TRAFFIC CONGESTION DETECTION USING VANET

by

Francisco M. Padron

A Thesis Submitted to the Faculty of
The College of Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Master of Science

Florida Atlantic University
Boca Raton, Florida
April 2009
Traffic Congestion Detection Using VANET

By

Francisco M. Padron

This thesis was prepared under the direction of the candidate’s thesis advisor, Dr. Imad Mahgoub, Department of Computer Science and Engineering, and has been approved by the members of his supervisory committee. It was submitted to the faculty of The College of Engineering and Computer Science and was accepted in partial fulfillment of the requirements for the degree of Master of Science.

SUPERVISORY COMMITTEE:

Imad Mahgoub, Ph.D.
Thesis Advisor

Shihong Huang, Ph.D.

Hari Kaila, Ph.D.

Robert S. Cooper
Borko Furth, Ph.D.
Chairman, Department of Computer Science and Engineering

Kari K. Stevens, Ph.D., P.E.
Dean, College of Engineering and Computer Science

Barry T. Rosson, Ph.D.
Dean, Graduate College

April 17, 2009
ACKNOWLEDGEMENTS

This work was completed thanks to the help and support of many individuals. I would like to thank everyone who helped me and motivated me along the way.

Thanks to my committee members, Dr. Shihong Huang and Dr. Hari Kalva, for their interest in my work and for the time they dedicated to review it and to understand it.

Thanks to Jean Mangiaracina, who guided me through all the administrative requirements and procedures, always going the extra mile.

Thanks to the members of the Florida Atlantic University Mobile Computing Group, for giving me great feedback and support, and for providing a place to bounce my ideas. My work would not have been the same without their contributions.

Great thanks to my advisor, Dr. Imad Mahgoub, for encouraging me to pursue what was just an idea, and helping me to develop it into this tangible body of work.

Lastly, I would like to thank my wife and my children, for supporting me through this period, and for enduring many weekends without me.
ABSTRACT

Author: Francisco M. Padron
Title: Traffic Congestion Detection Using VANET
Institution: Florida Atlantic University
Thesis Advisor: Dr. Imad Mahgoub
Degree: Master of Science
Year: 2009

We propose a distributed, collaborative traffic congestion detection and dissemination system using VANET that makes efficient use of the communication channel, maintains location privacy, and provides drivers with real-time information on traffic congestions over long distances. The system uses vehicles themselves, equipped with simple inexpensive devices, as gatherers and distributors of information without the need for costly road infrastructure such as sensors, cameras or external communication equipment. Additionally, we present a flexible simulation and visualization framework we designed and developed to validate our system by showing its effectiveness in multiple scenarios and to aid in the research and development of this and future VANET applications.
TRAFFIC CONGESTION DETECTION USING VANET

LIST OF TABLES ..................................................................................................... IX

LIST OF FIGURES .................................................................................................... X

1 INTRODUCTION ..................................................................................................... 1
  1.1 PROBLEM STATEMENT ................................................................................... 2
  1.2 CURRENT SOLUTIONS TO THE TRAFFIC CONGESTION PROBLEM ............... 4
  1.3 RELATED WORK ............................................................................................ 6
  1.4 CONTRIBUTIONS ............................................................................................ 9
  1.5 SCOPE ........................................................................................................ 10
  1.6 ORGANIZATION ............................................................................................ 10

2 AD-HOC NETWORKS ........................................................................................... 12
  2.1 MOBILE AD-HOC NETWORKS (MANET) ........................................................ 12
  2.2 VEHICULAR AD-HOC NETWORKS (VANET) ................................................... 12

3 TRAFFIC CONGESTION DETECTION .................................................................... 14
  3.1 PROPOSED SOLUTION .................................................................................. 14
    3.1.1 DISTRIBUTED VS. CENTRALIZED .............................................................. 15
    3.1.2 RELIABILITY .......................................................................................... 15
    3.1.3 EASE OF DEPLOYMENT AND COST EFFECTIVENESS ................................. 16
    3.1.4 COMPARISON WITH PREVIOUS WORK ..................................................... 16
3.2 COMPONENTS OF A DISTRIBUTED CONGESTION DETECTION SYSTEM .......... 18

3.2.1 INFORMATION GATHERING ................................................................. 18
3.2.2 INFORMATION SHARING ................................................................. 18
3.2.3 INFORMATION CONSOLIDATION ...................................................... 19
3.2.4 KNOWLEDGE CREATION ............................................................... 19
3.2.5 KNOWLEDGE PROPAGATION ......................................................... 19
3.2.6 KNOWLEDGE CONSOLIDATION ...................................................... 20

3.3 OTHER CONSIDERATIONS FOR CONGESTION DETECTION SYSTEMS ......... 20

3.3.1 BANDWIDTH UTILIZATION .............................................................. 20
3.3.2 VEHICLE PARTICIPATION ............................................................... 20
3.3.3 PRIVACY ....................................................................................... 21
3.3.4 SECURITY ....................................................................................... 22

3.4 PROPOSED ALGORITHM: VEHICULAR OVER-THE-AIR INFORMATION GATHERING (VOTING) ................................................................. 22

3.4.1 GENERAL IDEA ................................................................................ 23
3.4.2 INFORMATION GATHERING ............................................................. 26
3.4.3 INFORMATION PROPAGATION ........................................................ 26
3.4.4 MOVING TIME WINDOW ............................................................... 27
3.4.5 DETECTING AND MANAGING CONGESTIONS ................................. 27
3.4.6 CONGESTION SECTIONS ............................................................... 31
3.4.7 CONGESTION CHANGING OVERTIME ............................................. 35
3.4.8 INFORMATION FILTERING ............................................................. 42
3.4.9 PRIVACY ....................................................................................... 42
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4.2 FIELD</td>
<td>65</td>
</tr>
<tr>
<td>5.4.3 VOTING FIELD</td>
<td>66</td>
</tr>
<tr>
<td>5.4.4 NODE CLASSES</td>
<td>66</td>
</tr>
<tr>
<td>5.4.5 NODES</td>
<td>67</td>
</tr>
<tr>
<td>5.5 COMMUNICATIONS</td>
<td>68</td>
</tr>
<tr>
<td>5.6 ENCODING</td>
<td>69</td>
</tr>
<tr>
<td>5.7 NOTIFICATIONS</td>
<td>70</td>
</tr>
<tr>
<td>5.7.1 NOTIFICATION TYPES</td>
<td>70</td>
</tr>
<tr>
<td>5.8 COMMANDS</td>
<td>72</td>
</tr>
<tr>
<td>5.8.1 COMMAND TYPES</td>
<td>73</td>
</tr>
<tr>
<td>5.9 CONTROLLING SIMULATION TIME</td>
<td>74</td>
</tr>
<tr>
<td>6 SIMULATING AND VISUALIZING VOTING</td>
<td>78</td>
</tr>
<tr>
<td>6.1 VISUALIZATION WINDOW</td>
<td>78</td>
</tr>
<tr>
<td>6.2 CONTROL PANEL</td>
<td>79</td>
</tr>
<tr>
<td>6.3 RUNNING THE SIMULATION</td>
<td>80</td>
</tr>
<tr>
<td>6.4 VALIDATING VOTING</td>
<td>82</td>
</tr>
<tr>
<td>6.4.1 CONGESTION FORMATION</td>
<td>83</td>
</tr>
<tr>
<td>6.4.2 FALSE CONGESTION</td>
<td>85</td>
</tr>
<tr>
<td>6.4.3 DISSIPATING CONGESTION</td>
<td>85</td>
</tr>
<tr>
<td>6.5 SIMULATION RESULTS</td>
<td>87</td>
</tr>
<tr>
<td>7 CONCLUSION</td>
<td>95</td>
</tr>
<tr>
<td>7.1 FUTURE WORK</td>
<td>96</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>98</td>
</tr>
</tbody>
</table>
LIST OF TABLES

TABLE 1. COMPARISON OF CONGESTION DETECTION AND DISSEMINATION SYSTEMS.. 17
TABLE 2. CONGESTION ATTRIBUTES .......................................................................... 27
TABLE 3. DIFFERENT APPROACHES FOR CONSTRUCTION OF SIMULATIONS ........ 50
TABLE 4. FIELD PACKAGE COMPONENTS ................................................................ 54
TABLE 5. RADIO PACKAGE COMPONENTS ............................................................... 55
TABLE 6. MAC PACKAGE COMPONENTS .................................................................. 56
TABLE 7. NETWORK PACKAGE COMPONENTS .......................................................... 56
TABLE 8. ROUTING PACKAGE COMPONENTS ........................................................... 57
TABLE 9. TRANSPORT PACKAGE COMPONENTS ..................................................... 57
TABLE 10. APPLICATION PACKAGE COMPONENTS .................................................. 58
LIST OF FIGURES

FIGURE 1. WHY COMBINING CONGESTION BOUNDS IS NOT ENOUGH .............................................. 30
FIGURE 2. CONGESTION AREA SECTIONS ................................................................................... 31
FIGURE 3. A GROUP OF VEHICLES AND THEIR RECEPTION AREAS .................................... 33
FIGURE 4. MULTIPLE VIEWS OF A CONGESTION ....................................................................... 34
FIGURE 5. CONGESTION VIEWS BUILT OVER TIME ............................................................. 34
FIGURE 6. MOVING CONGESTION AFTER ONE BROADCAST CYCLE ...................................... 36
FIGURE 7. MOVING CONGESTION AFTER TWO BROADCAST CYCLES ..................................... 36
FIGURE 8. A MOVING CONGESTION CHANGING OVER TIME ............................................ 38
FIGURE 9. A CONGESTION DISSIPATING OVER TIME ........................................................... 41
FIGURE 10. JiST ARCHITECTURE ............................................................................................ 52
FIGURE 11. SWANS ARCHITECTURE ....................................................................................... 53
FIGURE 12. TEMPLATE FOR A SIMPLE NODE .......................................................................... 68
FIGURE 13. VISUALIZATION AND SIMULATION CONTROL MODULE ARCHITECTURE .......... 69
FIGURE 14. SIMULATION CONTROL SEQUENCE .................................................................... 76
FIGURE 15. FLOW CHART OF SIMULATION CONTROLLED BY MONITOR .............................. 77
FIGURE 16. VISUALIZATION SCREEN ...................................................................................... 78
FIGURE 17. CONTROL PANEL ................................................................................................... 79
FIGURE 18. START OF A CONGESTION ..................................................................................... 80
FIGURE 19. CONGESTION AS SEEN BY SELECTED VEHICLE .......................... 81
FIGURE 20. CONGESTION SECTIONS AS SEEN BY SELECTED VEHICLE ............... 81
FIGURE 21. DISSIPATING CONGESTION ................................................................ 82
FIGURE 22. CONGESTION CLEARED ..................................................................... 82
FIGURE 23. A CONGESTION STARTING TO FORM .............................................. 83
FIGURE 24. A CONGESTION IDENTIFIED AS SUCH AND BROADCASTED TO NEIGHBORS .... 84
FIGURE 25. VEHICLES GOING THE OPPOSITE DIRECTION ACTING AS CARRIERS .......... 84
FIGURE 26. A DETAILED VIEW OF A CONGESTION VIEWED BY ONE OF ITS MEMBERS .. 84
FIGURE 27. PROPAGATION OF INFORMATION OVER LONG DISTANCES ............... 85
FIGURE 28. A FALSE CONGESTION ...................................................................... 85
FIGURE 29. A CONGESTION STARTING TO DISSIPATE ........................................ 86
FIGURE 30. A SHRINKING CONGESTION CONTINUING TO DISSIPATE ..................... 86
FIGURE 31. A RESIDUAL CONGESTION NOT YET FLUSHED FROM THE SYSTEM ........ 86
FIGURE 32. THE CONGESTION FINALLY FLUSHED OUT OF THE SYSTEM ............... 86
FIGURE 33. CONGESTION FORMATION (100 VEHICLES) ....................................... 89
FIGURE 34. CONGESTION PROPAGATION (100 VEHICLES) .................................... 89
FIGURE 35. CONGESTION AWARENESS VS. CONGESTION SIZE (100) ..................... 90
FIGURE 36. DISTANCE TO CONGESTION VS. TIME TO DISCOVER (100) .................... 90
FIGURE 37. CONGESTION FORMATION (1000 VEHICLES) ..................................... 92
FIGURE 38. CONGESTION PROPAGATION (1000 VEHICLES) .................................. 93
FIGURE 39. CONGESTION AWARENESS VS. CONGESTION SIZE (1000) ................... 93
FIGURE 40. DISTANCE TO CONGESTION VS. TIME TO DISCOVER (1000) ................. 94
1 INTRODUCTION

Automobile traffic is a major problem in modern societies. Millions of hours and gallons of fuel are wasted everyday by vehicles stuck in traffic. According to the Texas Traffic Institute, drivers in the US wasted 4.2 billion hours of time, 2.9 billion gallons of fuel, and a total cost of 78 billion dollars in 2005 due to traffic delays [1].

Technology is at a point today in which vehicles themselves could be used to compile and analyze traffic data and relay it to the drivers in a format that will allow them to make smart decisions to avoid congested areas. Communications between vehicles can be achieved either through vehicle-to-vehicle (V2V) communications and/or vehicle-to-infrastructure (V2I). Vehicular ad-hoc networks (VANET) [2] are a form of mobile ad-hoc networks (MANET) that provide communications between nearby vehicles and nearby fixed equipment.

Congestion detection algorithms are designed to detect areas of high traffic density and low speeds. Each vehicle captures and disseminates information such as location and speed and process the information received from other vehicles in the network. Multiple approaches have been proposed to implement congestion detection in VANETs [3] [4] [5] [6]. Congestion detection is only one
of many applications of VANETs and it is not designed to be used as means for automated driving but rather as a tool to deliver information to the driver that will help him/her make decisions to avoid heavy traffic. Developing a traffic congestion detection system will have tremendous impact on the economy, the environment and society in general allowing us to spend less time stuck in traffic and more time doing more productive and enjoyable activities.

This work focuses on the development of a system for traffic congestion detection: Vehicular Over-the-air Traffic Information Gathering (VOTING), that is capable of detecting traffic congestion areas in real-time with data collected and disseminated by vehicles using V2V communications, without the need for any external infrastructure (such as antennas, satellites, etc.), and developing the tools for interactively simulating and visualizing the behavior of this and future congestion detection systems on a myriad of scenarios.

1.1 Problem Statement

Traffic congestions are formed by many factors; some are (somehow) predictable like road construction, rush hour or bottle-necks and some are unpredictable like accidents, weather and human behavior. Drivers, unaware of a congestion ahead eventually join it and increase the severity of it. The more severe the congestion is, the more time it will take to clear once the cause of it is eliminated or ameliorated.
The ability for a driver to know the traffic conditions on the road ahead will enable him/her to seek alternate routes saving time and fuel. When many drivers have this ability, traffic congestions, specifically those related to localized incidents such as accidents or temporary disruptions will be less severe and only the vehicle in the immediate vicinity of the incident, at the time of the incident, will be affected. This would lead to a much more efficient use of our road infrastructure.

Traffic congestions result from driver behavior and the lack of wide distance information. Current systems, such as helicopter traffic reports are effective because from the air we can get a good picture of a congestion area, where it starts, where it ends and how slow or fast is moving, however these reports require enormous resources and are therefore limited to major metropolitan areas. In order to provide drivers with useful information about traffic ahead a system must:

a) Identify the congestion, its location, severity and boundaries.

b) Relay this information to drivers within the congestion and those heading towards it.

These two requirements must be satisfied by any traffic congestion system. To identify the congestion an observer, like one riding on the helicopter, needs to see vehicles that are a long distance away from each other, and outside of each other’s line of sight. A visual picture of the congestion can only be obtained from a vantage point, well above the road surface. For vehicles within the congestion
to form their own picture of a congestion they need to collaborate using vehicle-to-vehicle (V2V) or vehicle-to-infrastructure (V2I) communication.

Once a clear picture of the congestion has formed, this information needs to be relayed to vehicles away from the congestion so that vehicles heading towards it can take evasive actions avoiding further escalation its severity.

1.2 Current Solutions to the Traffic Congestion Problem

Most current navigation systems are static and do not provide traffic information. Route selection is based solely on static map data which leads to the system that fails to give the driver the most efficient route to his/her destination. In the last year or so, some of these devices have incorporated “real-time” traffic information to aid in route selection. Such “real-time” traffic systems such as the services provided from NAVTEQ [7] and other commercial services today [8] [9] rely on humans and/or road infrastructure like traffic cameras and radars to maintain a central database of current traffic conditions. This limits their real-time accuracy and restricts its reach to major metropolitan areas. Monitoring traffic in this way, at all times, in all places is simply not cost-effective even in the most developed parts of the world. As an example, at the time of this writing, in New York City, a densely populated city of 305 square miles (790 km²) and 8 million inhabitants, only 81 traffic cameras were in place [10] and they only provide visual information that still needs to be translated into congestion information either by humans or specialized software.
The main characteristic of current systems is that they are one-way: vehicles only receive information, usually through FM radio or satellite. Requiring vehicles to transmit data in real-time to a central location would require an enormous communications infrastructure not widely available today across major highways, specially outside the major urban areas of the most industrialized countries. Two-way communication systems such as OnStar™ [11] rely on cell-phone communications which are not “always on” and are not designed for the volume of communications required by a real-time traffic-aware navigation system.

The work presented on this thesis uses a distributed approach that uses vehicles themselves as nodes in a vehicular ad-hoc network (VANET) as well as data-gathering devices that collect the necessary information to determine current traffic conditions and disseminate it over long distances through the use V2V communications. By integrating this information into the vehicle’s navigation system, the system will be able to provide the driver with traffic-aware routing recommendations using an efficient routing algorithm that takes into account traffic information [12]. A distributed approach to traffic congestion detection is the most efficient approach because it provides a greater degree of reliability and flexibility and does not require a major investment in infrastructure as the costs are distributed among many drivers.
1.3 Related Work

Much of the research in VANET focuses on simulating vehicular traffic [13] [14] [15] [16] [17] [18] and multi-hop routing [19] [20] [21] [22]. A few researches have studied the problem of using VANETs to discover and disseminate traffic congestion information [23] [24] [25]. Using vehicle based GPS systems we can create an ad-hoc wireless network that can find and disseminate traffic congestion information.

Collision avoidance systems [26] [27] are designed to detect a traffic incident in real-time and rapidly relay this information to nearby vehicles to prevent a collision. These systems are very different from traffic congestion systems, in the former, information needs to be relayed very fast over short distances and needs to be extremely reliable as it has a direct effect on life-and-death situations, while in the latter information remains current for a longer time, needs to be disseminated over long distances and is used for congestion amelioration.

In recent years, several researchers have addressed the issue of distributed detection and propagation of traffic congestion information. Fukumoto et al [3] proposed a system that uses vehicle based GPS systems to discover and disseminate traffic congestion information, the system is called COC for VANET. This system maintains and disseminates three types of information: Raw Information (level 1), density information (level 2) and congestion areas information (level 3). Higher levels contain aggregated information.
Donrbush et al [4] from the University of Maryland proposed a novel system for congestion detection in VANET: StreetSmart that uses clustering as a data aggregation technique to combine related recordings of unusually slow speed. StreetSmart uses clustering algorithms that work over a distributed network where each node analyzes the collected statistics eliminating the need for a central entity. Clustering is the process of combining data points that are similar to each other by some measure.

Yoon, Noble and Liu of the University of Michigan [6] proposed a system for traffic estimation that is based on road segmentation and focuses on complex inner-city traffic scenarios.

Some of these systems rely, either explicitly or implicitly, on having the location for all vehicles in the congestion available in order to make the determination that a congestion exists and where it is located. When congestion sizes exceed transmission ranges, common in freeway scenarios the use of multi-hop communications is required in order for all vehicles in the congestion to have knowledge of all other vehicles in the congestion. This presents a problem if we want to keep location information anonymous: a vehicle may receive fresh information directly from a vehicle as well as re-broadcasted information (older) from the same vehicle. Because vehicles constantly move, these two pieces of information will be undistinguishable from two pieces referring to two different vehicles. For this reason, these systems rely on unique vehicle IDs as a mechanism to identify the source of each piece of information and maintain a
unique location for each vehicle. The broadcasting of unique vehicle ID’s at the application level opens the door for location tracking raising major privacy concerns.

Most recently, companies have began to realize the potential of using vehicles as collectors of traffic information, Dash Navigation, Inc. [28] a start-up in Sunny Valley, CA started offering a service in 2009 called The Dash Driver Network that allows drivers to broadcast their location and speed in exchange for receiving updated traffic information compiled from other vehicles in the network. This system is centralized and relies on wireless internet connectivity which is not widely available on roads and highways around the globe. Because the collecting entity is a central, trusted location, privacy concerns are mitigated.

The CAR 2 CAR communication Consortium [29] an non-profit organization initiated by European vehicle manufacturers with the objective of improving road traffic safety and efficiency published in 2007 a manifesto in which it proposes standards for V2V and V2I communications among other things. Other organizations [30] initiated by industry, government and universities have started similar efforts in the last few years.

In 2008, the European Union took a major first step towards deployment of systems relying on V2V and V2I communications by reserving a radio frequency across the EU for vehicle applications aiming at enabling co-operative systems between carmakers [31]. The EU expects this action to lead to the eventual roll-
out of the first production examples early next decade with the first efforts expected to be focused in the area of road safety.

1.4 Contributions

There are three major areas in which this work makes a significant contribution:

- VOTING, a distributed congestion detection algorithm that constructs and disseminate traffic congestion information from data obtained directly from vehicles by forming an ad-hoc-network. This novel algorithm was developed in a way that does not require the use of unique vehicle IDs that may compromise location privacy. The algorithm also provides an efficient use of the communications channel by only re-transmitting aggregated data and not retransmitting location data for every single vehicle.

- A simulation environment for VANETs: Built on top of JIST/SWANS [32] [33], to allow simulations of Vehicular Ad-Hoc networks at all levels, from the radio layer to the application layer.

- A Visualization and Simulation Control Module that interacts with the simulation engine allowing the researcher to observe conditions of the nodes of the network in either real-time or as playback of a finished simulation run. This module also allows the researcher to control the simulation “time” and change properties of either the environment or particular nodes during a simulation run.
1.5 Scope

This work is focused on congestion detection for a simplified model of traffic, in particular, highway traffic on a straight highway segment with only one entry and exit points. More complex road layouts and traffic scenarios are a subject for future work, however the general ideas presented in this work for congestion detection and propagation, as well as the simulation and visualization tools developed can be extended to more complex traffic models.

Our research is directed to the application layer of a VANET network. In this work we are not concerned on low-level protocols for radio communications or routing, many such protocols have been developed [19] [20] [21] [22], some commercially available like IEEE 802.11 (Wi-Fi) [34] that can be adapted to V2V communications.

1.6 Organization

This rest of this thesis is organized as follows: Chapter 2 contains a brief background of Mobile Ad-Hoc networks (MANET) and Vehicular Ad-hoc Networks (VANETS). Chapter 3 contains an in-depth study of the problem of detecting traffic congestions and it includes our proposed solution for a distributed algorithm that finds congestion areas and disseminates this information so that it can be used by vehicles over long distances. Chapter 4 is dedicated to describing our simulation environment as well as the simulation
tools developed to support our research. In Chapter 5, we discuss the visualization modules developed to visualize the inner working of congestion detection algorithms as well as mobile networks in general. In Chapter 6, we present the results of integrating VOTING with the simulation and visualization modules. Finally, in Chapter 7, we discuss our conclusions and possible directions for future research.
2 Ad-Hoc Networks

2.1 Mobile Ad-Hoc Networks (MANET)

Mobile Ad Hoc Networks are flexible wireless networks that do not rely on any external infrastructure such as routers or radio towers, the network is formed by the nodes themselves, and messages are usually sent using multi-hop routing in which network nodes act as routers to deliver messages outside of the sender’s transmission range. The primary challenge for building a MANET is for each device to continuously maintain the information required to properly route traffic. Each MANET node may be able to move independently in a manner not necessarily predictable by other nodes. MANETs have been studied extensively [35].

2.2 Vehicular Ad-Hoc Networks (VANET)

Many car manufactures and research institutions are investigating ways of establishing vehicular networks. Because of the flexible nature of Mobile Ad Hoc Networks (MANET), they represent an attractive solution for inter-vehicular communications.
VANETs have some unique characteristics not shared by other types of MANETs:

- Vehicles move at high speed.
- Mobility patterns are somehow predictable as movement is constrained by road infrastructure. In some situations such as highway traffic, the mobility patterns become highly predictable.
- Large coverage area. Vehicles travel over long distances and traffic information may be useful to vehicles hundreds of miles away.
- Power consumption is not a major concern. Vehicles are mobile power plants.
- Vehicles have a high cost and therefore can be equipped with additional sensors without significantly impacting the total cost.
- VANET’s topology is extremely dynamic as vehicles go in and out transmission range quite rapidly.
- Vehicles travel long distances in a small amount of time when compared to other mobile networks.

Research of VANETs has been a topic of interest in recent years [36] [37].
3 TRAFFIC CONGESTION DETECTION

Congestion detection algorithms are designed to find areas of high traffic density and low speeds. Each vehicle disseminates the information it has obtained from its own hardware and from other sources and process the information received from other nodes in the network.

Congestion detection is only one of many applications of VANETs and it is not designed to be used as means for automated driving but rather as a tool to deliver information to the driver that will help him/her make decisions to avoid heavy traffic. Collision avoidance applications [26] [27], not covered in this work, use the same kind of GPS positioning information to alert drivers of possible dangerous situations such as a road hazard, incoming vehicle or an accident, in this kind of application response times need to be much smaller. A myriad of applications can be enabled by VANETs [38].

3.1 Proposed Solution

One important characteristic of congestions is that they move relatively slow and the vehicle density is high. We will use this to our advantage when designing our system for congestion detection and propagation.
By equipping vehicles with relatively simple devices that allow them to communicate with nearby vehicles, we can effectively turn them into data collectors. Distributed applications can be implemented over this infrastructure to detect congestions and propagate congestion information to vehicles outside of the congestion area making it possible for the driver to seek alternate routes to avoid the congestion. Routing algorithms integrated to the vehicle’s navigation system can use congestion information to produce the most efficient route to the desired destination [12].

3.1.1 Distributed vs. Centralized

When developing a congestion detection system, we can either rely on some central network and data gathering infrastructure or we can use hardware in the vehicles themselves to collect, analyze and disseminate this information. A distributed approach to traffic congestion detection is the most efficient because it provides a greater degree of reliability and does not require a major investment in infrastructure as the costs are distributed among many drivers, thus our research is focused on distributed solutions.

3.1.2 Reliability

By putting the devices on each vehicle, the reliability of the network as a whole is increased as the likelihood of many nodes failing all at once is less than that of a central component failing. Reliability is also increased by redundancy, vehicles
close to each other are all capturing and transmitting virtually the same information.

### 3.1.3 Ease of Deployment and Cost Effectiveness

Deploying of a distributed network can be made gradually; devices can be installed in vehicle factories or can be sold as after-market products such as today’s GPS navigation systems. Each user will pay for its parts and its maintenance and will benefit directly from its functionality making the cost-effectiveness equation very clear. As the number of users grows, the system becomes more and more useful and therefore more appealing to new users. A centralized system on the other hand requires large investments in infrastructure before the network can even begin to provide any service.

### 3.1.4 Comparison with Previous Work

Table 1 shows a comparison between the most relevant related work discussed in Section 1.3 and our proposed solution.
Table 1. Comparison of Congestion Detection and Dissemination Systems

<table>
<thead>
<tr>
<th></th>
<th>NAVTEQ</th>
<th>COC for VANET</th>
<th>StreetSmart</th>
<th>Dash Navigation</th>
<th>VOTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vehicles capture and transmit information</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Optional</td>
<td>Yes</td>
</tr>
<tr>
<td>Requires Vehicle ids</td>
<td>N/A</td>
<td>Yes</td>
<td>Yes¹</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Raw location information is rebroadcasted</td>
<td>N/A</td>
<td>Yes</td>
<td>Yes¹</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Only vehicles contributing use bandwidth</td>
<td>N/A</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Requires Human Intervention</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Requires Road Infrastructure</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Distributed (or centralized)</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

¹ Not specifically mentioned in the paper but inferred from algorithm description.
3.2 Components of a Distributed Congestion Detection System

3.2.1 Information Gathering

The most basic component of a congestion detection algorithm is the gathering of information from the environment. Each vehicle must be equipped with one or more devices that are capable of gathering data such as current location and speed. This data constitutes the building blocks of the system and changes very rapidly.

3.2.2 Information Sharing

The strength of any distributed system is in the numbers; therefore information must be shared between vehicles. Vehicles must be equipped with a device that allows them to transmit and receive information wirelessly to and from other vehicles in their vicinity by creating a vehicular ad-hoc network (VANET). The congestion detection algorithm will chose what data is disseminated and when, taking into consideration the relevance of such information and the network’s bandwidth.
3.2.3 Information Consolidation

As information from nearby vehicles as well as information relayed from far-away vehicles is received, it needs to be consolidated in order to eliminate inconsistencies and to keep a lid on the amount of information that is to be processed.

3.2.4 Knowledge Creation

As the information is consolidated it is converted into knowledge in the form of possible traffic congestion zones. This is the final product of the congestion detection algorithm and it is to be relayed to the vehicle’s operator through some visual interface that shows the relevant congestion areas on a digital map.

3.2.5 Knowledge Propagation

This knowledge is propagated so that vehicles away from the congestion zone, unable to make this determination with the information received from vehicles around them, can benefit from it. Knowledge is much more compact that raw information and it is also more meaningful. Once a determination is made that a traffic congestion exists at a certain location, this knowledge can be propagated to nodes far away because this information is more likely to be useful than raw data. Also since this knowledge is a condensed version of a larger amount of data, it has a lesser impact on the network’s workload.
3.2.6 Knowledge Consolidation

Knowledge received from other nodes is combined and re-broadcasted so that changes in the current traffic situation can be reflected in real-time. The congestion detection algorithm must provide reliable and current information in a timely fashion, in order to achieve this, information needs to be constantly verified and the knowledge obtained from it must be constantly re-evaluated.

3.3 Other Considerations for Congestion Detection Systems

3.3.1 Bandwidth Utilization

For a congestion detection algorithm to be practical it needs to make intelligent use of the available bandwidth, sending large amounts of information without concern for bandwidth limitations will overload the system making it inoperable. Smart consolidation of information and filtering of irrelevant and out of date data if of utmost importance. In VANET environments where messages are relayed from vehicle to vehicle it is important to consider ways of reducing redundant communications without significantly impacting the network’s effectiveness.

3.3.2 Vehicle Participation

In an ideal world, all vehicles are equipped with exactly the same hardware which allows gathering information and sharing it with others. A real-world application
has to be designed so that it can work even when a small percentage of vehicles on the road participate in the congestion detection system. Other considerations such as heterogeneous networks in which nodes have different capabilities must also be considered and are a subject for future research.

### 3.3.3 Privacy

When new cooperative networks systems such as VANETs are deployed, a major concern for users is the loss of privacy, after all nobody wants his/her vehicle to be telling everybody where it is at all times. When developing congestion detection algorithms over VANETs we have to be aware of these issues and include mechanisms to avoid or minimize them to a point that will be acceptable for most users. Most academic work on congestion detection in VANETs ignore this important fact and use unique vehicle IDs that persist over time making it possible for a malicious node to track the location of a vehicle, for this reason, the solution proposed in this work does not require vehicles to maintain unique IDs.

Removing unique vehicle IDs ensures privacy at the application level, however, privacy at lower levels can still be compromised. A vehicle location can still be tracked based on frequency of radio transmissions and other signatures. Research on MANET privacy at the protocol level such as proposed by Huang et al. [39] and more specific research on privacy for VANETs proposed by
Sampigethava et al [40] can be integrated to a congestion detection system to ensure location privacy.

### 3.3.4 Security

Along with privacy, security is a legitimate concern for real-world systems. Preventing users from abusing or sabotaging the system is paramount. Several initiatives have been developed in recent years to provide secure V2V communications [41] [42] [43] [44].

### 3.4 Proposed Algorithm: Vehicular Over-the-air Information Gathering (VOTING)

What is traffic congestion? Although this question may sound very simple, after all, we all have experienced traffic congestions in our own lives and when we are in one, we know it. However, as humans we have access to large amounts of information through all of our senses: we can see the vehicles around us, we know attributes of the road such as speed limit, road conditions, weather conditions, etc. For a vehicle mounted computer that only knows the vehicle location, speed and direction, it is a different story. To start with, the computer doesn't know what the intention of the driver is, if the car is going “too slow” is it because the driver wants it that way or because the there is something in the road (accident, large density of vehicles, bad weather) that’s forcing the driver to go slow? The later gives us a natural definition for traffic congestion: An area of
congestion is an area on which a large number of vehicles are going significantly slower than the drivers intend.

At the core of our research is the Vehicular Over-the-air Traffic Information Gathering system (VOTING for short). VOTING is based on a simple idea: decision by majority. A congestion area is formed when a large majority of vehicles going through a specific geographical area are going significantly slower than the maximum posted speed limit. Speed limits are available to the vehicle’s computer via archival map information such as that used in today’s vehicle navigation systems.

Vehicles transmit information every broadcasting interval. The broadcasting interval is fixed and equal for all participating vehicles, however, the algorithm does not require that vehicle’s clocks are synchronized; each vehicle can transmit at a different time as long as the time between transmissions is reasonably close to that of other vehicles in the network. In other words, an accurate clock is needed; the clock from the GPS system would be a good candidate for keeping time in a practical implementation.

### 3.4.1 General Idea

The detection of congestion areas in VOTING is done as follows:

- All vehicles broadcast their current location, speed and direction at fixed time intervals.
• A vehicle that is going “slower than normal” considers itself to be in a congestion. A vehicle that believes to be in a congestion will validate its congestion area with data received from other vehicles. This is where the name VOTING comes from: If the information received is consistent with the congestion, it is said to be in agreement. When the number of vehicles in agreement surpasses those in disagreement by a certain margin, and the congestion reaches a certain size, the congestion is then validated and broadcasted.

• Vehicles that are not in a congestion limit their participation to broadcasting their own information (location, speed and direction) and the congestion information they have received from other vehicles. They do not make any changes to congestion information.

These are the main characteristics of the VOTING algorithm:

• Vehicle raw information (location, speed, direction) is broadcasted and received by vehicles nearby. Raw information is never retransmitted. This minimizes bandwidth usage and eliminates the requirement for vehicle IDs.

• Only vehicles that are part of the congestion can characterize it and change congestion information. Vehicles outside of the congestion can only read it and re-transmit it.

• Speed measurements are taken as a moving average to eliminate stop-and-go noise.
Congestion information travels by hopping from vehicle to vehicle and is also carried by vehicles going in the opposite direction on the highway via delayed retransmission. Because congestions are intrinsically slow the small lag caused by this delayed retransmissions does not significantly degrade the accuracy of the system.

A vehicle going slower than normal is not on itself a cause for a congestion, the vehicle may be in the shoulder or disabled but not causing any major disruption to traffic around it. This type of “false congestions” are flushed out when other vehicles going in the same direction, going at normal speed “vote” against it, effectively invalidating the suspected congestion area and preventing it from being broadcasted to other vehicles in the network.

A vehicle that is in a real congestion receives multiple confirmations from vehicles nearby going also slower than normal allowing the vehicle to expand and validate its congestion area. Only after a suspected congestion area is validated by a reasonable number of vehicles it is then broadcasted. This reduces unnecessary network usage and false congestion from affecting the efficiency of the system.

Congestion areas may span longer than the transmission radius causing big congestions to be split. Vehicles believed to be in congestions use congestion information received from other vehicles to consolidate with its own congestion area producing a congestion area that encompasses its own congestion and
overlapping congestions received from other vehicles. This process of consolidation is at the core of VOTING and is explained in detail in the next few sections.

3.4.2 Information Gathering

Each vehicle is assumed to be equipped with a GPS device that provides with vehicle’s current location, a real-time clock and a wireless communication device such as a two-way radio that allows it to communicate with vehicles nearby.

Each vehicle collects and keeps the following information about itself:

- Current location from GPS. The accuracy of this location can be enhanced using map data as proposed by Dornbush et al [4].
- Current Speed (average over a small period of time to eliminate noise).
- Current Direction.

3.4.3 Information Propagation

At every broadcasting interval, each vehicle broadcasts the following information to all vehicles within the range of transmission:

- Its own raw information: Location, Speed, Direction.
- The list of congestion areas it is aware of.
3.4.4 Moving Time Window

When vehicles are, or believe to be, in a congestion they collect location information from nearby vehicles in order to form the picture of the congestion. In order to avoid duplicate information from being analyzed, a moving time window is used to keep the location of nearby vehicles. The size of the window is the size of the broadcasting interval (which is the same for every vehicle). Every location message is tagged with a timestamp. Every time a new message containing location information arrives, this new location information is stored and old location information is removed. At any given time, the collection of vehicle locations maintained by a vehicle spans no longer than one (1) broadcast interval. This is easily implemented using a priority queue.

3.4.5 Detecting and Managing Congestions

The attributes of a congestion are shown in Table 2.

<table>
<thead>
<tr>
<th>CongestionArea</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>CongestionDirection</td>
<td>(X,Y)</td>
</tr>
<tr>
<td>CreationTime</td>
<td>Timestamp</td>
</tr>
<tr>
<td>LastRefreshTime</td>
<td>Timestamp</td>
</tr>
<tr>
<td>InDisagreement</td>
<td>Integer</td>
</tr>
<tr>
<td>VehicleCount</td>
<td>Integer</td>
</tr>
<tr>
<td>AverageSpeed</td>
<td>Double</td>
</tr>
</tbody>
</table>

Table 2. Congestion Attributes
Every broadcast cycle, each vehicle assesses its status and classifies itself as being in a congestion or not by analyzing its own speed and raw data received from vehicles within the transmission range. If the vehicle’s own speed is below a certain percentage of the posted speed limit, the vehicle creates a congestion area containing only itself and it starts listening for location and speed information transmitted by nearby vehicles in order to determine, through a voting process, if its low speed is due to a congestion. At this point this is only a tentative congestion area and will not be broadcasted to other vehicles.

If the receiving vehicle is not in a congestion, it simply collects the congestion information received and stores it into its own memory. If one or more of the congestion areas received overlap those previously stored, the most recent one is kept and the older one is discarded.

A vehicle that is or believes to be part of a congestion builds its congestion area from data received from other vehicles. When a vehicle in this state receives raw information from another vehicle that is near its congestion area and going in the same direction, it updates the tentative congestion information according to the transmitting vehicle’s data: if the transmitting vehicle is also going slow it is said to be in agreement and the congestion area is expanded to include the transmitting vehicle. If in the other hand, the transmitting vehicle is going at
normal or above normal speed it is said to be in disagreement in which case, the
*InDisagreement* counter is incremented.

A congestion area is then built from two different sources of information:

- Raw location data from vehicles nearby that are considered part of the same
  congestion due to their proximity is collected and stored. Each piece of
  information is tagged with a timestamp that indicates the time at which it was
  collected. By using a moving time window of size equal to the broadcast
  interval, each vehicle involved may appear only once in this list, as new
  information is received old information is discarded. At any given time, the list
  of vehicle locations contains information that is one broadcast cycle old at the
  most. Only after the information contained in this detailed list is deemed to be
  enough to make the determination that the vehicle is in a congestion, this
  information is aggregated to build the congestion area to be broadcasted to
  other vehicles.
- Congestion areas known by other vehicles are combined to extend the reach.
  Knowing the bounds of the congestions is not enough to combine them.
  Simply combining two overlapping congestion areas into one without leaving
  any information about the source of each segment would cause these areas
  to perpetually expand not allowing for contraction or movement because
  feedback from previous rounds would stay in the system forever. Figure 1
  shows why combining congestion bounds is not enough to maintain moving
  congestions. In this example two vehicles moving from left to right share
congestion information. When the congestion moves forward the previously shared information remains in the system causing the back of the congestion to incorrectly continue to be part of it.

After vehicles receive data from each other bounds are merged

After the congestion moves vehicles have no way of knowing the tail part of the congestion is no longer there because it indistinguishable from the rest of the area, therefore this area remains in the congestion

**Figure 1. Why Combining Congestion Bounds is not enough**

In order to solve this problem, the consolidation of several congestion areas must be done in a way that maintains information about which part of the congestion area was built using first-hand information (raw location data received from nearby vehicles) and which parts were built from congestion areas built by other vehicles. A mechanism for merging congestion areas was developed that divides the congestion into sections. This mechanism is designed to maintain privacy (it
does not require each vehicle to transmit a unique ID) and to minimize network usage by transmitting aggregated information instead of information for each vehicle.

### 3.4.6 Congestion Sections

A vehicle $A$ that is part of a congestion, divides the congestion into sections as illustrated in Figure 2. The area in the center corresponds to the area occupied by vehicles in the congestion that communicated directly with vehicle $A$ and by itself. We call this section the **core section**. Expanding to the left and right, there is a section that corresponds to core sections of congestions created by vehicles that directly communicated with vehicle $A$. Notice that vehicles in this area are out of transmission range from $A$. We well call this section a **1-hop sections**. 1-hop sections received by vehicle $A$ become a **2-hop section** and so on.

![Figure 2. Congestion Area Sections](image)

Using this technique, congestion areas can be represented using only a small amount of bytes, thus reducing the load on the network. Even for congestion spanning a large geographical area, in which a large number of vehicles are
involved, the number of sections is much smaller, each section of about the size of the transmission area. When compared to transmitting the location of every single vehicle in the congestion area, this representation offers a significant advantage. For example, in a congestion where there is an average of one vehicle every ten meters in a four lane highway, a congestion spawning 1 Kilometer would contain 400 vehicles. Assuming a transmission range of 400 meters, the congestion area would have at the most three sections (for vehicles at the very front or back of the congestion).

Although all vehicles are in agreement as to the bounds of the congestion, each vehicle creates a different partition of sections. For example, let us considered a group of five vehicles as illustrated in Figure 3 moving from left to right, each of their transmission/reception areas depicted by a dashed circle.
Vehicle #1 is within communication range of Vehicles #2 and #3 but it is outside communication range of #4 and #5. Figure 4 shows how each vehicle has a different view of the same congestion. Each box represents a section and the number inside the box indicates the number of hops required to reach vehicles inside that particular section.

![Figure 3. A Group of Vehicles and their Reception Areas](image)
Figure 4. Multiple Views of a Congestion

This view is built overtime from the intermediate steps shown on Figure 5.

Figure 5. Congestion Views Built Over Time
3.4.7 Congestion Changing Overtime

When a congestion moves or changes, we need to make sure that these changes are known by all vehicles, but how do vehicles know that a congestion that is at a different position is indeed the same congestion they have stored in memory that has moved without transmitting individual IDs for every vehicle inside the congestion which would compromise privacy.

As previously shown in Figure 1, if we simply create a union of every congestion area received that intersect with our own, congestion areas would only grow. They would never shrink. The use of congestion sections solves this problem. Using the example in Figure 3, let us consider the following two scenarios:

**Scenario 1:** The congestion remains the same size but moves forward.

Assuming, without loss of generality, all vehicles update their information at the same time (this is not a requirement of the algorithm, the method works equally well when there is no synchronization), then after one broadcast interval, the congestion information stored on each vehicle corresponds to that presented in Figure 6.
Because vehicles #4 and #5 can communicate directly with the vehicles at the end of the congestion, they are the first to reduce the congestion area on the back side (left side in the figure). Vehicle #1 is the first one to expand its congestion area to the left. At this point in time, 0-Hop sections are up to date while the 1-hop and 2-hop sections are not. 2-Hop sections for vehicles #1 and #2 are too big and those for vehicles #4 and #5 are too small. They will get updated in the next few cycles:
After only a few broadcast cycles (one for each hop in the congestion area), the congestion area is correct for all vehicles. Since congestion move slowly by nature and vehicle density is high, the picture of the congestion any particular vehicle remains accurate at all times. Notice that since vehicles keep moving, the higher the number of hops, the larger the error. In order to minimize this error, congestion areas are tagged with a time stamp. When congestion areas with different time stamps are compared, the older congestion’s current location is predicted based on the time difference and the congestion’s average speed. Doing this in the example above produces updated congestion areas after only one broadcast cycle, the error will be limited to the difference between the actual speed and the average speed.

To see it more clearly, Figure 8 shows the congestion sections changing over time for a congestion moving from left to right.
Figure 8. A Moving Congestion Changing Over Time
One important thing to note is how areas at the back of the congestion that are no longer part of it are removed from the system. At $t + 3$ Vehicle #5 receives a section containing the back of the congestion, however, it discards this information because it is labeled as a 2-hop section but it is next to a 0-hop section. In general congestion areas must have a consecutive hop counts if they are adjacent to each other, otherwise the congestion with the higher hop count is discarded. The same technique is applied at $t + 4$ by Vehicle #3. After a few cycles (as many as the maximum number of hops in the congestion) the back of the congestion is effectively removed. This technique is extremely important in the next scenario.

**Scenario 2:** Congestion dissipates from the front.

Congestions don’t last forever, in time they dissipate and our system detects this condition and disseminates it. It is as important to build the picture of the congestion as it is growing as it is to update drivers when the congestion starts to dissipate. Removing congestions from the system is not as straight forward as building it because congestion information may already be stored in vehicles not participating in the congestion.

There are two mechanisms by which congestions are removed from the system when they no longer exist:

a) Vehicles inside the congestion use congestion sections to invalidate areas of the congestion that have dissipated.
b) Vehicles outside the congestion area that are carrying consolidated congestion information remove it after a certain time (called maximum congestion age). Every time an overlapping congestion (adjusted for lag time) is received, the newest congestion replaces the old one and the congestion age is reset to zero.

In a similar way that they provide a mechanism for detecting the movement of congestion areas, congestion sections provide the necessary information to remove areas of the congestion that have already dissipated. Figure 9 shows how a congestion dissipating from the front (vehicles at the front of the congestion start to move faster) is updated over time using congestion sections.
Like in the first scenario, sections which hop count is not consecutive to an adjacent congestion are ignored. At time $t + 1$, vehicle #1 receives a 2-hop section in front of a 0-hop section and ignores it preventing this information from propagating any further in time.

**Figure 9. A Congestion Dissipating Over Time**
3.4.8 Information Filtering

After the information consolidation is performed, each vehicle performs a clean-up of the information it has in its own memory. The following are removed from memory:

- Congestion areas in which LastRefreshTime makes it too old to be considered valid.
- Congestions too far away to be relevant.

3.4.9 Privacy

In order for a system such as VOTING to be effective, it has to be adopted by a large number of vehicles. Because of the distributed nature of the system, there is no trusted authority in which users may rely to guard their information. The continuous broadcasting of one’s location can be used by a malicious entity to track a person and discover their whereabouts on any particular moment.

Privacy is maintained by not assigning any unique identifier to a vehicle. As we have shown, the VOTING algorithm does not rely on this information being available, location information is broadcasted anonymously, every broadcast is independent from the previous one and there is no marker that identifies it as coming from the same vehicle.
A random variation on the start of the broadcasting interval can be introduced from time to time to avoid the identification of a vehicle by its exact broadcast times.

### 3.4.10 Parameters

VOTING can be fine-tuned by controlling the following parameters:

<table>
<thead>
<tr>
<th>VOTING Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaximumCongestionAge</td>
</tr>
<tr>
<td>MaximumCongestionDistance</td>
</tr>
<tr>
<td>MaximumGapTime</td>
</tr>
<tr>
<td>CongestionSpeepRatio</td>
</tr>
<tr>
<td>MinimumCongestionSize</td>
</tr>
<tr>
<td>AgreementVsDissagreementRatio</td>
</tr>
</tbody>
</table>
3.4.11 Formal Algorithms

3.4.11.1 Broadcasting

Every broadcast cycle congestion information is updated before sending. Vehicle has location, speed, direction and timestamp

BROADCAST (vehicle, congestion, knownCongestions)

1    maybeInCongestion ← vehicle.speed < MinimumNormalSpeed
2    inCongestion ← false
3    agreement ← 0
4    if maybeInCongestion then
5      if congestion is empty then
6        congestion ← new Congestion
7        congestion.AddVehicle(vehicle)
8        agreement ← congestion.size / congestion.Disagree
9      inCongestion ← (agreement > MinimumAgreementRatio)
10     congestion.disagree ← 0
11    else
12      myCongestion ← empty
13      vehicle.timeStamp ← CurrentTime
14      knownCongestions.RemoveOld()
15    if inCongestion then
16      Radio.Broadcast(vehicle, congestion, knownCongestions)
17    else
18      Radio.Broadcast(vehicle, knownCongestions)
19    return vehicle, congestion, knownCongestions
3.4.11.2 Receiving

- The parameters correspond to the information received

RECEIVE (vehicle, congestion, knownCongestions)

1. If not (this.congestion is empty) then
2. If vehicle.direction = this.direction and
3. this.congestion.IsNear(vehicle) then
4. If vehicle.speed > MinimumNormalSpeed then
5. this.congestion.Disagree ++
6. Else
7. ADD-ITEM(this.congestion.vehicles, vehicle)
8. If not (congestion is empty) then
9. If this.congestion.Intersects(congestion) then
10. ADD-ITEM(
11. this.congestion.intersectingCongestions,
12. congestion)
13. UPDATE-CONGESTION(
14. this.congestion.vehicles,
15. this.congestion.intersectingCongestions)
16. Foreach c in knownCongestions do
17. - Keep congestions in the same relative direction to
18. the sender or going in the opposite direction
19. If c.direction != this.direction or
20. vehicle.location-this.location = c.direction then
21. If c.Intersects(this.KnownCongestions) then
22. this.KnownCongestions.ReplaceIfNewer(c)
23. Else
24. this.KnownCongestions.Add(c)
3.4.11.3 Moving Time Window

Queue is a priority queue where the priority is the timestamp.

\[ \text{ADD-ITEM (queue, item)} \]
1. \( \text{queue.Enqueue(item)} \)
2. \( \text{minTime ← item.MaxTimeStamp - WindowSize} \)
3. while queue.Top.timeStamp < minTime do
4. \( \text{queue.Dequeue} \)

3.4.11.4 Maintaining Congestions

Updates a detailed congestion area from data in the moving time window. Vehicles contains a list of location, direction, speed and timestamp. Congestions contains a list of intersecting congestions. This function is called after new vehicle information is received.

\[ \text{UPDATE-CONGESTION (vehicles, congestions)} \]
1. \( \text{coreSection ← new Section} \)
2. \( \text{Calculate core section bounds} \)
3. \( \text{foreach v in vehicles do} \)
4. \( \text{elapsedTime ← CurrentTime - v.timeStamp} \)
5. \( \text{predictedLocation ← v.location + v.speed*elapsedTime} \)
6. \( \text{coreSection.ExpandToFit(predictedLocation)} \)
7. \( \text{sections ← \{ coreSection \}} \)
8. \( \text{sectionIndex ← 1} \)
9. \( \text{bounds ← coreSection.bounds} \)
10. \( \text{previousSection ← coreSection} \)
11. \( \text{loop} \)
12. \( \text{newSection ← new Section} \)
13. \( \text{foreach c in congestions do} \)
if \( \text{LENGTH}(c.\text{sections}) \geq \text{sectionIndex} \) then

\[ \text{elapsedTime} \leftarrow \text{currentTime} - c.\text{lastRefreshTime} \]

\[ \text{offset} \leftarrow c.\text{averageSpeed} \times \text{elapsedTime} \]

\[ \text{start} \leftarrow c.\text{sections}[\text{sectionIndex}].\text{start} + \text{offset} \]

\[ \text{end} \leftarrow c.\text{sections}[\text{sectionIndex}].\text{end} + \text{offset} \]

\[ \text{newSection.\text{ExpandToFit}}(\text{start, end}) \]

if \( \text{newSection.isEmpty} \) then exit loop

\[ \text{if not (previousSection at the start of bounds) then} \]

\[ \text{newSection.start} \leftarrow \text{bounds.start} \]

\[ \text{if not (previousSection at the end of bounds) then} \]

\[ \text{newSection.end} \leftarrow \text{bounds.end} \]

\[ \text{if newSection.bounds > previousSection.bounds then} \]

\[ \text{sections.Add(newSection)} \]

\[ \text{bounds.\text{ExpandToFit}}(\text{start, end}) \]

else exit loop

\[ \text{previousSection} \leftarrow \text{newSection} \]

return \( \text{sections, bounds} \)
In order to do research in traffic congestion we need to have a reliable simulation environment that allows us to simulate traffic patterns as well as wireless communications. Testing with real-life vehicles in real-life traffic situations should be done only after the algorithms have proven to work well in the simulated environment. A good simulation environment allows the researcher to test congestion detection algorithms in a myriad of real-life situations.

For the simulations done in this study, Java in Simulation Time (JiST) and the Scalable Wireless Ad hoc Network Simulator (SWANS) are used. This chapter provides an overview of these tools and how we adapted them to simulate vehicular traffic and VOTING. First JiST, the simulation engine, is presented. Then SWANS, the wireless network simulator built on top of JiST, is outlined in detail.

4.1 Flexibility

An important characteristic of a simulation environment is its ability to accommodate a diverse range of traffic patterns, driver behaviors, network configurations, information collected by the vehicles and many other parameters.
so that different algorithms or variations of the same can be tested and evaluated in a variety of environments with little effort.

4.2 Scalability

In order for congestion detection algorithms to be implementable, they need to be able to work with large amounts of vehicles over large geographical areas. The simulation environment must be capable of handling large amount of nodes.

4.3 Event Driven Simulation

There are two basic operations available for entities to interact with the simulation:

- **PostEvent**: Posts an event, at the current time in a priority queue. The event is processed at a later time.
- **Sleep**: To increment the simulation time by a certain amount. It is important to notice that a call to Sleep does not cause the simulation to pause, it merely causes events posted afterwards to be tagged with a later time.

The simulation engine is simply a loop that retrieves events from the priority queue (where the priority is given by the simulation time at which the vent was added to the queue), executes them and continues with the next event in the queue. The events are therefore processed in the order in which they occur in
simulation time which may be different than the actual order in which the events are queued. Simulation time is kept simply as a counter.

4.4 JiST

Java in Simulation Time (JiST) [32] is a Java-based discrete-event simulator. Traditionally, there have been three general design approaches used to create a discrete event simulator: kernels, libraries, and languages. Table 3 (extracted from JiST user manual) shows how JiST improves upon these three types of simulations. By running on a Java virtual machine, JiST brings out the best of each.

<table>
<thead>
<tr>
<th></th>
<th>Kernel</th>
<th>Library</th>
<th>Language</th>
<th>JiST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transparent</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Efficient</td>
<td>✓, x</td>
<td>✓, x</td>
<td>✓, x</td>
<td>✓</td>
</tr>
<tr>
<td>Standard</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Simulation kernels operate at the process boundary controlling process scheduling, inter-process communications, and the system clock, effectively virtualizing time for applications running under it. The advantage of this type of system is that they are completely transparent and unmodified programs can run under them. But this comes at a cost: the process boundary is a source of inefficiency.
Simulation libraries trade the transparency of the kernel approach for increased efficiency. Scheduling and time are handled by the library through explicit API calls from the program. This requires programs to be re-written and merges user code with simulation code in a way that does not guarantee isolation.

Simulation languages are designed to simplify simulation development by including constructs to manage simulation-specific concepts such as simulation time. The compiler can enforce constrains on simulation state and events and can perform optimizations. Unfortunately, simulation languages are domain-specific and lack the modern features of general programming languages. Another problem with simulation languages is the lack of portability.

JiST leverages the Java language and the Java Virtual Machine to bring simulation semantics to regular Java programs. JiST transparently converts regular method calls written in Java to simulation semantics, producing a system that is portable, flexible, efficient, and scalable. There is no need to litter the program with calls to a simulation library and isolation between the simulation kernel and the program is achieved through bytecode rewriting.

### 4.4.1 Architecture

The JiST simulator architecture (Figure 10) consists of four major components: the Java compiler, a bytecode rewriter, a simulation kernel and the Java Virtual Machine. A simulation program is written in plain Java and compiled using the standard Java compiler (1). The compiled classes are then processed by the
bytecode rewriter (2) so that it can run over the simulation kernel (also written in Java) and support simulation time semantics. The modified bytecode, along with the kernel code, can then be executed by the regular Java Virtual Machine (3).

Figure 10. JiST Architecture

The simulation kernel is an event-driven simulation engine as described in Section 4.3. Method calls to objects supporting a special marker interface are automatically converted by the bytecode rewriter to calls to the simulation kernel that enqueue the execution of this method at the current simulation time. Objects that implement this interface are called *entities*. The semantic of a method call to an *entity* object is therefore transformed. A direct call to the simulation kernel is needed to advance the simulation time which is completely independent of execution time.

It is important to note that JiST simulations run on a single thread, therefore there are no concurrency concerns when writing simulation code, however, because of
the semantics of simulation time, mutable data cannot be shared among multiple entities.

4.5 SWANS

Scalable Wireless Ad hoc Network Simulator (SWANS) [33] is a wireless network simulator written on top of JiST. Every SWANS component is encapsulated as a JiST entity, storing its own local state and interacting with other components through the use of events. Figure 11 shows the components in SWANS in a typical arrangement.

Figure 11. SWANS Architecture
4.5.1 SWANS Components

The SWANS architecture contains 7 packages of components: field, radio, MAC (data link), network, routing, transport, and application. They are discussed in detail in the following sections.

4.5.1.1 Field

Radios make transmission downcalls to the simulation “field” and other radios on the “field” receive reception upcalls. The field maintains a collection of radios and their location and uses propagation models to deliver messages between radios and mobility models to displace them.

Table 4. Field Package Components

<table>
<thead>
<tr>
<th>Class / Interface</th>
<th>Description</th>
<th>Implementations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>Provide communications between radio entities</td>
<td>Field</td>
</tr>
<tr>
<td>Fading</td>
<td>Determine signal fade along a path</td>
<td>None, Rayleigh, Rician</td>
</tr>
<tr>
<td>Pathloss</td>
<td>Determine signal degradation along a path</td>
<td>FreeSpace, TwoRay</td>
</tr>
<tr>
<td>Placement</td>
<td>Place nodes in the field</td>
<td>Random, Grid</td>
</tr>
<tr>
<td>Mobility</td>
<td>Instruct the field on how to move nodes</td>
<td>Static, RandomWaypoint, Teleport, RandomWalk</td>
</tr>
</tbody>
</table>
4.5.1.2 Radio

The radio package components implement the physical layer of the node protocol stack by emulating the behavior of a physical wireless transceiver. Radios are responsible for maintaining signal bit-error rate (BER) and signal power and reporting this information as necessary to the MAC layer.

Table 5. Radio Package Components

<table>
<thead>
<tr>
<th>Class / Interface</th>
<th>Description</th>
<th>Implementations</th>
</tr>
</thead>
<tbody>
<tr>
<td>RadioInterface</td>
<td>Interface between MAC and Field, report BER and power</td>
<td>RadioNoiseAdditive, RadioNoiseIndep</td>
</tr>
</tbody>
</table>

4.5.1.3 MAC

Medium Access Control (MAC) entities receive upcalls from the radio entity and deliver them to the network entity. SWANS includes two MAC implementations: 802.11 and “dumb” protocol. The 802.11 component implements the 802.11 data link layer specification [34], which is commonly used in real-world wireless networks. The “dumb” implementation only transmits a signal if the radio is currently idle.
Table 6. MAC Package Components

<table>
<thead>
<tr>
<th>Class / Interface</th>
<th>Description</th>
<th>Implementations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MacInterface</td>
<td>Implement link layer functionality</td>
<td>Mac802_11, MacDumb</td>
</tr>
<tr>
<td>MacMessage</td>
<td>Container class for MAC layer message</td>
<td>Rts, Cts, Ack, Data</td>
</tr>
</tbody>
</table>

4.5.1.4 Network

The network entity receives upcalls from the MAC entities and passes them to the appropriate packet handler. It also receives downcalls from the routing and transport entities, which it queues and eventually passes to the MAC entity. The network package contains an implementation of IPv4 and a no-drop message queue. Loopback and broadcast are implemented.

Table 7. Network Package Components

<table>
<thead>
<tr>
<th>Class / Interface</th>
<th>Description</th>
<th>Implementations</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetInterface</td>
<td>Manage the network layer, interface with message queues and routing logic</td>
<td>NetIp</td>
</tr>
<tr>
<td>MessageQueue</td>
<td>Store received packets before they are processed</td>
<td>NoDropMessageQueue</td>
</tr>
</tbody>
</table>

4.5.1.5 Routing

The routing entity receives upcalls from the network entity with packages that require next-hop information. It also receives upcalls that allow it to peek at all
packets that arrive at a node. It sends downcalls to the network entity when next-hop information becomes available. SWANS implements three wireless ad-hoc routing protocols: AODV, DSR, and ZRP. These protocols are among the most popularly used in ad hoc networking.

Table 8. Routing Package Components

<table>
<thead>
<tr>
<th>Class / Interface</th>
<th>Description</th>
<th>Implementations</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoutInterface</td>
<td>Determine packet routing</td>
<td>RouteAodv, RouteDsr, RouteZrp</td>
</tr>
</tbody>
</table>

4.5.1.6 Transport

The transport entity receives upcalls from the network entity and downcalls from the application entity which it passes to the network entity. SWANS provides three implemented transport protocols: UDP, TCP socket and TCP server. The transport entity is responsible for establishing and maintaining an end-to-end connection.

Table 9. Transport Package Components

<table>
<thead>
<tr>
<th>Class / Interface</th>
<th>Description</th>
<th>Implementations</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransInterface</td>
<td>Provide end-to-end error-free communication</td>
<td>TransTcp, TransUdp</td>
</tr>
<tr>
<td>SocketInterface</td>
<td>Implement a socket API</td>
<td>TcpSocket, TcpServerSocket</td>
</tr>
</tbody>
</table>
4.5.1.7 Application

At the top of the network stack, the application entity makes downcalls to the transport entity and receives upcalls from it. Any Java program can run as an application entity. Applications can open regular sockets to communicate, which transmits packets from the appropriate simulated node. The SWANS extension to the JiST bytecode rewriter rewrites the Java program calls to the standard Java language, network, and I/O libraries that are pertinent to the network simulation into their corresponding SWANS wrapper functions that call into the network simulator stack and JiST.

Table 10. Application Package Components

<table>
<thead>
<tr>
<th>Class / Interface</th>
<th>Description</th>
<th>Implementations</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppInterface</td>
<td>Load a Java application to drive the simulation</td>
<td>AppJava</td>
</tr>
</tbody>
</table>
4.6 Simulating Vehicular Traffic

4.6.1 Highway Mobility Model

SWANS supports the concept of mobility models. A mobility model controls how network nodes move over time. Examples of mobility models are Random Walk [45], Random Way Point [46] [47] and Random Trip [48]. These models are implemented as part of SWANS.

None of the mobility models implemented within SWANS at the time of this writing are adequate to simulate vehicular traffic. One particular characteristic of vehicular mobility that distinguishes it from other models is that in that a vehicle’s movement depends on the movement of other vehicles. The concept of field of vision was introduced to allow a vehicle to “see” others and change its speed and direction accordingly. The highway mobility model implemented simply adjusts the vehicle’s speed according to the following formula:

\[
\text{speed} = \min (\text{Speed Limit}, \ \text{Distance to Vehicle In Front} / \ \text{Minimum Following Time})
\]

Where the Minimum Following Time is the minimum time desired to separate two vehicles (usually about 3 seconds in real-world applications). The speed is constantly adjusted so that a vehicle never runs into the vehicle in front of it. At higher speeds, the minimum distance between vehicles becomes larger.
The highway mobility model moves vehicles in a straight line. There are no lane changes or changes in direction. These are topics of possible future work.

### 4.6.2 Congestion Detection Module

The congestion detection algorithm (VOTING) is implemented within the simulator as a separate module at the application layer. By posting events at constant time intervals, each vehicle can participate in the congestion detection. The standard radio communication entities are used to broadcast information from vehicle to vehicle.

### 4.6.3 Adding Interactivity to JiST/SWANS

The JiST / SWANS simulator is built as a batch process in which there is no relationship between simulation time and real time. In order to be able to visualize the status of a simulation as it happens and to control it, a mechanism was implemented to allow an external entity called a *monitor* to be notified of changes and to be able to control the simulation time so that it can be related to real time. In Section 5.9 we describe this mechanism in detail.

The modifications to JiST/SWANS were made so that the visualization and simulation control module can reside completely outside of the simulation environment. This provides several advantages:
• The visualization module can run on a different computer, not consuming valuable resources in the simulation environment. This is particularly important if we want to run the simulation in a cluster environment and require user interaction only sporadically.

• A visualization and simulation control module can be built using any platform or programming language.

• The relationship between simulation time and real time is controlled outside of the simulation.

It is important to note that the visualization module is only a special case of a simulation monitor and control module. The simulation code is agnostic of what is at the other end. We could, for example, develop an alternate implementation of a monitor module that simply stores all notifications sent to the monitor in a file for future playback and analysis.

4.6.4 Related Work

Other groups [49] [50], have made modifications to JiST / SWANS to support visualization. All of these implementations, however, are not designed to run the visualization and simulation control module as a separate entity. As discussed in Section 4.6.3, this characteristic is very important. Additionally due to the desired flexibility for the visualization module to adapt to any kind of network and application without the need for rewriting the code, the platform of choice was the
Windows Presentation Foundation WPF [51] and not Java. Section 5.3 explains in detail the reasons for this choice.

Research is currently on traffic patterns [14] [15] [16] [17] [18] and how inter-vehicular communication may affect those patterns [5] is an active topic of research.
5 VISUALIZATION AND SIMULATION CONTROL MODULE

We developed a visualization tool that allows researchers to determine the strengths and weakness of a particular congestion detection algorithm by showing them how the system is behaving under a particular situation. This is crucial for the development of effective congestion detection algorithms. Being able to control the pace of the simulation as well as the simulation, network and application parameters with immediate feedback allows the researcher to choose the best strategy or to discover flaws.

5.1 Agnosticism

One major requirement for the Visualization and Simulation Control Module (VSCM) is that it can be used for a wide variety of simulations outside of the VANET and Traffic Congestion Detection domain. The result is a system that allows researchers to customize which properties a node has and how those properties are mapped to visual attributes. The VSCM can be used with any simulator and it is not specifically designed for JiST/SWANS. In fact any simulator, regardless of the language in which it is written or the platform in which it is running the simulator can interact with the VSCM though standard communication protocols such as sockets.
5.2 Scalability

Real-life simulation of large vehicular networks must be handled by the visualization tools. There must be mechanisms to visualize at various levels. The researcher must be able to either see the whole network or a sub-area of it with more in-depth detail.

5.3 Platform of Choice

In order to provide a flexible environment that can adapt to a diverse range of simulation scenarios without the need for programming additional modules, WPF was chosen as the platform for visualization as it offers the ability of defining templates (styles) that are bound to properties that can be defined at run time.

5.3.1 Styling

Styling or templating [52] is the mechanism used by WPF to separate the code logic from the visual aspects of the application. Styling allows changing the look of an application without making changes to the code.

5.3.2 Data Binding

The mechanism by which data coming from the business logic can be bound to visual properties in a style is called data binding [53].
5.4 Architecture

The visualization module is composed of the following elements:

5.4.1 Properties

Properties are the most basic and important element of the VSCM, they provide the mechanism to link data coming to the simulation to visual attributes. The property system is completely flexible; properties are defined by the simulation and are attached to visual attributes through the use of styling and data binding.

5.4.2 Field

The field is the visual space in which nodes are represented. Bindings from node properties to field properties are use to position the nodes within the field. For example, a two-dimensional plane is used for VANET simulations. In this scenario, each node defines a location property consisting of X and Y coordinates within the field. Other implementations may require a three-dimensional space, others may not be aware of node location altogether. These bindings are specified in the field template in the form of a XAML [54] file. It is important to notice that location, is just another property like any other from the point of view of the visualization engine, it does not receive any special treatment.
5.4.3 VOTING Field

The field in VOTING represents a two-dimensional stretch of straight highway extending several miles and contains a list of nodes. Each vehicle (node) is placed in the field according to the value of its location property.

5.4.4 Node Classes

In order to give nodes a visual representation, the concept of node classes is introduced. A node class is simply a template that specifies how a node and some of its properties are going to be visually represented. Different node classes are used to represent nodes that look different than others. For example, in a VANET simulation we may want to represent cars and trucks differently. A node class is defined by the set of properties a node belonging to a class contains. The class a node belongs to is defined when the node is created and cannot be changed. A node class is immutable. When a node class is created it is assigned a template name which the visualization module uses to locate a XAML file that contains the template that defines the visual representation of that type of nodes.

5.4.4.1 VOTING Node Classes

We have only one node class for the visualization of VOTING: Vehicle. A Vehicle class is defined by the following properties:
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>Location2D</td>
<td>The location of the node within the field</td>
</tr>
<tr>
<td>Color</td>
<td>Color</td>
<td>The vehicle’s body color used to distinguish vehicles more easily</td>
</tr>
<tr>
<td>Selected</td>
<td>Boolean</td>
<td>A selected node displays more detailed information such as known congestion areas</td>
</tr>
<tr>
<td>Reception Radius</td>
<td>Integer</td>
<td>The radius of the radio reception area</td>
</tr>
<tr>
<td>InCongestion</td>
<td>Boolean</td>
<td>Indicates the vehicle is part of a congestion</td>
</tr>
<tr>
<td>KnowsCongestion</td>
<td>Boolean</td>
<td>Indicates the vehicle has information about one or more congestions</td>
</tr>
<tr>
<td>Groups</td>
<td>Rectangle[]</td>
<td>A list of known congestion areas</td>
</tr>
</tbody>
</table>

### 5.4.5 Nodes

A node is simply an instance of an object that contains a set of properties as defined by its node class. For example, a vehicle is a node that exposes its location and body color among other attributes relevant to the particular application. Every node has a unique ID used to identify it through the simulation. Notice that this ID is used for visualization and simulation control purposes only; it is not used in any way by the congestion detection algorithm. Figure 12 shows a simple template for a node written in XAML. The node is represented by a small circle with the node’s ID displayed inside and the fill color of the circle
determined by the *Color* property on the node. Notice the binding expressions to the *Color*, *ID* and *Info* properties of the node.

```xml
<DataTemplate x:Key="simpleNodeTemplate">
  <Canvas x:Name="nodeContainer">
    <Canvas.ToolTip>
      <StackPanel Background="Transparent">
        <TextBox Text="{Binding Path=Info}"/>
      </StackPanel>
    </Canvas.ToolTip>
    <Ellipse Width="16" Height="16" Fill="{Binding Path=Color}"/>
    <Ellipse Width="10" Height="10" Margin="3,3,3,3" Fill="White"/>
    <TextBlock Width="10" Height="10" Margin="3,3,3,3" TextAlignment="Center" Text="{Binding Path=ID}" Background="Transparent" FontSize="8" Foreground="Black"/>
  </Canvas>
</DataTemplate>
```

Figure 12. Template for a Simple Node

5.5 Communications

The VSCM is designed as a separate program that can run at a workstation, separate from the simulation module. This architecture is ideal for simulation of
large scale networks on which the simulation can be running on a cluster of powerful computers while the visualization module remains on the researcher’s desktop. Data is sent back and forth using sockets. Figure 13 shows how the VSCM communicates with the simulation through the use of notifications and commands discussed later in this chapter.

![Diagram of VSCM and Simulation Architecture](image)

*Figure 13. Visualization and Simulation Control Module Architecture*

### 5.6 Encoding

In order for the simulation and visualization modules to be independent of their respective platform, data sent between these modules is encoded using a standard format for which decoders are easily found or implemented. For the easy diagnostics and debugging, the encoding used today by the VSCM is JSON, a human readable representation of Java Script objects. Faster encoders may be used in the future to improve performance on large scale simulations.
5.7 Notifications

Notifications are sent from the simulation to the VSCM via the communication channel to make updates to the visual representation of the current state. Notifications are sent every time a state changes in either the simulation as a whole or in a particular node. Notifications are queued prior to sending so that all notifications that occurred at the same simulation time are delivered in one package.

5.7.1 Notification Types

The following are the different notification types supported by the VSCM:

<table>
<thead>
<tr>
<th>Table:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Simulation Started:</strong></td>
<td>Used to initialize the visualization and simulation control module. It specifies information about the field and the nodes.</td>
<td></td>
</tr>
<tr>
<td><strong>TemplateName</strong></td>
<td>String</td>
<td>File containing the template for the field and nodes.</td>
</tr>
<tr>
<td><strong>NodeClasses</strong></td>
<td>NodeClass []</td>
<td>List of classes describing node attributes.</td>
</tr>
</tbody>
</table>
**NodeClass**: Describes a class of node.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TemplateName</strong></td>
<td>String</td>
<td>Name of visual template for this node. This node template must be defined in the template file indicated in SimulationStarted.</td>
</tr>
<tr>
<td><strong>Properties</strong></td>
<td>NodePropertyInfo []</td>
<td>Properties for this node class</td>
</tr>
</tbody>
</table>

**NodePropertyInfo**: Specifies a default value for a property

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>String</td>
<td>Unique name for the property</td>
</tr>
<tr>
<td><strong>DefaultValue</strong></td>
<td>Object</td>
<td>Default value for the property</td>
</tr>
</tbody>
</table>

**Node Created**: Lets the monitor know that a new node is being added to the simulation.

<table>
<thead>
<tr>
<th>Class</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class</strong></td>
<td>String</td>
<td>Node class name. Must reference one of the classes specified at SimulationStartedNotification</td>
</tr>
<tr>
<td><strong>ID</strong></td>
<td>String</td>
<td>Unique ID for the node.</td>
</tr>
<tr>
<td><strong>Properties</strong></td>
<td>PropertyValue []</td>
<td>List for initial property values for the node</td>
</tr>
</tbody>
</table>

**PropertyValue**: Specifies an initial value for a property

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>String</td>
<td>Unique name for the property</td>
</tr>
<tr>
<td><strong>Value</strong></td>
<td>Object</td>
<td>Initial value for the property</td>
</tr>
</tbody>
</table>
**Node Deleted**: Lets the monitor know that a node has been removed from the simulation.

<table>
<thead>
<tr>
<th>ID</th>
<th>String</th>
<th>Unique ID for the node</th>
</tr>
</thead>
</table>

**Property Changed**: Lets the monitor know that a property has changed for a particular node.

<table>
<thead>
<tr>
<th>ID</th>
<th>String</th>
<th>Unique ID for the node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td>Property Name</td>
</tr>
<tr>
<td>Value</td>
<td>String</td>
<td>New Value</td>
</tr>
</tbody>
</table>

**Pause**: Instructs the monitor to pause for a period of time to reflect the passing of simulation time.

<table>
<thead>
<tr>
<th>Period</th>
<th>Long</th>
<th>Time to pause in nanoseconds</th>
</tr>
</thead>
</table>

### 5.8 Commands

Commands are sent from the VSCM to the simulation module through the communication channel. Commands are used to control simulation parameters as well as node properties. Commands can be directed to the simulation environment; for example a *tick* command instructs the simulation to process the pending events; or to specific nodes; for example a command to instruct a
vehicle to be disabled so that a congestion can start to form is sent to the
selected vehicle.

5.8.1 Command Types

5.8.1.1 Generic Commands

Generic commands are not specific to any application. There is only one
command in this category:

Clock Tick: Instructs the simulation to continue processing messages from
the message queue until a new notification

5.8.1.2 VOTING Commands

The following list of commands is specific to VANET simulation and VOTING:

| **Scale Speed**: Instruct the simulation to change the speed of a specific
  vehicle by multiplying its current speed by a factor. This command allows to
  simulate a vehicle breakdown or accident and produce a congestion. |
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ID</strong></td>
</tr>
<tr>
<td><strong>Factor</strong></td>
</tr>
</tbody>
</table>
**Select Node:** Change the selected state of the node. Selected nodes typically send more details about their status to the monitor.

<table>
<thead>
<tr>
<th>ID</th>
<th>String</th>
<th>Node ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selected</td>
<td>Boolean</td>
<td>Selected state</td>
</tr>
</tbody>
</table>

**Suspend Visualization:** Suspend all notifications to the monitor until a certain simulation time is reached.

| Period | Long | Time to suspend in nanoseconds |

5.9 Controlling Simulation Time

In order to visualize the simulation while it happens, a synchronization mechanism was created: when the simulation module is configured to “hold for monitor” it waits for a command from the monitor before starting to process events from the simulation event’s queue. When a `ClockTickCommand` is received the simulation continues running and queuing notifications. When the simulation time advances ahead of pending notifications held in the notification queue, the simulation sends the pending notifications, flushes the queue and waits again for a `ClockTickCommand`. This puts the monitor (the visualization and simulation control module in this case) in control of the pace of the simulation time.
When a package of notifications is sent from the simulation to the monitor, the simulation time elapsed since the previous package sent is included at the beginning of the package in the form of a *PauseNotification*. The visualization module uses this information to give the user feedback as to the speed of these changes in state. For example, if a notification is received indicating that a node's location has changed, the *PauseNotification* information can be used to visualize the real speed at which the node moved. A clock multiplier in the visualization module allows the user to speed up or slow down the simulation by given a different interpretation to the simulation time. The monitor instructs the simulation to continue by sending a *ClockTickCommand* after the pause time (transformed by the clock multiplier) has elapsed.

The step-by-step mode allows the user to manually control the *Tick* commands going to the simulation. The UI for controlling the clock is defined in the main window template and can be customized to suit specific needs.

Figure 14 shows the sequence of events, in chronological order from top to bottom, that allow the user to control the simulation pace. Figure 15 shows the logic the simulation monitor uses to control the simulation time according to the instructions received from the VSCM.
Monitor Starts

Simulation Starts

Initialization

Nodes Created

Field area initialized

Wait for Tick

Tick

Process events from the queue while queuing-up notifications that occur at the same simulation time.

When a notification with a later simulation time needs to be queued, the notifications in the queue are sent to the monitor, the notification queue is emptied.

A Pause notification is included at the end of the package sent to the monitor. Pause Time: Pause = Current Time – Time of Last Notification

Notifications

Update Visuals via Data-Binding.

Sleep for Pause Time

Tick

Wait for Tick

Tick

Process messages from queue until time advances

…

Figure 14. Simulation Control Sequence
Figure 15. Flow Chart of Simulation Controlled by Monitor
6 SIMULATING AND VISUALIZING VOTING

6.1 Visualization Window

The Visualization Window provides a visual representation of the field and nodes by linking the field template to data coming from the simulation. The simulation control window provides basic functionality such as zooming and scrolling and can be customized using a template. The following screenshots show the simulation window customized to visualized vehicular traffic for the visualization of VOTING.

Figure 16. Visualization Screen
6.2 Control Panel

The control panel provides an area on which specific User Interface (UI) elements are linked to simulation commands. Figure 17 shows the control panel created for simulating VOTING. UI for general simulation commands for clock control allow the user to control the simulation speed, pause and resume the simulation, going step-by-step and suspend visualization for a period of time. We also created UI for showing current simulation time. A push button for simulating a vehicle breakdown sends a command that is specific to the selected vehicle and instructs it to reduce its speed by a factor (we use 90% in our simulation), creating a congestion. A vehicle can be selected by simply clicking on it.

Figure 17. Control Panel
6.3 Running the Simulation

By letting the clock run, the simulation runs continuously at the desired speed. In this example we created a congestion by disabling two vehicles (at the front of the congestion). We can observe the congestion starting to form in Figure 18.

![Figure 18. Start of a Congestion](image.png)

We identify vehicles that are part of a congestion by an orange circle drawn around them. Vehicles that know about a congestion but are not part of one are identified by a yellow circle (they are also labeled in Figure 18). We can also see that at this point in time some vehicles, away from the congestion are still not aware of it; they eventually receive this information when getting within transmission range of a vehicle that knows about it, that vehicle may be one travelling from the opposite direction or a vehicle near the tail of the congestion.
Figure 19 shows the congestion area (large red rectangle) as seen by the selected vehicle (shown with headlights on and surrounded by a big circle representing its transmission radius). This UI allows the user to inspect the different views of the congestion different vehicles have at any given time.

Figure 19. Congestion As Seen By the Selected Vehicle

Figure 20 shows the congestion sections (discussed in 3.4.6) as seen by a vehicle inside the congestion. We can notice two sections a 0-hop section and a 1-hop section as the congestion area exceeds the reception range.

Figure 20. Congestion Sections As Seen By the Selected Vehicle
Figure 21 shows the congestion dissipating after the vehicles in front are returned to normal operation. Figure 22 shows the congestion finally flushed out of the system.

6.4 Validating VOTING

We validated VOTING against different scenarios. The following set of screenshots show how VOTING can recognize real congestions and is able to track their movement over time. We also show how false congestions are treated properly and how dissipating congestions are flushed out of the system.
We ran several simulations to evaluate the effectiveness of VOTING, we present here the most important scenarios and show a sequence of how congestions were detected, disseminated and updated.

6.4.1 Congestion Formation

The first group of figures (Figure 22 - Figure 26) corresponds to the formation of a congestion. The figures are presented in chronological order and belong to the same simulation run.

![Figure 23. A Congestion Starting to Form](image)

In this simulation, the transmission radius is set such that vehicles not in a congestion cannot directly reach a vehicle in front of them (we analyze the results of a scenario when the reception radius is big enough to reach vehicles in front of them in Section 6.5), therefore they only become aware of the congestion when one of the following two conditions is met:

- The vehicle reaches a vehicle near the tail of the congestion as shown in Figure 24.
• The vehicle reaches a vehicle traveling in the opposite direction that already went by the congestion (within transmission range) as shown in Figure 25.

Figure 26 shows how sections are formed (as explained in Section 3.4.6). Figure 27 shows vehicles that are far away from the congestion getting aware of it.

Figure 24. A Congestion Identified as Such and Broadcasted to Neighbors

Figure 25. Vehicles Going the Opposite Direction Acting as Carriers

Figure 26. A Detailed View of a Congestion Viewed by One of its Members
6.4.2 False Congestion

We studied how false congestion affects VOTING. Figure 28 shows a simulation in which a number of vehicles are pulled over the shoulder. Although the number of slow vehicles is enough to define a congestion, the congestion gets constantly invalidated by vehicles passing by it at normal speed (as discussed in Section 3.4.5) and therefore does not get broadcasted to vehicles away from that area.

6.4.3 Dissipating Congestion

The following group of screenshots shows how a dissipating congestion is properly handled by VOTING. After the congestion is completely cleared it remains in the memory of vehicles that can no longer communicate with vehicles travelling through that area, as seen in Figure 30. Eventually, because the congestion is never “refreshed” again it gets flushed out of the system after it reaches a certain age (Figure 31).
Figure 29. A Congestion Starting to Dissipate

Figure 30. A Shrinking Congestion Continuing to Dissipate

Figure 31. A Residual Congestion Not yet Flushed from the System

Figure 32. The Congestion Finally Flushed out of the System
6.5 Simulation Results

In this section we present two separate simulation runs with different parameters. Simulation Run 1 was designed in such a way that vehicles cannot reach via radio the vehicle in front of them. Congestion information can only be obtained as the vehicle approaches the congestion or when a vehicle travelling in the opposite direction, that came in contact with the congestion, reaches it. These are the specific parameters used for Simulation Run 1:

<table>
<thead>
<tr>
<th>Simulation Run 1</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Vehicles: 100</td>
</tr>
<tr>
<td>Initial inter-vehicular separation: 30m with 20% chance of double gaps</td>
</tr>
<tr>
<td>Transmission Radius ≈ 20m</td>
</tr>
<tr>
<td>Speed Limit: 20 m/s (about 45 mph)</td>
</tr>
<tr>
<td>Broadcast Interval: 0.2 seconds</td>
</tr>
</tbody>
</table>

The following group of charts (Figure 32 - Figure 35) show how the propagation of congestion information over time and distance occurs. Figure 33 shows how the congestion grows in number of vehicles over time once it was formed (around t = 5 sec). Figure 34 shows how the congestion information is rapidly propagated to other vehicles. The drop around time 40 sec. corresponds to a big gap left in the simulation between vehicles leaving the highway and vehicles entering. We
can observe that this drop is rapidly recuperated as vehicles entering the highway become aware of the congestion. Figure 35 shows the relationship between the number of vehicles in the congestion and the number of vehicles aware of it, as the size of the congestion grows the number of vehicles aware of it grows rapidly. Figure 36 shows the relationship between the distance to the congestion and the time it took for a vehicle to be aware of it. We can see two groups of points, one representing the vehicles going towards the congestion (darker circles) and one representing the vehicles moving in an opposite direction (lighter squares). Vehicles on the opposite side of the highway act only as carriers of this information but they are not affected by it, since there are no vehicles carrying the information to them (the front of the congestion is clear) they become aware of it only when they come within transmission range of the congestion (about 20m in this example), the former is the group of interest. We can observe that for this group the relationship shows to be linear. The division seen in this group corresponds to the particular way vehicles enter and leave the highway during the simulation: new vehicles entering the highway (upper group) find a congestion that is already been active for some time, their discovery time can therefore be no lower than this initial age.
Figure 33. Congestion Formation (100 Vehicles)

Figure 34. Congestion Propagation (100 Vehicles)
Figure 35. Congestion Awareness vs. Congestion Size (100)

Figure 36. Distance to Congestion vs. Time to Discover (100)
In the second simulation run (Simulation Run 2) we increase the number of vehicles and the vehicle density; we also increase the radio power in a way that information can now travel for large stretches of highway without the need for vehicles traveling in the opposite direction carrying it. A large percentage of vehicles (around 60%) can reach a vehicle in front of them directly through the radio. The following parameters were used for Simulation Run 2:

<table>
<thead>
<tr>
<th>Simulation Run 2</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Vehicles: 1,000</td>
</tr>
<tr>
<td>Initial inter-vehicular separation: 25m with 40% chance of double gaps</td>
</tr>
<tr>
<td>Transmission Radius ≈ 30m</td>
</tr>
<tr>
<td>Speed Limit: 20 m/s (about 45 mph)</td>
</tr>
<tr>
<td>Broadcast Interval: 0.2 seconds</td>
</tr>
</tbody>
</table>

We run this simulation for one minute, and then eliminate the cause of the congestion. Then we run it for another minute. As Figure 37 shows, by that time the congestion is already dissipated.

In Figure 38 we can see that only 10 seconds after the congestion was formed, most of the vehicles in the system know about the congestion. Figure 39 shows that after the congestion size reaches about 14 vehicles, most of the vehicles in the system are aware of it. Once the congestion starts to clear ($t \approx 60$ sec), the number of vehicles aware of it starts to decrease slowly. Once the congestion is completely dissipated and the maximum congestion age is reached the
congestion gets flushed away from the system ($t \approx 140$ sec). In Figure 40 we can see that in this case the behavior of vehicles in both sides of the highway is similar, this is because transmission ranges and density allow vehicles to be aware of the congestion before they reach it even if they are travelling in the opposite direction.

**Figure 37. Congestion Formation (1000 Vehicles)**
Figure 38. Congestion Propagation (1000 Vehicles)

Figure 39. Congestion Awareness vs. Congestion Size (1000)
Figure 40. Distance to Congestion vs. Time to Discover (1000)

When comparing Simulation Run 1 with Simulation Run 2 we can see that the effect of increased radio coverage and/or vehicle density is extremely important. In a real-world system, statistical data must be used to find the optimal cost-effectiveness formula that will yield good results while maintaining reasonable requirements on the hardware.
7 CONCLUSION

We presented VOTING: a distributed, collaborative algorithm for traffic congestion detection using VANETs that preserves driver privacy, conserves bandwidth and does not require any external infrastructure but uses vehicles themselves, an abundant resource in today's highways, as gatherers and distributors of information. We showed the effectiveness of the system in a variety of scenarios through the use of a flexible framework for simulation and visualization designed and developed to aid in the research of VANETs and other types of networks.

We showed how VOTING improves upon existing traffic congestion solutions that rely on road and network infrastructure in terms of cost, ease of deployment and reliability. We also show how our solution addresses location privacy concerns and makes an efficient use of the network channel by transmitting aggregated data in the form of congestion sections. Additionally, we presented a flexible simulation and visualization system that extends JiST/SWANS, a powerful wireless network simulator, by providing support for VANETs and by adding visualization and interactivity, allowing researchers to visualize the simulation, control simulation time and interact with it while it is running.
Traffic congestion will continue to be a problem in modern societies for the foreseeable future. The work on this study shows how efficient systems that detect and disseminate traffic congestion information can help us move towards a future where the driver is empowered with real-time traffic information that enables efficient routing and produces a more efficient use of our road infrastructure.

7.1 Future Work

We see many areas for future work that can expand this research. We identify four different areas for more research and where existing research can be integrated into ours:

- Complex traffic modeling and driving behaviors (mobility models) that incorporate lane changing and multiple entry and exit points can be integrated to our simulation framework to validate and evaluate our algorithm in more complex scenarios, taking them closer to real world applications.
- Inner-city traffic where more complex topologies exist and external events such as traffic lights need to be considered.
- Efficient broadcasting protocols for VANETs including hybrid protocols that use V2V and V2I communications.
- To study the effect of message losses, vehicle participation, transmission power and other physical characteristics of the underlying network may have in the effectiveness of congestion detection.
• Use of congestion information for efficient routing, including the use of congestion characteristics like size, age and number of vehicles as well as statistical data to forecast future traffic.


10. New York City Department of Transportation. *nyctmc.org.* [Online]


42. **Kargl, Frank, Ma, Zhendong and Schoch, Elamar.** Security Engineering for VANETs. *Ulm University, Institute of Media Informatics*. 2006.


