</td>
<td>54</td>
<td>42</td>
<td>Node:2</td>
<td>2 Node:1&amp;3</td>
<td>5</td>
</tr>
<tr>
<td>22</td>
<td>Node: 1</td>
<td>41</td>
<td>32</td>
<td>Node:3</td>
<td>8 Node:0&amp;2</td>
<td>8</td>
</tr>
<tr>
<td>23</td>
<td>Node: 2</td>
<td>15</td>
<td>15</td>
<td>Node:0</td>
<td>0 Node:1&amp;3</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>Node: 3</td>
<td>45</td>
<td>18</td>
<td>Node:1</td>
<td>7 Node:0&amp;2</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 4-5: Starvation Avoidance.

Figure 4.6 below reiterates the same thing this time giving a pictorial view as a bar graph of all the nodes and how starvation was avoided.
We finally explain pictorial using the pie charts given in Figure 4.7, that despite the fact the nodes did not get equal number of green times, the algorithm was able to maintain a balance junction in terms of node weights. It simply means being denied a right of way is not negligence or favoritism but an ingenious way of handling an adaptive transit priority strategy. A snapshot of the output of the requests.csv output file is analyzed and given.
As can be seen from the pie chart Node 1 enjoyed the lion’s share of the green time with 37% or the requests being granted as compared to Node 2 and 3 put together with 34%.

**Space and Time complexity of the algorithm**

We conclude this section by looking at the time complexity of the algorithm. Looking back to the pseudo code given. We note that the algorithm flow control if done by the outer while loop and the inner for loop iterating for the number of nodes. Supposing some variable n is passed to outer while loop to control the number of iterations the program is going to loop and that inner for loop has the iterator running up to m the number of nodes or wings of an intersection, it becomes very easy to note that performs at least n*m executions. We categorically, state that the algorithm follows a Big O(n*m).
The space complexity of the algorithm is stated small $o(m)$ since the algorithm only needs to store as many values as the number of nodes corresponding to node weights that are used in the program execution.

Summary

This section outlined the presentation analysis and findings of this research, beginning with the user interface right through to the output files that this research used. Some statistical methods have been used to provide strong evidence of research findings and proposals. We wind up this chapter by looking at how the system was able to resolve the challenges that were posed to it in the sense of starvation. The space and time complexity analysis for the algorithm is given last concluding this section.
Chapter Five  Conclusion and Recommendations

5.1 Summary
This research has noted remarkable success in its findings; intentions are to integrate such findings and control logic into a network simulator such as ns-2 and or any other microscopic simulator for Vanets such as MITSIMLab so that pseudo real conditions can be tested for the protocol have been outlined. Appendix B gives a framework, from the researchers perspective, in which such a research can be performed with in a network simulator such as NS2 . A controller outside simulation has got to be developed and integrated into vehicles and a case study taken to verify the findings of this research in real world scenarios. Work in the proposed simulator has been tabled as future work and the researchers proposed steps are hinted in this document that accompanies this research.

The research has mainly focused on traffic control at road intersections in a drive to provide intelligent transportation system that would reduce delay per person for transit vehicles and provide an adaptive priority strategy that would not neglect other travelers on the road as bus or transit vehicles and any other specialized vehicles are prioritized. The research designs resulted in a generic control logic that allows the simulation of control strategy for different types of road intersection. To cater for filters and other complex intersections, the wings defining a node at an intersection can be conceptual changed to refer to each vehicle stream as a node. So an intersection will be treated as simply as a four or three winged intersection, but could be five or more depending on the in cooperation of filters. Instead of requiring each signal phase to be specified explicitly, the generic logic allows each vehicle stream to be controlled independently, thus allowing more flexibility in the control logic specification.

This logic of this generic controller is based on a set of basic logic elements that are common to all signal control strategies. These logic elements define actions that can be initiated and the criteria that must be met in order for these actions to occur. Inputs for the logic include internal timers, detector states, and the states of other signals and controllers. Specific control strategies can be simulated by combining and ordering these basic elements. This flexibility in
defining the logic allows the generic controller to simulate both pre-timed and actuated control and to model different control strategies, such as isolated intersection control, arterial coordination, and network control. While the generic controller supports a wide range of controller types, its modular structure also allows additional logic elements and functions to be added easily. This allows specialized functions of specific control strategies to be implemented, and it also provides a framework for implementation of other advanced traffic management strategies designed to operate within the constraints of coordinated signal operation, with the objective of providing transit priority with as little disturbance to the coordination as possible. The control logic algorithm is used as a foundation for the modeling of this strategy. Its implementation in C++ resulting in executable results in the simulation laboratory is used to evaluate the effectiveness of the algorithm under various conditions and to compare its performance under alternate parameter settings.

5.2 Findings
In the study, the C++ implementation of the proposed protocol or controller, logic for simulated conditions showed a strong positive correlation between node weight and the number of times green time was given to the node. Regardless of starting values of the node weights, the simulation being allowed to run for a given time, the weights of the node tend to converge and follow a steady growth of a linear function, which is interpreted as a fair balance or consideration of all network node in the way that green time or right of way is given to a road network segment or node.

The simulation has been able to show the random nature in which vehicles can leave and join intersection or node and showed even if the nodes do not get equal amounts of green time, the implementation was able to show fairness with all the trends of the variation of node growth with time giving a Pearson correlation coefficient of 0.93 for all the nodes.

A general conclusion from the evaluation is that the effectiveness of transit signal priority is very site- and condition-specific. Therefore, results from one application may not necessarily apply in different situations. For this reason, simulation is a valuable tool for the design and
evaluation of priority strategies, as it can be used to determine if and how such strategies can be most effectively implemented is also designed as a research tool for the evaluation of advanced control strategies. Adaptive control is a key area of research that can benefit from evaluation with simulation, as the cost of implementing such systems can limit the scope of field evaluations.

5.3 Future Research
The simulation for this research have been implemented within a C++ simulation and is currently able to simulate most basic control types and strategies that are in standard use. However, this controller is also designed as a research tool for the evaluation of advanced control strategies. Adaptive control is a key area of research that can benefit from evaluation with simulation, as the cost of implementing such systems can limit the scope of field evaluations. Similarly, network control strategies can be evaluated in a simulation environment, where measures of effectiveness can be collected much more easily than in the field.

An evaluation of a network with a longer bus corridor would be beneficial, as the longer travel times would allow better measures of schedule adherence and would allow studies of behavior such as bus bunching. Expanding the network to include multiple bus routes would also allow the modeling of more complex situations, such as the management of simultaneous priority calls on conflicting signal groups.

The use of more advanced transit priority strategies is another area in which simulations using MITSIMLab or NS2 can be applied. Conditional priority logic, for example, can be considered, with buses being granted priority on the basis of factors such as schedule deviation and passenger load. Adaptive signal control needs to be researched into and field-tested in the Harare city, and incorporating bus priority and emergency vehicles into such a system. MITSIMLab together with NS2, with the enhanced control capabilities provided by the generic controller, can now serve as a laboratory for the design and evaluation of these and other advanced traffic management systems. Appendix B at the end of this document gives a framework in which simulations can be performed within NS2.
Appendix A.
The complete Source Code and Implementation of the Algorithm

1. `# include <iostream.h>`
2. `# include <conio.h>`
3. `# include <stdlib.h>`
4. `# include <fstream.h>`
5. `# include <time.h>`
6. `# define BLOCKED 0`
7. `# define GO 1`
8. `# include <windows.h>`
9. `void delay (float timeout) //function declare to create 5s delays`
10. {
11.    time_t start;
12.    time_t current;
13.    time(&start);
14.    do{
15.        time(&current);
16.        while(difftime(current,start) < timeout);
17.    } while(difftime(current,start) < timeout);
18. }
19. class Junction;
20. class Node
public:

int weight;
int ID;
int status;

void request(ofstream& of, int x);
void go(ofstream& of, int, Junction& j);
void stop();

};

class Junction
{

public:

friend void setWeight(Junction x, int y);
friend int getGreatest(int noOfWings);
static int nodeWeight[4];
Node wing[4];

void add(ofstream& of, int x);
Junction (int);
int getMax(int)const;

};

int Junction::nodeWeight[4]={0,0,0,0};

int main ()
```c
55. {
56. float timeout;
57. ofstream fout;
58. ofstream freq;
59. ofstream fLeavers;
60. system("mkdir C:\thesis");
61. fout.open("c:\thesis\WeightsAddedVsTime.csv");
62. freq.open("c:\thesis\requests.csv");
63. fLeavers.open("c:\thesis\WeightLoss.csv");
64. if (fout.fail() || freq.fail() || fLeavers.fail())
65. {
66. cout<<"\nError creating file";
67. exit(1);
68. getch();
69. }
70.
71. int noOfWings;i,weight,timer;
72.
73. cout << "---------------------------------------------------------------------------------";
74. cout << "\nAn Intersection can have 3 or 4 Nodes or Wings";
75. cout << "\nSpecify whether it is a four or three way Junction :";
76.
77. cin >> noOfWings;
78. cout << "\nSpecify sampling time i.e 1 for 1s intervals :";cin>>timeout;
79. cout << "\nNumber of Samples: ";cin>>timer;
80. int noOfsamples=timer;
81.
82. cout << "---------------------------------------------------------------------------------";
83.
```
Junction roadIntersection(noOfWings);
for (int i=0; i<noOfWings; i++) fout << "Node:" << roadIntersection.wing[i].ID << ",";
fout << \\
; 
flavers,"<<"Before,"<<"Weight Lost,"<<"After"<<",SAME ROAD,CARS LEFT,AVOIDED STAVARTION" << \\
;
while (timer--)
{
    for ( i =0; i< noOfWings; i++ )
    {
        roadIntersection.add(fout,noOfWings);
        setWeight(roadIntersection,noOfWings);// resets the weights for all the wings of the intersection 
        roadIntersection.wing[i].request(freq,roadIntersection.getMax(noOfWings));
        roadIntersection.wing[i].go(flavers,noOfWings,roadIntersection);
        setWeight(roadIntersection,noOfWings);// resets the weights for all the wings of the intersection 
        roadIntersection.wing[i].stop();
    }
    delay(timeout);
}
fout.close();
freq.close();
flavers.close();
cout<<" \n!!!!!!!!!!!!Progam Halt !!!!!!!! !!!!!!!";
110.  getch();
111.  return 0;
112.
113.  }  
114.  
115.  
116.  Junction::Junction (int x)
117.  {
118.  
119.  for (int i=0; i<x; i++)
120.  {
121.  
122.    this->nodeWeight[i] = 0;
123.    this->wing[i].ID = i;
124.    this->wing[i].weight = 0;
125.    this->wing[i].status = BLOCKED;   // Create a junction and make sure all vehicles
126.    are not moving
127.  }
128.  
129.  
130.  }  
131.  
132.  int Junction::getMax(int noOfWings)const
133.  {
134.  
135.    cout<<"\n--------- Called Mukuru -------------" ;
136.    int mukuru= 0;
137.    for ( int i=0; i< noOfWings-1; i++)
138.    {
139.    
140.    if (Junction::nodeWeight[mukuru] > Junction::nodeWeight[i+1])
else mukuru=i+1;
}
else

cout << "\nNode: mukuru Returned with greatest Weight ";
return mukuru;
}

void Node::request(ofstream& fo, int x)
{

cout<"\n--------Node:"<<this->ID<" Requesting for Right of Way--------------------";

if (this->ID==x)
{

cout<"\nNode: this->ID has requested ";

if ( this->status==BLOCKED)
{

cout<"\nNode: this->ID has been granted RIGHT OF WAY ";

this->status=GO;

fo<"Node:"<<this->ID<<"; fo<"\n;

else

}
cout << "\nNODE: "<<this->ID << " ALREADY HAS RIGHT OF WAY ";
fo<<"Node:"<<this->ID<<","<<1<<'\n';
}

else  fo<<"Node:"<<this->ID<<","<<0<<",Node:"<<x<<","<<1<<'\n';//requested but denied
}

void Node::go(ofstream& of, int noOfWings, Junction& junction)
{
    cout<<"\n---------------- Called Go ----------------"  ;
    cout << "\nnode " <<this->ID<< " STATUS:"<<this->status;
    int sameGroup=rand()%10+1;
    int avoidStarvation= rand()%10+1;
    switch(noOfWings)
    {
    case 3:

    switch(this->ID)
    {
    case 0:

    if (this->status== GO)
    {

int vehiclesLeaving = 5+rand()%20; // a minimum of five vehicles leave the intersection

if(junction.wing[2].weight<sameGroup)sameGroup=junction.wing[2].weight; // vehicles in the same road as the one with right of way

cout<<"\n-----------------------------------Wing[2] weight before "<<
junction.wing[2].weight;

if (this->weight < vehiclesLeaving)vehiclesLeaving =this->weight;
cout << "\nnode "<<this->ID" : " << vehiclesLeaving << " vehicles leaving ";
cout<<"\nNode 2: "<< sameGroup << " Vehicles Leaving: RATIONALE: In the Same Road as Node 0";

of << "Node: "<<this->ID"", " << this->weight"<<"," << vehiclesLeaving<<""," ;
this->weight -= vehiclesLeaving;
junction.wing[2].weight-=sameGroup;
if (this->weight<0)this->weight=0;     //making sure that weight is never negative
cout << "\n node "<<this->ID << " has aggregated weight of "<< this->weight
<<"\n ";
cout<<"\n-------------------------------------------Wing[2] weight before "<<
junction.wing[2].weight;

}
else
{
int vehiclesLeaving=0;
sameGroup=0;
avoidStarvation= 0;
cout << "\nnode "<<this->ID " : DENIED ";
of << "Node: "<<this->ID"", " << this->weight"<<"," << vehiclesLeaving<<""," ;

}
junction.wing[1].weight-=avoidStarvation;

of<<this->weight<<
  ",Node:2,"<<

  sameGroup<<",Node

  1;"<<avoidStarvation<<"\n;"
break;
case 1:
if (this->status==GO)
{
int vehiclesLeaving = 5+rand()%20; // a minimum of five vehicles leave the
intersection

if (this->weight < vehiclesLeaving)vehiclesLeaving =this->weight;
cout << "\nnode "<<this->ID<<" : " << vehiclesLeaving << " vehicles leaving ";

of << "Node: "<<this->ID<<", " << this->weight<<"," << vehiclesLeaving<<",";
this->weight -= vehiclesLeaving;
if (this->weight<0)this->weight=0;  //making sure that weight is never negative
cout << "\n node "<<this->ID << " has aggregated weight of "<< this->weight
  <<"\n ";

}
else
{
int vehiclesLeaving=0;
sameGroup=0;
avoidStarvation= 0;
cout << "\n node "<<this->ID <<": DENIED ";
of << "Node: "<<this->ID<<", " << this->weight<<"," << vehiclesLeaving<<",";
236. junction.wing[0].weight-=avoidStarvation;
237. junction.wing[2].weight-=avoidStarvation;
238. of<<this->weight<< ","<< sameGroup<<",Node 0 & 2:"<<avoidStarvation<<"\n";
239. break;
240.
241. case 2:
242.
243. if (this->status== GO)
244. {
245. int vehiclesLeaving = 5+rand()%20; // a minimum of five vehicles leave the
intersection
246. cout<<"\n-----------------------------------Wing[0] weight before "<<
junction.wing[0].weight;
247. if(junction.wing[0].weight<sameGroup)sameGroup=junction.wing[0].weight;//
vehicles in the same road as the one with right of way
248. junction.wing[0].weight-=sameGroup;
249. if (this->weight < vehiclesLeaving)vehiclesLeaving =this->weight;
250. cout < "\nnode "<<this->ID" : " << vehiclesLeaving << " vehicles leaving ";
251. cout<<"\nNode 0: "<< sameGroup " Vehicles Leaving: RATIONALE: In the Same
Road as Node 2";
252. of << "Node: "<<this->ID"", " << this->weight<<"", " << vehiclesLeaving<<"", ";
253. this->weight -= vehiclesLeaving;
254. if (this->weight<0)this->weight=0; //making sure that weight is never negative
255. cout < "\n node "<<this->ID << " has aggregated weight of "<< this->weight
<<"\n ";
256. cout<<"\n-----------------------------------Wing[0] weight before "<<
junction.wing[0].weight;
257. }
258. else
259. 
260. int vehiclesLeaving=0;
261. sameGroup=0;
262. avoidStarvation= 0;
263. cout << "\nnode "<<this->ID <<": DENIED ";
264. of << "Node: "<<this->ID<<", " << this->weight<<"," << vehiclesLeaving<<"," ;
265.
266. }
267.
268. of<<this->weight<<
269. 1;"<<avoidStarvation<<'\n';
270. break;
271. }
272. break;
273. case 4:
274. switch(this->ID)
275. {
276. case 0:
277. if (this->status== GO)
278. {
279. int vehiclesLeaving = 5+rand()%20; // a minimum of five vehicles leave the
280. intersection
281. if(junction.wing[2].weight<sameGroup)sameGroup=junction.wing[2].weight;// vehicles in the same road as the one with right of way
282. if (this->weight < vehiclesLeaving)vehiclesLeaving =this->weight;
283. cout << "\nnode "<<this->ID<<": " << vehiclesLeaving << " vehicles leaving ";
284. cout<<"\nNode 2: "<< sameGroup << " Vehicles Leaving: RATIONALE: In the Same
285. Road as Node 0";
of << "Node: "<<this->ID"" , " << this->weight"" , " << vehiclesLeaving"" , ";
this->weight -= vehiclesLeaving;
junction.wing[2].weight=sameGroup;
if (this->weight<0)this->weight=0;     //making sure that weight is never negative
cout << "\n node "<<this->ID  << " has aggregated weight of "<< this->weight
<<"\n ";
else
{
int vehiclesLeaving=0;
sameGroup=0;
avoidStarvation=0;
cout << "\nnode "<<this->ID  << ": DENIED ";
of << "Node: "<<this->ID"" , " << this->weight"" , " << vehiclesLeaving"" , ";
296. }
297. }
298. }
299. junction.wing[1].weight=avoidStarvation;
300. junction.wing[3].weight=avoidStarvation;
301. of<<this->weight<< ",Node:2,"<< sameGroup<<",Node
1&3:"<<avoidStarvation<<"\n; 
302. break;
303. case 1:
304. if (this->status== GO)
305. {
306. int vehiclesLeaving = 5+rand()%20; // a minimum of five vehicles leave the
intersection
307. 
308. if(junction.wing[3].weight<sameGroup)sameGroup=junction.wing[3].weight;//
vehicles in the same road as the one with right of way
if (this->weight < vehiclesLeaving) vehiclesLeaving = this->weight;

cout << "\nnode " << this->ID << " : " << vehiclesLeaving << " vehicles leaving ";

cout << "\nNode 3: " << sameGroup << " Vehicles Leaving: RATIONALE: In the Same Road as Node 1";

if (this->weight < vehiclesLeaving) vehiclesLeaving = this->weight;

cout << "\nNode " << this->ID << " weight = vehiclesLeaving;"

junction.wing[3].weight = sameGroup;

if (this->weight < 0) this->weight = 0; // making sure that weight is never negative

cout << "\n Node: " << this->ID << " has aggregated weight of " << this->weight << "\n";

else {

int vehiclesLeaving = 0;

sameGroup = 0;

avoidStarvation = 0;

cout << "\nnode " << this->ID << ": DENIED ";

cout << "\nNode: " << this->ID << " weight = vehiclesLeaving;"

}

case 2:

if (this->status == GO) {

junction.wing[0].weight = avoidStarvation;

junction.wing[2].weight = avoidStarvation;

of << this->weight << ",Node:3," << sameGroup << ",Node 0&2," << avoidStarvation << "\n";

break;

}
int vehiclesLeaving = 5+rand()%20; // a minimum of five vehicles leave the intersection

if(junction.wing[0].weight<sameGroup)sameGroup=junction.wing[0].weight;// vehicles in the same road as the one with right of way

if (this->weight < vehiclesLeaving) vehiclesLeaving = this->weight;

cout << "\nNode: " << this->ID<<" : " << vehiclesLeaving << " vehicles leaving ";
cout<<"\nNode 0: "<< sameGroup << " Vehicles Leaving: RATIONALE: In the Same Road as Node 2";

this->weight -= vehiclesLeaving;
junction.wing[0].weight-=sameGroup;
if (this->weight<0)this->weight=0; //making sure that weight is never negative

cout << "\n Node: " << this->ID << " has aggregated weight of " << this->weight <<"\n ";

}

else
{
int vehiclesLeaving=0;
sameGroup=0;
avoidStarvation= 0;
cout << "\nnode " << this->ID << ": DENIED ";
ocout<<"\nNode: " << this->ID<<" , " << this->weight<<" , " << vehiclesLeaving<<" ," ;

}

junction.wing[1].weight-=avoidStarvation;
junction.wing[3].weight-=avoidStarvation;
of<<this->weight<<","Node:0","sameGroup<<","Node 1&3","avoidStarvation<<\n";
break;
case 3:
if (this->status == GO) {
    int vehiclesLeaving = 5 + rand() % 20; // a minimum of five vehicles leave the
    // intersection
    if (junction.wing[1].weight < sameGroup) sameGroup = junction.wing[1].weight; //
    // vehicles in the same road as the one with right of way
    if (this->weight < vehiclesLeaving) vehiclesLeaving = this->weight;
    cout << "node " << this->ID << " : " << vehiclesLeaving << " vehicles leaving ";
    cout << "Node 1: " << sameGroup << " Vehicles Leaving: RATIONALE: In the Same
    Road as Node 3";
    of << "Node: " << this->ID << " , " << this->weight << ", " << vehiclesLeaving << " , ";
    this->weight -= vehiclesLeaving;
    junction.wing[1].weight -= sameGroup;
    if (this->weight < 0) this->weight = 0; //making sure that weight is never negative
    cout << "\n Node: " << this->ID << " has aggregated weight of " << this->weight << "\n ";
} else {
    int vehiclesLeaving = 0;
    sameGroup = 0;
    avoidStarvation = 0;
    cout << "\nnode " << this->ID << ": DENIED ";
    of << "Node: " << this->ID << " , " << this->weight << ", " << vehiclesLeaving << " , ";
}
junction.wing[0].weight-=avoidStarvation;
junction.wing[2].weight-=avoidStarvation;
of<<this->weight<<
    ",Node:1,"<<
    
sameGroup<<",Node
0&2,"<<avoidStarvation<<"\n';

break;

break;
}
break;
}

void Node::stop()
{

cout<"\n----------------- Called Stop -----------------" ;
if (this->status==GO)
{
    this->status=BLOCKED;
    cout << "\nnode "<<this->ID << " is blocked ";
}
}

void Junction::add(ofstream& of, int noOfWings )
{


cout<"\n----------------- Adding Vehicles  -----------------" ;
int addedVehicles[noOfWings];
for (int i=0; i<noOfWings; i++)
{

int moreVehicles = rand() % 10; // this adds at most 9 cars
addedVehicles[i] = moreVehicles;
this->wing[i].weight += moreVehicles;
Junction::nodeWeight[i] = this->wing[i].weight;
cout << "\nAdded weight for Node: " << this->wing[i].ID << " is " << moreVehicles;
}
for (int i = 0; i < noOfWings; i++)
{
of << addedVehicles[i] << ";"
}
of << '\n';
cout << "\n---------------------------------------------------------------";

void setWeight(Junction junct, int noOfnodes)
{
cout << "\n--------- Setting Weight for Nodes ";
for (int i = 0; i < noOfnodes; i++)
{
Junction::nodeWeight[i] = junct.wing[i].weight;
cout << "\nSet weight for Node " << i << " to " << junct.nodeWeight[i];
}
}

int getGreatest(int x) // returns the nodeWeight of the given node i.e. the greatest node weight
{
    // This is a friend function definition to Junction Class
    return Junction::nodeWeight[x];
}
Appendix B

In this annexure we outline a framework in which the algorithm can be implemented within an Network simulator such as NS-2. Please note that this is just a proposal and note a tried and tested implementation. This is only to give a framework with in which future work can be picked up from. The algorithm can be implemented as new protocol or an application for NS-2. We start by proposing a new header file as follows.

The header file
In new header file 'roadtraffic.h', we first declare the data structure for the new traffic packet header which is going to carry the relevant data.

```
struct hdr_roadtraffic {
    int senderID;
    int weight;
};
```

The int ‘senderId’ is going to be used to identify the node segment that sends the packet. The int weight is the node weight representing the vehicles in that road segment. The senderId can still be dropped from the definition since the nodes already have id in NS-2. The algorithm will thoughtfully and carefully be implemented in file roadTraffic.cpp.

Necessary changes
The following changes will be necessary : a new packet type for the roadtraffic-agent is needed, so the first step is to edit the file 'packet.h'. There you can find the definitions for the packet protocol IDs (i.e. PT_TCP, PT_TELNET, etc.).

Add a new definition for PT_ROADTRAFFIC there. Your edited version of packet.h's last few lines of enum packet_t {} should look like the following code
enum packet_t {
    PT_TCP,
    PT_UDP,
    ......
    // insert new packet types here
    PT_TFRC,
    PT_TFRC_ACK,
    PT_ROADTRAFFIC, // packet protocol ID for our ROADTRAFFIC-Agent
    PT_NTYPE // This MUST be the LAST one
};

Edit the p_info() in the same file to include "roadTraffic".
You then run the 'make depend' before the 'make', to recompile these files.

Edit the file 'tcl/lib/ns-default.tcl'. This is the file where all default values for the Tcl objects are defined.

insert the following line to set the default packet size for Agent/ROADTRAFFIC.

Agent/RoadTraffic set packetSize_ 64

You have to add an entry for the new roadTraffic packets in the file 'tcl/lib/ns-packet.tcl' in the list at the beginning of the file. It would look like the following piece of code

```tcl
{ SRMEXT off_srm_ext_}

   { roadTraffic off_roadTraffic_ }}

set cl PacketHeader/[lindex $pair 0]
The last change is to the ‘Makefile’.

Add the file ‘roadTraffic.o’ to the list of object files for ns. In your version the last lines of the edited list look like this:

```
sessionhelper.o delaymodel.o srm-ssm.o \
srm-topo.o \
roadTraffic.o \
$(LIB_DIR)int.Vec.o $(LIB_DIR)int.RVec.o \
$(LIB_DIR)dmalloc_support.o \
```

Recompile ns now simply by typing 'make' in the ns directory.
References

[1]. Yi Yang, Rajive Bagrodia Mobile Systems Lab Computer Science Department University of California, Los Angeles Los Angeles, California 90095-1596 {yangyi, rajive}@cs.ucla.edu : Evaluation of VANET-based Advanced Intelligent Transportation Systems* 

[2]. Davol, Angus: Modeling of traffic Signal Control and transit Signal Priority Strategies in a Microscopic Simulation Laboratory; September, 2001 


[4]. Tian, Zhili: Capacity Analysis of Traffic-Actuated Intersections; September, 2002 


