8-2008

A REAL-TIME TRAFFIC CONDITION ASSESSMENT AND PREDICTION FRAMEWORK USING VEHICLE-INFRASTRUCTURE INTEGRATION (VII) WITH COMPUTATIONAL INTELLIGENCE

Yongchang Ma
Clemson University, ma_yongchang@hotmail.com

A REAL-TIME TRAFFIC CONDITION ASSESSMENT AND PREDICTION FRAMEWORK USING VEHICLE-INFRASTRUCTURE INTEGRATION (VII) WITH COMPUTATIONAL INTELLIGENCE

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Civil Engineering

---

by
Yongchang Ma
May 2008

Accepted by:
Dr. Mashrur Chowdhury, Committee Chair
Dr. Adel Sadek
Dr. Wayne Sarasua
Dr. Jennifer Ogle

ABSTRACT

This research developed a real-time traffic condition assessment and prediction framework using Vehicle-Infrastructure Integration (VII) with computational intelligence to improve the existing traffic surveillance system. Due to the prohibited expenses and complexity involved for the field experiment of such a system, this study adopted state-of-the-art simulation tools as an efficient alternative.

This work developed an integrated traffic and communication simulation platform to facilitate the design and evaluation of a wide range of online traffic surveillance and management system in both traffic and communication domain. Using the integrated simulator, the author evaluated the performance of different combination of communication medium and architecture. This evaluation led to the development of a hybrid VII framework exemplified by hierarchical architecture, which is expected to eliminate single point failures, enhance scalability and easy integration of control functions for traffic condition assessment and prediction.

In the proposed VII framework, the vehicle on-board equipments and roadside units (RSUs) work collaboratively, based on an intelligent paradigm known as ”Support Vector Machine (SVM),” to determine the occurrence and characteristics of an incident with the kinetics data generated by vehicles. In addition to incident detection, this research also integrated the computational intelligence paradigm called “Support Vector Regression (SVR)” within the hybrid VII framework for improving the travel time prediction capabilities, and supporting on-line leaning functions to improve its performance over time. Two simulation models that fully implemented the functionalities

of real-time traffic surveillance were developed on calibrated and validated simulation network for study sites in Greenville and Spartanburg, South Carolina. The simulation models' encouraging performance on traffic condition assessment and prediction justifies further research on field experiment of such a system to address various research issues in the areas covered by this work, such as availability and accuracy of vehicle kinetic and maneuver data, reliability of wireless communication, maintenance of RSUs and wireless repeaters.

The impact of this research will provide a reliable alternative to traditional traffic sensors to assess and predict the condition of the transportation system. The integrated simulation methodology and open source software will provide a tool for design and evaluation of any real-time traffic surveillance and management systems. Additionally, the developed VII simulation models will be made available for use by future researchers and designers of other similar VII systems. Future implementation of the research in the private and public sector will result in new VII related equipment in vehicles, greater control of traffic loading, faster incident detection, improved safety, mitigated congestion, and reduced emissions and fuel consumption.

DEDICATION

This dissertation is dedicated to my father Weixian Ma and mother Zunli Ge.

ACKNOWLEDGMENTS

I would like to take this opportunity to express my deepest appreciation and gratitude to my advisor Dr. Mashrur Chowdhury, for believing in my abilities, for furnishing me with the required resources, and for his assistance, guidance, understanding and support throughout this entire dissertation process. I have learned so much, and without you, this would never have been completed in a satisfactory manner. Thank you so much for a great experience.

I would also thank Dr. Adel Sadek, Dr. Wayne Sarasua, and Dr. Ogle for serving in my dissertation committee. Thank you for your insights and guidance throughout the past three years. Show of gratitude also goes out to Dr. Kuang-Ching Wang for all his time, effort and expertise in guiding me to complete this dissertation.

Appreciation next goes out to my fellow graduate students, for their friendships and support. These include Devang Bagaria, Parth Bhafsa, Ryan Fries, Imran Inamdar, Liz Stephen, Carol Hamlin, and Yan Zhou.

Last but not least, to my parents and family in general, for their never-ending love and support throughout my life

TABLE OF CONTENTS

Table of Contents (Continued)

Page

LIST OF TABLES

LIST OF FIGURES

List of Figures (Continued)

Figure Page

List of Figures (Continued)

Figure Page

List of Figures (Continued)

Figure Page

CHAPTER 1

INTRODUCTION

Ensuring that the highway transportation system remains a productive part of the nation's infrastructure in the coming decades without costly expansion is of paramount importance. Increasing population, more vehicles, and urban sprawl now impact a highway system already overburdened and inadequately maintained. Given the anticipated increase in highway traffic in coming decades, America can expect problems of traffic management to continue to grow (Cambridge Systematic and TTI 2005). Therefore, the United States and other developed nations must create technologies that reduce the burden on overtaxed road ways that support national and global economies, while enhancing national security and simultaneously improving environmental quality.

1.1 Background and Motivation

Many countries have been using technologies and systems to better manage and control their surface transportation network under the umbrella of Intelligent Transportation Systems (ITS). The operation of numerous key components of ITS, such as incident management, real-time traffic management, traveler information, and hazard evacuation, relies heavily on the support of an effective and efficient highway traffic surveillance system. For example, maintaining the flow of traffic requires continuously monitoring the highway network for any problems and taking quick action to mitigate the impacts of those problems. Recent advances in computational intelligence, embedded

systems and wireless communication technologies, can make this process more effective and efficient.

An opportunity exists for developing the next generation highway traffic surveillance system in the use of "Vehicle-Infrastructure Integration (VII)" system, an emerging frontier in ITS. The VII-enabled vehicles, which are equipped with on-board processors, positioning systems and communication interfaces, are able to collect, process and disseminate traffic data and information to roadside units (RSUs). In this envisioned VII system, vehicle on-board units are expected to work autonomously to collect traffic statistics, such as vehicle locations, following distances and driver maneuvering at programmed intervals and transmit to RSUs. Additional statistics including vehicle speeds and accelerations can be converted and derived from these individual statistics. Research has shown that vehicle-generated data can provide reliable estimates of traffic conditions, including identifying incidents and congestion (Crabtree et al. 2006; Cheu et al. 2002; Sermons et al. 1996; Qi et al. 2002).

The existing concepts of VII are currently based on a centralized architecture. It is extremely difficult, if not impossible, to transmit, aggregate, and process massive amounts of information that are expected to be generated from VII-enabled vehicles at a central point. However, a distributed system without centralized control is complicated to implement system wide control and optimization. A hybrid VII framework, which is envisioned to monitor, process and control local traffic conditions locally through the collaboration and coordination of vehicles, RSUs and controllers, is expected to be an improvement over centralized only or distributed only operations.

1.1.1 Networking and Processing Architecture

The majority of today's highway traffic surveillance systems rely on the roadside sensors, which are connected by copper wires, fiber-optic cables, or cellular wireless network to a centralized control point. At this center, human operators are responsible for continuously monitoring and analyzing large amounts of data acquired from sensors, such as loop, radar and video detectors, to make the appropriate traffic control decisions. These decision-makers select the response strategies adapted to the information provided by sensors. The implementation commands are then conveyed to the field personals and equipments via the same communication infrastructure.

However, there are several problems associated with this existing centralized highway traffic surveillance network. First and foremost, the required dedicated communication infrastructure is prohibitively expensive as a system grows in coverage and number of sensors, making it difficult for wider deployment and expansion into broader suburban and rural areas. Also, the communication infrastructure and control center of these systems are vulnerable to both terrorist attacks and natural disasters. Furthermore, the response time of these centralized decision making systems is prohibitively long, which is a critical issue for any type of quality incident management system. Finally, human operators who monitor the sensors endure high working stress, which in turn decreases the system reliability.

Though the centralized architecture is prevalent for highway traffic surveillance system, distributed control concepts are not new to traffic signal control systems. To locally optimize traffic delays, traffic signal controllers have long been organized into

local clusters. The state-of-the-art such traffic signal control systems include the Split, Cycle, Offset Optimization Technique, or SCOOT (Siemens 2006); the Sydney Coordinated Adaptive Traffic System, or SCATS (Tyco Integrated Systems 2006); and the Real-time Hierarchical Distributed Effective System, or RHODES (Mirchandani and Head 1998). While all are quite effective traffic control systems, they are limited to only the scope of signal control, and fixed signal clusters. Moreover, they also require expensive communication infrastructures.

To improve the current highway traffic surveillance, Coifman and Ramachandran (2004) outlined a vision of deploying intelligent sensors along highways that could engage in distributed sensing and local data processing to report only concise information to traffic management center (TMC) or other responsible controllers when anomalies are detected (Coifman and Ramachandran 2004). The strength of this approach lies in the ability of sensors and controllers to make collaborative decisions without human intervention.

A hybrid framework, which is expected to improve the overall networking and processing performance, is the likely solution for large area on-line highway traffic surveillance systems. The envisioned hierarchical hybrid system is comprised of multiple layers of components such as TMC, controllers, RSUs, and vehicles. In each layer, components work in distributed fashion, while the immediate upper level supervisor manages its employee in a centralized fashion. The number of levels and the classification varies with the specific traffic network characteristics and application cases (Kochhal et al. 2003; Subramanian and Katz 2000).

1.1.2 Vehicle-Infrastructure Integration

The recent development of Vehicle-Infrastructure Integration (VII) proposes to equip vehicles and roadside infrastructures, such as traffic sensors, signals and message signs, with wireless communication interfaces to communicate with each other. These emerging VII technologies create great opportunities for the next generation highway traffic surveillance systems.

Since 2003, FHWA has sponsored a variety of efforts that led to the development of the national Vehicle Infrastructure Integration (VII) architecture and its functional requirements (FHWA 2005). Currently, the USDOT is conducting a research program called the Mobility Applications for Vehicle Infrastructure Integration initiative (National VII Coalition 2007). In that program, researchers are studying the potential for transmitting information between infrastructure and vehicles to provide mobility benefits. Several states including California (PATH 2007) and Michigan (MIDOT 2005) are testing various methods for implementing these types of programs (ITS America 2007).

Multiple sensors and computers in modern cars have the ability to access several hundred data types (FHWA 2005). These devices make it possible to provide constantly changing data, such as speed, acceleration/deceleration, position, and maneuver data to the traffic surveillance system. The VII system is in turn expected to substantially improve information availability for highway management, thereby increasing the safety and efficiency of large-scale highway systems (National VII Coalition 2007). Although previous and current research has demonstrated the great potential of using VII to benefit highway and intersection collision avoidance, traveler information dissemination, and

driver based incident reporting system, only limited research has been undertaken regarding the feasibility of using VII for real-time highway traffic surveillance.

VII California (2006) demonstrated the efficacy of using VII for on-line traffic condition assessment, while other studies modeled VII traffic for road and weather condition assessment (Petty and Mahoney 2007; Tanka and Piotrowicz 2007). Rather than just collecting data from VII, it is possible to use VII for more detailed on-line traffic condition assessment with an emphasis on incident detection. Various studies discovered encouraging data showing an increase in the ability of VII to detect highway incidents using vehicle-generated microscopic data (Qi et al. 2002; Sermons and Koppelman et al. 1996; Cheu et al. 2002). Crabtree et al. (2007) and Tanikella et al. (2007) illustrated that travel time data generated from VII can provide reliable estimates of traffic conditions and identify incidents. However, those studies, based on simple threaded classification or statistical analysis, did not take full advantage of microscopic traffic statistics available from VII. The VII system is expected to enable vehicle on-board units to work autonomously to collect microscopic traffic statistics, such as vehicle trajectory, space gap and lane changing behavior at programmed intervals. Along with its unprecedented powerful data collection capability, VII is a suitable and promising tool for real-time traffic condition assessment and prediction.

1.1.3 Computational Intelligence

Existing incident detection algorithms include spatial measurement-based algorithms and automatic incident detection (AID) algorithms, which use point-based or/and link-based data. Spatial-based algorithms include video image processing, and

AID algorithms include pattern recognition based (e.g. California algorithm (Payne and Tignor 1978)), Catastrophe theory based (e.g. McMaster algorithm (Persaud and Hall 1989)), statistical based (e.g. ARIMA (JHK and Associates 1993), DELOS (Stephanedes and Chassiakos 1993), Bayesian (Levin and Krause 1979), SSID (Antoniades and Stephanedes 1996)), and Artificial Intelligence (AI) based. Point-based data collection, which use traffic flow measurements made at a point, are common types of applications that employ existing sensor technologies, such as inductive loop detectors, microwave radar, infrared, ultrasonic, and acoustics detectors (Ozbay and Pushkin 1999). Conversely, a link-based data collection system uses individual vehicles as probes to assess the roadway link statistics such as travel time and average speed.

Detection rates, false alarm rates, and time to detection have traditionally been used to evaluate existing incident detection algorithms. While several algorithms reported detection rates of 100 percent (ARIMA, Bayesian, SSID), they also either had a high false alarm rate or longer detection time (Martin et al. 2001). Other algorithms have been found to provide low false alarm rates (less than 10 false alarms per hour) and short detection time (less than one minute) including those based on Artificial Neural Networks (ANN), which was able to achieve a detection rate over 90% (Martin et al. 2001). However, the performance of almost every AID algorithm is sensitive to the placement densities of the traffic sensors. Evidently, existing AID algorithms left room for improvement.

AI-based algorithms with their learning capabilities allow a sensor to improve detection performance over time to adapt to the changing traffic conditions. While ANNs

have been the most commonly applied AI tool for incident detection, Lin (2004) reported that the AI paradigm known as the Support Vector Machine (SVM) had a greater learning and prediction potential compared to ANN. In addition, SVM requires less computation resources and avoids the over-fitting problems of ANN.

The SVM paradigm family includes one-class SVM for distribution function estimation, Support Vector Classification (SVC) for pattern classification and Support Vector Regression (SVR) for regression or function estimation. The underlying theories behind them are similar. SVM is a collection of algorithms that achieve nonlinear regression by mapping the training samples onto a high dimensional kernel-induced feature space, followed by linear regression in that space. Since the kernel mapping is implicit, depending only on the inner or dot product of the input data vectors, it is possible to map the data into high dimensions and still keep the computational cost low.

Thus far, SVM has had limited applications in the transportation field. Previous examples include use for travel time, traffic speed and traffic flow predictions, and incident detection in the context of ITS applications (Wu et al. 2004; Vanajakshi and Rilett 2004; Ding et al. 2002; Cheu et al. 2003). SVM is also computationally efficient since only the cases corresponding to the support vectors add to the computational cost. Depending on the chosen kernel function, SVM models need only a subset of the complete case library (as low as 1% of cases) to train the SVM function (Vanajakshi and Rilett 2004). In addition, Sun et al. applied SVM for vehicle detection using extracted features from Gabor filters (Sun et al. 2002). This study compared the integrated application of SVM and Gabor filters with a different approach involving artificial neural

networks, demonstrating the superiority of the SVM approach. Further, Chowdhury et al. (2006) and Bhavsar et al. (2007) found that SVM is successful for travel time prediction and suitable for the hierarchical intelligence with respect to low memory and processing requirements.

In this research, the author propose to use a multi-class SVC model for traffic condition assessment and a SVR model for travel time prediction. The SVM-based VII highway traffic surveillance system that emphasizes incident detection and travel time prediction will comprise vehicle and infrastructure elements interconnected with a mixture of available short-range and wide-area wireless data-link protocols. This system uses the SVC model to recognize a particular pattern that appears in the microscopic traffic data during an incident and applies SVR algorithm to relate the current traffic condition with the future travel time. The SVM-based VII system is expected to provide better incident detection and travel time prediction performance by using more detailed and continuous data generated by vehicles, than existing systems. This system, using the traffic data in VII-enabled vehicle through vehicle-to-vehicle and vehicle-to-infrastructure communication, can assess and predict traffic conditions where traffic sensors are either absent or sparsely placed. It has unique advantages for use in rural and low flow highway networks with few drivers, weak cell phone signals, and where expensive highway traffic surveillance infrastructure construction is not cost effective.

1.1.4 Platform for VII Modeling

Although the Federal Highway Administration has developed and published an architecture and functional requirement for VII (FHWA 2005), the details for building a

function VII system are still quite vague. For instance, while this document outlines several potential communication protocols available for VII, guidelines for choosing the correct standard for determining capacities for a particular purpose are undefined or derived through qualitative analysis.

Due to the complexity and cost involved in conducting field experiments for a VII system, using the simulation platform that integrates traffic and communication simulator is a cost effective and efficient alternative to facilitate design and evaluation of any VII-based highway traffic surveillance system. Furthermore, detailed and realistic simulation of both traffic and communication interaction can assist researchers in testing various architecture designs, implementation algorithms and parameter configurations, eliminating the need for collecting field data after the implementation of a particular strategy.

Several studies have envisioned an integrated simulation platform connecting traffic and communication simulators. Earlier work on integrated traffic and communication simulations has been used to create simplified models of communication characteristics (Hsin and Wang 1992; Sukthankar et al. 1998; Ghaman et al. 2003). More recently, simulators integrating microscopic traffic and detailed network protocol modes were developed for vehicle-to-vehicle communication (Fujimura and Hasegawa 2005; Choffnes and Bustarnarnte 2005). The authors of these papers made a convincing case that an integrated traffic and network simulator revealed important findings that were not otherwise observed. However, none of these studies address communication involves fixed field equipments. Following this same vision, a simulation platform is necessary to

modeling a VII system that is capable to assess and predict traffic conditions in a real time fashion.

The author proposes to use an integration of traffic simulator PARAMICS and network simulator ns-2 to evaluate the proposed VII framework. PARAMICS, a time-step, behavior-based microscopic traffic simulation software, will be used to realistically model the traffic flow of the selected test network. The extensive Application Programming Interface (API) functions of PARAMICS and the microscopic modeling capability makes it most suitable for customizing software for research applications (Quadstone Limited 2002). Previous studies have also found PARAMICS to be superior in its detailed traffic modeling, which closely corresponds to real-world scenarios (Boxill 2000 and Chowdhury et al. 2006). The ns-2 simulator, which is used for communication network analysis, is an open-source software that can be coded to customize specific applications. Both the PARAMICS API and ns-2 model are C-based programmable and have open architecture, making it convenient to synchronize and transfer data, i.e. communicate between these two software packages. Specifically, the interaction between infrastructure and high speed vehicles, as well as vehicles generated microscopic data will be simulated in PARAMICS. In parallel, the real-time vehicle-to-vehicle and vehicle-to-infrastructure communication including addressing, routing, and scheduling solutions will also be modeled in the ns-2 synchronously and cooperatively.

1.2 Research Objectives

There remain important technological gaps to be to be filled before a real-time highway traffic surveillance system using VII becomes a reality. By far, there has been limited work that addresses the following problems:

1) There is a lack of an effective and efficient platform that comprehensively addresses traffic and network characteristics for the development and evaluation of VII network;
2) While centralized architecture suffers from scalability and single point failure issue, distributed architecture encounters difficulty on system wide control and optimization, quantitative study of the networking and processing architecture is needed for VII highway traffic surveillance system;
3) The intelligent algorithms that take advantage of state-of-the-art AI paradigms, for online traffic condition assessment and prediction using VII do not exist;
4) It warrants the development of a simulation model that implements fully functionality of real-time highway traffic surveillance before the real world experiment could be conducted.

To bridge these technical gaps, the *primary objective* of the proposed research project is to develop a hybrid VII framework that is capable to assess and predict traffic conditions accurately and reliably in a real-time fashion using computational intelligence. The author expects to achieve the primary objective by pursuing the following tasks:

- Develop an integrated simulation platform
- Select networking and processing architecture for highway traffic surveillance
- Develop a VII simulation model that integrates intelligent algorithms for online traffic condition assessment
- Develop a VII simulation model that integrates intelligent algorithms for real-time travel time prediction

The immediate impact of the research could be a *reliable alternative to traditional traffic sensors network* to assess the condition of the transportation system. This system can assess conditions where traffic sensors are not present using the RSUs collecting microscopic traffic statistics from VII-enabled vehicles through vehicle-to-vehicle and vehicle-to-infrastructure communication. This research will produce a platform, for evaluating the VII system, by integrating a traffic and communication network simulator for use in evaluating VII concepts and associated communication methods.

1.3 Research Hypothesis

The research hypothesis is built on three premises. The first premise is that the data provided through VII-enabled vehicles will improve on-line highway traffic surveillance performance. In this proposed VII system, vehicle on-board units are expected to work autonomously to collect microscopic traffic statistics such as vehicle trajectory, speed, spacing gap and lane change behavior at programmed intervals. Additional statistics can be converted and derived from these individual statistics. For instance, the speed and acceleration profiles can be derived from vehicle trajectory, and

the time series statistics can be converted to statistics over space coordinates using the vehicle trajectory with time stamp. This microscopic and continuous data generated by vehicles in the VII network is expected to provide faster traffic condition assessment and lower false detection rates in comparison to existing highway traffic surveillance systems that rely on traditional traffic.

The second premise is that the data provided by VII-enabled vehicles and processed in a hybrid framework, is expected to be an improvement over centralized only operations. In existing on-line centralized traffic management systems, communication links continuously send data from traffic sensors to a staffed centralized traffic management center (TMC) for assessment. As this data frequently requires no traffic management action, unnecessary communication costs are incurred. In addition, these systems are vulnerable to single point of failure and suffer from scalability issues.

The third premise is that a computational intelligence in each component can unify the benefits of centralized and distributed management and contribute to the understanding of the complex highway network. The learning ability of such computational intelligence as SVM will improve the system performance over time to accommodate the continuously changing traffic conditions and system parameters.

These three premises, combined with the proposed hierarchical grouping of infrastructure devices with their computational intelligence and learning capabilities, will significantly improve the operational efficiency of existing on-line traffic condition assessment and prediction system. The integrated simulation methodology and open source software will provide a valuable tool for design and evaluation of any real-time

traffic surveillance and management systems. Also, the developed VII simulation model will be made available for use by future researchers and designers of other similar VII systems. Future implementation of the research in the private and public sector will result in new VII related equipment in vehicles, greater control of traffic loading, faster incident detection, improved safety, mitigated congestion, and reduced emissions and fuel consumption.

1.4 Dissertation Structure

The dissertation contains categorized chapters for easy understanding and organized reference to the conducted research. Chapter 1 presents the background, motivation, significance and objectives of this research. Previous studies that are relevant to the various modules of the research conducted for this dissertation are presented in Chapter 2. Chapter 3 includes the methodology employed in conducting the research and Chapter 4 presents the study results and analysis. Chapter 5 concludes the research with contributions and recommendations for its use and future work. Appendices include supporting data, programming code and information used during the analysis, which could provide valuable recourses for practitioners and researchers involved in VII-related work.

CHAPTER 2

PREVIOUS STUDY

The wide scale deployment of extensive highway traffic surveillance systems is expected to expand at an ever-increasing pace, which poses a great demand and challenge for the new technologies and operational concepts used in these traffic monitoring systems. Recent advances in embedded systems and wireless sensor network technologies have made the integration of infrastructure elements with data processing units and wireless communication interface possible. This in turn has resulted in the formation of ad hoc data communication networks that require no additional communication infrastructure. The communications between vehicle and infrastructure over this ad hoc network as envisioned in Vehicle-Infrastructure Integration (VII) will provide enormous opportunities for improving existing traffic monitoring functions. These future systems, with their built-in intelligence, and decision making abilities that have a scope of unprecedented flexibility, are expected to have the capability to monitor the entire segments of highways effectively and efficiently.

This chapter seeks to summarize the evolution of state of the knowledge in real time traffic condition assessment and prediction in the following areas:

- Simulation platform for online traffic operations
- Networking and processing architecture
- Highway traffic surveillance technology
- Computational intelligence for highway traffic surveillance

2.1 Simulation Platform for Online Traffic Operations

Much effort has been devoted to developing realistic simulation models of online traffic operations in both traffic and communication domain. These studies are categorized into three genres according to their ability to adopt explicit or implicit models for simulating vehicular traffic and communication networking.

Implicit-traffic-explicit-communication refers to simulation platforms that adopt detailed communication protocol models but simplified vehicular traffic models such as the random way point model (Zang et al. 2005; Hasegawa 2005; Xu and Barth 2004; Fujimura and Sato et al. 1999). Marc et al (2006) investigated and compared various routing/forwarding strategies under the realistic channel model for vehicular ad hoc networks. In (Marc 2007), a promising position-based message forwarding strategy was proposed to disseminate time-critical safety information. While the randomized node movement and message generation models is the common practice of the mobile ad hoc network research community in validating networking protocols for generic applications, they are inadequate for real-time validation of specific vehicular traffic operations.

Explicit-traffic-implicit-communication simulators model the effects rather than process of communication through adopting simplified models of communication characteristics (Sukthankar et al. 1998; Ghaman et al. 2003). In (Ewing et al. 1996), intelligent vehicles are assumed to have cellular communication links to the TMC for downloading traffic statistics and finding optimal paths. In (Mirchandani and Wang 2005; Lee et al. 2004) researchers assumed that distributed traffic sensors have direct communication links (wired or wireless) to relay their measured data to central servers

for constructing hierarchical traffic models. In (Jayakrishnan and McNally 2006), short range communication was assumed for vehicles to download their trip history to any roadside sensor they encounter for analyzing traffic patterns based on distributed observations. In (MIT 2002), implicit communication was assumed to support various traffic management operations. In order to overcome scalability limitations of discrete event-based network simulators, Killat et al. (2007) proposed an approach using statistical models to simulate data packets communication, thereby to reduce the numbers of scheduled transmission. This approach had apparent advantages for fast validation of different operational concepts without concerning the details of communication efficiency and reliability, which allowed inevitable omission of fine-grain random effects in network communication process.

Choffnes and Bustarnarnte (2005) analyzed the performance of vehicular ad hoc networks (VANETs) on their traffic model platform STRAW. They demonstrated that accurately simulating the interaction between an ad hoc network and the traffic environment, instead of using a simple random way point model, is critical to the success of developing and evaluating VANETs. They were the most convincing in advocating the development of an integrated vehicular network and traffic simulators.

Explicit-traffic-explicit-communication simulators adopt detailed models for simulating both the traffic and communication aspects of a network. This genre of simulators achieves higher accuracies in both traffic and communication simulation at the cost of higher complexities. Such simulators either integrate mature simulators from each domain (Hsin and Wang 1992; Fitzgibbons et al. 2004; Yin et al. 2004; Schroth et

al. 2006; Eichler et al. 2005; Biswas et al. 2006) or completely compose both functions to meet study-specific requirements (Avila et al. 2005; Mangharam et al. 2005; Killat 2007).

In the context of Intelligent Vehicle Highway Systems (IVHS) which precedes ITS, Hsin and Wang (1992) describe the architectural issues and exemplary implementations following each approach: integrating commercial traffic simulator MODSIM and communication simulator COMNET, or modeling macroscopic traffic and communication characteristics using state machines. More recently, Fitzgibbons et al. (2004) integrated the CORSIM traffic simulator and the QUALNET communication network simulator to model vehicle ad hoc networks. For inter-vehicle communication in vehicle ad hoc networks, Yin et al. (2004) implemented software objects for modeling traffic flows, driver behavior, and network protocols, while Avila et al. (2005) adopted traffic models based on realistic street maps and simplified communication models based on state machines.

Eichler et al. (2005) developed and implemented a concept for coupling the traffic simulator CARISM and the network simulator ns-2. This concept provides a fast and reliable connection which exchanges synchronization data between the two simulators at certain points in virtual time to ensure cross-platform interoperability and easy extensibility. Schroth et al (2006) used CARISMA and ns-2 to form a combined traffic and network simulation environment, evaluated driver reactions. This research indicated that in order to precisely identify the effects of the inter-vehicle communication to the traffic flow, the road traffic, the wireless ad hoc communication and the application logic must be considered separately. Eichler et al. (2006) also did research in simulation

strategies for context-adaptive message dissemination in vehicular ad hoc networks. This study proposed an altruistic communications which consider each nodes interests in information. Traffic information must only be transported according to the needs and potential benefits. Biswas et al. (2006) demonstrated a simulation of Dedicated Short Range Communication (DSRC) based vehicle to vehicle wireless communication for the highway safety improvement, especially focusing on collaborative collision avoidance. Scgmidt-Eisenlohr et al. (2007) implemented a simulation framework analysis the inter-vehicle communications protocols in different transmission power and packet generation rate.

Although fixed infrastructure devices, such as the inductive loop detectors, are the most common components for communication networking, none of these previously discussed studies address communication among field sensors, and no explicit-traffic-explicit-communication simulator that integrated state-of-the-art traffic and communication simulation software has been reported.

Prevalent modern simulators used for communication studies and intelligent vehicle/sensors in their respective domains include Network Simulator version 2 (ns-2) (ISI 2001), Glomosim (Zeng et al. 1998), Jsim (Sobeih et al. 2005), Qualnet (Scalable Network Technologies 2006), and OPNET (OPNET Technologies 2006). Ns-2 provides the most comprehensive open source support of communication protocols. CORSIM (McTrans 2006) and Synchro (Trafficware 2006) are standard choices for vehicular traffic simulation, but are limited in their ability to interface with other programs. PARAMICS is a microscopic traffic simulation program that features a flexible

Application Programming Interface (API) for customized interface with other programs. In this dissertation, ns-2 and PARAMICS are adopted to build an *explicit-traffic-explicit-communication* integrated simulator to realistically model traffic data collection, exchange, and distributed processing on sensors and controllers, and to assess the instantaneous effects of sensor incident detection and control on the highway traffic flows. The platform is intended for use by interdisciplinary researchers. Traffic engineers can flexibly implement and insert advanced incident detection algorithms, distributed decision making, and real-time traffic management methods in PARAMICS, while wireless network researchers can evaluate different communication protocols and network parameters in ns-2.

2.2 Networking and Processing Architecture

Transportation authorities have long used a wide variety of sensing technologies to monitor vehicular traffic and driving conditions constantly on major highways throughout the United States. There are existing systems that use sensors to measure traffic statistics such as speed, volume and vehicle classification. State-of-the-art systems such as the Maryland's Coordinated Highways Action Response Team (CHART) program, Virginia's Smart Road System, California's Freeway Performance Measurement System (PeMS) (UC Berkeley 2006), Traffic.com's TrafficPulseSM system, Michigan's Remote Traffic Microwave Sensors, and ENSCO's Remote Monitors, all use sensor collected information. Sensor data is relayed to a control center via individual wired or wireless communication links. These control centers, staffed with designated personnel in turn collect and process the received data, thusly enabling traffic

managers to detect incidents, dispatch incident response teams, distribute precautionary alerts, and assist in real-time and long-term traffic management.

2.2.1 Centralized System

State-of-the-art highway traffic surveillance systems around the world have been built with an emphasis in observing and controlling from a central location (USDOT 2006, New South Wales Road and Traffic Authority 2006, Wang et al. 2005, City of Cape Town 2005, Tokuyama 1996). Transportation agencies deploy as many sensors as affordable along the highway infrastructure and establish Traffic Management Centers (TMCs) at central locations that collect data from sensors for making centralized control decisions. Substantial investments have been made to connect all sensors to central or regional controllers with dedicated communication links. Roadside sensors transmit data to TMCs following predetermined schedules, while human operators identify possible incidents from the continuous data streams and initialize reaction decisions.

One of today's largest examples of a centralized sensor system in the U.S. is the Freeway Performance Measurement System (PeMS) in California (UC Berkeley 2006), linking more than 25,000 (as of 2004) loop detectors to a TMC. The system collects data from each sensor every 30 seconds, accumulating more than 2 GB of data per day in a central database. While the majority of sensors are connected with fiber optic cables, 250 solar powered radar sensors have recently been connected via General Packet Radio Service (GPRS) cellular wireless links (SpeedInfo Inc. 2005), similar to those using Cellular Digital Packet Data (CDPD) cellular wireless links (FHWA 2004).

The majority of today's highway traffic surveillance systems rely on a connected web of roadside sensors. For freeway management, human operators detect abnormal conditions through the surveillance or screening of sensor data. Incidents are then resolved by dispatching human response teams, rerouting approaching traffic, and re-timing traffic signals (Chowdhury and Sadek 2003). This centralized control methodology, which imposes an enormous responsibility on operators, is critically dependent upon an extensive and costly communications backbone.

2.2.2 Distributed System

Though these centralized monitoring and control practices are now prevalently being used for freeway control, decentralized and hierarchical control methods have long been used for traffic signal control (Papageorgiou, et al., 2003). Traffic engineers carefully group traffic signals on closely related road segments, so that signal timings are consistently controlled to minimize delay and optimize road capacity. By extracting hierarchical traffic characteristics from sensor data at a given intersection, numerous real-time signal timing adaptation methods have been created, e.g. SCOOT (Siemens 2006) SCATS (Tyco Integrated Systems 2006) and RHODES (Mirchandani and Head 1998).

In early 1980s, the UK Transportation Research Laboratory (TRL) developed and implemented a global, real-time, rule-based expert system named as SCOOT (Siemens 2006). A huge rule base is maintained for the traffic signal network, with which SCOOT performs global optimization on signal timing to minimize delays. Unfortunately, SCOOT's global optimization approach is known to be slow and unable to deal with local changes in real time.

The Sydney Coordinated Adaptive Traffic System (SCATS), developed by Australian transportation authority in the late 1970s, is a distributed hierarchical system that optimizes traffic signal timing using volume data detected by sensors at signal stop-lines (Tyco Integrated Systems 2006). The system aims at optimizing the saturation flow based on optimizing individual regions in the network. However, trained specialists are often needed to properly define the region boundary and offset parameters for each region.

In early 1990s, the University of Arizona developed a real time adaptive control system called RHODES (Mirchandani and Head 1998). The system constructs stochastic traffic flow models to predict the expected condition over the next few minutes. While it's hierarchical architecture is conceptually applicable to network wide operation, its algorithms demand exponential complexity and network wide real-time communication, rendering its practical use to a very limited network scope.

With advances in ad hoc wireless sensor network technology, it is envisioned that the future traffic control system would consist of intelligent sensors and controllers capable of automated incident detection and traffic control (Estrin et al. 2001). However, there are as yet no been practical solutions for turning this vision into a reality. Only recently, researchers considered unleashing one level of freedom for the traffic sensors in California highways by removing the fiber optic cables of some sensors. Instead, the proposed Power Efficient and Delay Aware Medium Access (PEDAMACS) protocol allows the TMC to maintain links to only a number of gateway devices, each of which will collect data from their nearby sensors using multi-hop wireless network forwarding

(Coleri and Varaiya 2004). PEDAMACS does not alter the centralized control method, since all data are still delivered to TMCs for monitoring and control. In a more confined scope, sensors with wireless interface have been placed on specialized highways and vehicles for automated vehicle steering, which have been highlighted in the 1997 and 2003 automated highway system demonstration in San Diego, CA.

In (Coifman and Ramachandran 2004) the authors outlined the vision of deploying intelligent sensors along highways for distributed sensing, local data processing, and reporting only concise information to TMC or other responsible controllers if an anomaly is detected. The strength of the envisioned approach lies in the ability of sensors and controllers to make collaborative decisions without human intervention. As described in (Coifman and Ramachandran 2004), sensors, organized in a hierarchy to facilitate incident detection and verification, collaborate in clusters to verify incidents and initiate responses. The unique strength of the envisioned approaches lies in their ability of collaborative decision making. This study, which also estimated different levels of communication requirements and data precision, concluded that wireless sensors are more cost-effective than traditional wired sensors in rural areas and most locations adjacent to a central business district.

2.2.3 Hybrid System

Though rare in highway traffic surveillance, hybrid systems integrating centralized and decentralized control strategies do exist. Somers (1996) proposed using intelligent agents to develop a hybrid management network integrating centralized and distributed control. With respect to air traffic control systems, Feron (2003) applied a

hybrid of distributed and centralized decision making algorithm to solve the conflicting air traffic scheduling problem at the National Airspace system (NAS). Wall et al. (2007) implemented a combined centralized control for vehicle signal and distributed control for a pedestrian countdown signal system, which improved operations. Chiu and Mahmassani (2003) developed a hybrid dynamic traffic assignment (DTA) model that integrated centralized and decentralized DTA frameworks. Kurfees et al. (1995) presented a hybrid solution for modernizing the urban signal control system, consisting of a centralized control system for central business district (CBD) and distributed control system of outlying areas.

2.2.4 Evaluation of Communication Alternatives

There are a variety of efforts intended to help transportation agencies obtain better understanding of different communication alternatives including medium and architecture for ITS applications. Among them, the *Communication Handbook for Traffic Control System*, developed under Federal Highway Administration (FHWA) sponsorship, was a survey of various available communication medium and architecture for traffic control applications (Gordon et al. 1993). Another study sponsored by FHWA evaluated the performance of various Digital Subscriber Line technologies (xDSL) with both laboratory experiments and field tests (Jones 2002). The study implemented high speed data services (e.g., 2 Mbps) with xDSL on the existing twisted pair wire for transferring traffic video images, and their field studies showed that the xDSL technologies were able to maximize the DSL throughput and subsequently optimize the video motion/quality relation.

The Texas Department of Transportation sponsored the development of a reference guidebook and training workshop to establish a fundamental level of understanding of wired communication concepts and technologies among state transportation engineers and an evaluation framework for wired communication alternatives (Brydia et al. 2005). This guidebook recommended the different criteria for choosing wired technologies (e.g., serial, ISDN, DSL, T1/T3 Twisted Pair, and Fiber), based on the number of devices, bandwidth, latency, distance, and cost.

The California Department of Transportation (CalTrans) and FHWA conducted a field operational test (FOT) between June 1994 and September 1998 to evaluate the benefits of a *mobile surveillance* system with wireless communication interface (Kimberley et al. 1999). In 1998, the Philadelphia Satellite Communication Demonstration project evaluated the effectiveness of using very small aperture terminal (VSAT) Ku-band satellite communications for traffic and incident management on I-95 corridor (Habesch et al. 1998). Compared to terrestrial-based copper and fiber optic-based closed circuit television (CCTV) systems, the VSAT-based CCTV system was found to be superior in terms of quality of service assurance, delay and jitter control.

In 2002, the Kentucky Transportation Center at the University of Kentucky implemented and evaluated a base station based wireless communication technology as part of the TRIMARC traffic management system (Hunsucker, 2002). This study investigated the use of a 220MHz wireless communication system to transmit traffic measurements from field sensors to traffic management center to support real-time traffic management in Louisville, Kentucky, finding that this 220MHz communication system

was equal to or better than the leased phone line in terms of functional reliability and cost effectiveness. The Wisconsin Department of Transportation (WisDOT) has implemented a statewide digital microwave backbone infrastructure that is used to transport voice and traffic data for 161 public safety agencies throughout the state (Verhyen 2005).

Among the existing evaluation efforts, some measures of effectiveness (MOEs) were recognized as the most important indicators of the performance of the communication system. Gordon et al. (1993) summarized possible attributes such as bandwidth, signal attenuation, latency, power consumption, signal to noise ratio, bit error rate, error control technique as the fundamental MOEs for evaluating performance of the communication network. The authors also suggested that reliability, maintainability, and expandability were also important for overall effectiveness of communication system. In addition, quality of service assurance, delay and jitter control of video motion image were also widely used MOEs to assess the performance of communication network (Habesch et al. 1998). Kimberley et al. (1999) found that the portability and reliability of a communication system was a key factor to realizing the expected functionality of the mobile surveillance system. Hunsucker (2002) evaluated the owned and leased wireless network in terms of functional reliability and cost effectiveness. Jones (2002) considered throughput and video image/motion quality as the MOEs for evaluating communication systems supporting traffic surveillance systems using CCTV. Texas DOT identified the number of devices, communication link bandwidth and latency as the important criteria for evaluating communication alternatives (Brydia et al. 2005).

2.3 Highway Traffic Surveillance Technologies

One of the key tasks of highway traffic surveillance is to quickly and reliably identify incidents and obtain as much detailed information about the incident as possible. Effective incident detection and verification is important for the timely and appropriate initialization of real time traffic management. In 2002, the Fatality Analysis Reporting System identified 42,815 highway fatalities in U.S. (FHWA 2004). If these incidents had been detected and verified more quickly, medical assistance would have been able to provide faster treatment, thus possibly saving many lives that were regrettably lost. The time required to detect and identify incidents impact the consequent phases of incident management, i.e. incident response, incident clearance, and real time traffic management during incidents is a key factor in determining incident duration. A 1998 study found that if a crash is detected faster, for example after two minutes, instead of four, the incident response personnel travel will through a shorter queue to reach the incident location. The recovery time for traffic to return to normal also decreases due to the shorter queue formed (Skabardonis et al. 1998). Improvement on incident detection and verification performance will drastically reduce motorist delays, business losses, fuel consumption from sitting in accident related tie-ups, vehicle emissions, and possible secondary crashes.

Existing incident detection systems can be categorized into two types: sensor based and human based technologies. Sensor based technologies can be further grouped by Automatic Incident Detection (AID) algorithms that are based upon traffic sensor measurements, video camera image processing, and mobile probe sensors technology.

Human based monitoring systems are non-automatic systems that include operators monitoring closed-circuit television (CCTV) or processing anecdotal information reported from drivers or other resources. Often a combination of these systems can be used to achieve the best traffic management performances. However, the VII system with its capability to continuously monitor the entire roadway network is expected to be a promising tool for fast and accurate incident detection and mitigation.

2.3.1 Sensor-Based Highway Traffic Surveillance Technology

Automatic Incident Detection (AID) algorithms use point-based or/and link-based data to detect and verify incident without human intervention. Link-based data is usually collected by individual vehicles as probes to assess the roadway link statistics (e.g. travel time and average speed). Link based AID algorithms include the MIT algorithm (Parkany and Bernstein, 1993; 1995), the ADVANCE algorithm (Sethi et al. 1995), the TTI algorithm (Balke et al. 1996), and the TRANSMIT algorithm (Mouskos et al. 1999; Niver et al. 2000). Point-based data collection technologies, which use traffic flow measurements such as presence, speed, flow, etc. made at a point, are common types of applications that use existing sensor technologies. Currently, Inductive Loop Detector (ILD) technology is the most widely used traffic sensor technology today. Unfortunately, ILD systems suffer from several drawbacks. Their overall lifetime cost (Klein 2001) including installation, maintenance and repair is quite high, they are short-lived due to their vulnerability to pavement maintenance activity and heavy traffic conditions, and the installation and maintenance of ILD systems disrupts regular traffic flow.

In contrast to ILD systems, several novel non-intrusive sensing technologies have emerged over the last few years (Klein 2001). Some, like active infrared and acoustic array sensors are quite good but very costly. Others, such as passive infrared and ultrasonic sensors, are somewhat cheaper but quite weather sensitive. One of the most promising technologies may be microwave radar which is environmentally insensitive and multi-lane capable with moderate a cost between $700 and $3300. However, microwave radar cannot detect stopped vehicles which are the most prevalent during times of traffic incidents and subsequent heavy congestion. Moreover, the implementation of most traffic sensor technologies may not be cost effective in rural areas in which there is less traffic and communication infrastructure support.

The densely deployed sensor network makes it possible to develop AID algorithms which use one or more sensor measurements to detect and verify incidents without human assistance. For the latter, neighboring sensors must collaborate to efficiently detect and verify a traffic incident. AID algorithms can be classified as comparative (e.g. California algorithm (Payne and Tignor 1978) and APID), traffic flow theory (e.g. McMaster algorithm (Persaud and Hall 1989)), statistical forecasting (e.g. ARIMA (JHK and Associates 1993), DELOS (Stephanedes and Chassiakos 1993), Bayesian (Levin and Krause 1979), SSID (Antoniades and Stephanedes 1996)), and computational intelligence.

The comparative algorithm, among the earliest of the developed AID algorithms, compares traffic sensor measurements such as flow, occupancy, and speed with pre-set threshold values. Once the measurement exceeds the threshold, an incident alarm is

reported. Comparative algorithms include the entire California algorithm family (Payne 1976; Payne et al. 1976; Payne and Knobel 1976; Tignor and Payne 1977; Payne and Tignor 1987; Levin and Krause 1979a, b), pattern recognition (PATREG) algorithm (Collins et al. 1979), and the All-Purpose Incident Detection (APID) algorithm (Masters et al. 1991).

The statistical forecasting algorithm conducts statistical analysis to determine if the predicted traffic statistics are significantly different from those measured. When the algorithm detects a deviation from the forecasted values, which is computed based on historical data, an incident alarm occurs. The Standard Normal Deviate (SND) algorithm (Dudek et al. 1974), the Bayesian algorithm (Levin and Krause 1978; Tsai and Case, 1979), the Autoregressive Integrated Moving-Average (ARIMA) algorithm (Ahmed and Cook 1977; 1980; 1982), and the High Occupancy (HIOCC) algorithm (Collins et al. 1979) are all examples of statistical forecasting algorithms. Researchers applied various filtering and smoothing techniques to remove the noise in the traffic statistic signal to improve the accuracy of forecasting. The products of these efforts include the Double Exponential Smoothing (DES) algorithm (Cook and Cleveland 1974), the Low-Pass Filter (LPF) algorithm (Stephanedes et al. 1992; Stephanedes and Chassiakos 1993a, b; Chassiakos and Stephanedes 1993), and the Discrete Wavelet Transform and Linear Discriminate Analysis (DWT-LDA) algorithm (Samant and Adeli 2000; Adeli and Samant 2000).

Traffic flow theory algorithms compare the traffic parameters estimated by traffic flow theory with the measured parameters to detect incidents. The representatives of

traffic flow theory consist of the Dynamic Model (Willsky et al. 1980), the Catastrophe Theory Model, also known as the McMaster algorithm (Gall and Fall 1989; Persaud and Hall 1989; Persaud et al. 1990; Forbes and Hall 1990; Forbes 1992; Hall et al. 1993), and the Low-Volume (LV) Incident Detection algorithm (Fambro and Ritch 1979; 1980).

The Computation Intelligence Algorithm refers to applying those advanced computation paradigms for pattern recognition or parameter estimation in incident detection procedures. Different techniques were developed to classify incident and/or to determine incident characteristics, including artificial neural network (ANN) (Ritchie and Cheu 1993; Cheu and Ritchie 1995; Stephanedes and Liu 1995; Dia and Rose 1997; Abdulhai and Ritchie 1999; Adeli and Samant 2000), fuzzy logic (FL) (Chang and Wang 1994; Lin and Chang 1998), combination of ANN and FL (Hsiao et al. 1994; Ishak and Al-Deek 1998), and wavelet analysis (Samant and Adeli 2000; Adeli and Samant 2000). Among these techniques, ANN is the most popular and the Multi-layer Feed Forward Neural Network (MLF) and Probabilistic Neural network (PNN) are the two commonly used method.

As the traffic camera for highway traffic surveillance is passive sensor technology, the light condition dictates the accuracy. Also, the traffic cameras require extensive high cost infrastructure support such as wide bandwidth communication and intensive computation for image processing. There are two types of image processing techniques. The first derives traffic statistics such as occupancy and volume from video images to feed the AID algorithm, and the other directly detects slow moving or stuck vehicles in the camera range to detect incidents. Traficon is an example of the first and

(Versavel 2000) the Autoscope Incident Detection Algorithm (AIDA) is an example of the second (Michalopoulos 1991; Michalopoulos et al. 1993; Blosseville et al. 1993).

Compound algorithms integrating two or more data resources or techniques have also been proposed for use in improving the incident detection perfomance. Westerman et al. (1996) integrated data from loop detector and probe vehicles, whereas Ivan and colleagues (Ivan et al. 1995; Ivan and Chen 1997; Ivan 1997; Ivan and Sethi 1998) used MLF ANN to fuse fixed sensor and probe vehicle data during incident detection. Thomas (1998) also proposed to apply Bayesian discrimination and multiple attribute decision making techniques to integrate information from multiple sensors and probe vehicles. Bhandar et al. (1995) took this process a step further by attempting to fuse information from loop detectors and probe vehicles as well as driver reports.

In addition to the incident detection, another important task of highway traffic surveillance system is to predict travel time for traveler information dissemination and traffic management. Numerous technologies and algorithms have been developed using sensor measurements, such as loop detector and probe vehicles. Park and Rilett (1998) compared the performance of Kalman filter and feed-forward neural network (FNN) models for travel time prediction. Zhang and Rice (2003) presented an easy to implement short-term freeway travel time prediction algorithm based on linear model. Though FNN is popular in travel time prediction (Huisken and Van Berkum 2003; Park and Rilett 1998; Innamaa 2001; Park and Rilett 1998), Van Lint (2006) proposed state-space neural network (SSNN) model to explicitly consider the prediction of travel time in each section to derive the future travel time of the entire segment. This dissertation will develop travel

time prediction algorithms using detailed microscopic vehicle statistics, instead of sensor measurements as presented by previous researchers.

2.3.2 Human-Based Monitoring Technology

Unlike the sensor based AID technology, human based monitoring technology is a non-automatic process involving human operators to monitor CCTV images or process reports from drivers or other witness of incidents. Researchers advocating driver based incident detection system (e.g. enhanced 911 services) argue these systems provide quick and accurate detection, rich and interactive information, broad spatial and temporal coverage, and less capital, maintenance and operational costs, as opposed to other incident detection technologies (Xie and Parkany 2002). Many simulation studies (Mussa 1997; Mussa and Upchurch 1999; 2000) and field experiments (Skabardonis et al. 1998; Walters et al. 1999) were performed to evaluate the effectiveness and performance of driver based incident detection technologies. In one nationwide survey on incident detection system of Traffic Management Centers (TMC) operators, it was found that CCTV monitoring and driver based incident detection system are the primary detection approaches in most TMCs. Indeed, many implemented AID algorithms in these systems were turned off due to poor performance or difficulty to use.

2.3.3 VII for Highway Traffic Surveillance

In addition to the use of roadway traffic statistics measurements to detect traffic incidents, methods involving the use of vehicle kinetics have also been developed. Petty et al. (1997) and Qi et al. (2002) developed an algorithm to detect freeway incidents

using speed and the acceleration profiles of probe vehicles. Other studies also discovered encouraging data showing an increase in the ability of VII to detect highway incidents using vehicle-generated microscopic data (Sermons and Koppelman 1996; Cheu 2002). Crabtree et al. (2007) and Tanikella et al. (2007) illustrated that the travel time data generated from VII can reliably estimate traffic conditions and identify incidents. In a recent paper, Torrent-Moreno (2007) presented a position-based message forwarding strategy between vehicles for exchanging information on time critical safety risks. VII California (UC Berkeley 2006) presented a field experimental study on the potential for using VII for real time highway traffic surveillance. In that study, individual vehicles were used as probe vehicles that sent their location, speed, direction, and time stamp to a centralized processing center for highway traffic surveillance and traveler information dissemination.

In addition to the state level research and study, the USDOT is currently conducting a research program called the Mobility Applications for Vehicle Infrastructure Integration initiative (National VII Coalition, 2007). In that program, researchers are studying the potential for transmitting data from the roadside to warn drivers to avoid collision at an intersection, or to see if individual vehicles, serving as data collectors, can transmit traffic and road conditions from every major road within the transportation network, and even notify drivers if their car is under recall. Although it is recognized that communications between roadside infrastructure and vehicles can improve safety and mobility, there is a lack of standard to guide the design of the communication network support particular VII application.

Federal Highway Administration has developed and published an architecture and functional requirement for VII (FHWA 2005). That document is in high level, the detailed design guidance remains vague and under various assumptions. For instance, it mentioned there are several potential communication protocols available for in VII. However, the guideline to choose the right standard and determine capacity for particular purpose is undefined or based qualitative analysis.

2.4 Computational Intelligence for Highway Traffic Surveillance System

In 1995, Vladimir Vapnik and his colleagues at AT&T Bell Laboratories developed the Support Vector Machine (SVM) algorithm based on the statistical learning theory ((Vapnik 1995; Sewell 2005). The theory was developed to help characterize properties of learning machines that enable the system to generalize predictive information. SVM includes a set of supervised learning algorithms from the field of machine learning applicable to classification as well as regression problems. They use kernels to map the input data into a high dimensional feature space where linear classification becomes feasible. SVM algorithms are based on the principal of Structural Risk Minimization (SRM) and the statistical learning theory developed by Vapnik and co-workers at AT&T Bell Laboratories (Vapnik 1982).

Although SVMs are popular for their applicability in the problem of pattern classification, Smola and Scholkopf (Smola and Scholkopf 1998) promoted Support Vector for Regression (SVR) as a different formulation of SVM. This SVR model depends only on a subset of the training samples, because the cost function for building the model ignores the training samples inside the epsilon-tube (a certain threshold

distance from the prediction). SVR has been successfully applied in diverse areas, such as haptic data prediction, illumination analysis, and financial forecasting (Clarke et al. 2003; Funt and Xiong 2004; Cao and Tay 2001).

Regression algorithms based on the underlying theory of Support Vector Machines are termed Support Vector Regression (SVR) algorithms. SVR achieves nonlinear regression by similarly mapping the training samples into a high dimensional *kernel induced* feature space, followed by linear regression in that space. Since the kernel mapping is implicit (depending only upon the dot product of the input data vectors), it is possible to map the data to a very high dimension and keep computational costs low. Figure 1 gives an overview of support vector regression. In this study, Radial Basis Function (RBF) kernel was used. As shown in Figure 1**,** the SVR model depends on a subset $\boldsymbol{S}$ of the training samples, Support Vectors coefficients $\boldsymbol{C_s}$ and a constant $\boldsymbol{b}$.

The underlying theory behind Support Vector Machines (SVM) and Support Vector Regression (SVR) is similar. SVM is primarily used for pattern classification, whereas SVR is used for regression or function estimation. Thus far, SVR has had limited applications within the transportation field. Previous examples SVR applications to transportation problems include use for travel time, traffic speed and traffic flow predictions, and incident detection in the context of ITS applications (Wu et al. 2004; Vanajakshi and Rilett 2004; Ding 2002; Cheu 2003). In addition, Sun et al. applied SVM for vehicle detection using extracted features from Gabor filters (Sun 2002). Their comparison of the integrated application of SVM and Gabor filters using an approach involving Neural Networks demonstrated the superiority of the SVM approach.

2.5 Summary of Previous Work

Previous study evidently demonstrated that an explicit-traffic-explicit-communication simulator was critical for simulation study of real-time traffic operations. Existing research on integrated simulation platform either concentrated on simulating vehicular ad hoc network (VANET) or did not integrated state-of-the-art traffic simulator. With the similar vision, this research adopted two state-of-art traffic and communication simulator PARAMICS and ns-2 to study a VII network composed of vehicles and infrastructure devices.

The current prevailing centralized highway traffic surveillance system suffers from scalability, single point failure, and reliability issue due to the requirement expensive infrastructure and single point control by human operator. On the other hand, it is difficult to implement system wide control and optimization in a pure distributed system. A hybrid networking and processing architecture, which could be exemplified by a hybrid framework and integrate distribute processing within each hierarchy and centralized control between neighboring hierarchy, is needed. Therefore, it warrants detailed evaluation of different communication alternatives that would lead to the selection of appropriate communication medium and architecture.

The highway traffic surveillance system has not expand to broader suburban and rural areas due to the requirement of expensive infrastructure facilities and drawbacks of existing technologies, such as driver-based reporting system, traffic camera system, and sensor-based AID system. A VII-based highway traffic surveillance system is expected to provide a feasible, effective and efficient alternative. Though ANN is the most common

AI tool used for highway traffic surveillance, this research proposed SVM for real-time incident detection and travel time prediction for its less computation resource requirement, greater learning ability and prediction potential.

CHAPTER 3

METHODOLOGY

The research method was formulated to attain the objectives of the dissertation. The first objective required developing an integrated simulation platform and the second objectives included evaluating different communication architectures to support on-line traffic management. The third and fourth objectives are related to VII for traffic condition assessment and travel time prediction. The author developed two intelligent algorithms: support vector machine (SVM) for incident detection and support vector regression (SVR) for travel time prediction. The following sections are organized into the following four primary categories according to each of the four objectives: develop integrated simulation platform, evaluate communication alternatives, develop VII simulation model with online traffic condition assessment function, and develop VII simulation model with real-time travel time prediction function. As shown in Figure 3.1, the integrated simulator provided a platform for evaluating communication alternatives and developing the VII simulation model. Then the VII simulation model was again evaluated and revised based on the analysis.

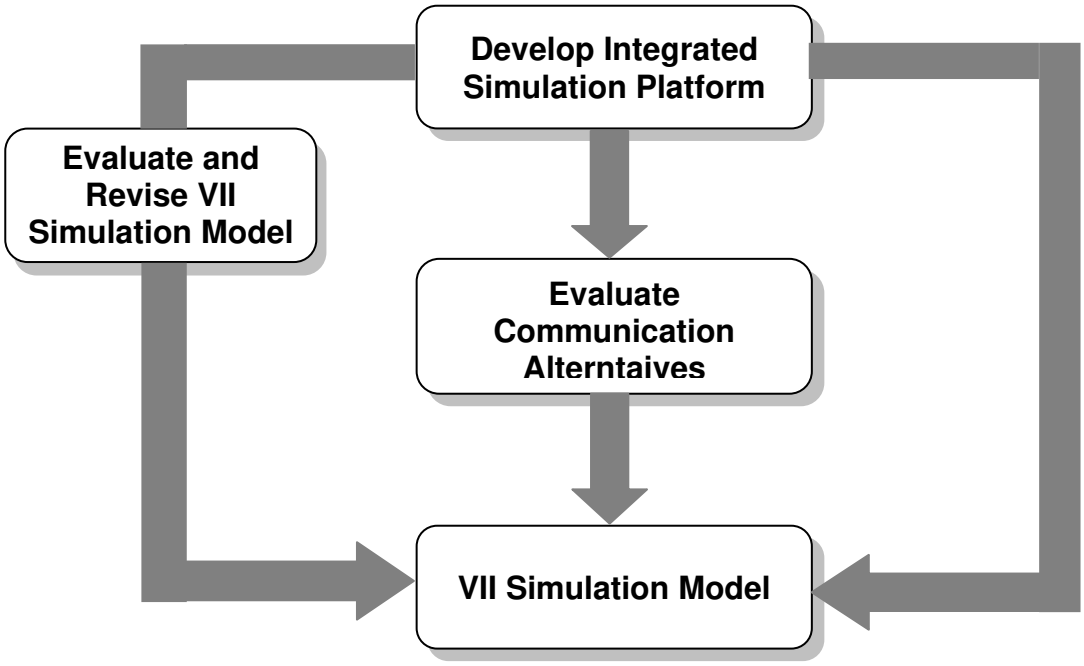


Figure 3.1 Research approach in this dissertation

3.1 Develop Integrated Simulation Platform

As the first step shown in Figure 3.1, the author developed an integrated simulation platform that integrates state-of-the-art microscopic traffic simulator PARAMICS and packet-level networking simulator ns-2 for accurate evaluation of the effectiveness, efficiency, and reliability of traffic management methods and networking protocols.

3.1.1 Traffic Simulation

The microscopic traffic simulation model was used to create a realistic traffic environment to test the effectiveness of various traffic condition assessment tools. Since this study would be extremely expensive and complex to complete through field test, traffic simulation offers an efficient opportunity to collect extensive amounts of

information in a controlled environment that accurately reflects real-world traffic conditions.

PARAMICS traffic simulation software (Quadstone 2006) was used to model the traffic flow of the test network in South Carolina. PARAMICS is a time-step, behavior-based microscopic traffic simulation model, which can incorporate detailed network and traffic control information to provide a realistic representative of traffic operation conditions. In PARAMICS, many different Driver Vehicle Units (DVUs), including VII-enabled vehicles, interact in the simulation model to realistically represent the traffic conditions in the real world. DVUs allow a reasonable distribution of different vehicle and driver types, e.g. sports cars with excellent acceleration characteristics or cautious drivers awaiting a large gap in traffic. An accurate representation of these interactions is especially important during traffic incidents, such as when the vehicles traveling on a lane that is blocked ahead must slow down and seek opportunity to change to a non-blocked travel lane.

Since the VII-enabled vehicles were assigned their own DVU type, the model allowed only the VII-enabled vehicles to communicate their collected microscopic traffic data as well as traffic information and control messages with networking simulation software ns-2. PARAMICS also provided quantifiable measurements of macroscopic traffic statistics such as occupancy of loop detectors, average speed, flow rate and network delay.

The Application Programming Interface (API) is an add-on module which allowed users to modify many features of the underlying PARAMICS models. The API

also allowed the modeling of advanced traffic management strategies such as automatic incident detection and travel time prediction. The case study for evaluation of the integrated simulation platform applied an API to randomly generate incidents and simulate the realistic operation of incident detection and response.

With PARAMICS, network building began with collection of field data including geometric, traffic control, and traffic volume data. The network was then calibrated through comparison between the simulated volume output and the field traffic counts data, as well as via comparison between the simulator animations and site observation. The validation process compared site-collected queue lengths and travel times to those produced by the simulation model. After many iterations and adjustments to the road network and driver behavior parameters, the simulation model accurately reflected the observed travel times within one percent and no significant difference was observed between the field-collected and simulated queue lengths at the bottleneck segment, which were at the signalized off ramp intersections.

3.1.2 Communication Simulation

The communication networking simulation software ns-2 simulates various protocols in each hierarchical layer as the internet architecture at packet-level among nodes for a specified network topology. The simulated layers for this study are summarized in Table 3.1. Network protocols are developed or modified with individual source files in C++ and corresponding changes in OTCL library and header file. For example, user-defined application such as incident signaling in sensor was inserted at the application layer with a function of C++ source codes. Another example was that the

developed hierarchical message routing scheme at each fixed nodes such as repeaters, sensors and controllers was implemented as a new routing agent class with several member functions in the network layer. To start the communication networking simulation, network topology, nodes parameter configuration, simulation initialization and tracking were specified in OTCL language.

Table 3.1 Simulated Protocol Hierarchy Stack

Layer	Protocol	Implement ation	Remark
Application	VII	Customized	Implement various VII application
Transport	UDP	Embedded / Customized	Modified UDP protocol to support VII application
Network	IP & VII Routing	Embedded / Customized	Added VII routing protocols to support hierarchal routing
MAC + Physical	IEEE 802.11	Embedded	Configured for different bandwidth and range for wireless communication

Through ns-2 modeling, real-time effects in the communication networking domain can be modeled accurately with explicit constraints and variations such as:

- Finite and variable communication bandwidth and latency;
- Random errors and transmission conflicts;
- Synchronization effects in communication and control;
- Out-of-order messages and event effects.

On the other hand, in collaboration with PARMICS, realistic communication patterns and requirements can be modeled including:

- Communications induced by topology-dependent mobile nodes movement pattern;
- Communications scope due to RSU and controller placement;

- Communications load with respect to traffic volume, traffic surveillance strategy, and incident probability;
- Communications induced by message exchange for realistic application of traffic monitoring application.

Modeling of such effects enhances accuracy in performance evaluation and assessment of deployment strategies beyond that is achievable with simplified communication assumption in traffic simulation tools.

3.1.3 Integrated Simulation

The integrated simulation platform is based on the PARAMICS traffic modeler and the ns-2 network simulator. PARAMICS simulates a transportation system using a number of network files that define all aspects of a transportation system, including its infrastructure geometrics, traffic control methods, ITS components, driver characteristics, and traffic demands. Those text file provides easy method to assess and modify every aspect of the traffic network. In addition, user-defined functions are programmable via a plug-in C++ source file and a group of API functions, which connect PARAMICS's internal modeling core with external customization and software (Quadstone 2006). This API interface allows the possibility of implementing traffic statistics logging, synchronized modeling time step, generating vehicle movement pattern file, and exchange data and control commands with ns-2. Ns-2's open-source architecture gives great freedom for incorporating newly developed protocol components and interfacing with other software (Sobeih et al. 2005).

A synchronization file was used to act as switcher to control the sequential running of PARAMICS and ns-2. Since PARAMICS needs a warm up time to load the traffic into network and ns-2 needs an initialization period to discover the hierarchical architecture for hierarchical routing, the synchronization file defines the synchronization start up time for the two simulators to perform synchronized, locked-step executions for simultaneously modeling traffic dynamics and communication networking. This setup of running two simulator separately first and synchronized later is beneficial to the simulation efficiency because synchronization is much more resource intensive than individual runs. At the end of each synchronized period, PARAMICS updates the mobile nodes movement and messages sending command in TCL language. At the beginning of each synchronization step, ns-2 load and push those events transferred from PARAMICS into its scheduler for execution. Ns-2 is also able to log the real time data and commands into a set of log file, one for each fixed node, such as sensor and controller, to feed the data for traffic management simulation.

In the current practice, sensors are connected by repeaters that are within communication range determined by the physical layer protocol configuration. The typical communication range for IEEE 80.11 a/b/g is 200-300 meters. The placement of these repeaters also assures that wherever a vehicle sends a message, there will be at least one repeater that can receive it. To initiate a simulation, in PARAMICS, users build, calibrate, and validate a traffic network, while in ns-2, users define the wireless networking protocol stack, the network topology, and the execution time and interval. Figure 3.2 shows the simulator execution flow chart.

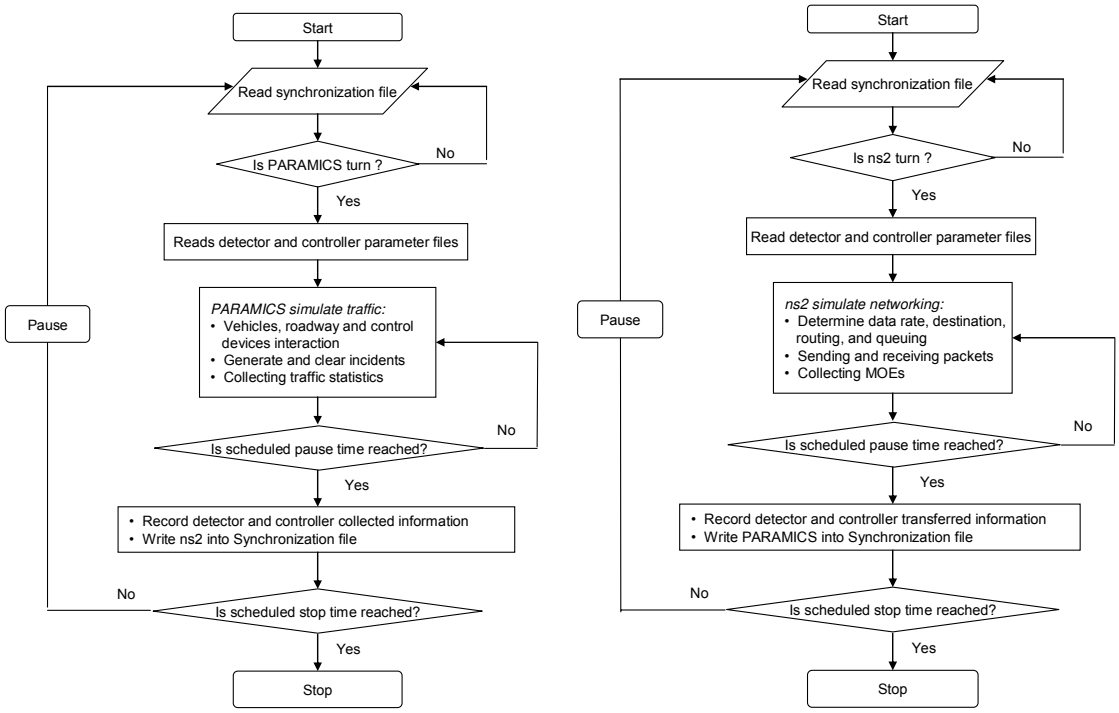


Figure 3.2 Integrated simulator process execution flow chart

3.1.4 A Case study to Evaluate the efficacy of Integrated Simulator

As an example of the usage of the hierarchical networking development and the integrated simulation platform, a hypothetical distributed incident detection and resolution strategies is designed and simulated as shown in Figure 3.3. In this example, sensors placed at regular distances along a highway measure vehicle speed and traffic volume. The distributed detection algorithm consists of three phases: detection, verification, and notification. For detection, each sensor independently carries out a *shockwave detection algorithm* (Chowdhury and Sadek 2003). When a "possible" incident is *detected locally*, the sensor invokes verification by sending a query to its

adjacent sensor on each side. If any queried sensor has already observed a corresponding shockwave, or will see one within a specified time frame, the incident is *verified*. The verifying sensor will proceed to *notify* its local cluster (parent) controller. Upon receiving the detection notification, the cluster controller determines its response with an *incident resolution strategy*. In this example, the cluster controller immediately notifies its upstream controller to perform traffic diversion.

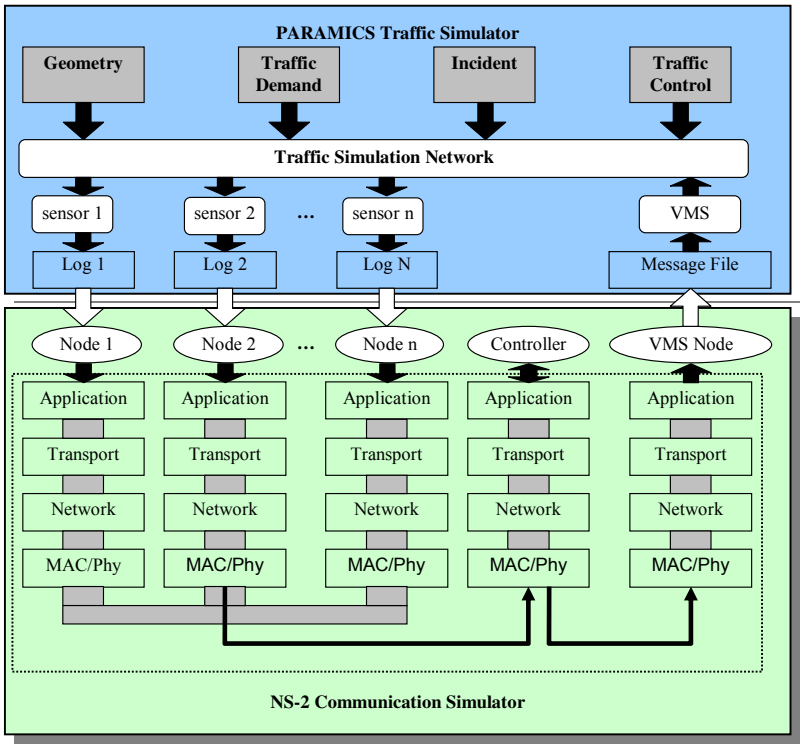


Figure 3.3 Architecture of a hypothetic incident detection and management system

The shockwave detection algorithm is based on the fact that an incident causes changes in the upstream and downstream traffic flows' volumes, densities, and speeds. The effects propagate upstream and downstream from the incident location, such that the affected zone (that observes a change in the flow) expands like a propagating wave,

namely, the shockwave (Chowdhury and Sadek 2003, May 1990). A backward moving shockwave progresses upstream of the incident location against the flow of traffic, as queues started to develop due to decreasing speeds and flow. A forward moving shockwave progresses downstream as the decreasing number of vehicles traveling past the incident location reduces demand to the downstream freeway. The sensors detect these shockwaves by identifying abrupt changes in the instantaneous flow/volume, density and speed.

The hypothetical incident management system presented in this dissertation exercises the distributed and collaborative processing ability of the traffic sensor network through two-phase incident detection including local detection and clustered verification. The system is modeled in the integrated simulator to assess its feasibility and functionality.

The test freeway network is selected in Spartanburg, South Carolina, containing 3 freeway corridors I-85, I-26, and I-85 Business that meet and form a triangle. Figure 3.4 shows this network as it appears in the PARAMICS interface. The I-85 segment between exit 68 and exit 70 has high traffic volumes and a high occurrence rate of incidents that block all lanes; hence, it was identified as the main link for incident generation. The other two corridors serve as the alternative routes to the main link. With PARAMICS, network building began with collection of field data including geometric, traffic control, and traffic volume data. The network is then calibrated through comparison between the simulated volume output and the field traffic counts data, as well as via comparison between the simulator animations to the site observation. The validation process

compared site-collected queue lengths and travel times to those produced by the simulation model. The process iterated with adjustments to the road network and driver behavior until the travel times were within one percent and there is no significant difference between the field observed queue lengths and simulation generated ones.

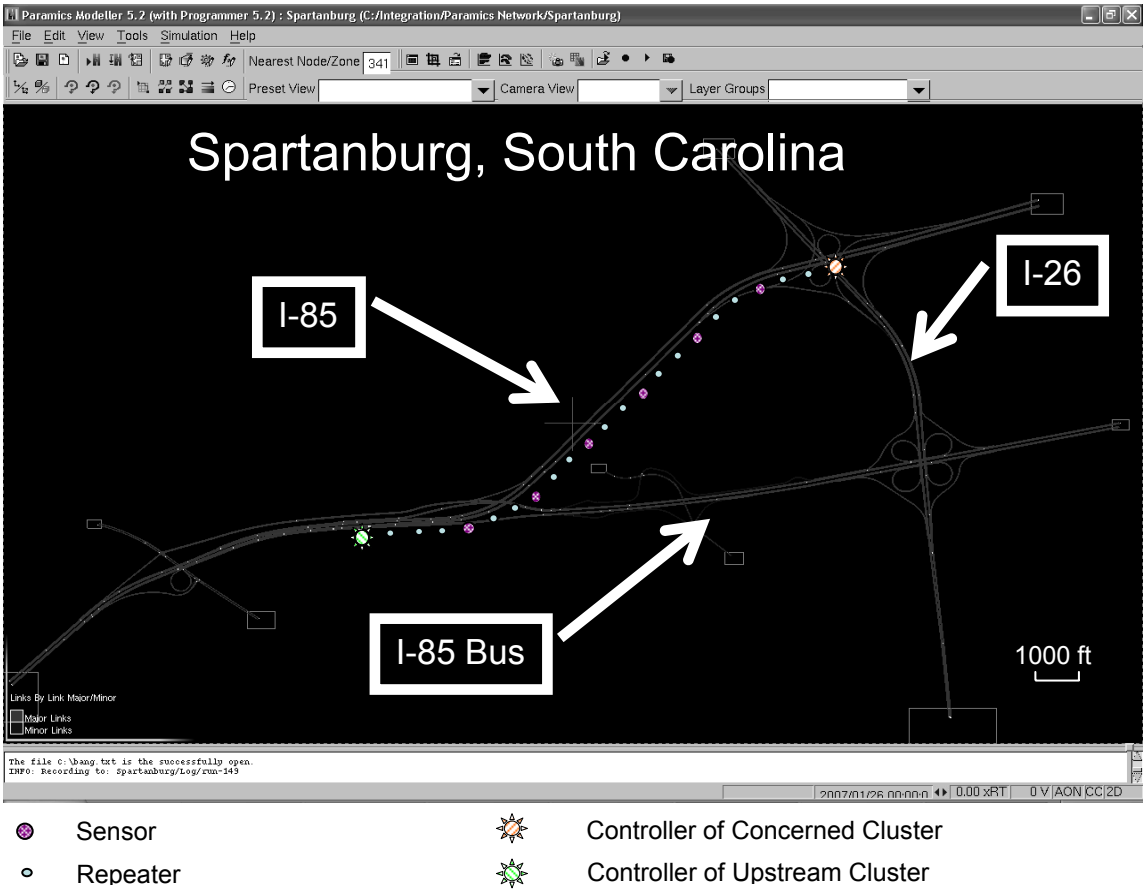


Figure 3.4 Simulated freeway and placement of sensors and controllers

3.2 Evaluate Communication Alternatives

This research sought to develop a systematic approach to evaluate different communication alternatives for real-time traffic surveillance system. This approach includes identifying alternative architecture, such as distributed or centralized, and

identifying communication mediums including wired, wireless or a combination of both. Then, a set of important measures of effectiveness (MOE) for making objective comparisons between alternatives based on the performance of the communication systems related to real-time traffic time surveillance was identified. The simulation platform developed in the previous task was used for efficiently evaluating communication alternatives with different architecture and mediums by generating important MOEs.

A case study was performed for a test network in Greenville, South Carolina. The authors followed the proposed evaluation approach to identify four communication alternatives, namely the centralized-wired, distributed-wired, centralized-wireless and distributed-wireless, to generate the selected MOEs, such as throughput, delivery ratio, and throughput cost ratios, for comparing and analyzing these alternatives.

3.2.1 Alternative Identification

The communication infrastructure for a real-time traffic surveillance system can adopt either a centralized or distributed architecture. To choose among alternative communication architectures, one must evaluate the advantage and disadvantage for each, and carefully balance the trade offs between them. A centralized communication infrastructure allocates dedicated bandwidth to connect a central controller with a set of controlled field devices, which are in general referred to as *sensors* in this dissertation. A distributed communication infrastructure, on the other hand, makes no distinction among central controllers and field devices. Each device is connected to nearby peer devices for relay, sharing, and confirmation of their sensing information. Using traffic camera

system as an example, a centralized solution aggregates all traffic surveillance data to one place for centralized processing. In contrast, a distributed solution assigns each device with certain distributed decision making ability and each device adjust the data generation rate based on such decisions on traffic conditions. Moreover, in distributed systems, multiple sensors and controllers may share the bandwidth of communication links among them. Here, competition for communication resources might occur. Figure 3.5 illustrates a sketch of the typical topology of centralized and distributed communication networks.

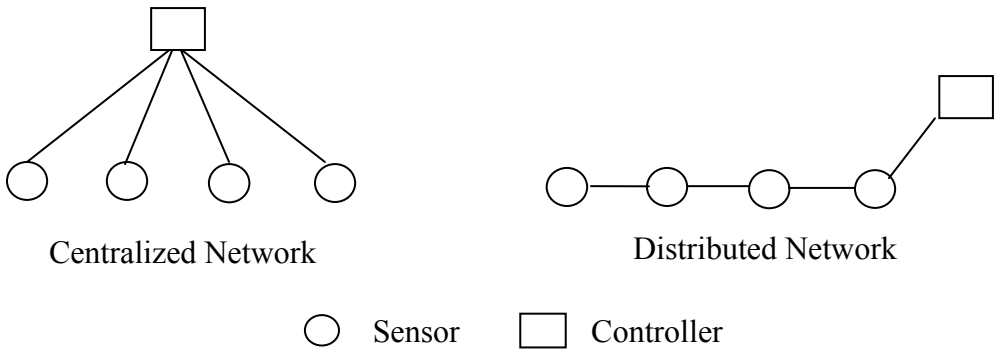


Figure 3.5 Topology of centralized and distributed communication network

For selecting the communication medium, although wired lines are the most prevalent communication medium used in vehicular traffic control system, wireless communication has become a popular technology for recent ITS applications. Wired communication can be very costly for large scale implementation. It also causes inconvenience for maintenance and system extension. However, wireless communication can be affected by environmental conditions such as adverse terrain and weather conditions.

3.2.2 MOE Selection

The performance and costs for different communication alternatives must be evaluated with respect to the specific communication needs of the ITS application in question. The problem of quantifying, measuring, and controlling the performance metrics of a network has been studied extensively in the context of Quality of Service (QoS) analysis (Peterson et al. 2003). The MOEs for the ITS communication system must therefore be selected in terms of the proper QoS metrics with respect to the application requirements.

An ITS communications system must transfer information from field components to the traffic operations center, which will then transmit responses and commands to various field components (Gordon et al. 1993). According to the respective components' functionalities, MOEs for the communication system can include its bandwidth and data rate, where bandwidth of a network is given by bits that can be transmitted over the network in a certain period of time (Peterson et al. 2003). The reliability of timely monitoring and response operations is also of crucial importance. Reliability is affected by environmental (terrain, weather) as well as human factors. For example, while wired communication is typically considered reliable, its communication can completely break down due to physical damage to the wires during construction or adverse weather, and such damages are time consuming to locate and repair. Wireless communication is sensitive to terrain and weather conditions even during its off-peak operation. Communications can occasionally be lost or contain errors. The degree of such errors/loss increases as the adverse conditions worsen, yet at all times, a fraction of

communications can be made successfully, which poses a significant opportunity to enhance the system's reliability under all conditions.

The communication system is the most expensive part of a traffic surveillance system (Gordon et al. 1993). The cost of a communication alternative must be justified with respect to its QoS requirement. Hence, MOEs must also be defined to quantify the relationship between costs and performances, such as throughput per unit cost (e.g., Megabytes per dollar), to facilitate the system planning process.

3.2.3 Simulation Study

The integrated simulation platform based on the PARAMICS traffic modeler and the ns-2 network simulator is used to conduct simulation study for evaluation of communication alternatives. PARAMICS is a detailed microscopic simulator that provides realistic traffic flow and detector modeling, with an extensive API for plugging in customized control procedure and external interface (Quadstone 2006). Ns-2 is an open-source, packet level and event-driven network simulator, allowing modular incorporation of newly developed protocol components and interface with other software (Sobeih et al. 2005). User-defined functions are programmable via an API add-on module in PARAMICS and a plug-in C++ source file, with which the integrated simulator implements traffic statistics-logging, synchronize sensor data and exchange control commands with ns-2.

3.2.4 A Case study to Evaluate Communication Alternatives

To illustrate the efficacy of the approach to analyze and evaluate the performance of different communication alternatives, a case study on selecting the best communication alternative for a real-time traffic surveillance system in Greenville, South Carolina, was conducted.

3.2.4.1 Alternative Identification

Greenville is one of the largest cities in South Carolina. With 3 major national freeways, I-85, I-185, and I-385 passing by, one of the four SC transportation management centers (TMC) is located here to enhance the traffic management and operation.

The author first started the alternative identification with collecting data, which included highway topography, ITS asset locations and traffic volumes. Greenville is one of the major ITS hubs in South Carolina. The existing ITS equipments includes traffic cameras, traffic detectors, count stations, a traffic management center, variable message signs (VMS), and highway advisory radio (HAR) stations. The infrastructure information in the database includes the locations of facilities and such attributes as cost, bandwidth, latency and power requirements. Table 3.2 shows the list of the existing ITS equipment.

Table 3.2 Example of list assets for communication analysis

	Asset	Resource
1	Traffic Cameras	SCDOT
2	Traffic Detector	SCDOT
3	Count Station	SCDOT
4	Traffic Management Center	SCDOT
5	HAR Transmitter	SCDOT
6	VMS	SCDOT
7	Traffic Signal	SCDOT and operation city/county
8	Drop Cabinets	SCDOT and operation city/county
9	DOT Owned Fiber	SCDOT
10	Leased Fiber	Commercial carriers
11	Fiber Node/Hub	SCDOT
12	Coaxial Cable	SCDOT and commercial carriers

The obtained ITS infrastructure and communication system information was geo-coded into the GIS software in different layers, each containing the available attributes (location, bandwidth, cost, etc.) of a particular type of ITS device. This can help analyze existing and proposed communication alternatives. Figure 3.6 shows a GIS map with a layout of the existing ITS infrastructure of Greenville.

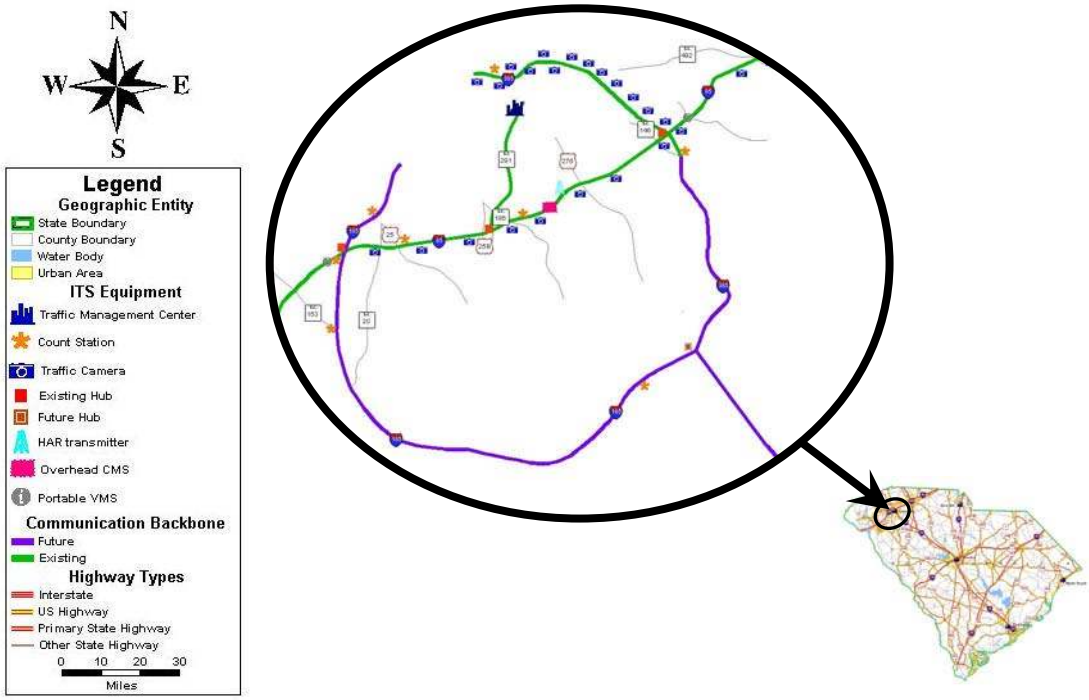


Figure 3.6 GIS map of ITS and communication infrastructure in Greenville, South Carolina

Considering the choice over two system architectures (i.e. distributed and centralized) and two communication media (i.e. wired and wireless), four alternative communication architecture were studied to support the traffic surveillance system comprised of traffic cameras and a traffic management center. The four alternative architectures considered were wired centralized, wireless centralized, wired distributed, and wireless distributed.

3.2.4.2 MOE Selection

To select the MOEs for this study, the ITS application requirements and the various system variables for the four communication alternatives were analyzed. To

evaluate both the performance and the cost effectiveness of the alternatives, the study selected three MOEs: the peak achievable throughput, the successful delivery ratio, and the throughput per unit cost. Figure 3.7 illustrate the MOE selection process. Delay was not selected as a MOE because the magnitude of communication latency is far less than the time magnitude of the traffic surveillance events when the required data rates are below the provisioned capacity of the communication network.

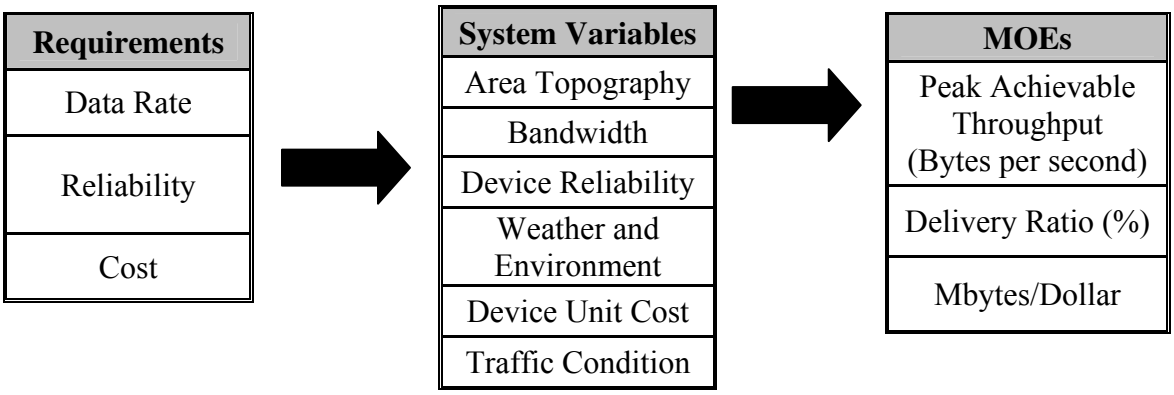


Figure 3.7 Relationships between communication requirements and evaluation MOEs

3.2.4.3 Simulation Study

The I-85 corridor in Greenville, South Carolina was selected as the study network, which consists of approximately 11 miles of freeway and 6 interchanges. This segment of I-85 is the major corridor connecting Atlanta, Georgia, and Charlotte, North Carolina. It serves the traffic to and from the Greenville metropolitan area with a population of 601,986 according to the 2006 census estimate.

After site selection, the author used the PARAMICS microscopic traffic simulation software to build, calibrate, and validate the roadway network. Network building began by collecting various data including geometry, traffic control, and traffic

volume. The geometric layout data for the roadway network was obtained from South Carolina Department of Natural Recourses in GIS format. Next, aerial photos from multiple sources and information collected from site visits were used to verify correct geometric conditions, such as number of lanes, lane widths, lane allocation, and curvature. The specific location of each traffic camera was also added to the network according to the South Carolina Department of Transportation (SCDOT) GIS data base. The author requested and received the traffic volume and incident data from the SCDOT, and local planning organizations. The SCDOT provided hourly and average daily traffic count data, traffic signal timing data, and incident location, severity and duration data. The local planning organizations provided a planning model for use in predicting the origins and destinations matrix of the future network traffic. Other data needs such as speed limits, rights of way, and stripping, were met through observation during site visits. All this information was used to build the traffic simulation model in PARAMICS.

To ensure that the simulation model reflects traffic conditions accurately, the calibration and validation steps are of the utmost importance. The calibration steps involve "face validation" of the traffic model animation and comparison of simulated and measure traffic volume. The validation of the system performance output was carried out by comparing observed travel times and queue length with the simulated ones. After many iterations and adjustments to the road network and driver behavior parameters, the expert opinions from the local traffic management centers' staff confirmed that the traffic model was a realistic representative of the real world. In addition, the overall simulated vehicular traffic volumes were within one percent of the measured, the highest individual

volume error was no more than ten percent, and most of the individual volume errors were less than five percent. Furthermore, the simulation model accurately reflected the observed travel times within one percent and there was no significant difference between the observed and simulated queue lengths at the bottleneck segment, which were at the signalized off ramp intersections.

The average annual daily traffic was obtained from the SCDOT and converted to hourly volume according to the typical traffic volume profile of an average weekday. The traffic scenario for this study was PM peak period during an average weekday because the peak traffic flow occurred between 4:30 PM and 6:30 PM at the study site. The simulations were started at 4:00 PM and allowed at least half an hour of warm up time. After the traffic volumes were fully loaded into the network, incidents were generated at random locations and random times between 4:30 PM and 5:00 PM.

The ns-2 communication simulator implemented the T1 data links with a bandwidth of 1.544MHz as the medium for wired centralized and distributed alternatives. For the wireless centralized system, the author assumed that traffic surveillance operating agencies will lease the CDMA2000 data links with a bandwidth of 1.25 MHz. For the wireless distributed alternative, IEEE 802.11b protocol with a bandwidth of 11MHz is assumed for communication among sensors and controllers in the field.

The study considered traffic surveillance data generated at constant bit rate and sent across the network using the User Datagram Protocol (UDP). Different data rates were simulated to examine the capacity of the four alternatives. For the study scenario with incidents, the vehicular traffic simulator randomly generated incidents on the

segments under surveillance of traffic cameras during the AM peak hours through PARAMICS Programmer's API interface. The program selected various incident occurrence times, locations and severities accounting for the effects of different incident scenarios. PARAMICS also determines the duration of incidents through the realistic simulation of interaction between DVUs including the vehicles involved in incidents and the vehicles in the queue. The duration of incidents, which is defined as the time period between incident occurrence and the return to normal traffic condition, directly affects the communication cost in terms of data rate, which can be altered by the ns-2 during the simulation. In a centralized system, each device continuously generates constant rate data at the rate of 384Kbps no matter there is an incident or not. On the contrary, in a distributed system, the devices send stationary images, which generate a consequent date rate of 24Kbbs to the controller at a low frequency during the normal condition. Once an incident was identified or suspected, the corresponding traffic camera transmits full motion videos with a data rate 384Kbps, which is the same as the constant data rate of the centralized system, to the controller. Within the two hours simulation period, throughputs of centralized and distributed system for various incident durations were examined to compare their communication costs.

3.3 Develop VII Simulation Model

This section discusses the approach used for developing a hybrid architecture, building a test network, developing a SVM/SVR model for VII system, and evaluating performance of the VII model. The first step of the approach involved a hybrid architecture that integrates centralized and distributed architecture.

3.3.1 Design Hierarchical Architecture for VII Model

The hybrid networking concept in the on-line traffic condition assessment and prediction framework was exemplified with the hierarchical model proposed in (Mirchandani and Head 1998), leveraging the incremental scopes of road segments, intersections, and networks. The architecture effectively extracts traffic dynamics at the various levels, with which optimal control methods were derived.

As shown in Figure 3.8, the hierarchical architecture includes multiple hierarchies, with each of which is comprised of one type of such components as vehicles, RSU, and various level of controllers. The traffic sensing, processing, and networking are composed in a distributed way within its own hierarchy and in a centralized way between the parent hierarchy and its child hierarchy. From the road level, individual vehicles collect and process their individual microscopic traffic statistics, and then report the processed information to the RSU when they are approaching them or through the relay of wireless repeaters or other vehicles. Each RSU performs its assigned function with corresponding computing resources. In the proposed system, RSUs in each cluster working in a distributed fashion, receive data from the vehicles, perform analysis on traffic conditions along road segments and report data to its controller. Centralized control functions are also implemented with underlying message exchanging between RSUs and controllers, or between child controllers and parent controllers. Therefore, an ad hoc wireless network is formed to support the hybrid framework for on-line traffic condition assessment. A cluster is a logical grouping of roadside agents and controller agents in which agents in each level communicate with its lower level agents or upper

level controllers. The number of levels and the classification varies with the specific traffic network characteristics and application cases (Kochhal, et al., 2003, Subramanian and Katz 2007). In addition to assess information from the lower level entities in its cluster, controllers can utilize information from sources other than vehicles, such as cell phone calls from drivers, weather reports, road condition and various events. The controllers then interface with other traffic control entities such as the traffic signal control systems and freeway ramp metering for on-line response. Figure 3.9 presented an example set up of the functional elements for VII model implemented in Spartanburg, South Carolina.

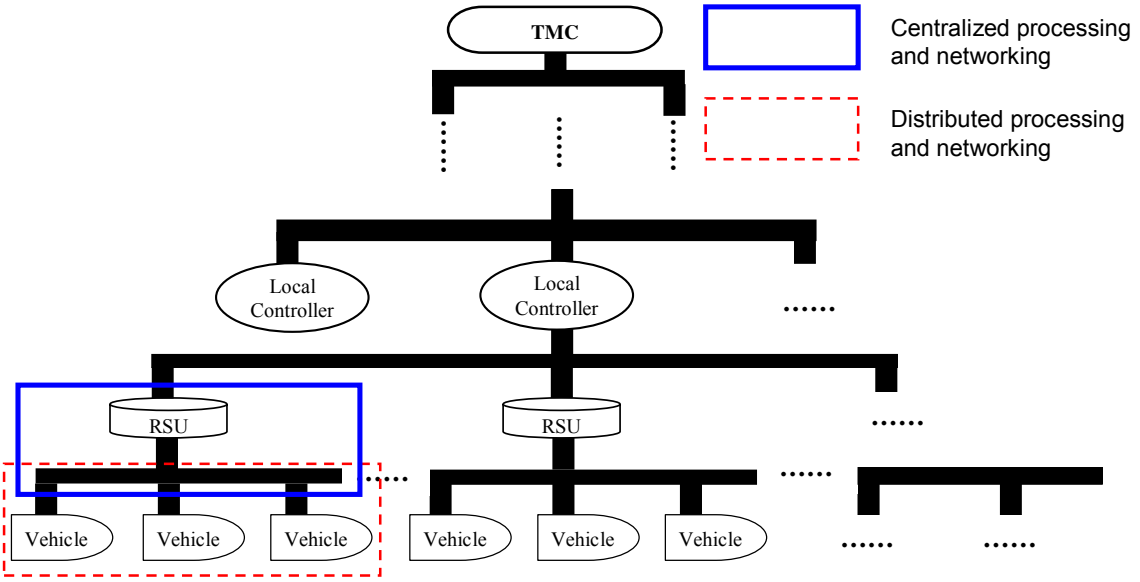


Figure 3.8 Hybrid architecture for VII Model

The addressing scheme of the hierarchical system must facilitate the location assisted routing of all kinds of data packets. As shown in Figure 3.9, the addresses of vehicles, RSUs and controllers are in the format of [RID, Mileage, Direction, Level]. Following the convention of the U.S. highway reference system (Tokuyama 1996), each

highway or major arterial road has a unique road identification number (RID). The location of each identity on the road is uniquely identified with its mileage from the road's starting point. The RSUs or controllers can have one or multiple addresses, according to its location (on one or multiple highways or an intersection/interchanges), its monitoring scope (overseeing one or both sides of a road), and its association with one or multiple clusters and task levels.

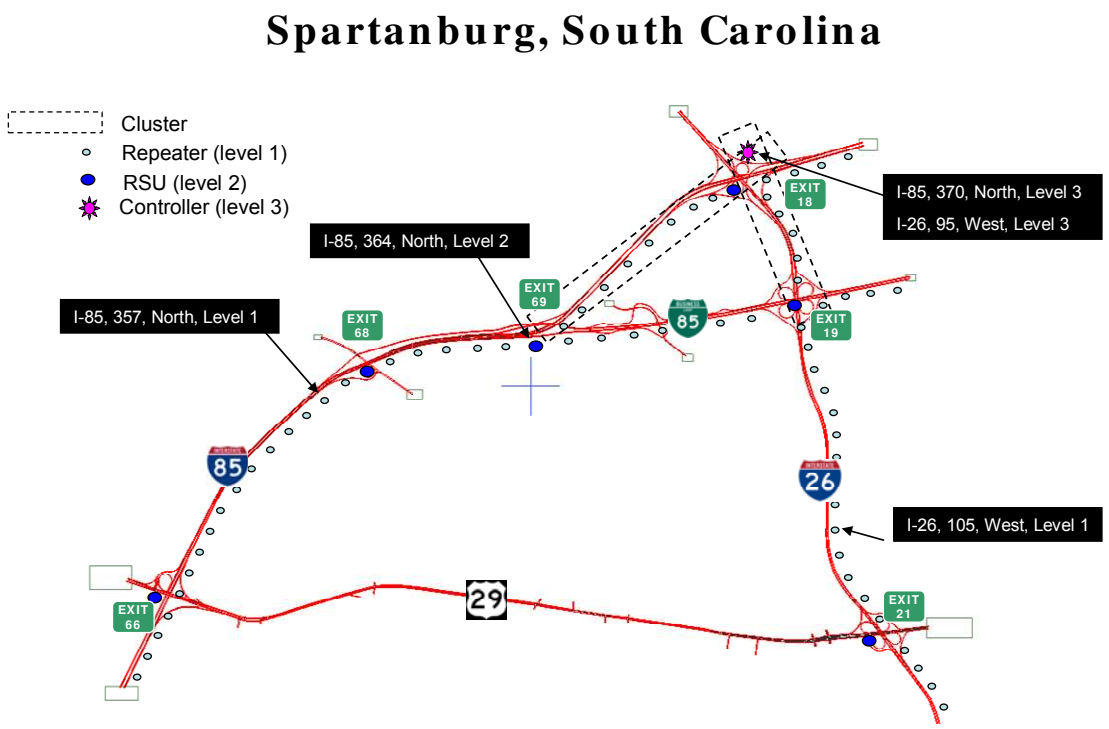


Figure 3.9 Functional elements set up with addressing configuration examples for the VII model implemented in Spartanburg, South Carolina

Message routing among sensors and controllers is done in the hierarchical address space, with tailored emphasis on simplicity for vehicles and repeaters and intelligence at RSU and controllers. Message forwarding routes discovery is based on local broadcasts: each repeater (level 1) discovers, records, and registers its immediate neighbors at the

same level and its supervisor RSU at the upper level (level 2). Iteratively, each RSU continues to discover, record, and register its immediate adjacent RSU and its controller at level 3, which usually located at interchanges of multiple major highway intersect. This process iterates until the top level controllers are reached. The hierarchical routing procedure always forwards messages along the physical roads. RSUs route messages up or down the same road (if destination is on the same road), or towards its parent controller (if destination is on a different road). Controllers route messages in one of four ways: 1) up or down the same road, 2) to an adjacent controller, 3) to its parent controller, or 4) to its supervised RSUs.

RSUs acquire and process data from vehicles, and participate in collaborative functions with nearby RSUs and controllers. Collaborative functions are implemented with underlying message sending and receiving functions for exchanging information in the hierarchical addressing scheme. The hierarchical, process-based programming semantics is consistent with existing traffic control and distributed system design practices.

3.3.2 Develop Computational Intelligence Model

The author developed a SVM algorithm for online traffic condition assessment and a SVR algorithm for real-time travel time prediction using VII system, as SVM is quite suitable in pattern recognition and classification, while SVR performs well on parameter estimation and regression. Using the individual vehicle dynamics measured by each VII-enabled vehicle, the traffic condition assessment and travel time prediction

module of VII model was able to indentify the occurrence, locations and severity of incidents, and predict the travel time in a real-time fashion, respectively.

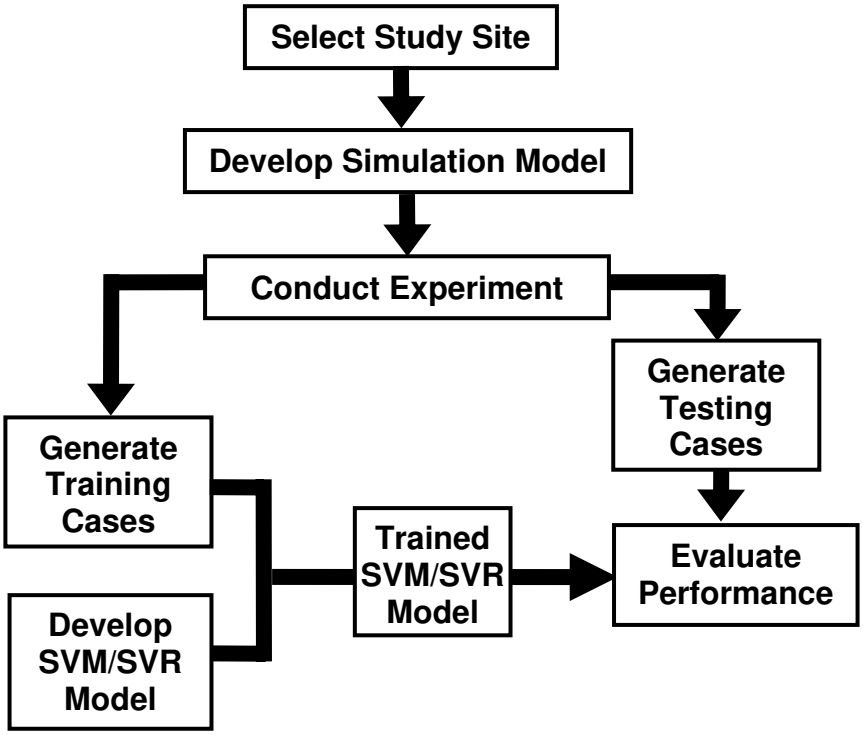


Figure 3.10 SVM/SVR model development and evaluation

As shown in Figure 3.10, the first step of this process was to select a test network and develop a detailed microscopic simulation model for the network. After calibration and validation, the traffic simulation model was applied to generate training and testing cases. The development of computational intelligence model for VII was an interactive process that included designing vehicle data collection plan, cross validation of the training sets and grid searching of optimal parameters for the SVM/SVR model. The trained SVM/SVR model was applied to the VII-enabled vehicles generated statistics in a real time fashion to evaluate the performance of the developed intelligent algorithm.

SVM and SVR is a collection of algorithms based on the similar underlying theory to achieve nonlinear classification or regression by mapping the training samples onto a high dimensional kernel-induced feature space, followed by linear classification or regression in that space. Since the kernel mapping is implicit, depending only on the inner or dot product of the input data vectors, it is possible to map the data into high dimensions and still keep the computational cost low.

Figure 3.11 gives an overview of the concepts of SVM and SVR. In this study, kernel functions such as radial basis function (RBF) will be used (Vanschoenwinkel and Manderick 2006). As shown in Figure 3.11, the SVM / SVR model depends on a subset of the training samples ***s***, support vector coefficients ***Cs*** and a constant ***b***.

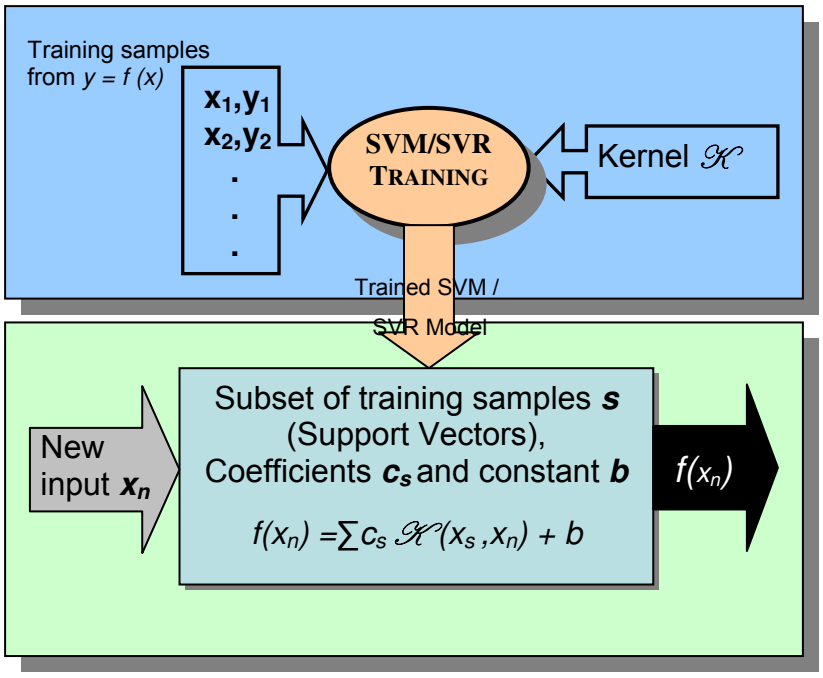


Figure 3.11 Concept of SVM and SVR

3.3.2.1 Study Sites and Simulation Model Development

The microscopic traffic simulation model PARAMICS was used to create a realistic traffic environment to develop and evaluate the VII model for on-line traffic assessment framework. The author selected a freeway network in Spartanburg, South Carolina, (see Figure 3.4) as the study site for develop the incident detection functionality of VII model. As the I-85 segment between exit 68 and exit 70 has three lanes in each direction with high traffic volumes and a high occurrence rate of incidents, it was chosen as the link for the experiment of evaluation of proposed VII incident detection system. The author simulated incidents blocking 1, 2 and 3 lanes, and recorded their impacts on vehicle kinetics on the study segment along I-85 North. The Application Programming Interface (API) program in this study was developed to continuously collect microscopic traffic statistics and apply SVM algorithm for classification of the collected data.

A freeway network in Greenville, South Carolina was selected as the study site for developing real time travel time prediction functionality of VII model. Figure 3.6 shows a layout of the network. Simulation model development procedure was already explained in section 3.2.4.3.

3.3.2.2 Case Generation

With the simulation model developed, the next step was to generate the cases required for developing and evaluating the proposed incident detection and travel time prediction model using computational intelligence.

At first, the cases for SVM algorithm were generated. The idea of using the microscopic traffic data from an individual vehicle to detect incidents was based on the assumption that when an incident occurs, the kinetics of passing vehicles site would be affected. These kinetics (i.e. speed drops and increases, increased lane changing, and significant acceleration and deceleration) could then be recorded by VII-enabled vehicles. This study identified the speed profile and lane changing behavior over selected time step s_t to recognize the patterns that indicate the occurrence of incidents. Specifically, an API program was developed for each VII-enabled vehicle to log an array of six speed values and six lane change indicators for each time slice s_t. Table 3.3 shows a sample of each vehicle's data log, which will be the input for SVM algorithm. The VII-enabled vehicles were assigned as specific types with varying portion to entire traffic population depending on the penetration rate of VII-enabled vehicles.

Table 3.3 Sample Data Log in Vehicle On-Board Units

Kinetics	Time Instant					
	t	t-s_t	t-2s_t	t-3s_t	t-4s_t	t-5s_t
Speed (mph)	52.6	62.7	22.4	36.5	52.5	66.0
Lane Change	0	1	0	0	0	0
Decision	+1					

The decision of the instance y_i is either +1 for a vehicle passes an incident site, or +2 a vehicle stops in the queue, or -1 for a non-incident scenario. After 10 minutes warm up time, an incident was generated along the segment between Exit 68 and Exit 70 on I-85 as shown in Figure 3.4. When a VII-enabled vehicle either passed by the stopped in queue, the speed and lane changing data was stored into the training set file. A total of

129 vehicle cases from 12 incident experiments were recorded. For the other case, 179 vehicles traveling under normal condition from 12 non-incident experiments were recorded. The non-incident scenario was able to record more data than during the incident scenario because VII-enabled vehicles will more easily travel through the network and transmit their recorded data.

Similar to SVM, the cases for SVR model include a series of vectors (x_i, y_i), where y_i is the target value and x_i is the input vector that has three member variables. The target value is the average travel time of the vehicles that depart the start point in the next time interval. The input variables include the average travel time collected at the end of the study segment, the number of VII-enabled vehicles and number of VII-enabled vehicles entered the study segment during current time interval.

For the experiment scenario, four weeks of weekday travel data were collected. The traffic demand profile for each weekday was different to represent the day-to-day travel time pattern. Among them, two-week data was randomly selected as training data and the remaining two-week data were used for testing.

When the experiment scenario varied with different penetration rate, the SVR model needed to be trained again, therefore new cases needed to be generated to accommodate the changes in penetration rate because the VII-enabled vehicle volume varied significantly for different penetration rate.

3.3.2.3 Develop SVM algorithm

The training sets for SVM algorithms, in which (x_i, y_i), i=1, …, l where $x_i \in R^n$ and y_i is the classifier. In the following section, a two-class classification problem, where $y \in \{-1,1\}^l$, will be introduced first as it is the basis of the multiple classification problem. Note that this study followed the classical SVM or so called C-SVC (Boser et al. 1992; Cortes and Vapnik 1995) for two-class classification. x_i is the input of vehicle microscopic statistics (i.e. time series of speed and lane change indicator) and n is determined by the time window size and decision time steps. For example, a given time step is 4 seconds and the time window is 24 seconds; then $n = 2*(24/4) = 12$, to provide a time series representing both vehicular speed and lane change. y_i is the classifier of normal ($y_i = -1$) or abnormal ($y_i = 1$) condition. The objective of the training is to find the prediction function $f(x_i) = w * x_i + b$ that optimizes the minimum distance between the classification hyper-plane for any sample of the training data. This is expressed using the following formula (Stitson *et al.* 1996; Chang and Lin 2007).

$$\min_{w,b} \quad \frac{1}{2} w^T w$$

$$subject \;\; to \quad y_i (w^T x_i + b) \geq 1 \qquad\qquad i = 1,...,l$$

Equation 3.1

Considering the non-separable data to allow training errors, one can incorporate an error term ζ multiplying a penalty parameter C. The objective of the prediction function objective can be achieved by solving the following optimization problem (Hsu *et al.* 2007).

$$\min_{w,b,\xi} \quad \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i$$

$$subject \quad to \quad y_i (w^T \phi(x_i) + b) \geq 1 - \xi_i \qquad i = 1,...,l \text{ and } \xi_i \geq 0$$

Equation 3.2

Note that only the transformation coefficients ø of support vectors are not zero and most of the error terms ξ are zero.

Here, training vectors x_i are mapped into a higher dimensional space by the function ø, enabling SVM to find linear separating the hyper-plane with the maximal margin in this higher dimensional space. C>0 is the penalty parameter of the error term. Furthermore, $K(x_i,x_j) = \phi(x_i)^T \phi(x_j)$ is called the kernel function. As research shows that radial basis function (RBF) generally performs well in many scenarios (Vanschoenwinkel and Manderick 2006), RBF are selected as the kernel function of this research.

$$K(x_i, x_j) = \exp(-\gamma \| x_i - x_j \|^2) \quad , \quad \text{Here, } \gamma \text{ is the kernel parameter.}$$
$$\gamma > 0$$

Equation 3.3

The results of the classification would be the sign of the decision function:

$$w^T \phi(x) + b$$

Equation 3.4

If $w^T \phi(x) + b > 0$, then y =1. On the other hand, $w^T \phi(x) + b < 0$ will induce y=-1.

For the multi-class classification problem like in this study $y \in \{-1,1,2\}^l$, a "one-against-one" (Knerr et al. 1990; Friedman 1996; Kreβel 1999) approach was applied to

construct multiple two-class classifier and classify any two different classes using the method similar to the two-class SVM as shown in following:

$$\min_{w^{ij},b^{ij},\xi^{ij}} \quad \frac{1}{2}(w^{ij})^T w^{ij} + C\sum_{i=1}^{l}(\xi^{ij})_t$$

$$\textit{subject to} \quad (w^{ij})^T \phi(x_t) + b^{ij} \geq 1 - (\xi^{ij})_t, \text{ if } x_t \text{ is in the } i\text{th class}$$

$$(w^{ij})^T \phi(x_t) + b^{ij} \leq -1 + (\xi^{ij})_t, \text{ if } x_t \text{ is in the } j\text{th class}$$

$$t = 1,...,l \text{ and } \xi_{ij} \geq 0$$

Equation 3.5

After the results of multiple two-class classification were available, voting strategy was used to predict the class for specific input vector. When the votes for two classes are the same, the class with smaller index was selected for simplicity (Chang and Lin 2007). For example, there are three classes {-1,1,2} needed to classified, then 3*(3-1)/2=3 two-class classifier was constructed and applied respectively. Then, if the results of the 3 two-class classification come out to be that 1 for -1 VS 1, 2 for -1 VS 2, and 1 for 1 VS 2, the final decision of this multi-class classification will be 1. If each class got 1 vote, the input vector would be classified as -1 for conservative consideration of minimizing false alarm rates.

3.3.2.4 Develop SVR Model

Similar to SVM, SVR trains the training data set to identify the support vectors and mapping function coefficients and constants. The difference between SVM and SVR are that instead of having finite number of classifier in SVM, SVR has infinite number of

target output in the training data set. As a consequence, SVR would give any possible value in the output space from a group of input vectors. Given a training data set of (x_i, y_i), i=1, …, l where $x_i \in R^n$ and y_i is the target output, the objective of the training by applying ε-SVR is to find the prediction function and mapping function:

$$\min_{w,b,\xi,\xi^*} \quad \frac{1}{2} w^T w + C\sum_{i=1}^{l} \xi_i + C\sum_{i=1}^{l} \xi_i^*$$

$$subject \quad to \quad w^T \phi(x_i) + b - y_i \geq \varepsilon - \xi_i$$

$$-w^T \phi(x_i) - b + y_i \leq \varepsilon - \xi_i^*$$

$$i = 1,...,l \text{ and } \xi_i, \xi_i^* \geq 0$$

Equation 3.6

where ε is parameter in ε-SVR represent the marginal error of regression. Due to its good performance, RBF was again selected as the kernel function for SVR model.

The prediction function for new input will be:

$$y = w^T \phi(x) + b$$

Equation 3.7

Though, the prediction function of SVR is in the same form of classification function for SVM, the predicted output of SVR can be any estimate in the output space instead of several pre-defined classifiers of SVM.

3.3.2.5 SVM and SVR Implementation

As noted by *(*Hsu et al. 2007; Sarle 2007*)*, scaling is important for the success of AI paradigms such ANN and SVM. Before training, all the data were linearly scaled to a

range of [0, 1] using a common range file. To maximally the usefulness of the training data and search for optimal parameters, the authors randomly divided them into 5 groups. Each time, four groups of data were used to train a SVM/SVR model with a possible combination of parameters, while the trained model was tested on the remaining group to estimate the prediction accuracy of this testing group. This process was repeated five times with the same combination, but with different testing groups, to obtain an average prediction accuracy in terms of percentage for SVM or mean squared error (MSE) for SVR.

The most important parameters for classifications using SVM with radial basis kernel are C and γ. The optimal parameters were identified through grid searching of 110 combinations in the range of [C, γ] = [2^{-5}: 2^2 : 2^5, 2^{-15} : 2^2 : 2^3]. The authors performed the experiment by increasing parameters in exponential order, i.e. 2^n, in the range of -5 to 5 for C and -15 to 3 for γ with a step of 2. The identified optimal parameters were then used to train the entire training set to generate a trained SVM algorithm. SVR model has one additional important parameter ε as shown in Equation 3.6. Similar to the procedure of finding optimal parameter combination in train SVM, SVR applied grid searching technique in the range of [C, γ, ε] = [2^0: 2^2 : 2^{10}, 2^{-2} : 2^2 : 2^8, 2^0 : 2^2 : 2^{10}] to identify the values of C, γ and ε.

This study used LIBSVM (Chang and Lin 2007), an open source implementation routine for SVM and SVR, to train and test the SVM/SVR model. The training time of the SVM/SVR model was less than two seconds in all the training cycles. The prediction

time for an unknown case was in milliseconds magnitude, which is reasonable and suitable for real-time application.

3.3.3 Baseline Algorithm Selection and Development

In order to provide baseline algorithms for comparison with the developed intelligent algorithm, two popular and easy-to-implement algorithm, California automatic incident detection algorithm and instantaneous travel time prediction algorithm were developed and compared with SVM and SVR models, respectively. The comparison of the SVM/SVR and baseline algorithm was performed in the same network and same prevailing traffic conditions.

3.3.3.1 California Algorithm for Incident Detection

The author selected California #7, for its good performance and simplicity among the California algorithm family, as the baseline algorithm to compare with SVM incident detection algorithm (Payne and Tignor 1978). California #7 examined the occupancy data from two neighboring loop detectors to decide the incident states through a decision tree. There are totally 7 tests that examine three parameters: downstream occupancy (DOCC), spatial difference in occupancies (OCCDF) and relative spatial difference in occupancies (OCCRDF) to decide the incident state, which can be one of the four states: incident free, tentative incident, incident occurred, and incident continuing. The decision tree for California #7 is shown in Figure 3.12. The virtual loop detectors were placed at the density of one every quarter mile along the same Spartanburg freeway network in South Carolina. The occupancy data from the loop detector were aggregated into 1-minute

interval, average over all lanes. The decision interval was also one minute. Incidents that block one, two, or three lanes were created in the network after 10 minutes of warm-up time and 10 minute of no incident period. There were 100 experiments conducted for each type of incident. These data was used for off-line calibration to identify the parameter set in the decision tree.

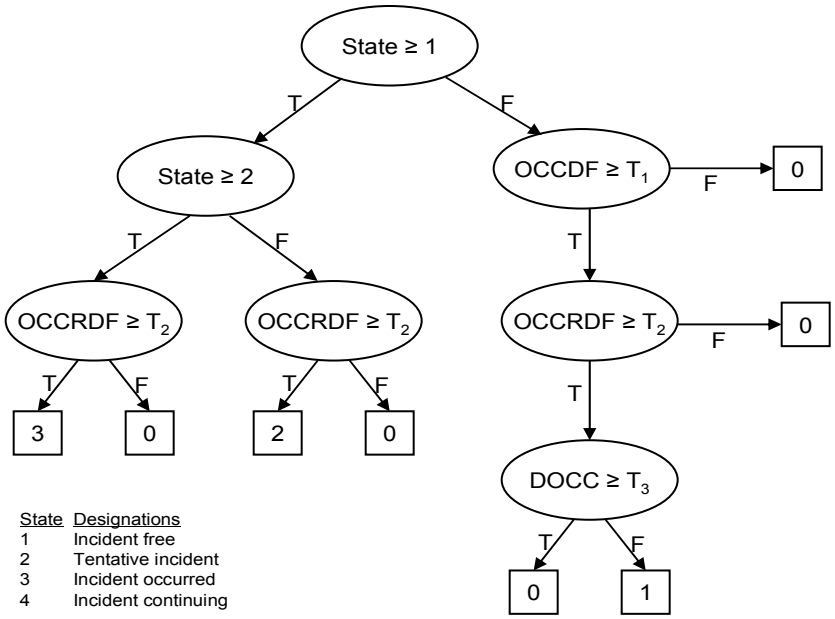


Figure 3.12 Decision tree for California #7 (Payne and Tignor 1978)

The author followed the parameter calibration method shown in Equation 3.8 (Payne and Tignor 1978) to identify a series of optimal parameter set for the required detection rates. The Matlab code for parameter calibration was shown in Appendix B.

$$\min_{T}\{\alpha(T) \mid \beta(T) \geq y)\}$$

Equation 3.8

where T is the parameter set in the decision tree, $\alpha(T)$ and $\beta(T)$ are the consequent false alarm rate and detection rate for specific parameter set T, y is the required detection rate threshold. The author varied the required detection rate threshold in a range between

60% and 99.5% to determine a series of the three parameters (T_1, T_2 and T_3) in the decision tree.

3.3.3.2 Instantaneous Algorithm for Travel Time Prediction

The instantaneous travel time prediction model assume that the travel time does not change for a short period. Therefore, it only uses the available travel time collected within the immediate previous time step to predict the travel of vehicles that will start within the immediate following time step. Since the VII system is able to collect the travel time directly, the averaged travel time of the VII-enabled vehicles arriving at the end point during each time interval will be considered as the predicted travel time of the vehicles departing the start point during the next time interval.

3.3.4 Traffic Condition Assessment

A traffic condition assessment model was established in each RSU when the VII-enabled vehicles transmitted the vehicle experienced traffic status to it. Each RSU divided its supervised section into several segments and estimated the traffic status indicator in each segment. The number of segment depends on the length of the section, average travel speed, and VII-enabled vehicles sending message interval. To ensure each VII-enabled vehicle sends exactly one message for each segment when traveling at normal speed, the number of segment can be determined by following equation:

$$n_i = L_i / (\bar{v}_i * s_t)$$

Equation 3.9

Where n_i is the number of segment in section i, L_i is the total length of segment i, $\bar{v}_i$ is the average travel speed in segment i at normal condition, and s_t is the periodic VII message sending interval for each VII-enable vehicle.

The RSU collected the traffic status classifier prediction from the latest 6 VII-enabled vehicles. Since the classifier has 3 possible values: {-1,1,2}, the scale of the estimated traffic status of each segment is in a range of -6 to 12. When one vehicle stays in one segment more than one interval, the RSU will use the average predicted classifier prediction value for that vehicle. Figure 3.13 shows a sample of traffic condition assessment in a time space diagram estimated at a RSU. The color bar on the right represents the assessed traffic condition in a range of -6 to 12, as introduced above. For example, the wide spread deep blue indicates that most segments are in normal condition at most times. In this case, an incident occurred at segment 6 at 1112 seconds. If the threshold value was set at -1, after approximate 110 seconds, the incident will be indentified by the RSU.

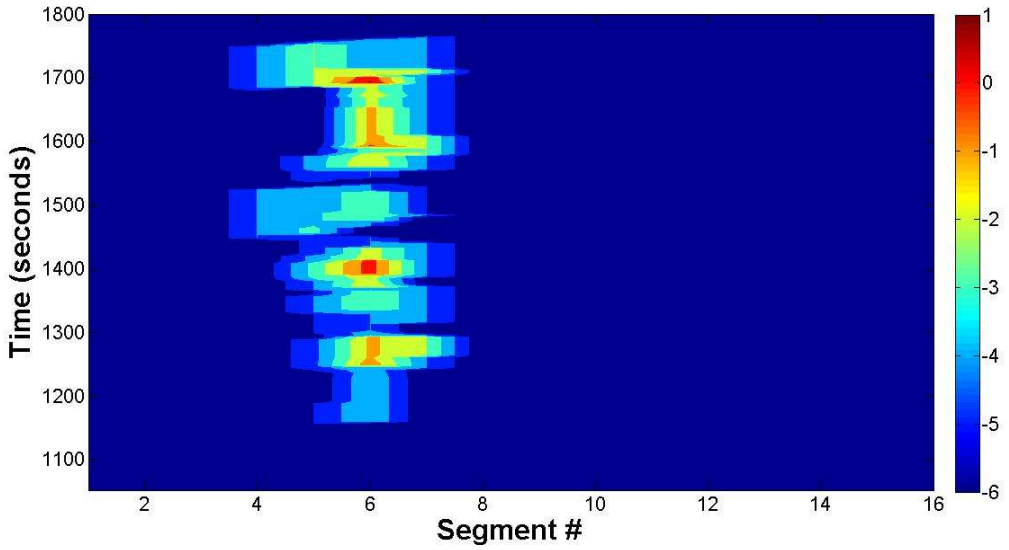


Figure 3.13 Sample contour map of the traffic condition assessment at a RSU

After obtaining the traffic status in time space diagram, the next task would be deciding the threshold to classify traffic status number into four statuses: incident free, incidents blocking one, two and three lanes.

3.3.5 Evaluate the VII Simulation Model

Using the trained SVM and SVR model, each VII-enabled vehicle traveling on the study segment on I-85 will continuously send message out with the time stamp, vehicle location, and the traffic condition based on the collected speed and lane change profile of itself. The RSU will receive those messages and would be able to assess the traffic condition and predict travel time.

If an RSU received threshold number of alarms i_t within maximum accumulation time t_{max}, then an incident was identified. To examine the impact of alarm number threshold and the maximum accumulation time window, the authors also conducted a

sensitivity analysis varying the value of i_t and t_{max}. Incident detection performance was assessed to determine the best combination of them.

The authors also tested different penetration rates (i.e., the percentage of VII-enabled vehicles in the total traffic in the study link or segment) to evaluate the effectiveness of the incident detection and travel time prediction model. For each selected percentage of VII-enabled vehicles in the network, 100 incident scenarios and 10 hours of non-incident scenario were tested for incident detection functionality of VII model and 2 weeks of weekday afternoon peak period were tested for travel time prediction functionality. The measures of performance for traffic condition assessment framework were detection time, detection rate, and false alarm rate. The detection time is defined as the time difference between the incident occurrence and the time it was correctly identified. The detection rate is the percentage of incidents that are correctly detected over the total number incidents occurred. The false alarm rate is defined as the number of false alarms per hour for the incident free period. In addition, the delivery ratio and communication latency as explained in section 3.2.2 are also selected as MOE in the communication domain.

Let t_i be the true value and y_i be the predicted value, then $e_i = y_i - t_i$ is defined as the prediction error and $re_i = e_i / t_i$ is the relative error. Statistical analyses were performed to examine if the true values and the predicted values were the same. Since the actual and predicted values were not independent with each other, a t-test for the difference of mean of two paired samples was conducted. The procedure of the hypothesis test with 95% confidence level was defined as following:

Hypothesis

H_0: $\mu_e = 0$

H_A: $\mu_e \neq 0$

Level of Significance

$\alpha = 0.05$

Test Statistics

$$t_{OBS} = \frac{\bar{e}}{s_e / \sqrt{N}}$$

Rejection Region

Two tails of a t-distribution with degree of freedom *N-1*

P-Value

$$2 * P(t > |t_{OBS}|)$$

where *N* is the number of predictions, μ_e is the true mean of the prediction error, $\bar{e}$ is the average prediction error, and $s_e = \sqrt{(\sum_{i=1}^{N}(e_i - \bar{e})^2)/(N-1)}$ is the standard deviation of the prediction error.

Additionally, the MOEs for evaluation of the accuracy and variation of prediction for the computational intelligence model, such as incident location prediction and travel time prediction model, are defined in Equation 3.10 through Equation 3.13.

Root mean of squared error proportional (RMSEP) in percentage:

$$\frac{100}{\bar{t}}\sqrt{\frac{1}{N}\sum_{i=1}^{N}(e_i)^2} \quad \text{with } \bar{t} = \frac{1}{N}\sum_{i=1}^{N} t_i$$

Equation 3.10

Mean relative error (MRE) in percentage:

$$\frac{100}{N}\sum_{i=1}^{N} re_i$$

Equation 3.11

Mean absolute relative error (MARE) in percentage:

$$\frac{100}{N}\sum_{i=1}^{N} |re_i|$$

Equation 3.12

Standard deviation of relative error (SRE) in percentage:

$$100\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(re_i - MRE/100\right)^2}$$

Equation 3.13

where N is the number of experiments for specific scenario.

CHAPTER 4

ANALYSIS AND RESULTS

This chapter presents the analysis to test a series of research objectives that hypothesizes: 1) if an integrated simulation platform can support the design and evaluation of online traffic surveillance system; 2) if distributed architecture provides better communication efficiency for both wireless and wired mediums; 3) if a hybrid framework, in combination with a VII system and computational intelligence improves the incident detection performance; and 4) if a hybrid framework, in combination with a VII system and computational intelligence, improves the accuracy of online travel time prediction. The following sections are organized into four primary categories according to each of these four hypotheses: integrated simulation platform, evaluation of different communication alternative, performance of the VII model for online traffic condition assessment, and performance of the VII model for real-time travel time prediction.

4.1 Integrated Simulation Platform

The following section presents a case study for the application of integrated simulation platform developed in this dissertation in evaluating an incident detection and responding system using a traffic sensor network. The simulation platform that integrates the state-of-the-art microscopic traffic simulator PARAMICS and the packet-level wireless network simulator *ns-2* was expected to accurately evaluate the effectiveness, efficiency, and reliability of traffic management and networking protocols. To

demonstrate the functionalities of the proposed integrated simulation platform, a wireless sensor network was modeled over a freeway section in Spartanburg, South Carolina for incident detection.

In this case study, traffic sensors with wireless interfaces were placed every quarter mile along the highway, where incidents would be randomly generated, to measure vehicle speed and traffic volume. Since a quarter mile distance was beyond the typical communication range of existing short range wireless communication protocols such as IEEE 802.11 a/b/g, wireless repeaters were placed in-between to relay messages. The incident detection algorithm consists of three phases: individual detection using the shockwave algorithm (Chowdhury and Sadek 2003), collaborated verification between adjacent sensors, and notification to cluster (parent) controller by the sensor verifying the incident. Upon receiving the detection notification, the cluster controller immediately notifies its upstream cluster controller to warn the drivers of incidents.

Performance metrics collected were incident detection rates, false alarm rates, and wireless network communication latencies. The interdependencies among the performance metrics and the sensor and controller placement were examined to predict the system's operation and optimality prior to the actual deployment. Since the shockwave algorithm is the basis for the incident detection algorithm used in the case study, its basic concepts are first reviewed.

4.1.1 The Shockwave Algorithm

The shockwave caused by an incident changes traffic flow parameters, such as speed, flow and density, both upstream and downstream of the incident location. As

shockwaves reach them, the sensors that are upstream and downstream of this incident observe these changes, which are the basis for the algorithm applied in this study to detect and verify incidents. This concept is shown in Figure 4.1 with an example of a density contour map of the studied freeway segment during an incident.

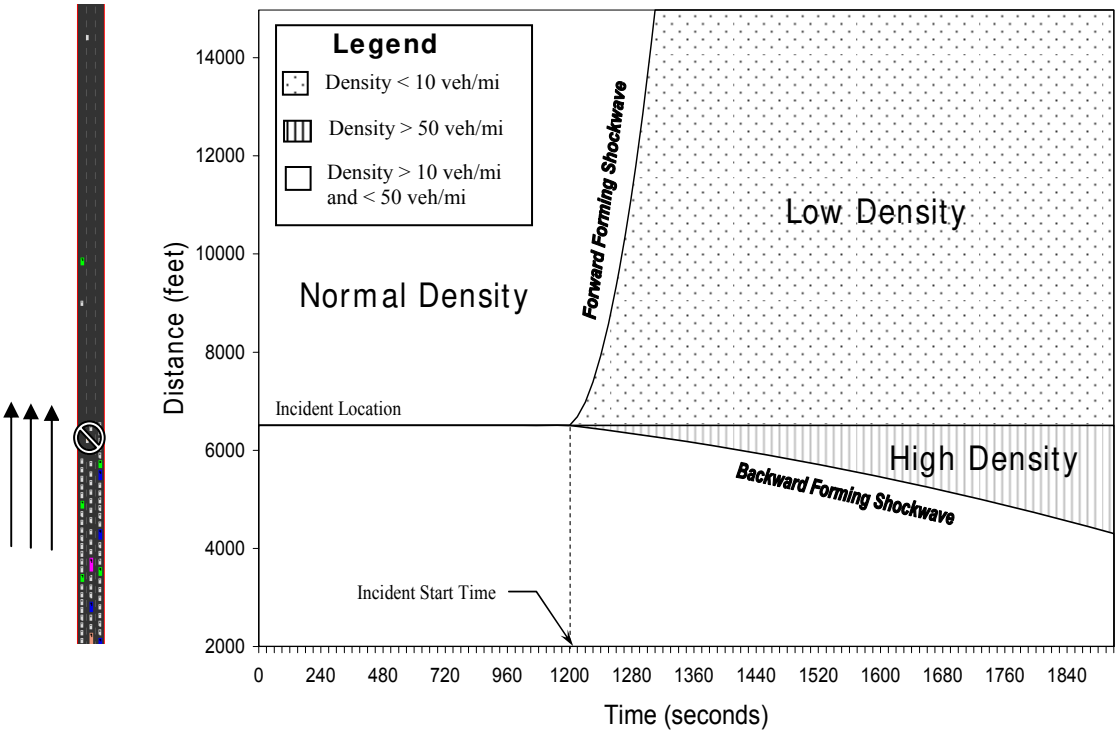


Figure 4.1 Density contour map for the studied freeway network when the incident occurred. An overview of the highway is shown on the left with the incident marked with a "prohibited" symbol.

As shown in Figure 4.1, an incident that blocked all lanes was created in the simulated network 1200 seconds after the simulation began making two shockwaves to begin propagating backwards and forwards at different speeds. The forward shockwave was the boundary between the low-density traffic immediately downstream (of the incident) and the normal density traffic further downstream (of the incident). Similarly,

the backward shockwave is the boundary between the high-density traffic immediately upstream and normal density traffic further upstream. In another word, the forward and backward shockwave represent the boundary of traffic conditions with and without the impacts of the incident at downstream and upstream, respectively. The shockwave propagation curves and the incident location line divided the density contour map into four regions: immediate downstream to incident location low density, further downstream normal density, immediate upstream to incident location high density, and further upstream normal density. In Figure 4.1, the normal density ranged between 10~50 vehicles per mile; below 10 vehicles per mile is considered a low density while over 50 vehicles per mile is considered a high density. The exponential shape of the forward moving shockwave curve indicated that the boundary of low and normal density traffic propagated at a rapidly increasing speed. Meanwhile, the backward moving shockwave propagated at a much slower but also increasing speed. The phenomenon in which the slope appeared to be flat initially and became sharp as time lapsed agreed with the fact that the queue accumulated faster as it went further upstream. In this example, an incident occurred at the location of 6508 feet with immediate downstream and upstream sensors located 546 feet and 855 feet away from the incident location. It took approximately 30 seconds for the forward moving shockwave to travel 855 feet to reach the immediate downstream sensor. The backward moving shockwave required 233 seconds to reach the immediate downstream sensor. Though the downstream sensor was the closer to the incident location, the backward shockwave took longer time to reach a sensor than the

forward shockwave. The backward and forward shockwaves determined the time in which incidents were detected by the upstream and downstream sensors.

4.1.2 Detection Performance

Figure 4.2 presents the time between incident occurrence and notification to the upstream controller versus incident location distance to the upstream controller. Incidents were generated at different locations at a fixed interval of every 500 feet with varied distances to the upstream sensor. In Figure 4.2, the maximum and minimum times at each location are shown with a thin line, while a solid bar is used to indicate the 95 percent confidence bound. This time metric accounts for the time it took the sensors to detect and verify the incident, and the time to notify the controller over the network. This time period generally increases with the upstream sensor's distance from the incident location.

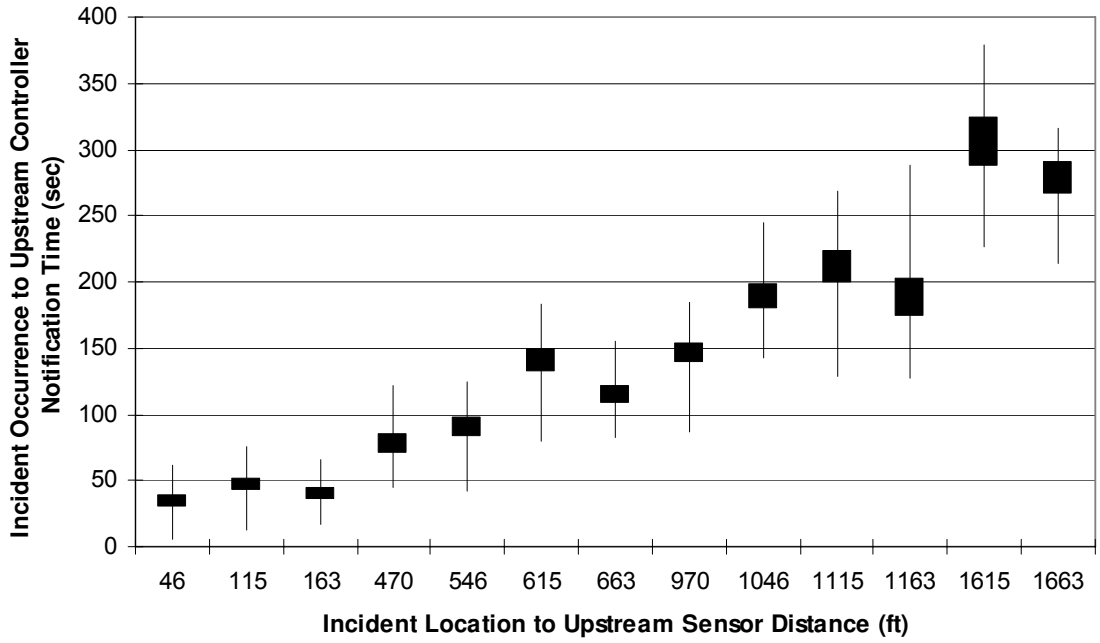


Figure 4.2 Time between incident occurrence and notification to upstream controller versus incident location to upstream sensor distance

The simulation was run for 8 hours, which generated a total of 394 experiments, each of which concludes with the sensor network generated decision on an incident being detected and verified. For each experiment, a warm up period of 10 minutes was used to assure the simulated vehicle flow approach stationary condition before any incidents were generated. Another 10 minutes were simulated to study the false alarm rate before incidents were generated with random start times and random locations along the freeway section after this warm up time. The detection rate was 100 percent. The false alarm rate was measured as the ratio of the number of verified detections to 1) the total number of detection attempts or 2) the total observation time, given that there is not an actual incident. Thus,

False alarm rate = 8 false alarms / (394*10*60/30) (number of decision intervals)

/ 8 (number of sensors)

= 0.0125%

OR

= 8 false alarms / (394 * 10 min)

= 0.12 false alarms per hour

Time-based false alarm rates have been more commonly adopted by incident management agencies.

4.1.3 Communication Metrics

Figure 4.3 shows the detecting-sensor-to-verifying-sensor communication time versus incident location expressed as distance from the downstream controller. The first three readings were due to boundary (between the sensor at the end of a cluster and the sensor at the beginning of the next cluster) effects of the most downstream sensor. Since the sensor had only upstream neighbors, upon incident detection, verifying queries only needed to be sent in one direction, hence halving the communication load and thereby achieving much lower delays. In practice, most sensors must have both upstream and downstream sensors to verify incidents. With all other incident locations, the time metric showed large deviations due to random message transmission latencies using the IEEE 802.11 protocol. Sensors in close range compete with each other for access to the common wireless channel when transmitting packets. As incident detection and verification caused multiple sensors to transmit in a close time, some messages had to delay their transmission due to this congestion.

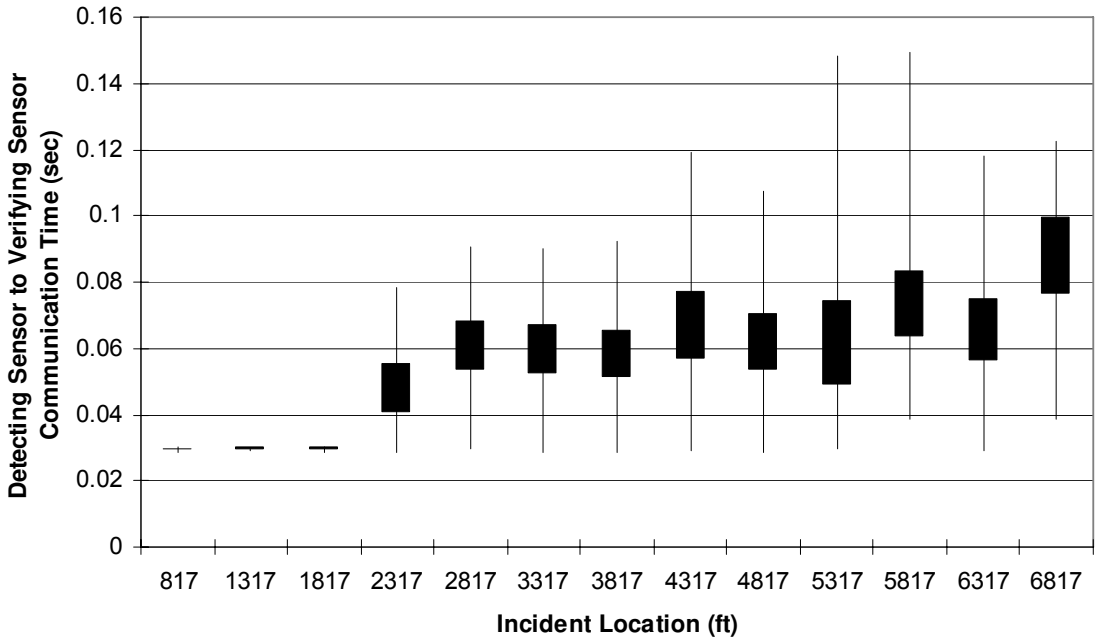


Figure 4.3 Detecting-sensor-to-verifying-sensor communication time versus incident location expressed as distance from the downstream local cluster controller

Figure 4.4 shows the verifying-sensor-to-local-controller communication time versus incident location expressed as distance from the downstream controller. The further the verifying sensor from the local controller, the more time needed for the notification to be relayed to the controller via multiple hops. At four incident locations, the communication times were much higher than at other locations. These higher communication times were due to two sensors detecting the same incident at nearly the same time and simultaneously initiating verification transmissions that resulted in transmission contentions and longer delays. While such events are random, they result in large deviations in the communication time among sensors and controllers. Special cases such as simultaneous detections must be taken into account when designing real-time distributed control methods in such a system.

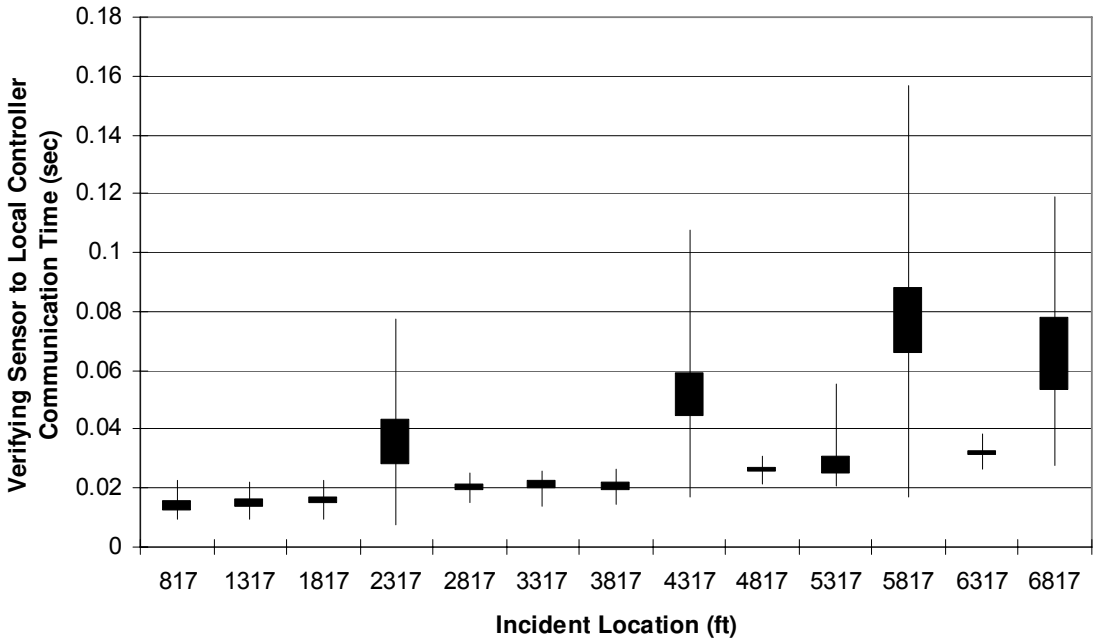


Figure 4.4 Verifying-sensor-to-local-controller communication time versus incident location expressed as distance from the downstream local cluster controller

Figure 4.5 shows the local-controller-to-upstream-controller communication time versus incident location expressed as distance from the downstream local controller. Since the distance between two local controllers was fixed, the communication time was expected to be independent of incident locations. Yet, the results showed a slightly larger delay when an incident (and therefore the detecting sensor) was closer to the local controller. A closer inspection of the results revealed that the cause was due again to transmission conflicts. The closer an incident was to the controller, the earlier the controller would send a notification to the upstream controller, and the higher the chance the message would conflict with the verification messages still being forwarded by nearby sensors.

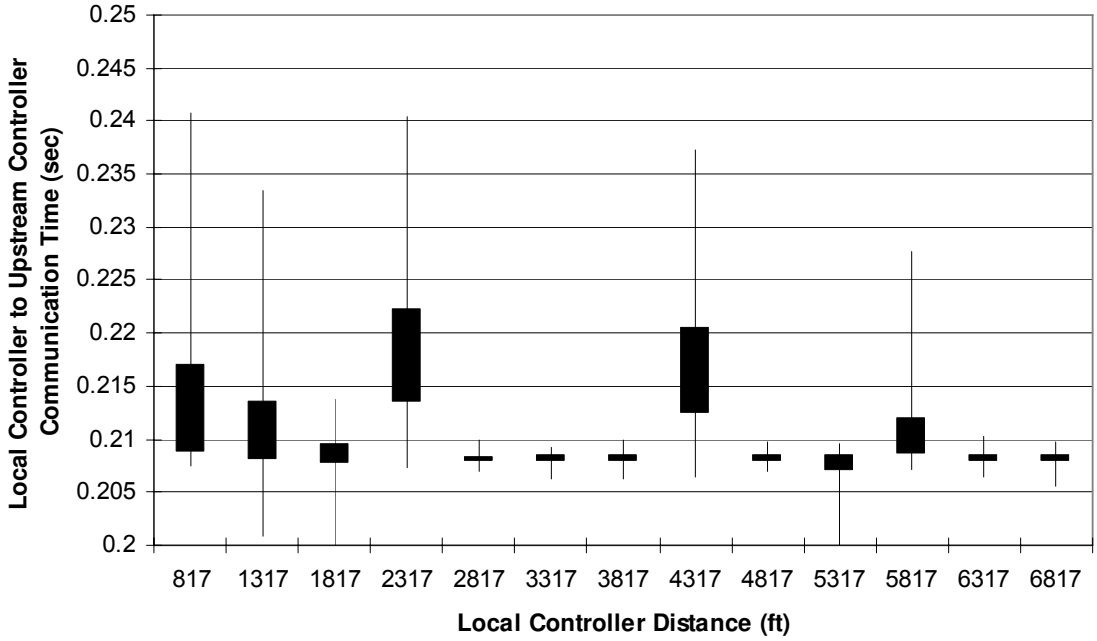


Figure 4.5 Local-controller-to-upstream-controller communication time versus incident location distance expressed as from the downstream local controller

4.1.4 Regression Analysis

A linear regression analysis relating the time between incident occurrence and notification to the upstream controller with the distance between the incident and an upstream sensor using Statistical Analysis Software (SAS) (SAS Institute Inc. 2005) was conducted to illustrate how the simulation results can guide the adjustment of design parameters. As Figure 4.6 shows, the total time that an upstream cluster controller needs to be notified increases linearly as the distance between an incident location and its upstream sensor increases. A comprehensive examination of the results suggested that the propagation speed of incident-generated shockwaves dominated the incident notification time. As the distances between sensors increased, the notification time and its variance increased, indicating that a higher sensor density would effectively enhance the detection

latency and performance predictability. Understanding the extent of communication time variations is essential for determining potential disturbances to the correct execution of the distributed algorithms, which shall become increasingly important as the system scales further increases. In the case study, despite their variations, communication times were tolerable with respect to the overall incident detection time, which depended more on the traffic flow's shockwave propagation speeds.

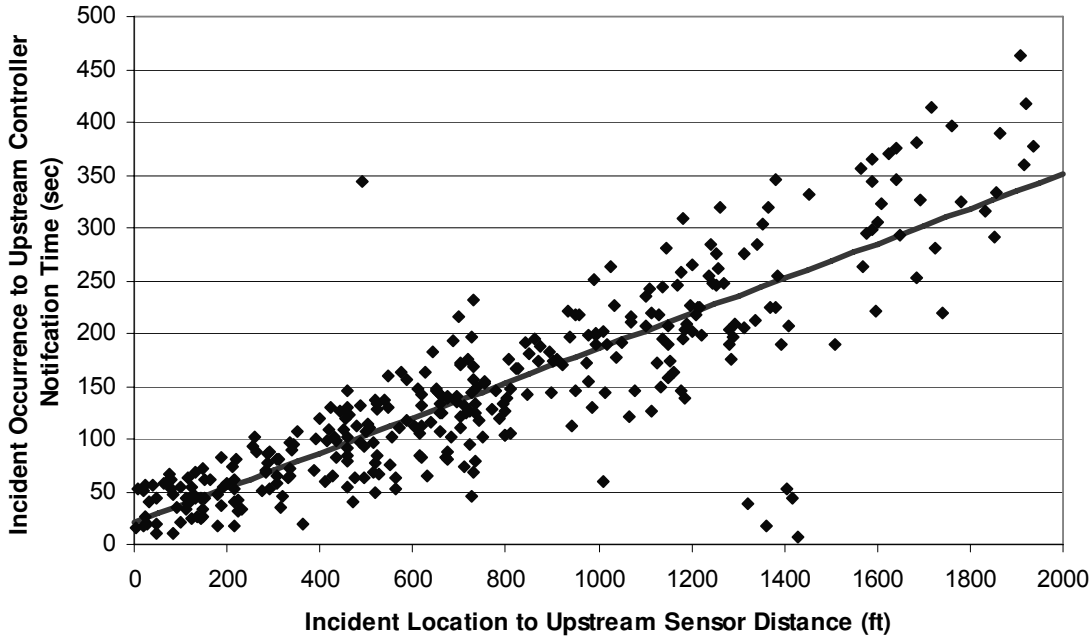


Figure 4.6 Linear regression model relating the time between incident occurrence and notification to upstream controller with the distance between incident location and upstream sensor

4.1.5 Summary of Case Study on Integrated Simulation Platform

The author developed an integrated traffic and networking simulation platform to facilitate the design and evaluation of online traffic surveillance system. As a case study, a reference design for a distributed incident detection and response system using a

wireless traffic sensor network was developed with a hierarchical network architecture, with its simulation model implemented in the integrated simulator. The detection rate and false alarm rate were assessed for a distributed detection algorithm based on a traffic shockwave theory and distributed network collaborations, detecting and verifying the presence of shockwaves caused by incidents. While simultaneous detections caused unforeseen communication latency, the communication times were found to be tolerable for the case study. Communication latency, ordering, and reliability will become more crucial once a larger system is in place. Statistical dependency of detection performance on sensor placement was evaluated, showing opportunities and direction for improvement.

The integrated simulation platform presented in this study provided a valuable tool to facilitate a detailed, objective and efficient evaluation of automatic incident detection system. In addition, it can be applied to evaluate a wide range of online traffic condition assessment and prediction tools involving complex traffic control strategies and communication requirements, such as vehicle-to-vehicle and vehicle-to-infrastructure communications as will be presented in Sections 4.3 and 4.4.

4.2 Evaluation of Communication Alternatives

This study sought to test the hypothesis that distributed communication architecture provides better communication efficiency. The author selected important measures of effectiveness (MOE) for making objective comparisons between alternatives based on the performance of the communication systems related to real-time traffic condition assessment.

A case study was performed and presented in the following sections for a test network in Greenville, South Carolina. Four communication alternatives, namely the centralized-wired, distributed-wired, centralized-wireless and distributed-wireless, supporting the real-time traffic surveillance system that generates data traffic with constant bit rate were modeled using the previously developed integrated simulation platform to generate the selected MOEs, such as throughput, delivery ratio, and throughput cost ratios, for comparing and analyzing these alternatives.

4.2.1 Capacity of Communication Alternative for Traffic Surveillance System

The following analyses were done by varying the data rates generated by traffic surveillance system to examine the capacity of the four alternatives. Figure 4.7 through Figure 4.10 presents the performance measure of throughput and delivery ratio for the communication alternatives with different architecture and medium.

Figure 4.7 shows the throughput and delivery ratio of the wired centralized network at various data rates. The simulated centralized network ensures each camera with a bandwidth of 1.544MHz no matter how many cameras deployed per mile. The capacity of the T1 centralized network is 1546Kbps. The delivery ratios for data rates under the capacity were 100%. As the data rates approach the capacity, an increasing number of packets were dropped. The delivery ratio decreased dramatically and the throughput remained at 1546Kbps.

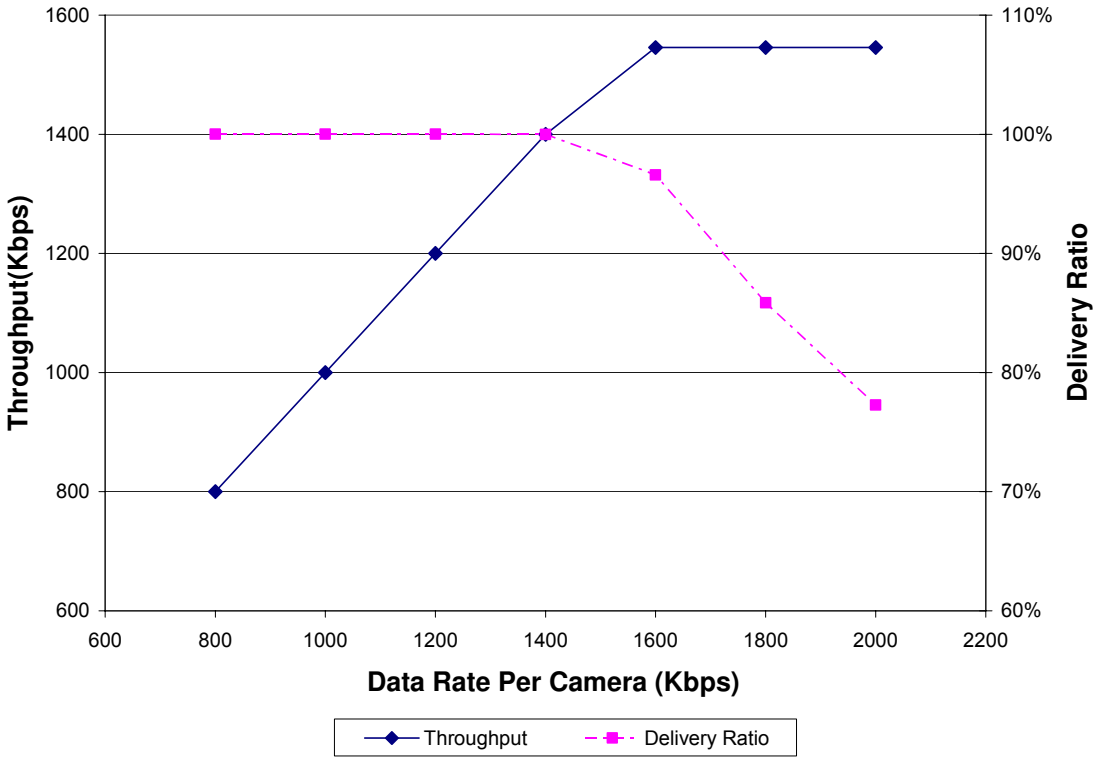


Figure 4.7 Throughput and delivery ratio of the wired centralized network

Figure 4.8 shows the throughput and delivery ratio of the wired distributed network at various data rates. For traffic surveillance system with a density of 1 camera per 1.5 miles, the T1 distributed network was able achieve a maximum throughput of 512Kbps at the capacity data rate of 512Kbps. When the data rate was over the capacity, the delivery ratio decreased from 100% to 43% with data rate of 512Kbps and 1216Kbps, respectively, and the throughput slightly increased up to 520Kpbs when the data rate is over 960Kbps. Similarly, the wired distributed network reached maximum throughput at capacity data rate of 256Kbps when the camera density increased to 1 camera per 0.6 mile.

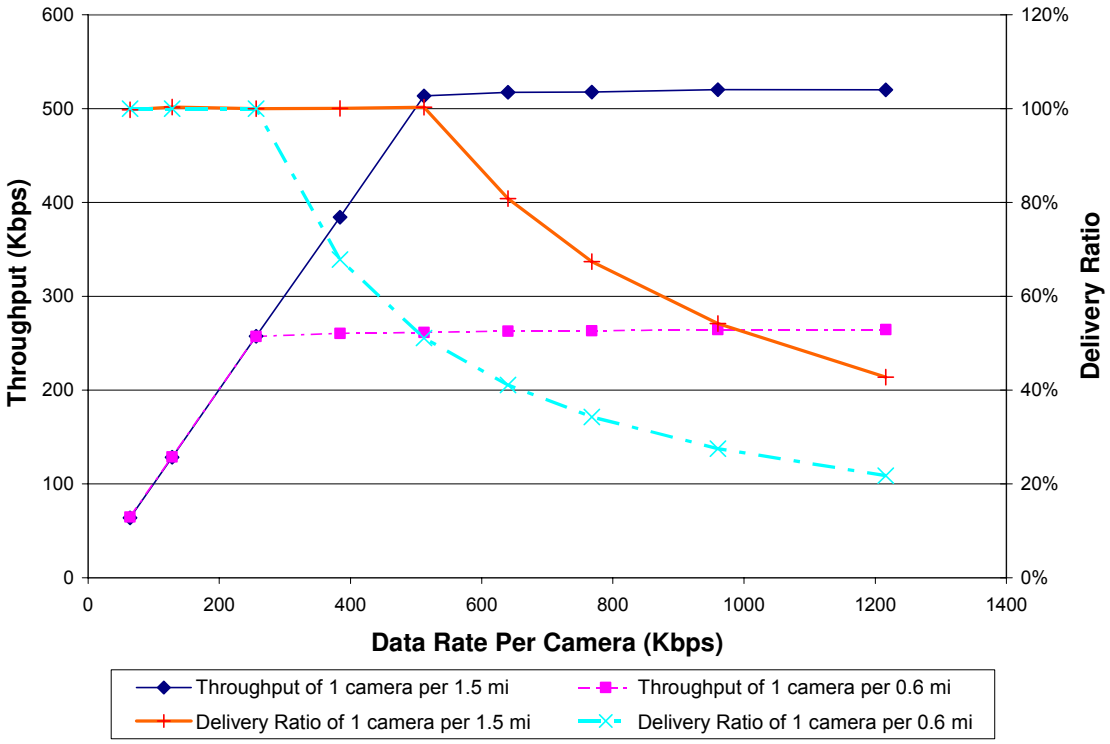


Figure 4.8 Throughput and delivery ratio of wired distributed network

As shown in Figure 4.9, the throughput and delivery ratio of the wireless centralized network revealed a trend similar to that of the wired centralized network, but achieved a smaller maximum throughput of 1252Kbps due to its assumed 1.25MHz link bandwidth.

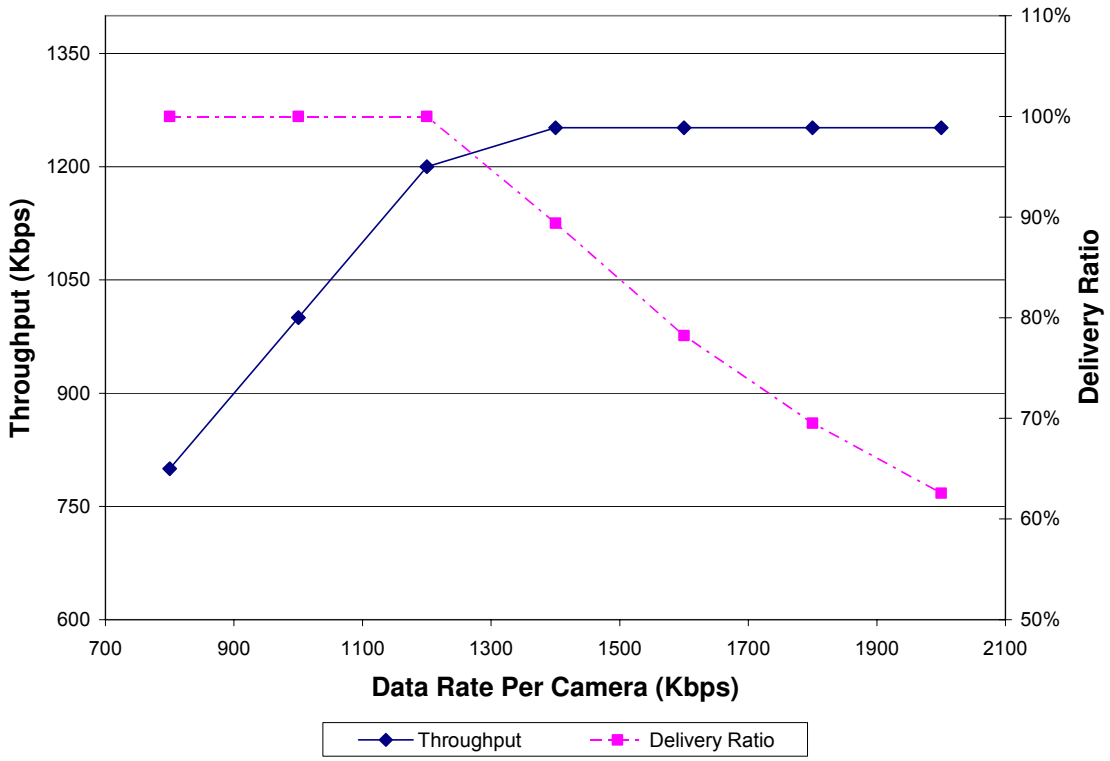


Figure 4.9 Throughput and delivery ratio of wireless centralized network

Figure 4.10 shows the throughput and delivery ratio of the wireless distributed network at various data rates. For a traffic surveillance system with a density of 1 camera per 1.5 miles, though the system was able to achieve a maximum throughput of 435Kbps, the delivery ratio began to drop from 100% when the data rate was 384Kbps. Therefore, when the rate is over 384Kbps, the surveillance system might encounter considerable delay and jitter effects. Therefore, the capacity rate for this scenario was 384Kbps. When the camera density increased to 1 camera per 0.6 mile, the system could only support 128Kbps data rate, which is lower than the typical full motion video data rate of typical traffic cameras. Increasing the wireless link bandwidth from 11MHz to 54MHz, which is compatible with the IEEE 802.11g standard, can possibly enhance the capacity. Another solution is to partition the sensors into smaller groups with each group communicating

with a different radio channel, such that bandwidth contention within each group is reduced. A feature of the wireless distributed system that differs from the other alternatives is its decrease in throughput upon reaching a peak value due to the increased random transmission collision at high per-device transmission rates.

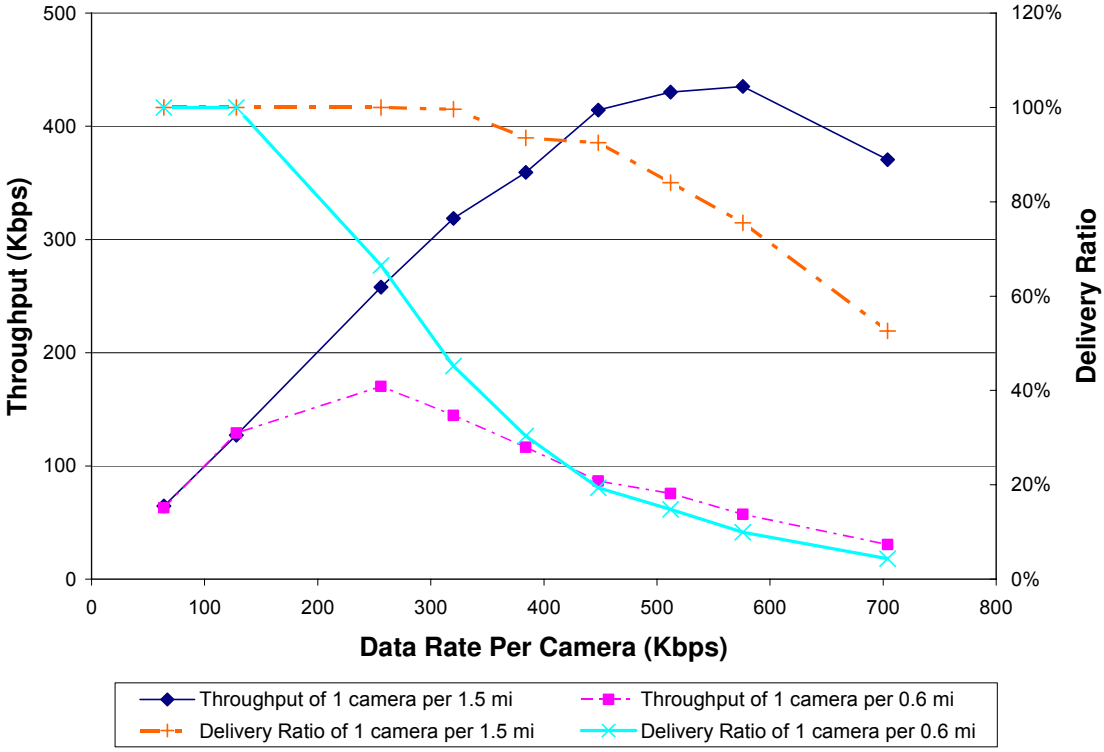


Figure 4.10 Throughput and delivery ratio of a wireless distributed network

4.2.2 Cost Effectiveness of Communication Alternatives

In order to assess the cost effectiveness of different communication alternatives, the author selected and examined the throughput to communication infrastructure deployment cost ratio as the performance measures.

For cost analysis, installation and operation costs for the different communication infrastructures have been estimated with best effort according to vendor advertisements

and are summarized in Table 4.1. The cost of various resources including FHWA ITS online database (USDOT 2007), communication device vendors and communication service carriers, were converted into 2007 dollars using an inflation rate of 3%. In order to calculate the throughput cost ratio, the costs were further converted to dollar per operation second based on the life cycle of devices and operation schedule of a traffic surveillance system.

Table 4.1 Cost Estimate in 2007 Dollar of the Communication Infrastructure

	Element	Life-time (years)	Unit Cost		Installation Cost		O&M Cost ($/year)	
			Low	High	Low	High		
Wired Distributed/ Centralized	Fiber Optical Cable	20	172/100ft		5000	15000	1000	2000
	Transmitter	10	1000	4000	150	250		
	Optical Regenerator	10	Optional		Optional			
	Optical Receiver	10	800	1200	150	250		
Wireless Distributed	Wireless Access	20	400	1000	200	900	1000	
Wireless Centralized	Monthly Service		60	140				
	Modem	10	200	400	50	150		

The throughput to cost ratios of four communication alternatives are computed and presented in Figure 4.11. The wired alternatives were less cost effective than the wireless ones; particularly for a camera density of one camera per one and a half miles, the wireless centralized and distributed has comparable cost effectiveness for camera rates up to 480 Kbps, where the distributed throughput saturates. With the centralized solution, per-camera rates can be as high as 1400 Kbps. The wireless distributed network, however, increases its cost-effectiveness as the density of supported devices increases. The per-camera capacity reaches the inflection point at a capacity of 480 Kbps.

Compared to the wireless alternatives, the per-camera rates in wired-centralized networks can be 1700Kbps, while the same rates reach the maximum capacity of 512 Kbps in wired distributed networks.

Figure 4.11 Throughput to cost ratio of different network architectures

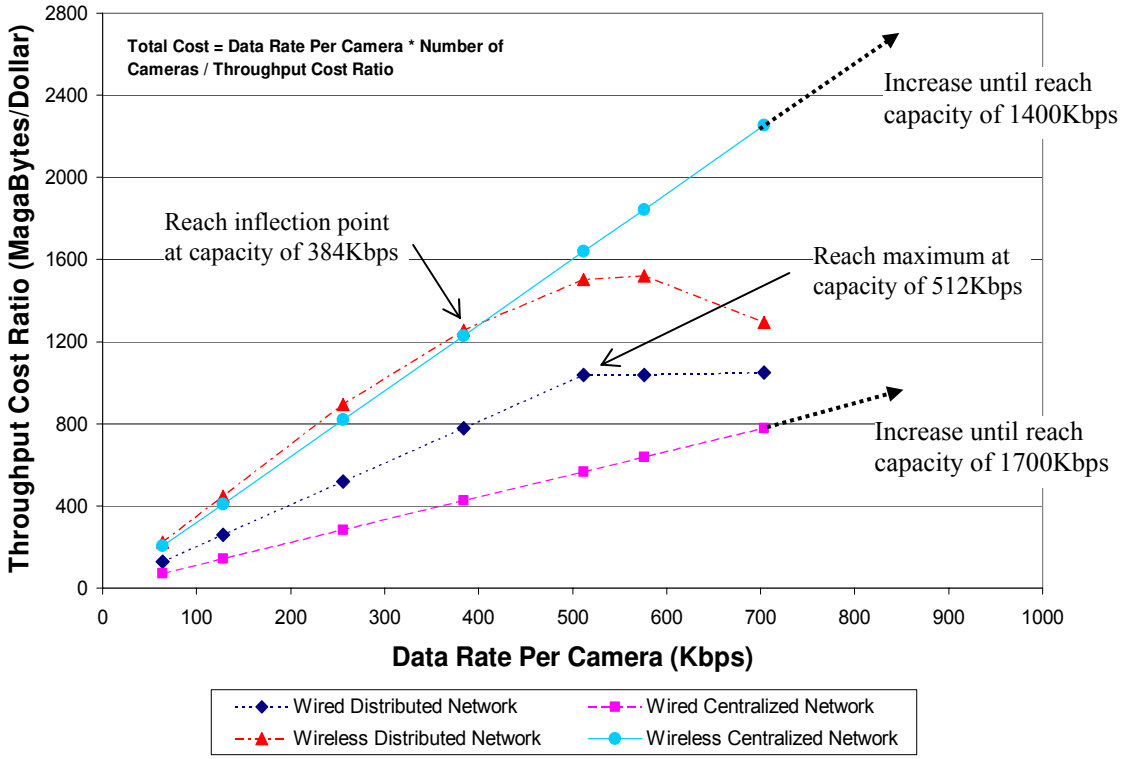


Figure 4.12 shows the throughput-to-cost ratio for different camera densities for the wireless distributed system. The ratio increases substantially when the density increases, and the optimal density depends upon the desired camera data rate. If the expected data rate is low while the density is high, a wireless distributed solution is economically more preferable. The throughput-to-cost ratio also increases if higher rates with IEEE 802.11g are achievable in the field.

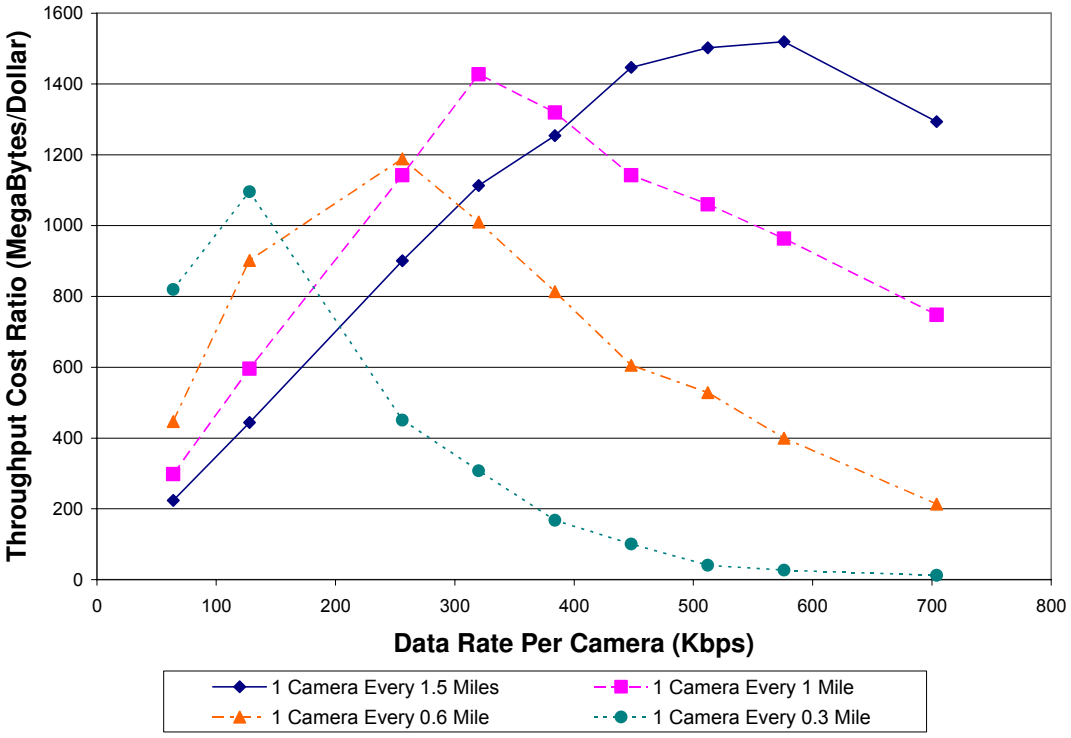


Figure 4.12 Throughput to cost ratio of wireless-distributed network

4.2.3 Efficiency of Communication Alternatives during Incidents

For the study scenario with incidents, the vehicular traffic simulator generated incidents with random occurrence times, locations and durations on the segments under surveillance of traffic cameras during the AM peak period through PARAMICS Programmer's API interface. The duration of incidents directly affects the communication cost in terms of data rate, which can be altered by the ns-2 during the simulation. In a centralized system, each device continuously generates constant rate data at the rate of 384Kbps no matter there is an incident or not. On the contrary, in a distributed system, the devices send stationary images with a consequent date rate of 24Kbps to the controlling center at a low frequency during the normal condition. Once an

incident is identified or suspected, the corresponding traffic camera transmits full motion videos with a data rate 384Kbps to control center. Within the two-hour simulation period, throughputs of centralized and distributed system for various incident durations were examined to compare their communication costs.

As evident in Figure 4.13, the centralized and distributed wireless systems revealed significant differences in throughput during the simulated peak vehicular flow period. The solid line that represents the centralized system maintains a high throughput of 384Kbps throughout the entire period. Conversely, the dashed line indicates that the average the throughput of distributed network increases linearly as the incident duration increases. The reason is because the incident duration determines the portion of time that needs high data rate. While Figure 4.13 only displays the results for wireless distributed and centralized alternatives, similar results were found when analyzing the throughput difference between wired distributed and centralized alternatives with the incidents occurred.

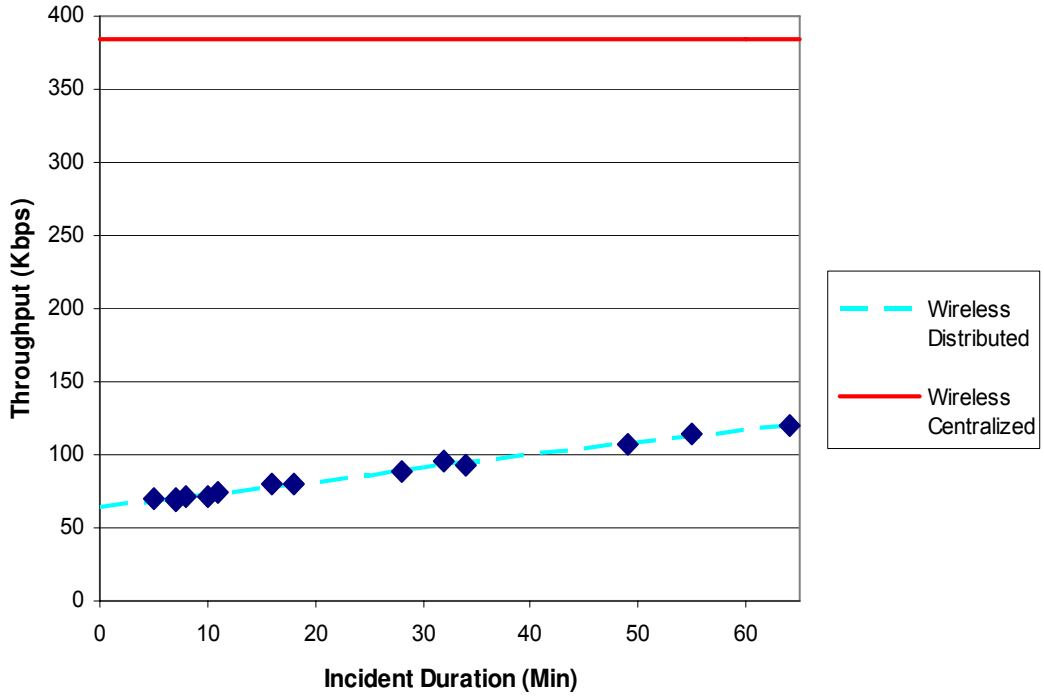


Figure 4.13 Throughput of centralized and distributed networks during an incident

4.2.4 Summary on Evaluation of Communication Alternatives

The presented study attempted to establish the advantage of distributed architecture in terms of communication efficiency by evaluating the performance of different communication alternatives under different traffic demands and conditions. The results showed that the wired alternatives are less cost effective than the wireless ones in both centralized and distributed communication topologies. For the particular camera density (1 camera per 1.5 miles), the wireless centralized and distributed alternative has comparable cost effectiveness before per camera rate goes up to saturation throughput. The wireless-distributed network, however, increases its cost-effectiveness as the density of supported devices increases. Throughput-to-cost ratio for different camera densities

increases substantially when the density increases, and the optimal density depends upon the desired data rate. Therefore, a wireless-distributed solution will be economically more preferable if the expected data rate is moderate while the camera density is high. In addition, the distributed architecture outperformed the centralized architecture in terms of savings in communication costs as minimum amount of data was transferred during normal vehicular traffic condition.

4.3 Traffic Condition Assessment Framework

The author developed a VII simulation model with a SVM-based intelligent algorithm for incident detection. Using the individual vehicle dynamics measured by each VII-enabled vehicle, the intelligent incident detection algorithm was tested for its ability to identify the occurrence, locations and severity of a highway incident. As a case study, the following sections present the implementation details and evaluation results of the proposed VII model on a calibrated and validated simulation network in Spartanburg, South Carolina. The RSUs were placed at every interchange with many wireless repeater placed between them to forward messages between vehicles and infrastructure devices, as well as between infrastructure devices and infrastructure devices. The intelligent algorithm SVM was implemented in each VII-enabled vehicle to classify traffic conditions. The RSU detected incidents of the highway segments by collecting and assessing the traffic condition estimations from individual vehicles.

The selected MOEs for incident detection capabilities include detection and false alarm rate, detection time, accuracy of prediction on incident locations and number of

lanes blocked by incidents. The communication metrics, such as latency and delivery ratio, of the VII model were also examined and presented.

4.3.1 Parameter Adjustments for the SVM Algorithm

An important step in developing an SVM algorithm involves determining the optimal parameters for the algorithm. Figure 4.14 shows the grid searching efforts for optimal parameters (cost coefficient C and kernel function parameter γ) in a range of $C=2^{-5}\sim2^{15}$ and $\gamma=2^{-15} \sim 2^{3}$ with a contour map. Each contour line represents a specific combination of C and γ that produces the same prediction accuracy in percentage (shown as numbers in each contour line). The contours were used to identify the parameter combination that yielded the highest prediction accuracy. As shown in Figure 4.14, when [C, γ] falls within a triangle area, a greater than 95% prediction accuracy rate is achieved. The program determined that the optimal parameter was in the $C=2^{5}$ and $\gamma = 2^{-1}$, which gave a cross validation prediction accuracy rate of 98.04%.

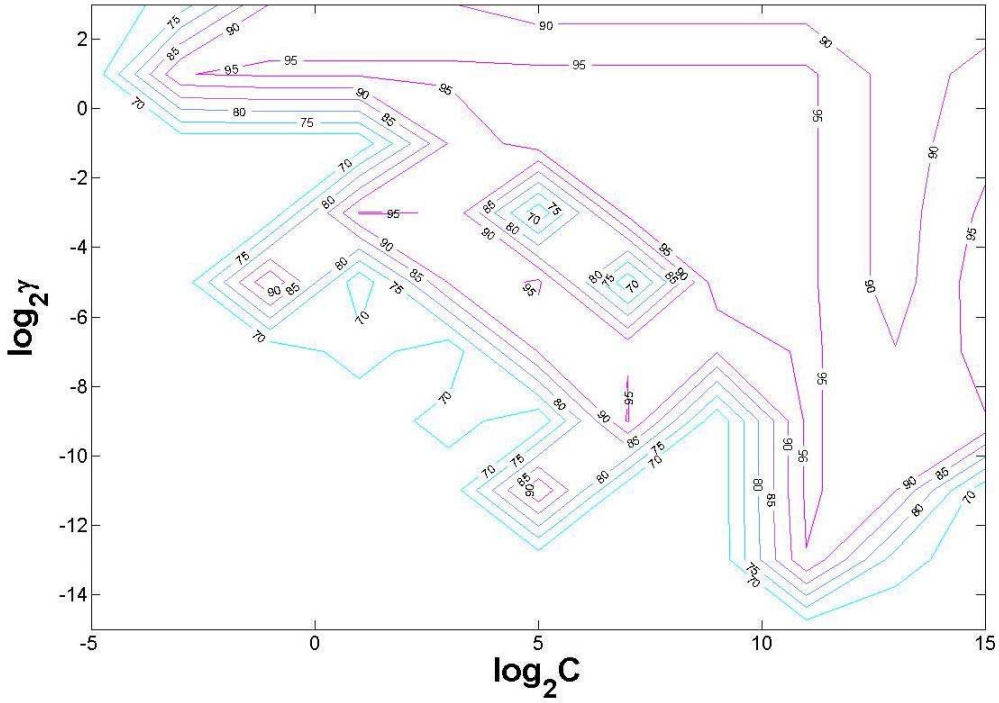


Figure 4.14 Prediction accuracy contour of parameters combination for developed SVM algorithm

The authors performed a sensitivity analysis between the threshold number of alarms from vehicles (i_t) and the maximum accumulation time (t_{max}) in relation to the detection rate (D_rate) and the false alarm rate (F_rate in terms of false alarms per hour) for incidents blocking one lane. As Table 4.2 shows, i_t and t_{max} affect the incident detection performance of the proposed VII system. Lower selected threshold number of alarms and longer maximum accumulation times will result in increase in both detection rates and false alarm rates. The detection rate for percentage of VII-enabled vehicles over 15% is not affected by changes in i_t and $t_{max,}$ while the false alarm rate might increase significantly. The sensitivity analysis indicated that i_t = 3 and t_{max} = 3 min can achieve a reasonable tradeoff between detection rate, and false alarm rate.

Table 4.2 Sensitivity Analysis of Threshold Number of Alarms by Vehicles and Maximum Accumulation Time by Infrastructure Agents

i_t	t_{max} (min)		Penetration Rate					
			5%	10%	15%	20%	25%	30%
2	3	F_rate	0.06	0.12	0.6	0.84	0.9	1.2
		D_rate	86%	99%	100%	100%	100%	100%
2	5	F_rate	0.06	0.18	0.72	0.9	1.02	1.32
		D_rate	87%	100%	100%	100%	100%	100%
3	3	F_rate	0	0.06	0	0.18	0.3	0.3
		D_rate	72%	96%	100%	100%	100%	100%
3	5	F_rate	0	0.06	0.18	0.18	0.3	0.36
		D_rate	73%	96%	100%	100%	100%	100%
4	3	F_rate	0	0.06	0	0.06	0.06	0.06
		D_rate	53%	93%	100%	100%	100%	100%
4	5	F_rate	0	0.06	0	0.06	0.18	0.12
		D_rate	62%	93%	100%	100%	100%	100%

4.3.2 Incident Detection Performance the VII Model with SVM Algorithm

4.3.2.1 Comparison of the SVM Algorithm and California Algorithm

The authors applied a well known incident detection algorithm known as California Algorithm #7 (Payne and Tignor 1978) in the same test network as used in the development of the SVM algorithm. The incident detection performance of these two incident algorithms were compared in the following analysis.

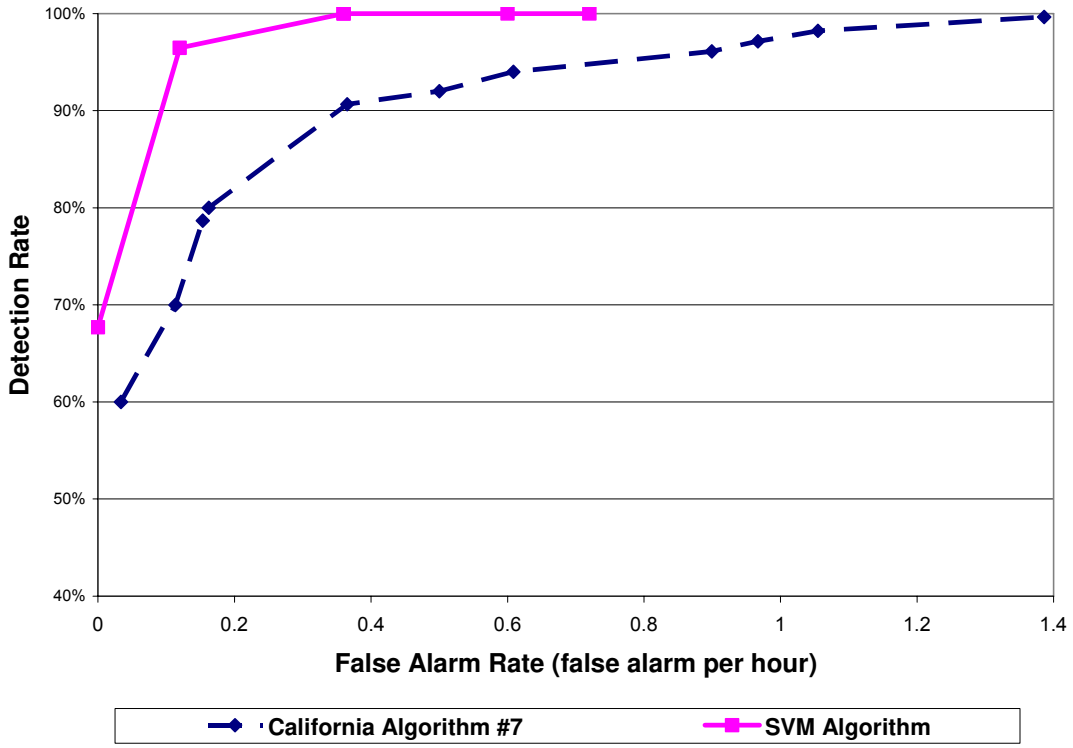


Figure 4.15 Comparison of California and SVM Algorithm for detection rate and false alarm rate

As shown in Figure 4.15, the SVM algorithm is superior over the California #7 in terms of detection rate and false alarm rate under identical traffic conditions. The developed SVM achieved a 100% detection rate at very low false alarm rate, while the California algorithm approached 100% detection rate with the cost of substantial increases for the high false alarm rate.

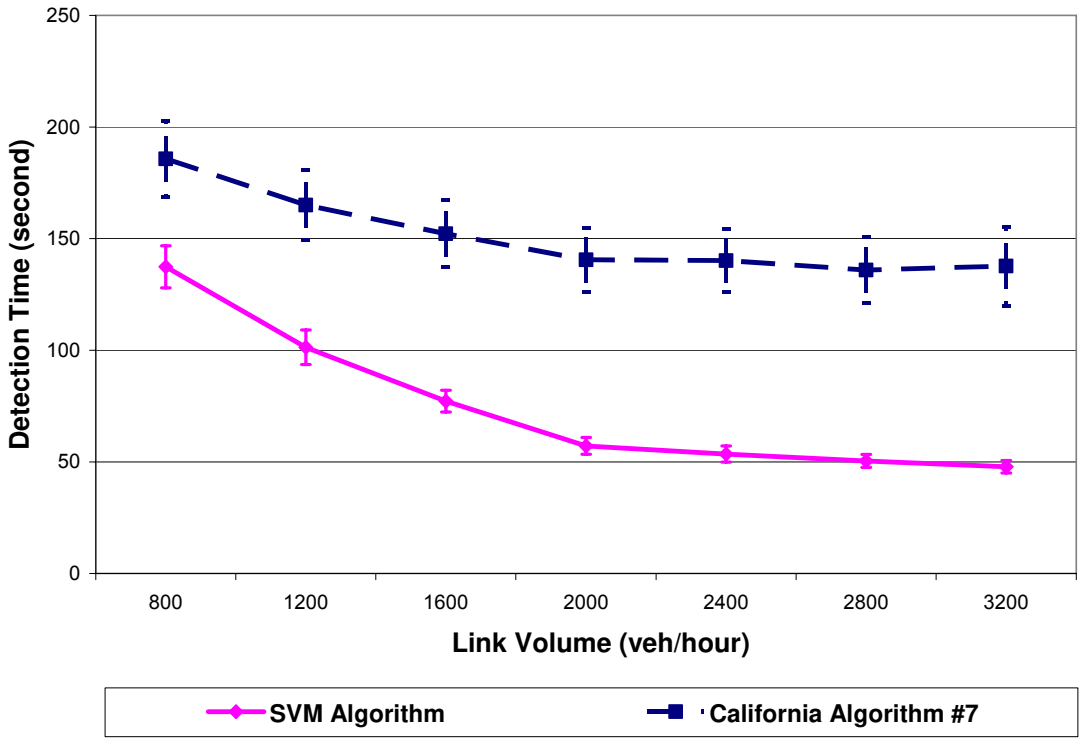


Figure 4.16 Comparison of California algorithm and SVM algorithms for detection time

Figure 4.16 presents the detection time of the California algorithm and SVM algorithm under various traffic volumes. The detection time of both the California and SVM algorithm decrease as link volume increases, but the effects diminish when the link volume is over 2400 veh/hour. The detection times of SVM algorithm are much less than that of the California algorithm under all traffic conditions. Note that the SVM algorithm assumed a 20% penetration rate. The California algorithm used the parameter sets that can achieve a 92% detection rate and a 0.5 false alarm per hour for calibration data sets. The actual detection rate varies between 80% and 92%, while the false alarm rate varies in a range between 0.12 and 0.6 false alarm per hour. The traffic detector density is

approximately 1 detector every quarter mile. As the detector density decreases, the detection time is expected to increase.

4.3.2.2 Incident Detection Rate and False Alarm Rate of the VII Model

As is evident in Table 4.3, the results on detection rates and false alarm rates are encouraging. Even with a penetration rate is as low as 5%, the SVM incident detection algorithm can achieve a detection rate between 75% and 100% depending on the number of lanes blocked by incidents. When the penetration rate is above 15%, almost all incidents were detected by the VII incident detection system. The false alarm rate slightly increased as the penetration rate increased but was still within the acceptable range, which were 10 false alarms per hour reported by a nation wide survey of the real time traffic management agencies (Martin et al. 2001). However, the false alarm rate will increase as the network size increases. The detection rate was the percentage of incidents correctly detected by the proposed VII system over the total number of incidents that occurred. The false alarm rate was the number of false alarms per hour reported by the system in which no actual incidents occurred.

Table 4.3 Detection Rate and False Alarm Rate of the VII Model

		Penetration Rate					
		5%	10%	15%	20%	25%	30%
Detection Rate	incidents blocking one lane	75%	89%	99%	100%	100%	100%
	incidents blocking two lanes	98%	99%	100%	100%	100%	100%
	incidents blocking three lanes	100%	100%	100%	100%	100%	100%
False Alarm Rate (false alarms per hour)		0.00	0.00	0.00	0.00	0.00	0.10

Table 4.4 shows the detection rate and the false alarm rate under for the SVM incident detection algorithm with 20% VII-enabled vehicles under various traffic volumes. When the penetration rate is as high as 20%, the detection rate maintains 100% for any vehicular traffic volume except 800 vehicles per hour. However, as the link volume increased to over 2400 vehicles per hour, the false alarm rate increased considerably. This increase was due to the fact that the incident identification mechanism implemented in the RSU was designed and tuned up for the moderate to low traffic volumes. As the traffic volume increases, the parameter sets should be adjusted accordingly. The two parameters, which have a significant effect on the detection performance, are the threshold number of alarms (i_t) from vehicles to signal an incident and the number of VII-enabled vehicles whose data are used by RSUs to assess traffic conditions.

Table 4.4 Detection Rate and False Alarm Rate of the VII Model with 20% VII-enabled Vehicles for Different Traffic Volumes

		Link Volume (veh/hr)						
		800	1200	1600	2000	2400	2800	3200
Detection Rate	incidents blocking one lane	93%	100%	100%	100%	100%	100%	100%
	incidents blocking two lanes	100%	100%	100%	100%	100%	100%	100%
	incidents blocking three lanes	100%	100%	100%	100%	100%	100%	100%
False Alarm Rate (false alarms per hour)		0.00	0.00	0.00	0.00	0.10	0.50	0.85

4.3.2.3 Incident Detection Time of the VII Model

Figure 4.17 presents the detection time for different penetration rates of VII-enabled vehicles to detect incidents blocking one, two and three lanes. The boxes in Figure 4.17 indicate the mean detection time and the upper and lower limit of the bars show the range of expected detection with 95% confidence level. More severe incidents, in terms of greater number of lanes blocked, will be detected faster as they quickly affect more traveling vehicles. Thus, there is a higher chance for VII-enabled vehicles to detect them. Detection time decreases as penetration rate increases, but the extra benefits diminish, as the penetration rate is larger than 25%. When the penetration rate is as low as 15%, the detection time of the proposed VII system is comparable or superior to most existing AID algorithms.

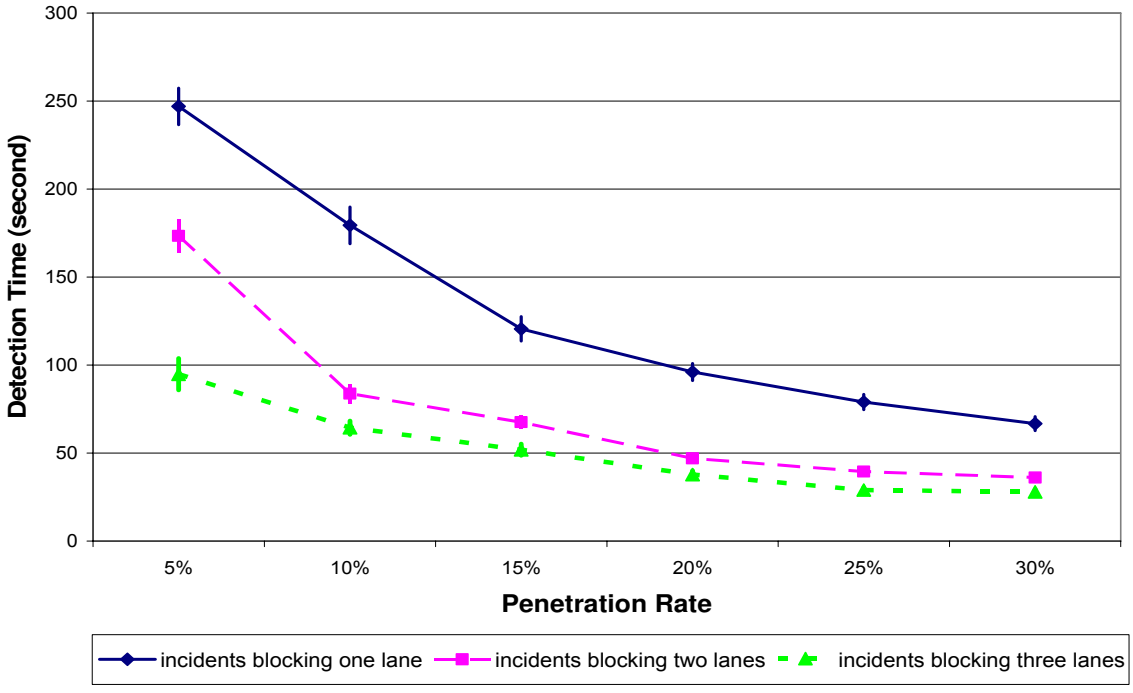


Figure 4.17 Incident detection time of the VII Model with various penetration rates of VII-enabled vehicles

Figure 4.18 shows the detection time of the SVM incident detection algorithm for different traffic volumes in the study segment with a 20% VII-enabled vehicle on the link. The peak hour traffic volume on the study segment was 1600 vehicles per hour, so the authors varied this value within a range of -50% to +200% to represent dynamic characteristics of a traffic network. As shown in Figure 4.18, an increase in traffic volume will have a positive impact on the detection time with the increase in the number of VII-enabled vehicles. However, after the traffic volume increases to a threshold level for incidents blocking one, two and three lanes, the detection times do not differ significantly.

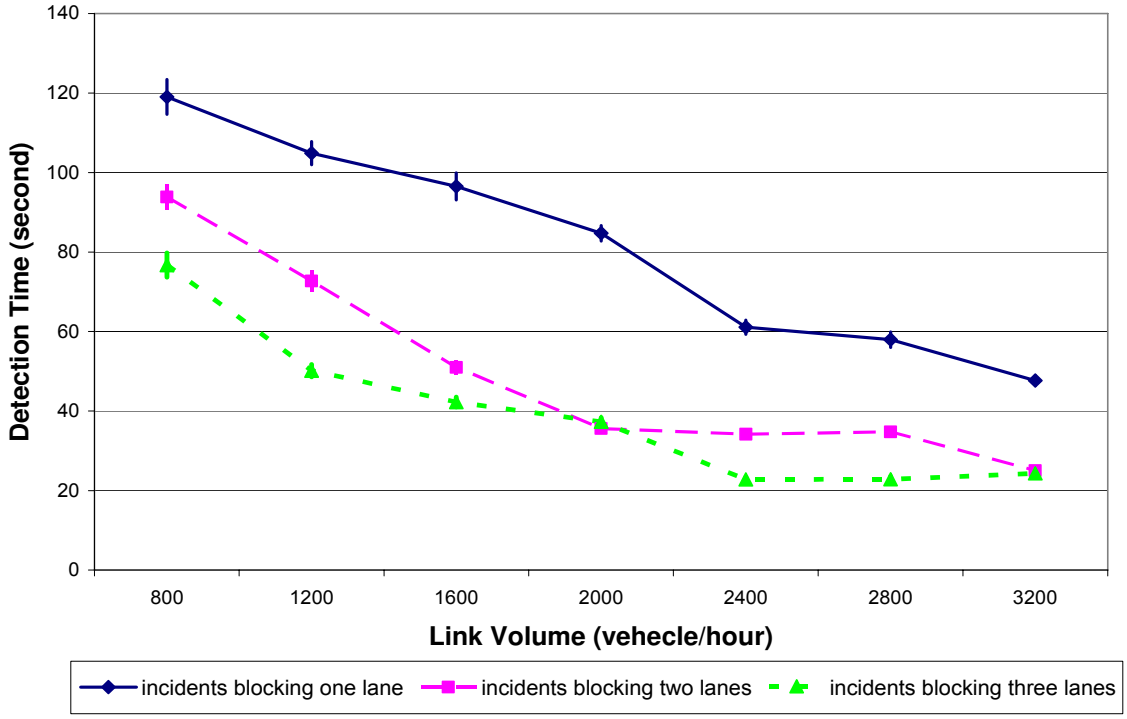


Figure 4.18 Incident Detection Time of the VII Model with 20% VII-enabled Vehicles for Different Traffic Volumes

4.3.2.4 Prediction on Number of Lanes Blocked

As explained in section 3.3.3, the proposed VII model developed the functionality of predicting the number of blocked lanes due to incidents. Figure 4.19 shows the accuracy of such predictions on the number of lanes blocked by incidents. The prediction accuracy for incidents blocking two or three lanes increased as the percentage of VII-enabled vehicles increased. On the other hand, the prediction accuracy for incidents blocking one lane increased as the penetration rate increased from 5% to 10%, and the accuracy kept decreasing as the penetration further increased.

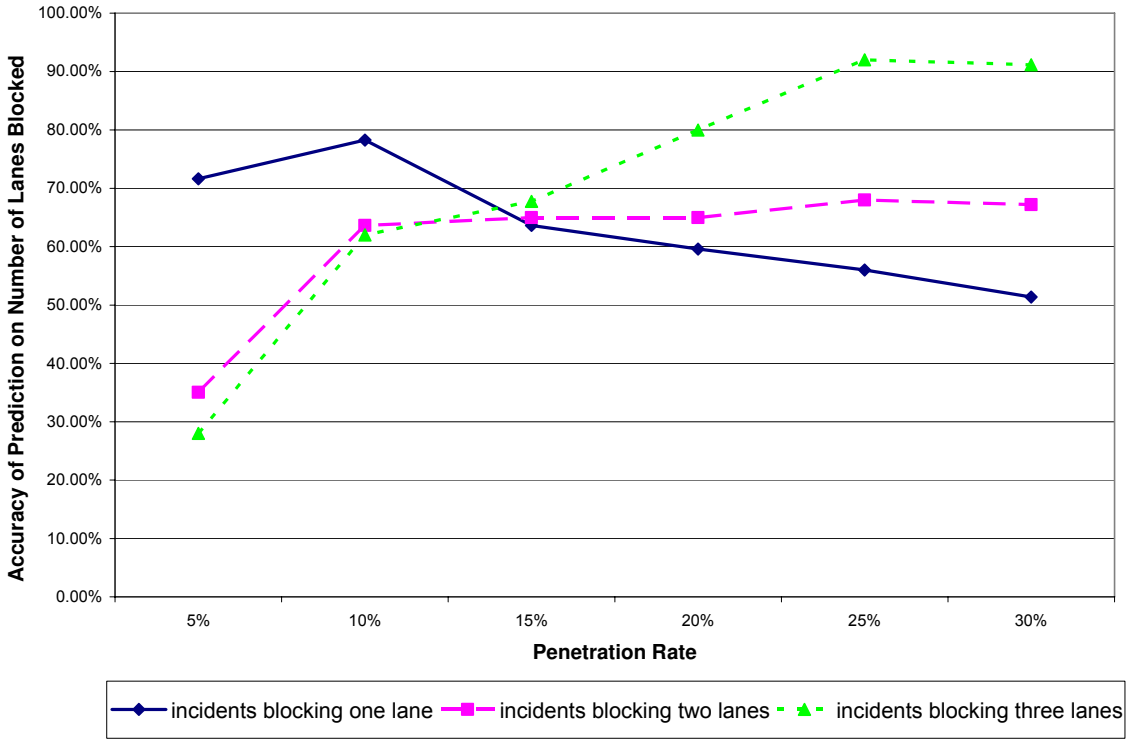


Figure 4.19 Prediction accuracy on number of lanes blocked of the VII model with various penetration rates of VII-enabled vehicles

Figure 4.20 shows an example of the distribution of prediction on number of lanes blocked for VII system with 15% VII-enabled vehicles. The different pattern in each vertical bar represents the percentage of predicted specific number of lanes blocked by incidents over the total number of that type of incidents occurred. For example, the left bar shows that for the incidents actually blocking one lane, around 63% are correctly predicted as such, while 30% and 7% are wrongly predicted as incidents blocking two or three lanes.

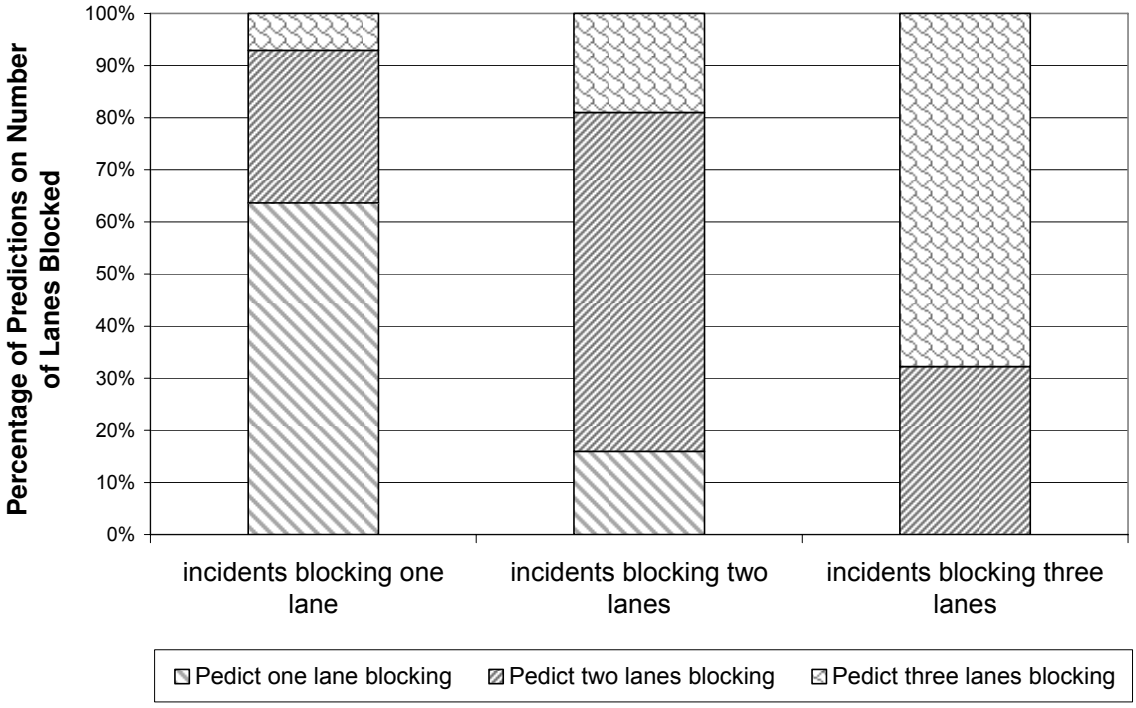


Figure 4.20 Distribution of prediction on number of lanes blocked of the VII model with 15% VII-enabled vehicles

Figure 4.21 shows the prediction accuracy of the number of lanes blocked for different traffic volumes with 20% of the vehicles in the network VII-enabled. The lane blockage prediction did not work well. As the link volume is over 3200 vehicle/hour, the

prediction accuracy for incidents blocking one or two lanes dropped to under 30%. The model predicting lane blockage accurately for incidents blocking three lanes as the model was bias due to overestimation of the number of lanes blocked. Therefore, if the model predicts that one lane is blocked by an incident, it is very likely to be a minor incident blocking one lane.

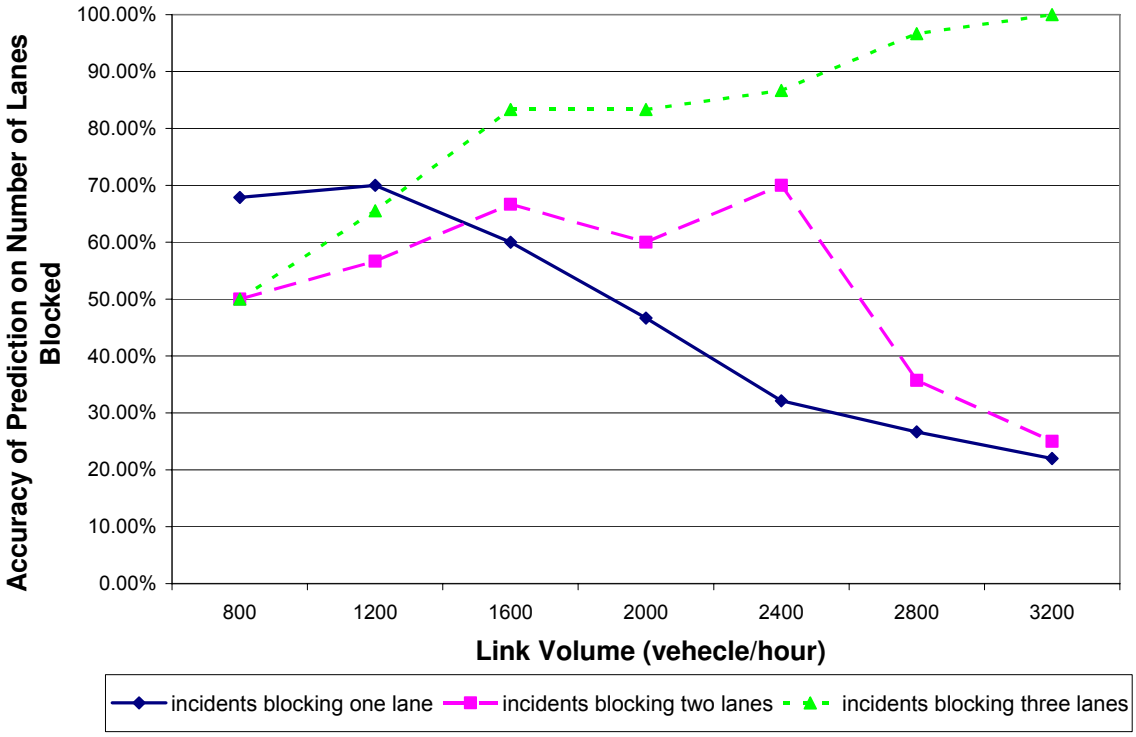


Figure 4.21 Prediction accuracy on number of lanes blocked of the VII model with 20% VII-enabled vehicles for different traffic volumes

4.3.2.5 Prediction on Incident Location the VII Model

As shown in Figure 4.22, most predicted incident locations were within 1000 feet of the actual incident sites. There were more cases of the predicted locations downstream of the actual location because many vehicles were not able to detect the incident prior to

passing the incident scene. Note that the VII model predicted the incident location based on the locations where VII-enabled vehicles reported an abnormality. To achieve a low false alarm rate, many vehicles only detected an abnormality after traveling great distances from the incident site. A possible improvement involves the use of the location with the lowest speed instead of the reported location to predict incident locations.

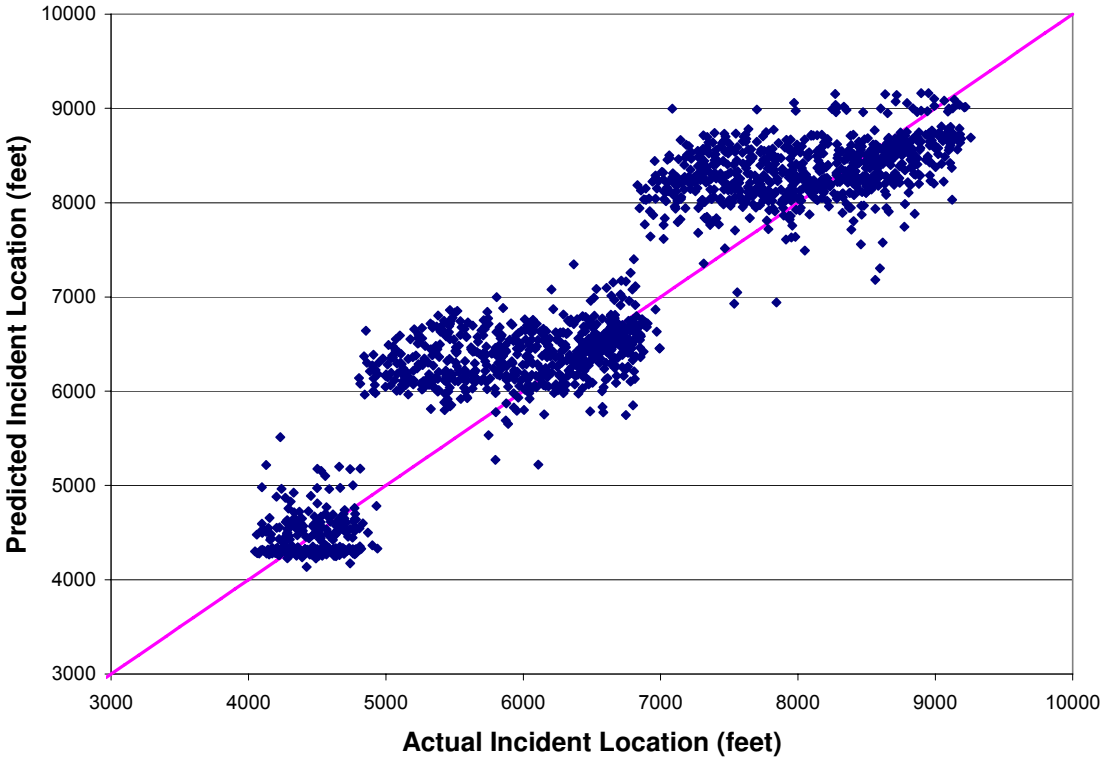


Figure 4.22 Prediction on incident location of the VII model

As shown in Figure 4.23, the RMSEPs of prediction on incident locations varied little with various penetration rates of VII-enabled vehicles for incidents blocking different numbers of lanes. The RMSEP falls within the range of 7% and 10.5%.

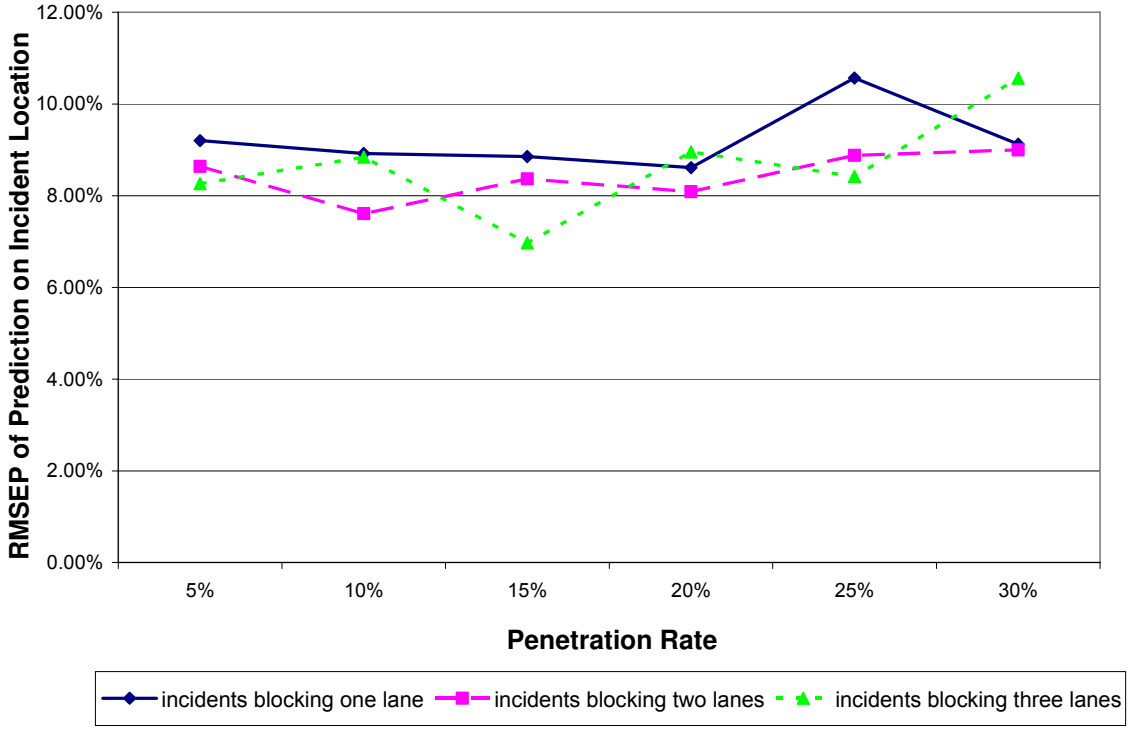


Figure 4.23 RMSEP of prediction on incident locations of the VII model with various penetration rates of VII-enabled vehicles

Figure 4.24 shows the RMSEPs of prediction on incident locations of the VII model with 20% VII-enabled vehicles for different traffic volumes. As expected, there was no significant difference in the prediction accuracy in terms of RMSEP among various traffic volumes. Here, the RMSEP varied between 5.4% and 10.2%.

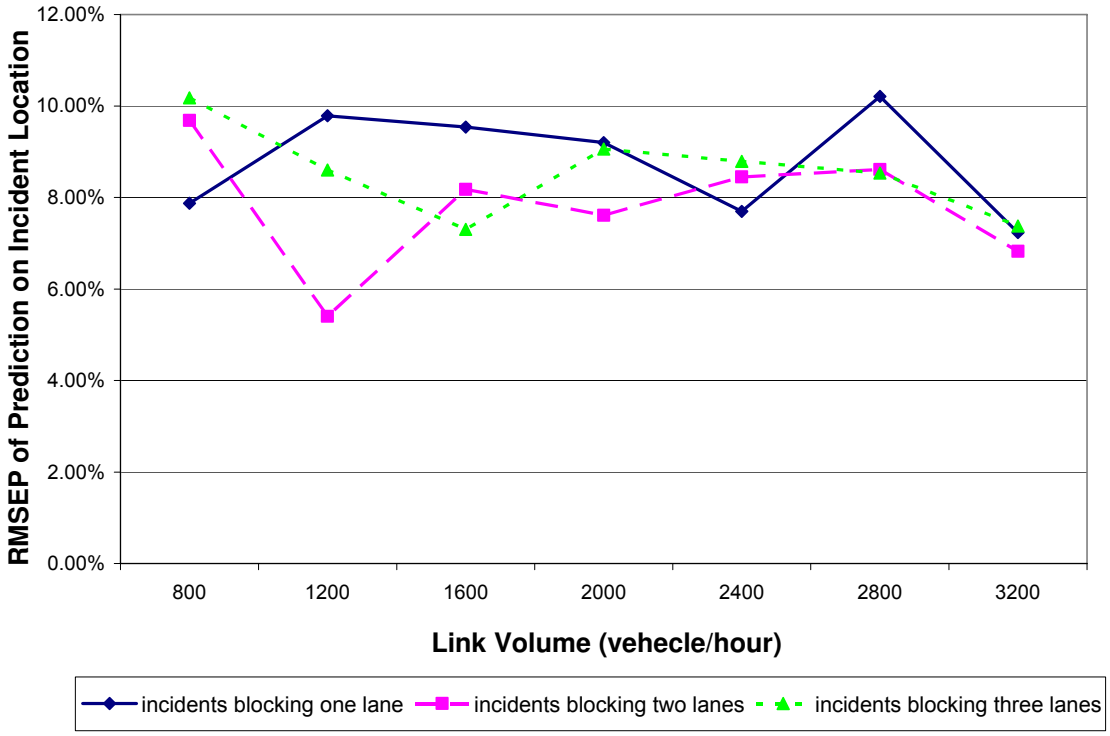


Figure 4.24 RMSEP of prediction on incident locations of the VII model with 20% VII-enabled vehicles for different traffic volumes

4.3.3 Communication Metrics of the VII Model

As shown in Figure 4.25, the number of packets sent increased linearly as the percentage of VII-enabled vehicles increased. On the other hand, the delivery ratios maintained a very high rate, which is close to 100%, regardless of the penetration rate. This guarantees the reliable operation of the proposed VII system.

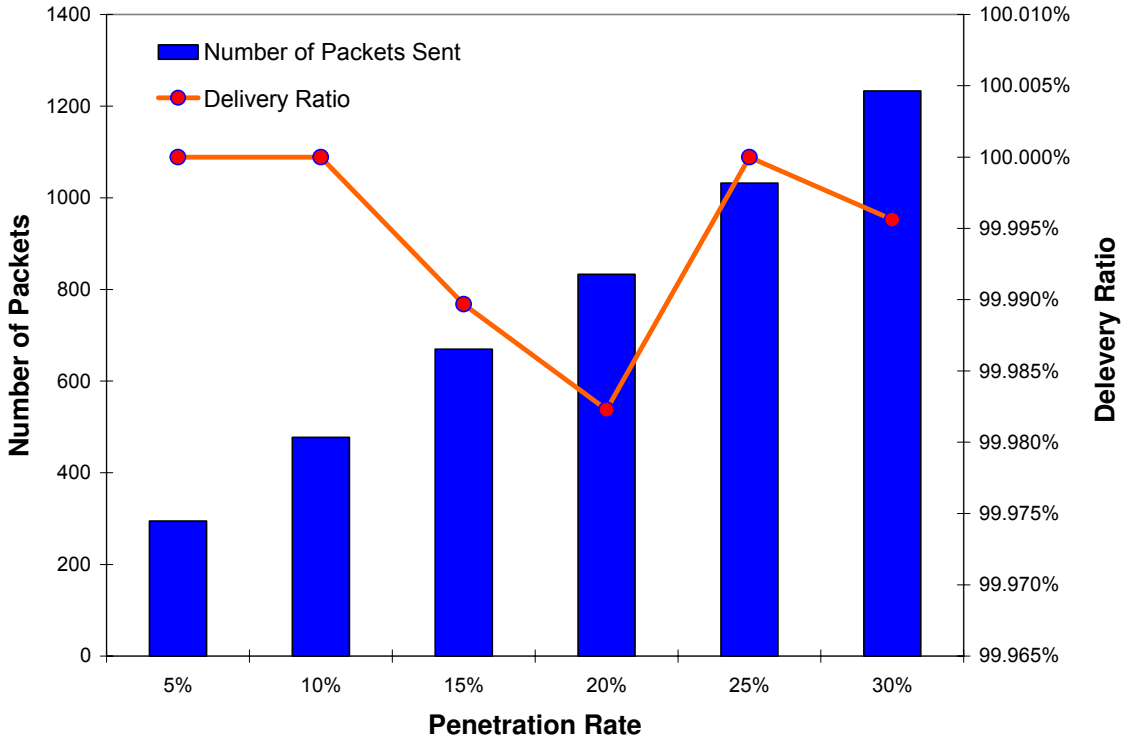


Figure 4.25 Number of packets sent and the delivery ratio of the VII model with various penetration rates of VII-enabled vehicles

Figure 4.26 shows the number of packets sent and the delivery ratio of the VII model with 20% of the vehicles VII-enabled for different traffic volumes. Since the number of packets sent is similar to that in Figure 4.25, the delivery ratios also maintains a high rate of close to 100% for any traffic volume. To detect the capacity of the communication network, more VII-enabled vehicles are needed.

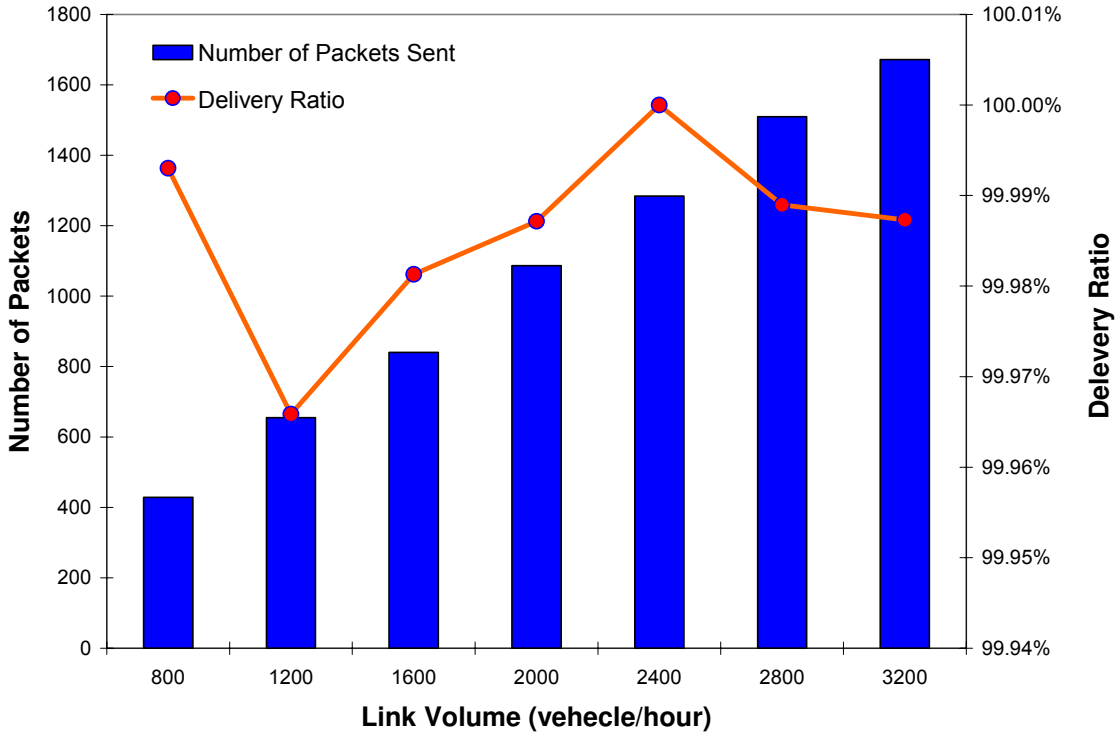


Figure 4.26 Number of packets sent and the delivery ratio of the VII model with 20% VII-enabled vehicles for different traffic volumes

Figure 4.27 presents the communication latency for transmitting a packet from a vehicle to a RSU. The lower limit of latency for vehicles to send a message at specific location was dependent on the number of hops a packet must travel for a vehicle to RSU. Both the mean and variation of latency increased as the distance between the vehicle and the RSU increased. This is because as the vehicle was far away from the RSU, there were more number of hops the packets needed to pass by, which resulted in an increase in the time needed for transmission and presented higher risks to encounter collision and retransmission.

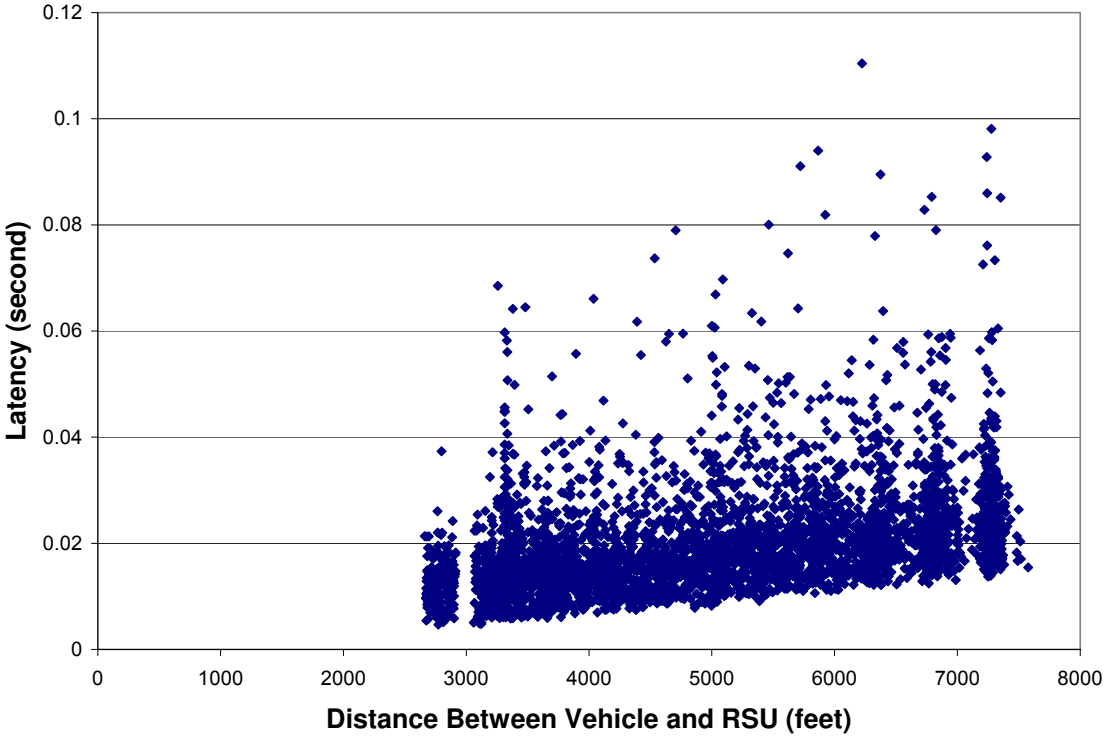


Figure 4.27 Communication latency of the VII model

4.3.4 Summary on Traffic Condition Assessment Framework

This section presents an analysis of the performance of a VII model on assessing real-time traffic conditions. The evaluation of the VII model, on a simulated network in Spartanburg, South Carolina using the integrated simulator, revealed that the SVM algorithm within the in-vehicle module successfully classified traffic conditions into three categories (normal conditions, passing a possible incident scene, and stopped in the queue) by using the vehicle kinetics data, such as speed profiles and lane changing behaviors. The RSU then reliably assessed different types of incidents, such as incidents blocking one or more lanes, with alerts from several vehicles on a selected time window. The incident detection and false alarm rates were quite encouraging. The detection time

was also superior to most existing automatic incident detection algorithms. In addition, the prediction on incident locations will be useful for the incident response team to identify incidents in the field. However, the prediction accuracy on the number of lanes blocked was by an incident was not satisfactory.

The study also found that the detection time decreased as percentage of the VII-enabled vehicles of the total traffic increased, but the extra benefits diminish as the proportion is greater than 25%. When the percentage of VII-enabled vehicles was as low as 15%, the detection time of the VII model was comparable or superior to most of the existing AID algorithms reported in the literature.

4.4 Online Travel Time Prediction Using VII Model

The author developed an online travel time prediction system by incorporating different traffic data, such as link travel time and traffic densities, from VII-enabled vehicles and the use of SVR in parameter estimation. A VII simulation model with the functionality of online travel time prediction was implemented and tested in calibrated and validated traffic simulation network in Greenville, South Carolina. The travel time prediction was performed on a highway corridor 11 miles long with 6 interchanges. The RSU placed at the each interchange was responsible for collecting travel time and traffic volume from each VII-enabled vehicle within its supervised segment. All data were aggregated at a master controller that performed SVR algorithms to predict the travel time of vehicles departing the start point at the next time step.

The prediction performance of the VII model is presented in the following sections in terms of selected MOEs, such as MARE, SRE and RMSEP, which were defined in section 3.3.4.

4.4.1 Travel Time Pattern at the Tested Network

The training and tested data used in this study were travel time data generated by a PARAMICS traffic simulation model of a freeway network in Greenville, South Carolina. In order to examine the performance of online travel time perdition algorithm using a VII model, a sequence of afternoon peak periods with recurrent congestion were generated by varying the travel demand profile. As shown in Figure 4.28, a wide range of variations exist in travel time patterns of ten weekdays with five different traffic demand inputs. Those travel time data created a test environment that included different traffic conditions. Note that the same traffic demand inputs may result in different travel time pattern due to the random nature of the microscopic traffic simulation.

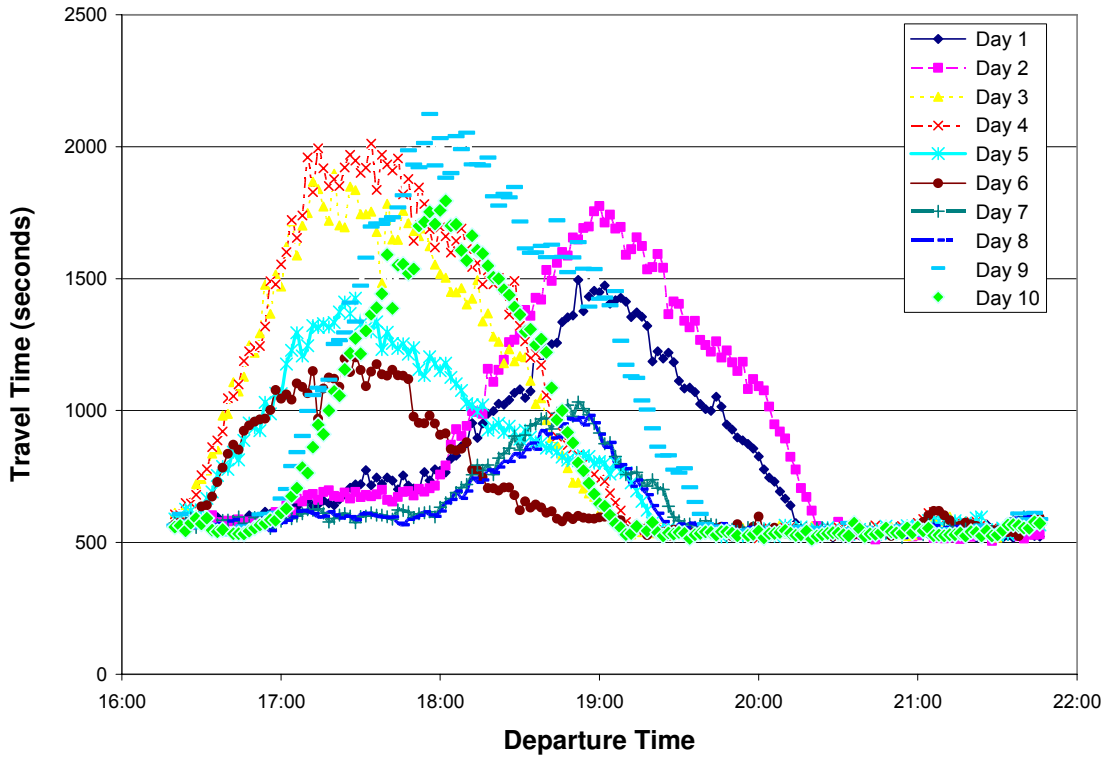


Figure 4.28 Travel time pattern with different demand inputs

4.4.2 Adjustment of the SVR Travel Time Prediction Model

4.4.2.1 Identifying the Parameters for SVR Model

Besides the cost coefficient C and kernel function parameter γ, an additional important parameter was introduced in the SVR model: ε in loss function of epsilon-SVR. Similar to identifying the optimal parameters combination procedure for SVM algorithm, the author again applied the grid searching technique.

Figure 4.29 shows the grid searching efforts for optimal parameters (cost coefficient C, kernel function parameter γ and loss function parameter ε) in a range of $C=2^0 \sim 2^{10}$, $\gamma=2^{-2} \sim 2^8$, and $\varepsilon=2^0 \sim 2^{10}$ with a sliced contour map. Each contour line

represents a specific combination of C, γ and ε that produces the same prediction performance in terms of mean squared error (MSE). The contours were used to identify the parameter combination that yielded the highest prediction accuracy. As shown in Figure 4.29, when [C, γ, ε] falls within a triangle area at level of C=8, MSE equal or lower than 5000 can be achieved. The program determined that the optimal parameter was in the C=2^8, $\gamma = 2^4$ and $\varepsilon = 2^4$, which gave a MSE value of 2411.

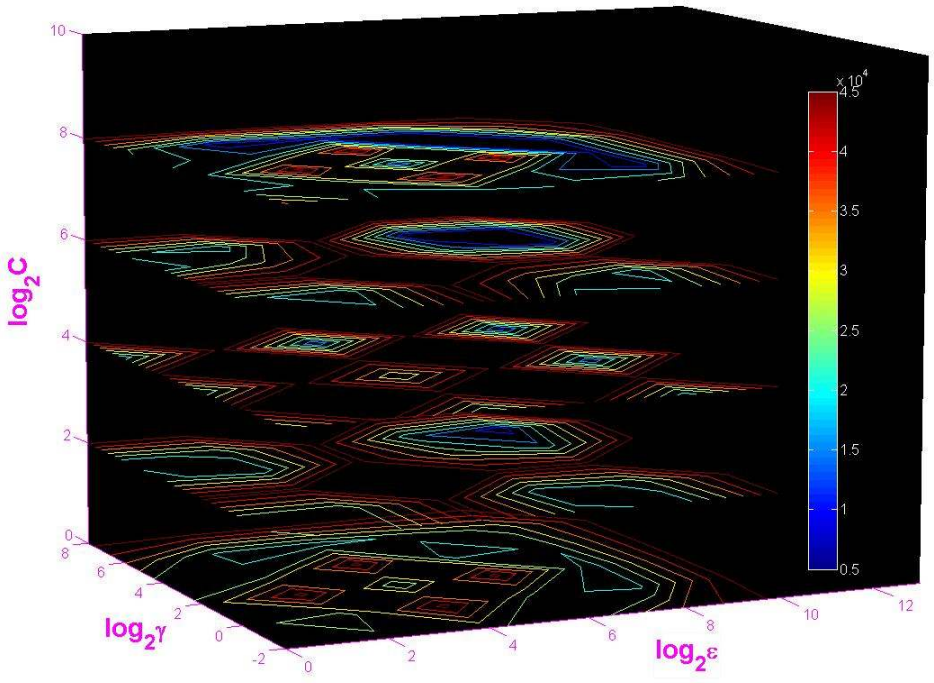
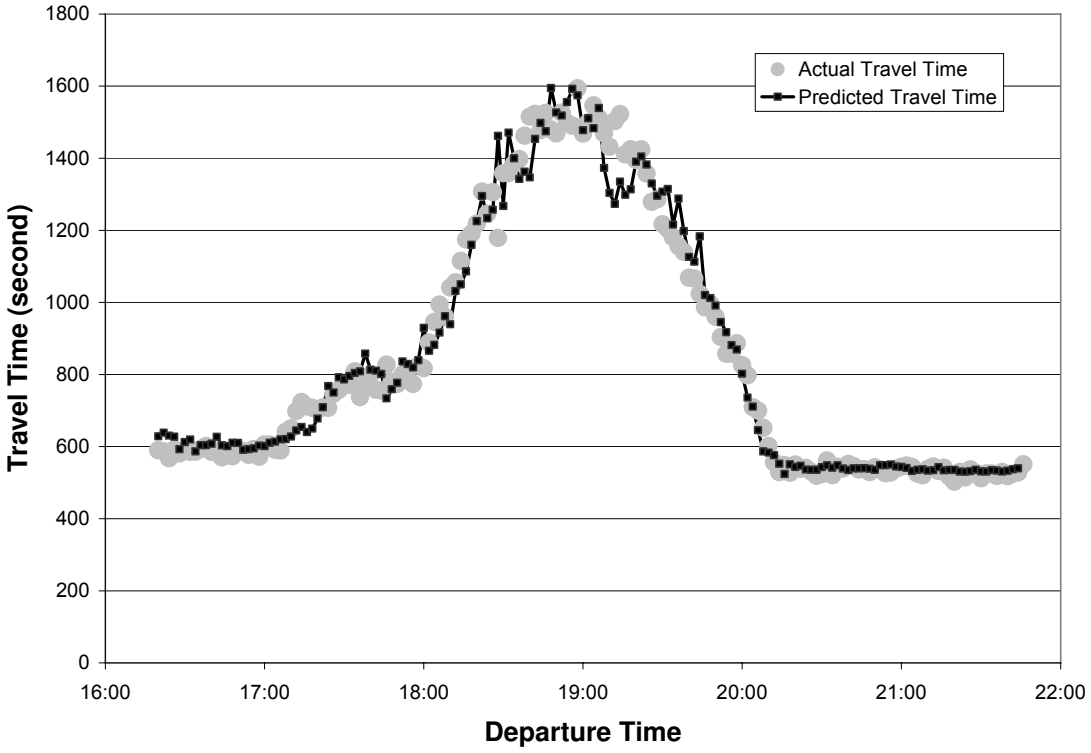


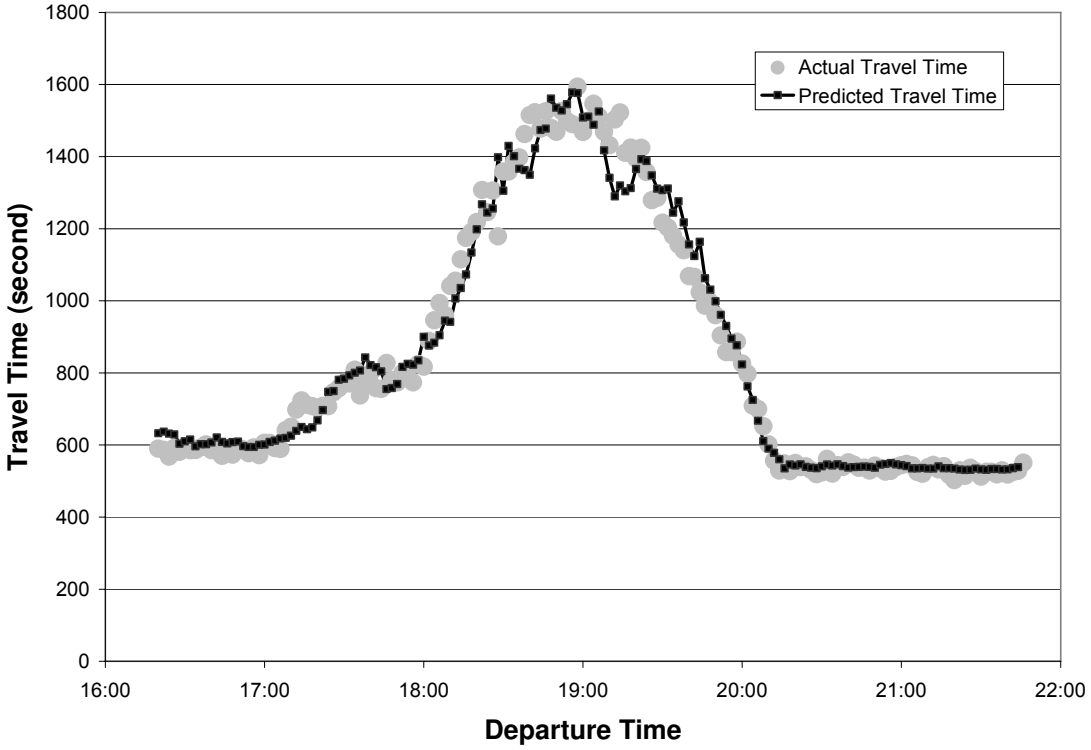
Figure 4.29 Prediction performance contour map of parameter combinations of the developed SVR model

4.4.2.2 Smoothing the Predicted Travel Time

Figure 4.30 (a) and (b) presents an original and smoothed travel prediction for one afternoon peak period with recurrent congestion, respectively. As shown in Figure 4.30, the smoothing functions reduced the variation of the predicted travel time, which resulted in a positive effect on SRE. Further analysis revealed that smoothing also improves the MARE and RMSEP.



(a)



(b)

Figure 4.30 Original (a) and smoothed (b) travel time prediction on an afternoon peak period with recurrent congestion

In order to identify the optimal smoothing factors adopted in the smoothing function, a sensitivity analysis was conducted to examine the performance of smoothed travel time prediction with different smoothing factor options. As shown in Figure 4.31, 7-3-0 and 7-2-1 are superior over other options. 7-2-1 yielded the highest accuracy while 7-3-0 retained the variation minimum. The author selected 7-2-1 as the smoothing factor for this study. Note that the 3 numbers connected by hyphen repented 3 parameters in the smoothing function as specified in section 3.3.3. For example, 7-2-1 means the smoothed travel time prediction will be the sum of 70% of current predicted travel time plus 20% of

one time step ago predicted travel time plus 10% of two time steps ago predicted travel time.

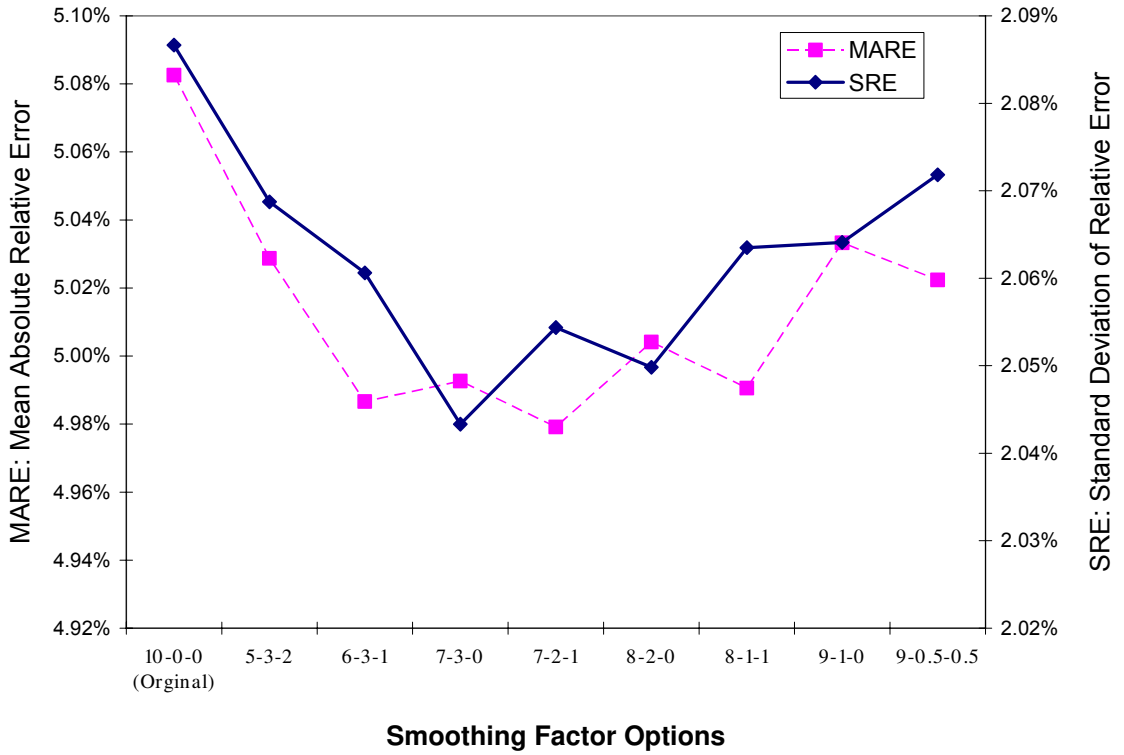


Figure 4.31 MARE and SRE of travel time prediction with different smoothing factors

4.4.3 SVR Algorithm for Travel Time Prediction

4.4.3.1 Comparison of SVR with Other Travel Time Prediction Model

Figure 4.32 showed an example of travel time prediction using the instantaneous prediction model, which was developed as a base line model for comparison with the SVR model. As shown in Figure 4.32, while the instantaneous predictive model worked well during non-congested period, there was a lag between the actual and predicted time during congestion. This was because the instantaneous model suffered from the

assumption that the travel times of vehicles departing from the start point at the specific time point would not vary significantly from the travel times of vehicles arriving at the end at the same time point, while the travel time changed frequently during congestion.

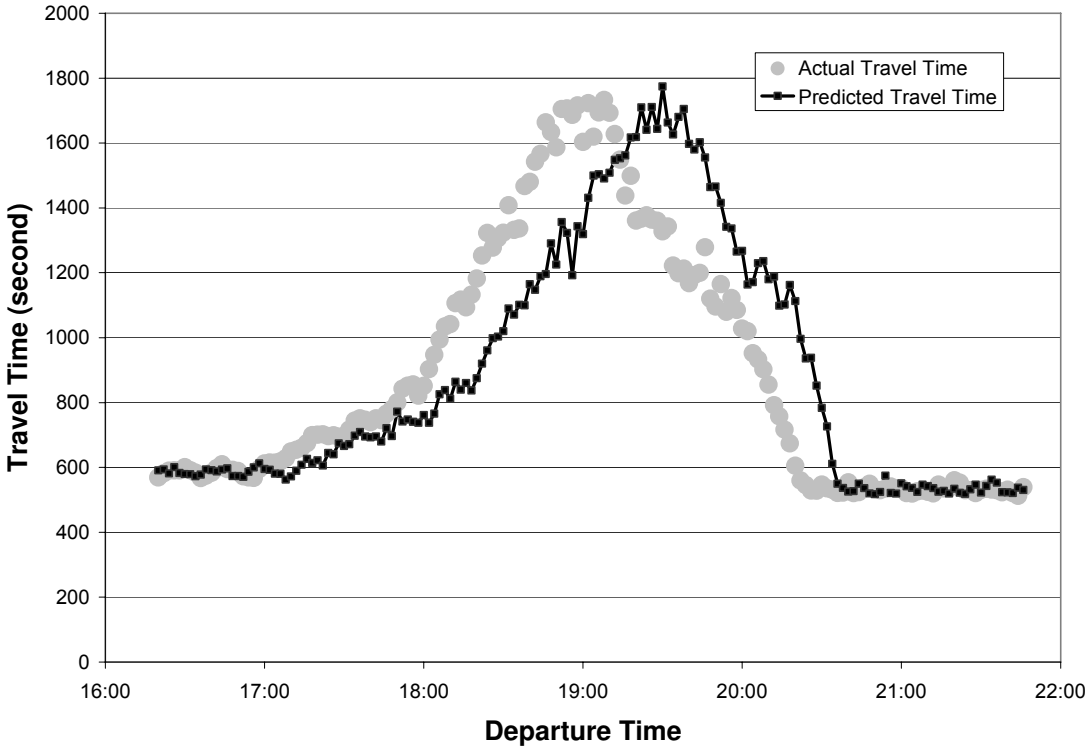


Figure 4.32 Travel time prediction using instantaneous prediction model

As shown in Table 4.5, the SVR model was much better than the instantaneous model in terms of the selected MOEs such RMSEP and MARE. There was little bias in the prediction for the SVR model while the MRE was close to 0, while the instantaneous model predicted a overall of 2.23% longer travel time than the actual travel time. Both RMSEP and MARE indicated that SVR model had much better accuracy than the instantaneous model. In addition, instantaneous model had a larger variation in its

prediction. The comparison of original and smoothed SVR, also in Table 4.5, show the smoothed model to be slightly superior over the original model in every aspect.

Table 4.5 Performance of SVR and Instantaneous Travel Time Prediction Models

Model	RMSEP	MRE	MARE	SRE
SVR (original)	8.35%	0.13%	5.03%	2.07%
SVR (smoothed)	8.26%	0.09%	4.98%	2.05%
Instantaneous	22.95%	2.23%	13.91%	7.35%

The SVR model developed in this dissertation was compared other reported travel time prediction model in the literature. As shown in Table 4.6, MARE between 4 and 6 can be considered to be superior and the developed SVR model is among the best. However, one should note that some models were not suitable for online prediction, as with the application of an SVR model to time series prediction on travel times, which generated very low MARE values without specifying the input time window (Wu et al. 2004). For a network 219 miles long, the travel time available at each time point was the one departed long time ago. It must be acknowledged that these prediction results varied greatly with the network length, traffic congestion level and travel time data availability and quality. Most other models used traditional travel time measurement tools, such as dual loop and cameras, which may be unreliable and need extra effort to deal with (Van Lint 2006). Many researchers applied ANN, which required large data sets for training, a very time consuming process not easily adaptable to ever changing traffic conditions. Conversely, SVR only requires a relative small size of data for training and can be updated to adapt to new scenarios easily. In addition, SVR is also advantageous in that its

structural risk minimization (SRM) mechanism that always ensures a return of the global minimum, whereas the empirical risk minimization (ERM) mechanism of ANN cannot guarantee the global minima (Vapnik 1995).

Table 4.6 Comparison of SVR Model with Other Models Reported in Literature

Model	MARE (%)	Network Length (mile)	Data Source	Training Data Set	Testing Data Set
SVR (this study)	5.0	11	VII	20 Peaks	20 peaks
FNN (Innamaa 2007)	4.6-4.9	6.3-17.5	Dual loop / Camera	4 months	2-3 weeks
SSNN (Van Lint 2006)	5.4	8.1	Dual loop	1071 peaks	118 peaks
SVR (Wu et al. 2004)	1.0-4.4	28-219	Dual loop	28 days	7 days
FNN (Huisken and Van Berkurn 2003)	4.6	6.3	Dual loop		13 peaks
Linear regression (Zhang and Rice 2003)	6-11	6.3	Dual loop / Probe Veh.	N/A	20 days
Kalman filter (Park and Rilett 1998)	6.2	17.3	AVI	131 days	100 days
Spectral FNN (Park et al. 1999)	7.2	17.3	AVI	131 days	100 days
Modular FNN (Park and Rilett 1998)	8.1	17.3	AVI	131 days	100 days
Regular FNN (Park and Rilett 1998)	9.0	17.3	AVI	131 days	100 days

Note: SSNN = state-space neural network; FNN = feed-forward neural network

4.4.3.2 SVR Travel Time Prediction with Different Penetration Rate

Figure 4.33 shows the MARE and SRE of the travel time prediction using VII model with different penetration rates. The increased number of VII-enabled vehicles positively affects the prediction accuracy and variation. When the penetration rate is low, the travel time and traffic volume data collected from VII-enabled vehicles, which were treated as a sample of the traffic population, becomes unreliable; the sample size is too

small and the deviation of measurement from the population is too high. As the penetration increases, the positive effects diminish with 20% to 25% of VII-enabled vehicles being good enough to yield accurate and reliable travel time prediction.

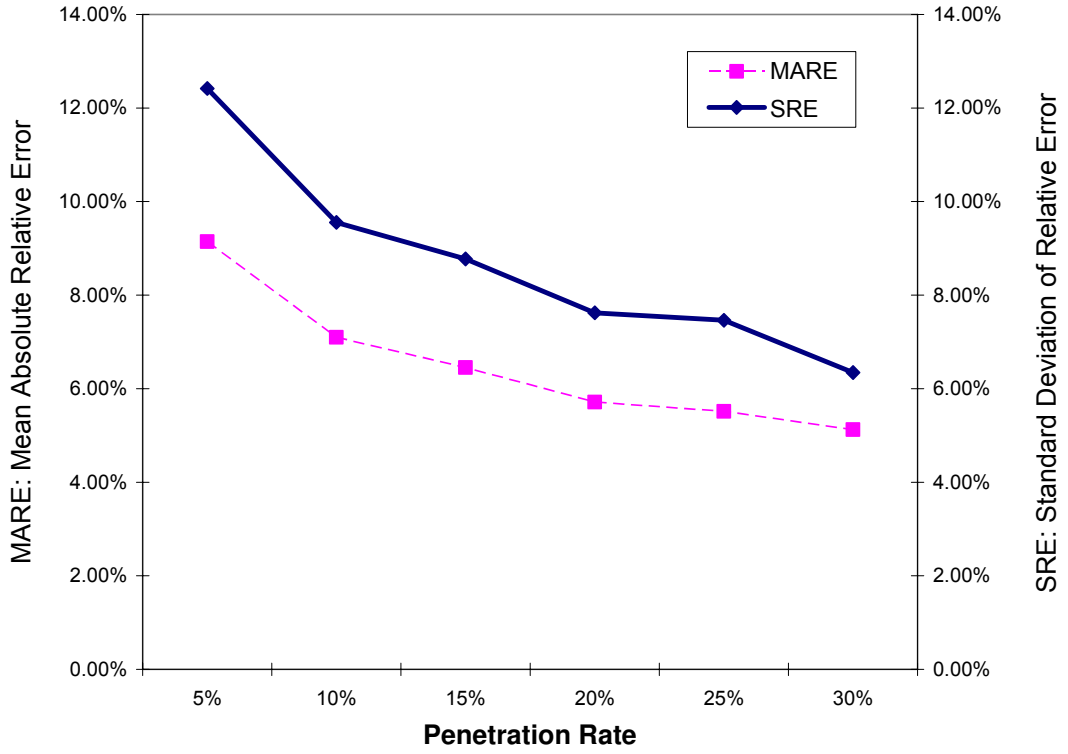


Figure 4.33 MARE and SRE of travel time prediction with different penetration rates

T-test for the difference in the average of actual and predicted travel time to conducted to further examine the accuracy of the prediction. As shown in Figure 4.34, the percentage of predictions that have no significant difference with the actual travel time generally increases as different penetration rates increases. As the penetration rates are greater than 25%, more than 80% of the prediction will have no significant difference with the actual travel time.

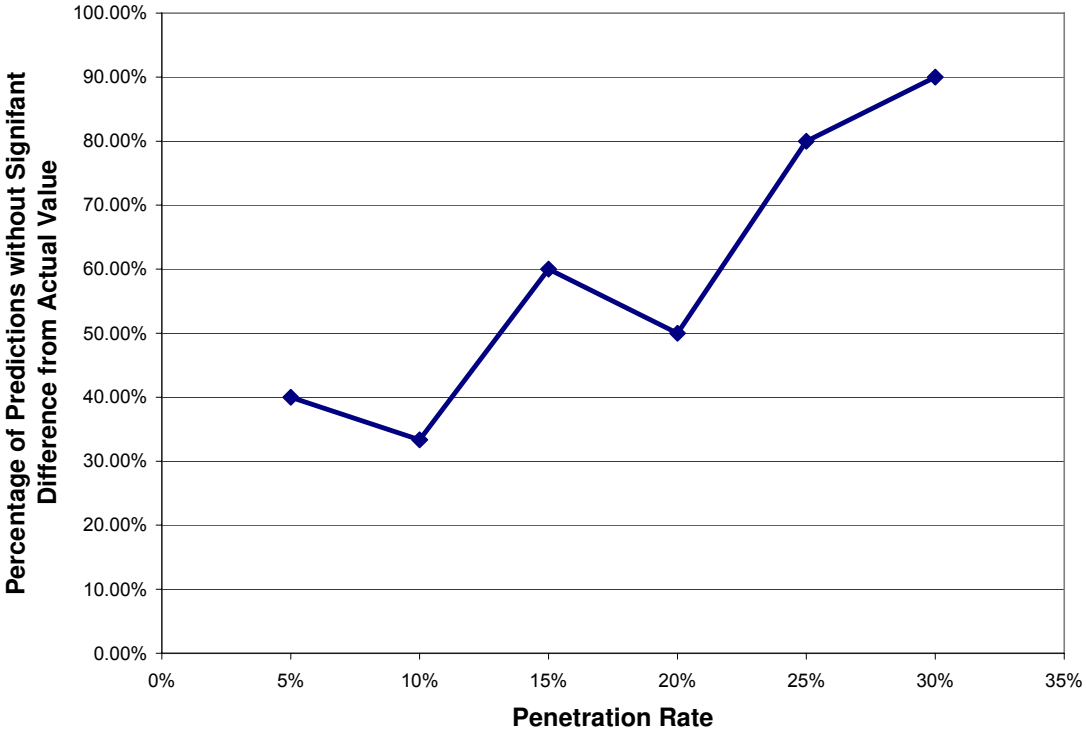


Figure 4.34 Percentage of predictions with no significant difference from actual travel time with different penetration rates

4.4.3.3 Performance of SVR Travel Time Prediction Model during Incident

Many ANN travel time prediction models failed to perform well during incident due to the rarity and insufficiency of non-recurrent congestion used to train the ANN model. However, SVR algorithm predicts travel times based upon vehicles with VII-enable systems that measure travel time and traffic volumes, which remain unaffected by congestion caused by either bottlenecks or incidents.

As shown in Figure 4.35, the developed VII model can predict the travel time for normal traffic conditions and traffic conditions during incidents. The diamond indicates

that the predicted travel time for normal condition and the traffic jams commenced at approximately 18:00 and lasted for approximately one hour and 45 minutes. Once an incident blocking two lanes for 30 minutes occurred at 16:35, the travel time pattern changed significantly. However, the SVR model was still able to accurately provide a good real time estimate of actual travel time.

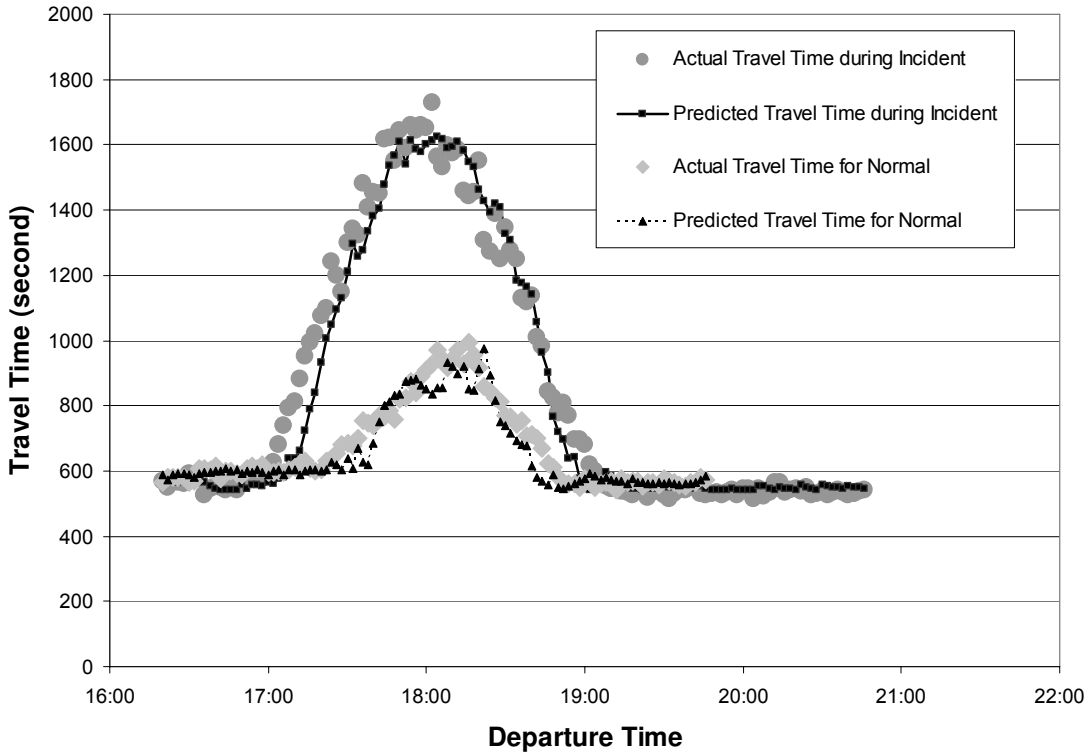


Figure 4.35 Travel time prediction in both normal traffic conditions and during incident

4.4.4 Summary on Travel Time Prediction

This study elucidated encouraging results of travel time predictability using both the VII system and SVR. The developed travel time prediction model outperformed the simple instantaneous prediction model, and the accuracy, in terms of MARE, of the

presented SVR model was among the best of the reported results in the literature. However, specific MOEs may be sensitive to variations in network characteristics between this and other sites. The smoothing function was found to be beneficial for both the accuracy and variation of the travel time prediction model. Additionally, increasing the penetration rate of VII-enabled vehicles was shown to positively affect accuracies and variations of the prediction. Unlike other prediction models, the proposed model performed fairly well even during non-recurrent congestion delays.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

The first part of this chapter presents conclusions developed based upon the results of analysis. The section, following the conclusions, presents recommendations for use of this research and further study on the areas covered in this dissertation**.**

5.1 Conclusions

This dissertation presented a real-time traffic condition assessment and prediction framework using vehicle-infrastructure integration (VII) system with computational intelligence. As a platform in support of the design and evaluation of the simulation model of such a framework, an integrated traffic and communication simulator was developed. Using this integrated simulation platform, various communication alternatives with combinations of distributed or centralized architecture and wired or wireless medium were evaluated to facilitate selection and design of appropriate processing and networking architecture for the proposed VII framework. Additionally, a hybrid framework utilizing the positive aspects of the centralized and distributed management was developed. It was able to eliminate the risks of single point failures, enhance scalability and integration of control functions, thereby was used to support a VII system to assess and predict traffic conditions in a real-time fashion. Moreover, this research also integrated two computational intelligence paradigms called “Support Vector Machine (SVM)” and “Support Vector Regression (SVR)” within the hybrid VII framework for

improving the incident detection and travel time prediction capabilities. Finally, this work developed and evaluated the simulation model of the proposed traffic condition assessment and travel time prediction framework.

Though the results of this research were quite encouraging, there were several potential drawbacks that warrant the attention of future researchers and practitioners. Foremost, one must keep in mind that this research was conducted using the simulation tool. In the real-world implementation, the performance of the models developed in this study may vary due to factors not considered in the computer simulation. Secondly, the false alarm rate of the proposed VII traffic condition assessment framework was expected to increase as the volume-to-capacity ratio increased. Future research should include experiment on a more congested network. While comparing wired and wireless medium, the reliability issues of wireless communication under the circumstance of various terrain and weather condition were not addressed in this research. Additionally, the assumption made in this research that vehicle kinetic and maneuver data could be accurately measured by a vehicle on-board system needs verification through further research. The maintenance problem of RSUs and repeaters in the VII system also requires carefully study and design to fulfill the requirement of real-world implementation.

The immediate impact of this research will provide a reliable alternative to traditional traffic sensors to assess and predict the condition of the transportation system. This system can assess traffic conditions where traffic sensors are not present, by using the roadside units (RSUs) collecting traffic data in individual VII-enabled vehicles through vehicle-to-vehicle and vehicle-to-infrastructure communication. The following

sub-sections summarized the research efforts in four aspects, which are correspond to each of the four research objectives.

5.1.1 Integrated Simulation Platform

The integrated simulator developed through this research can be utilized for evaluating different real-time traffic management applications. This research successfully met the objective of developing an integrated explicit-traffic-explicit-communication simulation platform based on ns-2 and PARAMICS software. The platform is capable of modeling both centralized, distributed or hybrid real time traffic management system. A notable advantage of integrating ns-2 and PARAMICS is that it allows users to customize the simulator to generate any selected MOE in both traffic and communication domains, such as incident detection rate, false alarm rate and detection time as metrics for traffic related impacts as well as communication latency and delivery ratio for the analysis of communication effectiveness and efficiency. The integrated simulation platform is expected to facilitate the design and evaluation of a wide range of online traffic surveillance and management system for inter-disciplinary research and development.

As a case study on the application and evaluation of the integrated simulation platform, a distributed incident detection and response system using wireless traffic sensor network was developed and implemented in the integrated simulator. The detection rate and false alarm rate were assessed for a distributed shockwave detection algorithm based on traffic flow theory and distributed network collaborations, detecting and verifying the presence of shockwaves caused by incidents. While simultaneous detections caused unforeseen communication latency, the communication times were

confirmed to be tolerable for the case study. Communication latency, ordering, and reliability shall become more crucial once a larger system is in place. The statistical dependency of detection performance on sensor placement was evaluated, showing opportunities and direction for improvement.

5.1.2 Evaluate Communication Alternatives

Using this integrated simulator, this study evaluated different communication alternatives with a combination of different architectures and mediums to facilitate the development of a hybrid framework for a VII model. The important MOEs, such as throughput, delivery ratio, and throughput cost ratios, showed that the wired alternatives are less cost effective than those wireless alternatives in both centralized and distributed communication topologies. For the particular camera density (1 camera per 1.5 miles), the cost-effectiveness of the wireless centralized and distributed system are comparable and larger than that of wired centralized and distributed system, before camera data rate goes beyond the saturation throughput. On the other hand, the wireless-distributed network increases its cost-effectiveness in terms of throughput-to-cost ratio as the density of supported devices increases when the camera data rate is within the capacity. Therefore, optimal density of traffic cameras in a wireless-distributed network depends upon the desired camera data rate. Given the fact that distributed architecture outperforms centralized architecture in terms of saving communication cost, a wireless-distributed solution will be economically more preferable if the expected data rate is moderate while the camera density is high.

5.1.3 Hybrid Framework for VII System

The proposed hybrid framework for a VII model was exemplified via a communication networking scheme that includes hierarchical addressing, routing and control. Without relying on any communication infrastructure and a central control point, the system was able to perform on-line traffic condition assessment using a group of ad hoc protocols. The performance of this ad hoc network was evaluated using the integrated simulation platform. As expected, the number of messages sent by vehicles increased linearly as the penetration rates of VII_enabled vehicles and traffic volume increased. The delivery ratio was maintained at a very high level (99.95%) and varied little for all experimental scenarios tested in this study. The latency of transmitting messages between vehicles and RSUs increased in mean value and variability as the distance between vehicles and RSUs increased because the packets had to travel more hops to reach the RSUs. The communication latency was small enough to be negligible.

5.1.4 Traffic Condition Assessment Framework

Another major contribution of this research was the development of a VII model in the integrated simulation platform that was able to detect highway incidents using data generated from vehicles, such as speed profile and lane changing behavior represented in vehicle kinetics, as envisioned in the Vehicle-Infrastructure Integration (VII) system. A SVM algorithm was developed to organize and process the vehicle-generated data, which was fed into an RSU to assess traffic conditions in each segment of the highway. The evaluation of the VII model on a simulated network in Spartanburg, South Carolina

revealed that the SVM algorithm within the in-vehicle module successfully detected different types of incidents, such as incidents blocking one or more lanes, by using the vehicle kinetics data, and the RSUs reliably validated the incident with alerts from several vehicles on a selected time window.

The module performance in terms of the detection and false alarm rates was encouraging. The detection time was also superior over most existing automatic incident detection algorithms reported in the literature. In order to compare the VII-based incident detection system with one most well known automatic incident detection algorithms called "California Algorithm", the latter was modeled and compared with the SVM algorithm under similar traffic and incident conditions. The SVM outperformed California Algorithm in terms of detection rate, false alarm rate, and detection time.

The study also found that while the detection time decreased as percentage of the VII-enabled vehicles in the total traffic increased, the extra benefits diminished, when the proportion of VII_enabled vehicles was larger than 25%. However, when the percentage of VII_enabled vehicles was as low as 15%, the detection time of the VII model was comparable or superior to most existing AID algorithms. In addition, the increase in traffic volume also had a positive impact on the incident detection performance.

The proposed VII model was also able to predict number of lanes blocked during incidents. As the penetration rate was over 15%, the prediction accuracy was between 50% and 90% depending on number of lanes blocked due to an incident. The penetration rate of VII-enabled vehicles had different effects on the prediction accuracy for different number of lanes blocked during an incident. With the increase in penetration rate of VII-

enabled vehicles, the model was able to predict incidents blocking all lanes with superior accuracy as compared with incidents blocking one or two lanes. On the contrary, the prediction accuracy for incidents blocking one lane generally decreased as penetration rate of VII-enabled vehicles increased. The prediction accuracy for incidents that blocked two lanes did not vary significantly when the penetration rate decreased significantly.

The developed VII model was also able to predict most incident locations within 1000 feet of the actual incident sites, which is expected to be helpful to incident management agencies tasked with identifying and responding to the incident site. The average statistical RMSEP of predicting incident location was 9%, and the penetration rate of VII-enabled vehicles and traffic volume had little effect on the prediction accuracy.

5.1.5 Travel Time Prediction Framework

An SVR-based travel time prediction model was developed that also used the VII system. The smoothed SVR model was able to predict travel time with a high level of accuracy. The fact that only a small amount of data set was needed for training suggests that SVR is possibly suitable for adaptation to the ever changing highway traffic conditions. The baseline travel time prediction model using the instantaneous algorithm was developed and compared with SVR model. In comparison with the baseline and other travel time prediction models reported in the literature, the developed SVR model was as good as or superior over others. The increase in the penetration rates of VII-enabled vehicles had positive effects on improving the travel time prediction accuracy.

5.2 Recommendations

The recommendations are organized in two subsections: recommendations for use of this research and recommendations for future research.

5.2.1 Recommendations for Use of this Research

The following recommendations are made regarding the use of this research for on-line traffic management:

- Future implementation of the research in the commercial sectors will result in new VII related equipments in vehicles to support connected vehicles and connected vehicle and roadway infrastructure concepts to improve highway safety and mobility.
- Transportation operation agencies can use the results of the distributed wireless ad hoc sensor network concept for greater control of traffic loadings, faster incident detection, improved safety, mitigated congestion, and reduced travel times without significant investment in the communication infrastructure.

5.2.2 Recommendations for Future Research

The following recommendations are made for further research on the areas covered in this study:

- Future research should evaluate the performance of the VII models presented in this research in other highway networks with larger training and testing dataset for implementing the SVM and SVR algorithms.

- Follow-up research should also evaluate the relationship between communication performance and various VII application configurations, such as the different message size, periodic message sending interval, RSU densities, as the success of the framework presented in this dissertation partially depends on real-time communication between vehicles and RSUs.
- With the knowledge of real-time traffic conditions in the network, the proposed VII framework could be expanded to optimally distribute the traffic loading through direct communication with vehicles through VII and continuous update of the loading distribution plan.
- Future research should conduct field study to evaluate and refine the proposed on-line traffic condition assessment and prediction framework. Additional comparison of the VII-based SVM incident detection and SVR travel time prediction model with other traffic condition assessment and prediction method should also be performed using the real world data.
- Other communication protocols that use vehicles to forward messages to RSUs should be researched and compared with the presented framework where the repeaters relay messages.

APPENDICES

The presented source codes in the following sections are also the research products of James E. Clyburn University Transportation Center, located at South Carolina State University, funded project titled Integrated Simulation Platform for Evaluating Wireless Traffic Sensor Network for Traffic Safety and Security.

A.1 Implementation of Integrated Simulation Platform in ns-2

The following sections present selected source codes for the implementation of integrated simulation platform in ns-2.

A.1.1 Application Layer

A.1.1.1 Header File "snet.h"

```
#include "timer-handler.h"
#include "packet.h"
#include "app.h"
#include "udp-snet.h"
#define MAX 20

class Snet;

// Sender uses this timer to
// schedule next application data packet transmission time
class SwitchTimer : public TimerHandler {
 public:
      SwitchTimer(Snet* t) : TimerHandler(), t_(t) {}
      inline virtual void expire(Event*);
 protected:
      Snet* t_;
};

class LandmarkSendTimer : public TimerHandler {
 public:
      LandmarkSendTimer(Snet* t) : TimerHandler(), t_(t) {}
      inline virtual void expire(Event*);
 protected:
      Snet* t_;
};


// Snet Application Class Definition
class Snet : public Application {
```

```
 public:
      Snet();
      void switch_control();  // called by SwitchTimer:expire (Sender)
      void send_landmark_packet();  // called by
LandmarkSendTimer:expire (Sender)
protected:
      int command(int argc, const char*const* argv);
      void start();       // Start sending data packets (Sender)
      void stop();        // Stop sending data packets (Sender)
      void send_msg(int purpose, char address[20], int scale, float
alertTime, int alertLocation);
      void read_files();
 private:
      virtual void recv_msg(int nbytes, const char *msg = 0);
      int number_of_address;
      int landmark_count;
      double next_landmark_time_;
      int nodenumber_;       // node number to which the application is
attached
      int roundnumber_;
      char type_exp_[MAX];
      double delta_t_;        // time interval delta t
    double interval_;      // Application data packet transmission
interval
      int pktsize_;          // Application data packet size
      int running_;          // If 1 application is running
      int scale_;            // Media scale paramete
      int purpose_;
      SwitchTimer swi_timer_;  // SwitchTimer
      LandmarkSendTimer landmark_timer_;  // LandmarkSendTimer
      char addr_mult[MAX][MAX];
      char sending_to[MAX];
      int sending_from;
      char temp_filename[50];
      char temp_filename1[150];
      int time_duration[MAX][2];
      int level, phase;
};
```

A.1.1.2 Main File “snet.cc”

```
#include <string.h>
#include "random.h"
#include "snet.h"
#include "rtp.h"
#include "udp-snet.h"
#include "packet.h"
#include "unistd.h"
#include "snetrout/snetrout_pkt.h"
#include "snetrout/snetrout.h"

#define HEAD_DISCOVERY 2
```

```
#define HEAD_BROADCAST 3
#define DATA_PKT 0
#define UDP_FLOW 1104
#define PRINT_TABLES 555
#define START 0
#define END 1
#define TOPOLOGY_DISCOVERY_WARMUP 80
#define ADDRESS_PKT 10
#define TOPOLOGY_DISCOVERY 1100
#define HIERARCHICAL_WARMUP 403
#define nodeNum 51
#define switch_timeStep 30
#define lanePred_ext 120

FILE *syn;
static char *syn_file="C:\\incident.txt";
FILE *rec;
static char *rec_file="C:\\VII_sim\\int_false\\fr.rtf";
FILE *lat;
static char *lat_file="C:\\VII_sim\\int_false\\latency.txt";
FILE *swi;
FILE *mp;
static char *swi_file="C:\\bang.txt";
static char *mp_name="C:\\nodeM";

static int
switch_startTime,inc_startTime,inc_location,laneBlock,penetration,deman
d;
static float interval[nodeNum];
float warmupTime=200,simDuration=1200;
float rdStatus[23][6][5];
float latency[5000][2];
int seg_length=4*40*3.28;
float chk_Status_int=10;
static float identifyTime;
static int det_laneBlock, det_incSeg,det_location,rep_ab[3];
static int Nveh_msgSent,Nveh_msgRecv;
static int stop_time=1770;
// Snet OTcl linkage class
static class SnetClass : public TclClass {
 public:
  SnetClass() : TclClass("Application/Snet") {}
  TclObject* create(int, const char*const*) {
    return (new Snet);
  }
} class_app_mm;

// When swi_timer_ expires call Snet:switch_control() to switch between
paramics and ns2
void SwitchTimer::expire(Event*)
{
  t_->switch_control();
}
```

```
void LandmarkSendTimer::expire(Event*)
{
  t_->send_landmark_packet();
}

// Constructor (also initialize instances of timers)
Snet::Snet() : running_(0), swi_timer_(this), landmark_timer_(this)
{
  for(int i = 0; i < 20; i++)
    {
      time_duration[i][0] = 0;
      time_duration[i][1] = 0;
    }
  bind("pktsize_", &pktsize_);
  bind("nodenumber_", &nodenumber_);
  bind("roundnumber_", &roundnumber_);
  bind("delta_t_", &delta_t_);
}

// OTcl command interpreter
int Snet::command(int argc, const char*const* argv)
{
  Tcl& tcl = Tcl::instance();

  if (argc >= 3) {
            if (strcmp(argv[1], "attach-agent") == 0) {
                  agent_ = (Agent*) TclObject::lookup(argv[2]);
                  if (agent_ == 0) {
                        tcl.resultf("no such agent %s", argv[2]);
                        return(TCL_ERROR);
                  }
                  agent_->attachApp(this);
                  return(TCL_OK);
            }
            else if(strcmp(argv[1], "type-exp") == 0) {
                  strcpy(type_exp_, argv[2]);
                  printf("SNET.CC : type is .......... %s \n",
type_exp_);
                  return(TCL_OK);
            }
            else if (strcmp(argv[1], "addr") == 0) {
                  number_of_address = argc;
                  for(int i=0;i<=number_of_address-3;i++)
                        strcpy(addr_mult[i],argv[i+2]);
                  return(TCL_OK);
            }
            else if(strcmp(argv[1], "filename") == 0) {
                  strcpy(temp_filename,argv[2]);
                  if(strcmp(temp_filename, "TEMPORARY") != 0)
                        printf("I am %d and file is %s \n",
nodenumber_, temp_filename);
                  return(TCL_OK);
            }
            else if(strcmp(argv[1], "time_duration") == 0) {
```

```
                level = atoi(argv[2]);
                if(strcmp(argv[3], "START") == 0) phase = START;
                else if(strcmp(argv[3], "END") == 0) phase = END;
                time_duration[level][phase] = atoi(argv[4]);
                return(TCL_OK);
        }
        else if(strcmp(argv[1], "send-to") == 0) {
                sending_from = 1;
                strcpy(sending_to, argv[2]);
                return(TCL_OK);
        }
        else if(strcmp(argv[1], "veh_msg") == 0) {
                send_msg(DATA_PKT, "IX1.0000.0.2",
atoi(argv[2]),atof(argv[3]),atoi(argv[4]));
                return(TCL_OK);
        }
  }
  return (Application::command(argc, argv));
}

void Snet::read_files()
{
      FILE *fk;
      if(strcmp(temp_filename, "TEMPORARY") != 0)
      {
            fk = fopen(temp_filename,"r");
            fscanf(fk,"%f\n", &interval[nodenumber_]);
            fclose(fk);
      }
      else  interval[nodenumber_]=16;
}

void Snet::start()
{
  for (int i=0; i<=number_of_address - 3; i++) {
      send_msg(ADDRESS_PKT, addr_mult[i], 0,0,0);
  }
  if(nodenumber_ == 0) {
        if((syn = fopen(syn_file, "r")) != NULL)
              fscanf(syn,"%d %d %d %d %d
%d",&switch_startTime,&inc_startTime,&inc_location,&laneBlock,&penetrat
ion,&demand);
        else
              printf("error open file %s.\n",syn_file);
        fclose(syn);
  }
  send_landmark_packet();
  switch_control();
  read_files();
  swi_timer_.resched((double)switch_timeStep);
}

void Snet::stop()
{
```

```
  if(addr_mult[0][9] != '1')  {
        send_msg(PRINT_TABLES, "XXX", 0,0,0);
  }
}

void Snet::send_landmark_packet()
{
  // DONE FOR EARLIER TOPOLOGY DISCOVERY
  if ((addr_mult[0][9] != '1') && Scheduler::instance().clock() <
TOPOLOGY_DISCOVERY_WARMUP)    {
      send_msg(TOPOLOGY_DISCOVERY, addr_mult[0], 0,0,0);
  }

  switch(addr_mult[0][9])     {
      case '0': level = 0;break;
    case '1': level = 1;break;
    case '2': level = 2;break;
    case '3': level = 3;break;
    case '4': level = 4;break;
    case '5': level = 5;break;
    case '6': level = 6;break;
    case '7': level = 7;break;
    case '8': level = 8;break;
    case '9': level = 9;break;
  }

  /*
   * Higher level controllers do broadcast and lower level controllers
reply back with head discovery message
   * Because of this warmup hierarchy is developed
   */

  if(Scheduler::instance().clock() > TOPOLOGY_DISCOVERY_WARMUP)   {
        if(Scheduler::instance().clock() > time_duration[level][START]
&& Scheduler::instance().clock() < time_duration[level][END])     {
              send_msg(HIERARCHICAL_WARMUP, "XXX", HEAD_BROADCAST,0,0);
        }
      if(Scheduler::instance().clock() > time_duration[level+1][START]
&& Scheduler::instance().clock() < time_duration[level+1][END]) {
              send_msg(HIERARCHICAL_WARMUP, "XXX", HEAD_DISCOVERY,0,0);
        }
  }

  /*
   * Sending message to the sensor with address "sending_to"
   */

  if (((sending_from == 1) && ((strcmp(sending_to, "everyone") != 0) &&
(strcmp(sending_to, "head") != 0))) && ((Scheduler::instance().clock()
> warmupTime) && (Scheduler::instance().clock() <
warmupTime+simDuration+1))) {
      send_msg(DATA_PKT, sending_to, 0,0,0);
  }
```

```
  next_landmark_time_ = interval[nodenumber_];
  landmark_timer_.resched(next_landmark_time_);
}

// Send application data packet
void Snet::send_msg(int purpose, char address[20], int scale, float
alertTime, int alertLocation)
{
  hdr_mm mh_buf;
  strcpy(mh_buf.address, address);
  mh_buf.nbytes = nodenumber_;
  mh_buf.scale = scale;
  mh_buf.purpose = purpose;
  mh_buf.a_time = alertTime;
  mh_buf.location = alertLocation;
  agent_->sendmsg(pktsize_, (char*) &mh_buf);  // send to UDP
  if((mh_buf.purpose == DATA_PKT) &&
Scheduler::instance().clock()>warmupTime && nodenumber_>=20)      {
      Nveh_msgSent++;
  }
  return;
}

void Snet::recv_msg(int nbytes, const char *msg)
{
      int i,j,stucked=0;
      static int
check_sta,writed,p_idx,ab_value,i_t,check_time,check_location;
      float rdStatus_sum=0,alertNum=0,rep_location;
      if(msg) {
            hdr_mm* mh_buf = (hdr_mm*) msg;
            int value = mh_buf->scale;
            int location = mh_buf->location;
            int sender = mh_buf->nbytes;
            float sendTime = mh_buf->a_time;

            if((mh_buf->purpose == UDP_FLOW) && (nodenumber_ == 19)) {
                  if(Scheduler::instance().clock()-check_time>180) {
                        if(i_t==2) i_t=1;
                        else i_t=0;
                        rep_ab[0]=rep_ab[1];
                        rep_ab[1]=0;
                        check_time=9999;
                  }
                  for(j=0;j<6;j++) {
                        if(sender==rdStatus[location/seg_length][j][0]
&& (Scheduler::instance().clock()-
rdStatus[location/seg_length][j][3])<80) {
                              stucked=1;

      if(rdStatus[location/seg_length][j][3]==sendTime) {
                                    break;
                              }
                              else {
```

```
                                    Nveh_msgRecv++;

      rdStatus[location/seg_length][j][2]=(rdStatus[location/seg_length
][j][2]*rdStatus[location/seg_length][j][1]+value)/(rdStatus[location/s
eg_length][j][1]+1);

      rdStatus[location/seg_length][j][1]++;

      rdStatus[location/seg_length][j][3]=sendTime;

      rdStatus[location/seg_length][j][4]=location;
                                    latency[p_idx][0]=abs(11825-
location);

      latency[p_idx][1]=(Scheduler::instance().clock()-sendTime);
                                    p_idx++;
                                    if(value>0 && i_t<3 &&
rep_ab[0]!=sender && rep_ab[1]!=sender && identifyTime==0) {
                                          if(i_t>=1 && abs(location-
check_location)>1800) {i_t=0;rep_ab[0]=0;rep_ab[1]=0;}
                                          rep_ab[i_t]=sender;
                                          i_t++;
                                          det_location+=location;
                                          ab_value+=value;
                                          if(i_t==1)
{check_time=sendTime;check_location=location;}

      if(Scheduler::instance().clock()>switch_startTime+210 && i_t==3)
{

      identifyTime=Scheduler::instance().clock();
                                                det_laneBlock=1;

      det_location=det_location/3;
                                          }
                                    }
                                    break;
                              }
                        }
                  }
                  if(stucked!=1) {
                        stucked=0;
                        Nveh_msgRecv++;
                        for(j=5;j>0;j--) {

      rdStatus[location/seg_length][j][0]=rdStatus[location/seg_length]
[j-1][0];

      rdStatus[location/seg_length][j][1]=rdStatus[location/seg_length]
[j-1][1];

      rdStatus[location/seg_length][j][2]=rdStatus[location/seg_length]
[j-1][2];
```

```
      rdStatus[location/seg_length][j][3]=rdStatus[location/seg_length]
[j-1][3];

      rdStatus[location/seg_length][j][4]=rdStatus[location/seg_length]
[j-1][4];
                        }
                        rdStatus[location/seg_length][0][0]=sender;
                        rdStatus[location/seg_length][0][1]=1;
                        rdStatus[location/seg_length][0][2]=value;
                        rdStatus[location/seg_length][0][3]=sendTime;
                        rdStatus[location/seg_length][0][4]=location;
                        latency[p_idx][0]=abs(11825-location);

      latency[p_idx][1]=(Scheduler::instance().clock()-sendTime);
                        p_idx++;
                        if(value>0 && i_t<3 && rep_ab[0]!=sender &&
rep_ab[1]!=sender && identifyTime==0) {
                              if(i_t>=1 && abs(location-
check_location)>1800) {i_t=0;rep_ab[0]=0;rep_ab[1]=0;}
                              rep_ab[i_t]=sender;
                              i_t++;
                              ab_value+=value;
                              if(i_t==1)
{check_time=sendTime;check_location=location;}

      if(Scheduler::instance().clock()>switch_startTime+210 && i_t==3)
{

      identifyTime=Scheduler::instance().clock();
                                    det_laneBlock=1;
                                    det_location=location;
                                    if(ab_value==6) det_laneBlock=3;
                                    }
                        }
                  }
                  if(Scheduler::instance().clock()>switch_startTime+210
&& (Scheduler::instance().clock()/chk_Status_int) > check_sta) {

      check_sta=Scheduler::instance().clock()/chk_Status_int+1;

      for(i=(int)4121/seg_length;i<=9214/seg_length;i++) {
                              rep_location=0;
                              for(j=0;j<6;j++) {

      rdStatus_sum=rdStatus_sum+rdStatus[i][j][2];
                                    if(rdStatus[i][j][2]>0) {
                                          alertNum++;

      rep_location+=rdStatus[i][j][4];
                                    }
                              }
```

```
                              if(rdStatus_sum>=-1 && identifyTime==0 &&
alertNum>0) {

      identifyTime=Scheduler::instance().clock();
                                    det_laneBlock=1;
                                    det_incSeg=i;
                                    det_location=rep_location/alertNum;
                              }
                              if(det_laneBlock<3 && rdStatus_sum>=4 &&
(Scheduler::instance().clock()-identifyTime)<lanePred_ext &&
identifyTime!=0) {
                                    det_laneBlock=2;
                                    det_incSeg=i;
                              }
                              if(rdStatus_sum>=11.5 &&
(Scheduler::instance().clock()-identifyTime)<lanePred_ext &&
identifyTime!=0) {
                                    det_laneBlock=3;
                                    det_incSeg=i;
                                    break;
                              }
                              alertNum=0;
                              rdStatus_sum=0;
                        }
                        if((Scheduler::instance().clock()-
identifyTime>lanePred_ext && identifyTime!=0) ||
Scheduler::instance().clock()>stop_time) {
                              if((syn = fopen(syn_file, "r")) != NULL)
                                    fscanf(syn,"%d %d %d %d %d
%d",&switch_startTime,&inc_startTime,&inc_location,&laneBlock,&penetrat
ion,&demand);
                              else printf("error open file
%s.\n",syn_file);
                              fclose(syn);
                              while(1) {
                                    if(writed==0 && (lat =
fopen(lat_file, "a")) != NULL) {
                                          for(j=0;j<p_idx;j++)
                                                fprintf(lat,"%5.0f
%9.5f \n",latency[j][0],latency[j][1]);
                                    }
                                    fclose(lat);
                                    break;
                              }
                              while(1) {
                                    if(writed==0 && (rec =
fopen(rec_file, "a")) != NULL) {
                                          writed=1;
                                          if(identifyTime!=0 &&
identifyTime<inc_startTime) inc_startTime=900;
                                          fprintf(rec,"%d %d %d %9.5f
%9.5f %d %d %d %d %d
%d\n",roundnumber_,penetration,demand,identifyTime,identifyTime-
```

```
inc_startTime,laneBlock,det_laneBlock,inc_location,(int)det_location,Nv
eh_msgSent,Nveh_msgRecv);
                                        }
                                        fclose(rec);
                                        break;
                                  }
                                  while(1) {
                                        if((swi = fopen(swi_file,"w")) !=
NULL) {
                                              fprintf(swi,"STOP\n");
                                              Tcl& tcl = Tcl::instance();
                                              tcl.evalf("$ns_ at %9.5f
\"finish\"",Scheduler::instance().clock()+5);
                                        }
                                        fclose(swi);
                                        break;
                                  }
                            }
                      }
                }
          }
          return;
}

void Snet::switch_control()
{
          char result[20];
          char line[100];
          strcpy(result,"para");
          int syn_time;

          if (nodenumber_ == 0 &&
Scheduler::instance().clock()>=switch_startTime) {
                Tcl& tcl = Tcl::instance();

                while(1)    {
                      swi = fopen(swi_file,"w");
                      if(swi != NULL) {
                            fprintf(swi,"para
%d\n",(int)Scheduler::instance().clock());
                            fclose(swi);
                            break;
                      }
                }
                printf("ns2 pause at time %5.0f for round %d, msg_sent %d
VS msg_rev %d\n",
Scheduler::instance().clock(),roundnumber_,Nveh_msgSent,Nveh_msgRecv);
                while(1) {
                      swi = fopen(swi_file,"r");
                      if(swi != NULL) {
                            fscanf(swi,"%s %d\n", &result,&syn_time);
                            if(strncmp(result,"ns2",3) == 0 &&
syn_time==30+(int)Scheduler::instance().clock()) {
                                  fclose(swi);
```

```
                                if((mp = fopen(mp_name, "r")) == NULL) {
                                        printf("Error opening movement
pattern file.\n");
                                        exit(1);
                                }
                                while( fgets(line, sizeof(line), mp) !=
NULL ) {
                                        tcl.eval(line);
                                }
                                fclose(mp);
                                break;
                        }
                fclose(swi);
                sleep(1);
                }
        }
      }
      swi_timer_.resched((double)switch_timeStep);
      return;
}
```

A.1.2 Transport Layer

A.1.2.1 Header File "udp-snet.h"

```
#ifndef ns_udp_snet_h
#define ns_udp_snet_h

#include "udp.h"
#include "ip.h"

// Packet Header Structure

struct hdr_mm {
      char address[20];     //Address being passed to network layer
    int purpose;     // purpose of passing this packet
      int nbytes;  // bytes for pkt
      double time; // current time
      int location;
      float a_time;

        int slot_array[1000];

        inline int& valscale() {return (scale);}
      // Packet header access functions
        static int offset_;
        inline static int& offset() { return offset_; }
        inline static hdr_mm* access(const Packet* p) {
                return (hdr_mm*) p->access(offset_);
        }
};
```

```
// Used for Re-assemble segmented (by UDP) MM packet
struct asm_mm {
      int purpose;     // mm purpose number
      int rbytes;  // currently received bytes
      int tbytes;  // total bytes to receive for MM packet
};


// UdpMmAgent Class definition
class UdpSnetAgent : public UdpAgent {
public:
      UdpSnetAgent();
      UdpSnetAgent(packet_t);
      virtual int supportMM() { return 1; }
      virtual void enableMM() { support_mm_ = 1; }
      virtual void sendmsg(int nbytes, const char *flags = 0);
      void recv(Packet*, Handler*);
      virtual void sendaddr(int a, int b, int c = 0);
protected:
      int support_mm_; // set to 1 if above is MmApp
private:
      asm_mm asm_info; // packet re-assembly information
};

#endif
```

A.1.2.2 Main File "udp-snet.cc"

```
#include "udp-snet.h"
#include "rtp.h"
#include "random.h"
#include <string.h>

char bhagwan[20];


int hdr_mm::offset_;

// Packet Header Class
static class SensornetHeaderClass : public PacketHeaderClass {
public:
      SensornetHeaderClass() :
PacketHeaderClass("PacketHeader/Sensornet",
                                          sizeof(hdr_mm)) {
            bind_offset(&hdr_mm::offset_);
      }
} class_mmhdr;


// UdpSnetAgent OTcl linkage class
static class UdpSnetAgentClass : public TclClass {
public:
```

```
        UdpSnetAgentClass() : TclClass("Agent/UDP/UDPsnet") {}
        TclObject* create(int, const char*const*) {
                return (new UdpSnetAgent());
        }
} class_udpsnet_agent;


// Constructor (with no arg)
UdpSnetAgent::UdpSnetAgent() : UdpAgent()
{
        support_mm_ = 0;
}

UdpSnetAgent::UdpSnetAgent(packet_t type) : UdpAgent(type)
{
        support_mm_ = 0;
}

//add address
void UdpSnetAgent::sendaddr(int a, int b, int c)
{
  //  target_->recvaddr(int a, int b, int c);
}
// Add Support of Sensornet Application to UdpAgent::sendmsg
void UdpSnetAgent::sendmsg(int nbytes, const char* flags)
{
    Packet *p;
        int n, remain;

        if (nbytes == 0) {
                target_->recv(p);
        }

        if (size_) {
                n = (nbytes/size_ + (nbytes%size_ ? 1 : 0));
                remain = nbytes%size_;
        }
        else
                printf("Error: UDPsnet size = 0\n");

        if (nbytes == -1) {
                printf("Error:  sendmsg() for UDPsnet should not be -1\n");
                return;
        }
        double local_time =Scheduler::instance().clock();
        while (n-- > 0) {
                p = allocpkt();
                if(n==0 && remain>0)
                        hdr_cmn::access(p)->size() = remain;
                else
                        hdr_cmn::access(p)->size() = size_;
                hdr_rtp* rh = hdr_rtp::access(p);
                rh->flags() = 0;
                rh->seqno() = ++seqno_;
```

```
            hdr_cmn::access(p)->timestamp() =
(u_int32_t)(SAMPLERATE*local_time);
            hdr_mm* mh = hdr_mm::access(p);
            if(flags) // MM header is passed as flags
                  memcpy(mh, flags, sizeof(hdr_mm));
            target_->recv(p);
      }
      idle();
}

// Support Packet Re-Assembly and Sensornet Application
void UdpSnetAgent::recv(Packet* p, Handler*)
{
      if(app_) {  // if MM Application exists
            // re-assemble MM Application packet if segmented
            hdr_mm* mh = hdr_mm::access(p);
            hdr_mm mh_buf;
            memcpy(&mh_buf, mh, sizeof(hdr_mm));
            app_->recv_msg(mh_buf.nbytes, (char*) &mh_buf);
            if(mh->purpose == asm_info.purpose)
                  asm_info.rbytes += hdr_cmn::access(p)->size();
            else {
                  //asm_info.purpose = mh->purpose;
                  asm_info.tbytes = mh->nbytes;
                  asm_info.rbytes = hdr_cmn::access(p)->size();
            }
            // if fully reassembled, pass the packet to application
      }
      Packet::free(p);
}
```

A.1.3 Network Layer

A.1.3.1 Routing Header File “snetrout.h”

```
#ifndef cmu_snetrout_h_
#define cmu_snetrout_h_

#include "config.h"
#include "agent.h"
#include "ip.h"
#include "delay.h"
#include "scheduler.h"
#include "queue.h"
#include "trace.h"
#include "arp.h"
#include "ll.h"
#include "mac.h"
#include "priqueue.h"

#include "snet_rtable.h"
```

```
#if defined(WIN32) && !defined(snprintf)
#define snprintf _snprintf
#endif /* WIN32 && !snprintf */

typedef double Time;

#define MAX_QUEUE_LENGTH 5
#define ROUTER_PORT      0xff

class SnetRout_Helper;


//class SnetRoutTriggerHandler;
class SnetRout_Agent;


class SnetRout_Agent : public Agent {
      friend class SnetRout_Helper;
      friend class SnetRoutTriggerHandler;
public:
      SnetRout_Agent();
      Agent* agent1_;
      virtual int command(int argc, const char * const * argv);
      void lost_link(Packet *p);
      char addr_proto[20][20];
      char reachable_highways[10][10];
      char gateway_address[20][20];
      int rseqnumber;
      int rec_seqnumber;
      char comp_array[10];
      int flag_value;
      int flag_ack;
protected:
      void New_Packet(Packet* p, char source[20], Packet * p1, int
scope, int count, char discovery_dest[20], int seqnumber_);
      int difference(char address1[20], char address2[20]);
      Packet* snet_rtable(int);
      virtual void recv(Packet *, Handler *);
      void trace(char* fmt, ...);
      void sensor_node (Packet * p);
      void controller1_node (Packet * p);
      int hierarchical_head_discovery(double x1, double y2, double x2,
double y2);
      int hierarchical_head_bcast(double x1, double y1, char
source_address[20]);
      void startUp();
      void sendOutBCastPkt(Packet *p);
      Trace *tracetarget;       // Trace Target
      SnetRout_Helper  *helper_;    // SnetRout Helper, handles
callbacks
      SnetRoutTriggerHandler *trigger_handler;
      snet_RoutingTable *table_;     // Routing Table
      PriQueue *ll_queue;       // link level output queue
```

```
int seqno_;                // Sequence number to advertise with...
int myaddr_;               // My address...
int return_value;
int temp_i;
int possible;
int found_schedule;
int k_count;
int target_intersection;
int collected_intersection;
char controller_local_topology [6][50][30];
  int controller_local_topology_phy [6][50];
  double controller_xcor[6][50];
double controller_ycor[6][50];
int controller_array_found;
  int controller_array_count;
int number_of_highways;
  char xcor[200][100];
  char ycor[200][100];
  double xcor1[200];
  double ycor1[200];
int max_seg_intersection;
int mult_addr_send;
int max_seg_depth[8];
char my_controller_head[20];
char neighboring_controller[6][20];
int ncount;
char slot_array[200][200][20];
char curr[4][20];
int done_count;
int slot_current;
int status[4];
int starting_slot[4];
int slot[100];
int lane_number;
int c;
  int nseqnumber[4];
int change_address;
int mini_interval_;
  int maxi_interval_;
int local_scope;
int last_sensor;
int downstream_node;
  int upstream_node;
char upstream_addr[20];
  int upstream_node_controller[6];
  char upstream_addr_controller[6][20];
char downstream_addr[20];
int flag_dmac;
int delay_spec_;
int rate_, run_;
char type_exp_[20];
int estimated_time;
  int last_segment_time;
  int downstream[4];
```

```
      int upstream;
      int landmarkpacketnum;    // last packet seen from a landmark
        int querypacketnum;    // last packet seen from a query
      double p1, q1, r1;
      char myname [20];    // My name...
      char contname [20];  // Controller name...
      char head_addr_ [20];
      double destination_xcor, destination_ycor;
      int heard_head_flag;
      char *subnet_;             // My subnet
      MobileNode *node_;         // My node
      char *address;
      NsObject *port_dmux_;    // my port dmux

      Event *periodic_callback_;           // notify for periodic
update

      // Randomness/MAC/logging parameters
      int be_random_;
      int use_mac_;
      int verbose_;
      int trace_wst_;

      // last time a periodic update was sent...
      double lasttup_;        // time of last triggered update
      double next_tup;        // time of next triggered update
      //  Event *trigupd_scheduled; // event marking a scheduled
triggered update

      // SnetRout constants:
      double alpha_;  // 0.875
      double wst0_;   // 6 (secs)
      double perup_;  // 15 (secs)  period between updates
      int    min_update_periods_;    // 3 we must hear an update from a
neighbor
      int i;
      int j;

};


class SnetRout_Helper : public Handler {
  public:
    SnetRout_Helper(SnetRout_Agent *a_) { a = a_; }
      //    virtual void handle(Event *e) { a->helper_callback(e); }

  private:
    SnetRout_Agent *a;
};

#endif
```

A.1.3.2 Routing Header File “snetrout.cc”

```
extern "C" {
#include <stdarg.h>
#include <float.h>
};

#include "udp-snet.h"
#include "snet.h"
#include "snetrout.h"
#include "snetrout_pkt.h"
#include "priqueue.h"
#include "snet_rtable.h"
#include <random.h>
#include <stdio.h>
#include <time.h>
#include <cmu-trace.h>
#include <address.h>
#include <mobilenode.h>

#define CONTROLLER 2
#define HIGH_CONTROLLER 3
#define SENSOR 1
#define DATA_PKT 0
#define ADDRESS_PKT 10
#define nodeNum 51

#define TOPOLOGY_REQUEST 1101
#define TOPOLOGY_RESPONSE 1102
#define INITIATING_WARMUP 1100
#define RC_BROADCAST 2000
#define SET 2
#define H_BROADCAST "XXXXXXXXXX"
#define UDP_FLOW 1104

#define snetIP_DEF_TTL   332 // default TTTL

#define DESTINATION_PACKET              400
#define DESTINATION_UP                  401
#define DESTINATION_DOWN                402
#define TOPOLOGY_DISCOVERY              403
#define TOPOLOGY_DISCOVERY_PROP         404
#define DESTINATION_PACKET_SEC          500
#define DESTINATION_UP_SEC              501
#define DESTINATION_DOWN_SEC            502
#define DUMP_TABLE                      555
#define HIERARCHICAL_HEAD_DISCOVERY       2
#define HIERARCHICAL_HEAD_BCAST           3
#define TOPOLOGY_DISCOVERY_TIME          45

extern int devang_pri_array[500];

int hdr_snetrout_pkt::offset_;
```

```
static class SnetRoutHeaderClass : public PacketHeaderClass {
public:
      SnetRoutHeaderClass() :
PacketHeaderClass("PacketHeader/SnetRout",sizeof(hdr_snetrout_pkt)) {
      bind_offset(&hdr_snetrout_pkt::offset_);
      }
} class_SnetRouthdr;

void SnetRout_Agent::trace (char *fmt,...)
{
  va_list ap;

  if (!tracetarget)
    return;

  va_start (ap, fmt);
  vsprintf (tracetarget->pt_->buffer (), fmt, ap);
  tracetarget->pt_->dump ();
  va_end (ap);
}

/*
 * This function actually sends down the packet. The packet leaves the
Network layer at this point
 */
void SnetRout_Agent::sendOutBCastPkt(Packet *p)
{
  hdr_ip *iph = HDR_IP(p);
  hdr_cmn *hdrc = HDR_CMN (p);
  struct hdr_snetrout_pkt* rp = hdr_snetrout_pkt::access(p);

  iph->dport() = ROUTER_PORT;
  hdrc->direction() = hdr_cmn::DOWN;
  rp->prev_hop = myaddr_;
  if(strcmp(addr_proto[0], rp->destination_haddress) != 0)
        target_->recv(p, (Handler *)0);
  return;
}
/*
 * Packet received at the  network layer
 */
void SnetRout_Agent::recv (Packet * p, Handler *)
{
    hdr_ip *iph = HDR_IP(p);
      struct hdr_mm* mh = hdr_mm::access(p);
      struct hdr_snetrout_pkt* rp = hdr_snetrout_pkt::access(p);
      sprintf(myname, "dadar%d", myaddr_);
      sprintf(contname, "controller%d", myaddr_);
    hdr_cmn *cmh = HDR_CMN(p);
      int src = Address::instance().get_nodeaddr(iph->saddr());
      snet_rtable_ent rte;
      if((mh->purpose == DESTINATION_PACKET) || (mh->purpose ==
TOPOLOGY_DISCOVERY))
            strcpy(rp->destination_haddress, mh->address);
```

```
      if(mh->purpose != DATA_PKT && mh->purpose != ADDRESS_PKT && mh-
>purpose != TOPOLOGY_REQUEST && mh->purpose != TOPOLOGY_RESPONSE && mh-
>purpose!= INITIATING_WARMUP && mh->purpose!= DESTINATION_PACKET && mh-
>purpose!= DESTINATION_UP && mh->purpose != DESTINATION_DOWN && mh-
>purpose !=  TOPOLOGY_DISCOVERY && mh->purpose !=
TOPOLOGY_DISCOVERY_PROP && mh->purpose != DESTINATION_PACKET_SEC && mh-
>purpose != DESTINATION_UP_SEC && mh->purpose != DESTINATION_DOWN_SEC
&&    mh->purpose != DUMP_TABLE && mh->purpose !=
HIERARCHICAL_HEAD_DISCOVERY &&      mh->purpose !=
HIERARCHICAL_HEAD_BCAST &&    mh->purpose != TOPOLOGY_DISCOVERY_TIME)
      {
            mh->purpose = UDP_FLOW;
      }
      if ((mh->purpose == DATA_PKT)) {
            strcpy(addr_proto[i],mh->address);
            mh->purpose = UDP_FLOW;
      }
      if (mh->purpose == ADDRESS_PKT) {
            strcpy(addr_proto[i],mh->address);
            i++;
            return;
      }
    if (addr_proto[0][9] == '2') {
        if((mh->purpose == UDP_FLOW)) {
                  if(myaddr_ != iph->daddr()) {
                        controller1_node(p);
                  }
                  else {
                        agent1_->recv(p, (Handler *)0);
                        Packet::free(p);
                  }
            }
            if ((mh->purpose == INITIATING_WARMUP) || (mh->purpose ==
TOPOLOGY_REQUEST) || (mh->purpose == TOPOLOGY_RESPONSE)) {
                  if(Scheduler::instance().clock() <
TOPOLOGY_DISCOVERY_TIME) {
                        if(mh->purpose == INITIATING_WARMUP)
                              rseqnumber = rseqnumber + 1;
                        mult_addr_send = 0;
                        controller1_node(p);
                  }
            }
            if(mh->purpose == DESTINATION_UP || mh->purpose ==
DESTINATION_DOWN || mh->purpose == TOPOLOGY_DISCOVERY || mh->purpose ==
TOPOLOGY_DISCOVERY_PROP || mh->purpose == DESTINATION_UP_SEC || mh-
>purpose == DESTINATION_DOWN_SEC) {
                  if(mh->purpose == TOPOLOGY_DISCOVERY)
                        rp->version_val = mh->scale;
                  if((strcmp(rp->destination_haddress, addr_proto[0])
== 0) && (rp->version_val != HIERARCHICAL_HEAD_DISCOVERY)) {
                  }
                  else {
                        if(((strcmp(rp->dummy_destination_haddress,
addr_proto[0]) == 0) || (strcmp(rp->dummy_destination_haddress,
```

```
addr_proto[1]) == 0)) && (mh->purpose == DESTINATION_UP_SEC || mh-
>purpose == DESTINATION_DOWN_SEC)) {
                                printf("DUMMY GOT IT %s %s \n",
addr_proto[0], addr_proto[1]);
                                strcpy(rp->dummy_destination_haddress,
"BULL");
                                if(mh->purpose == DESTINATION_DOWN_SEC)
                                      mh->purpose = DESTINATION_DOWN;
                                else mh->purpose = DESTINATION_UP;
                          }
                          controller1_node(p);
                    }
              }
              if(mh->purpose == DUMP_TABLE)
                table_->PrintTable(myaddr_);
        }
        else if (addr_proto[0][9] == '1') {
              sensor_node(p);
        }
        else {
              if(mh->nbytes>=20 && myaddr_!=mh->nbytes)
                    Packet::free(p);
              else
                    sensor_node(p);
        }
        //Packet I'm originating...
      if(src == myaddr_ && cmh->num_forwards() == 0) {
              //Add the IP Header
        cmh->size() += IP_HDR_LEN;
              iph->ttl_ = snetIP_DEF_TTL;
        }
        //Packet I'm forwarding...
        else {
              // Check the TTL.  If it is zero, then discard.
              if(--iph->ttl_ == 0) {
                    drop(p, DROP_RTR_TTL);
                    return;
              }
        }
}
/*
 * The sensors execute this part of code
 */

void SnetRout_Agent::sensor_node (Packet * p)
{
  struct hdr_mm* mh = hdr_mm::access(p);
  struct hdr_snetrout_pkt* rp = hdr_snetrout_pkt::access(p);
  hdr_ip *iph = HDR_IP(p);
  double p5,q5,r5;
  //printf("I am %d getting purpose %d \n", myaddr_, mh->purpose);
  if(mh->purpose == UDP_FLOW) {
        if(rp->time_stamp < 5) {
                    rp->prev_hop = myaddr_;
```

```
                        iph->saddr() = myaddr_;
                        if(upstream_node!=0)
                              iph->daddr() = upstream_node;
                        rp->time_stamp =  Scheduler::instance().clock();
                        sendOutBCastPkt(p);
                        //agent1_->recv(p, (Handler *)0);
                  }
                  else {
                        rp->time_stamp1 = Scheduler::instance().clock();
                        rp->prev_hop = myaddr_;
                        iph->daddr() = upstream_node;
                        sendOutBCastPkt(p);
                  }
    }

    if(mh->purpose == TOPOLOGY_RESPONSE) {
                  strcpy(downstream_addr, rp->prev_hop_address);
                  downstream_node = rp->prev_hop;
                  rp->prev_hop = myaddr_;
                  iph->daddr() = upstream_node;
                  strcpy(rp->prev_hop_address, addr_proto[0]);
                  mh->purpose = TOPOLOGY_RESPONSE;
                  rp->pointer_to_address = rp->pointer_to_address + 1;
                  rp->phy_address[rp->pointer_to_address] = myaddr_;
                  strcpy(rp->address_topology[rp->pointer_to_address],
addr_proto[0]);
                  node_ = (MobileNode*)Node::get_node_by_address(myaddr_);
                  node_->getLoc(&p5, &q5, &r5);
                  rp->xcor[rp->pointer_to_address] = p5;
                  rp->ycor[rp->pointer_to_address] = q5;
                  sendOutBCastPkt(p);
        }
        if((mh->purpose == TOPOLOGY_REQUEST)) {
              int location = strcmp(rp->prev_hop_address, addr_proto[0]);
          if(strncmp(rp->prev_hop_address, addr_proto[0],3) == 0) {
                    if((((addr_proto[0][11] == '1') && (location < 0)) ||
((addr_proto[0][11] == '2') && (location > 0))) && ((rp-
>prev_hop_address[9] != '1') || (rp->prev_hop_address[11] ==
addr_proto[0][11])))
                    {
                          if(strncmp(upstream_addr,rp-
>prev_hop_address,2) != 0) {
                                strcpy(upstream_addr, rp-
>prev_hop_address);
                                upstream_node = rp->prev_hop;
                          }
                          else {
                                if(addr_proto[0][11] == '1') {
                                      int difference = strcmp(rp-
>prev_hop_address, upstream_addr);
                                      if(difference > 0) {
                                            strcpy(my_controller_head,
rp->prev_cntrl_address1);
```

```
                                                strcpy(upstream_addr, rp-
>prev_hop_address);
                                                upstream_node = rp->prev_hop;
                                          }
                                    }
                                    if(addr_proto[0][11] == '2') {
                                          int difference = strcmp(rp-
>prev_hop_address, upstream_addr);
                                          if (difference < 0) {
                                                strcpy(my_controller_head,
rp->prev_cntrl_address1);
                                                strcpy(upstream_addr, rp-
>prev_hop_address);
                                                upstream_node = rp->prev_hop;
                                          }
                                    }
                              }
                              rp->prev_hop = myaddr_;
                              iph->daddr() = IP_BROADCAST;
                              strcpy(rp->prev_hop_address, addr_proto[0]);
                              if(rp->seqnumber > rseqnumber) {
                                    sendOutBCastPkt(p);
                                    rseqnumber = rp->seqnumber;
                              }
                        }
                  }
            }

            if(mh->purpose == DESTINATION_PACKET) {
                  //printf("DESTINATION PACKET : coming to right place %s %s
\n", addr_proto[0], rp->destination_haddress);
                  if((strncmp(addr_proto[0],rp->destination_haddress,3)==0) &&
(addr_proto[0][11] == rp->destination_haddress[11])) {
                        if(strncmp(addr_proto[0],rp->destination_haddress,8)
< 0) {
                              if(strncmp(upstream_addr,downstream_addr,8) >
0) {
                                    mh->purpose = DESTINATION_UP;
                                    iph->daddr() = upstream_node;
                                    //printf("sending up first \n");
                                    sendOutBCastPkt(p);
                                    return;
                              }
                              else {
                                    mh->purpose = DESTINATION_DOWN;
                                    iph->daddr() = downstream_node;
                                    sendOutBCastPkt(p);
                                    return;
                              }
                        }
                        else {
                              if(strncmp(upstream_addr,downstream_addr,8) <
0) {
                                    mh->purpose = DESTINATION_UP;
```

```
                                iph->daddr() = upstream_node;
                                sendOutBCastPkt(p);
                                return;
                        }
                        else {
                                mh->purpose = DESTINATION_DOWN;
                                iph->daddr() = downstream_node;
                                printf("sending down second TO %d\n",
iph->daddr());
                                sendOutBCastPkt(p);
                                return;
                        }
                }
        }
        else {
                mh->purpose = DESTINATION_UP;
                iph->daddr() = upstream_node;
                sendOutBCastPkt(p);
                return;
        }
 }
 /*
* If the address prefix not same then send to controller
*/
 if((mh->purpose == DESTINATION_UP) || (mh->purpose ==
TOPOLOGY_DISCOVERY) || (mh->purpose == TOPOLOGY_DISCOVERY_PROP) || (mh-
>purpose == DESTINATION_UP_SEC))
 {
        char dest[20];
        if(mh->purpose == DESTINATION_UP_SEC)
        strcpy(dest, rp->dummy_destination_haddress);
        else strcpy(dest, rp->destination_haddress);

        //printf("UP:I am %d and the destination is %s \n",
myaddr_, dest);
        rp->scope = rp->scope + 1;
        if(strcmp(addr_proto[0],dest)==0)
                printf("My packet ... I received it ... \n");
        else {
                iph->daddr() = upstream_node;
                //printf("forwarding %d number %d\n", myaddr_, rp-
>seqnumber);
                sendOutBCastPkt(p);
                return;
        }
 }

 if((mh->purpose == DESTINATION_DOWN) || (mh->purpose ==
DESTINATION_DOWN_SEC)) {
        char dest[20];
   if(mh->purpose == DESTINATION_UP_SEC)
                strcpy(dest, rp->dummy_destination_haddress);
   else strcpy(dest, rp->destination_haddress);
```

```
                    printf("DOWN:I am %d and the destination is %s \n",
myaddr_, dest);
                    if(strcmp(addr_proto[0],dest)==0)
                          printf("My packet ... I received it ... \n");
                    else {
                          iph->daddr() = downstream_node;
                          sendOutBCastPkt(p);
                          return;
                    }
              }
        return;
}

void SnetRout_Agent::controller1_node (Packet * p)
{
        float recv_time;
        struct hdr_mm* mh = hdr_mm::access(p);
        struct hdr_snetrout_pkt* rp = hdr_snetrout_pkt::access(p);
        snet_rtable_ent *prte;
        snet_rtable_ent rte;
        snet_rtable_ent rte1;
      hdr_ip *iph = HDR_IP(p);
        hdr_cmn *cmh = HDR_CMN(p);
        double p5,q5,r5;
        if(mh->purpose == UDP_FLOW) {
                    rp->prev_hop = myaddr_;
                    iph->saddr() = myaddr_;
                    iph->daddr() = upstream_node_controller[0];
                    rp->time_stamp =  Scheduler::instance().clock();
                    FILE* fdelay1;
                    char name_of_file1 [300];
                    if(rp->time_stamp > 200) {
                          sprintf(name_of_file1,
"Type_%s.init_delay.Rate_%d.Run_%d.Final_%d", type_exp_, rate_, run_,
iph->saddr());
                          fdelay1 = fopen(name_of_file1,"w");
                          fprintf(fdelay1," %f %f\n",
Scheduler::instance().clock() - rp->time_stamp,
Scheduler::instance().clock());
                          fclose(fdelay1);
                    }
              rp->time_stamp1 = Scheduler::instance().clock();
                    //printf("controller %d send msg to %d at
%f\n",myaddr_,upstream_node_controller[0],Scheduler::instance().clock()
);
                    sendOutBCastPkt(p);
        }
      if(mh->purpose == INITIATING_WARMUP) {
                printf("STARTING GURUDEV FROM %d \n", myaddr_);
                mult_addr_send = 0;
             while(mult_addr_send < number_of_highways) {
                  Packet *p;
                  p = allocpkt();
                  struct hdr_mm* mh = hdr_mm::access(p);
```

```
                struct hdr_snetrout_pkt* rp =
hdr_snetrout_pkt::access(p);
                hdr_ip *iph = HDR_IP(p);
                hdr_cmn *cmh = HDR_CMN(p);
                rp->seqnumber = rseqnumber;
                rp->prev_hop = myaddr_;
                strcpy(rp-
>prev_hop_address,addr_proto[mult_addr_send]);
                strcpy(rp->prev_cntrl_address1,
addr_proto[mult_addr_send]);
                iph->daddr() = IP_BROADCAST;
                mh->purpose = 1101;
                iph->saddr() = myaddr_;
                rseqnumber = rp->seqnumber;
                sendOutBCastPkt(p);
                mult_addr_send++;
               }
      }
      /*
       * Function  called when the incoming packet is of type TOPOLOGY
REQUEST and was initiated by someone else
       */
      if((mh->purpose == TOPOLOGY_REQUEST) && (iph->saddr() !=
myaddr_)) {
          int seen_controller_flag;
          int address_number;
          int controller_found_flag;
          int location;
              printf("CONTROLLER:TOPO_REQUEST:SEEING PKT %d \n",
myaddr_);
              for(address_number=0;
address_number<max_seg_intersection; address_number++) {
                    if(strncmp(addr_proto[address_number], rp-
>prev_cntrl_address1,3) == 0) {
                          location = strcmp(rp->prev_hop_address,
addr_proto[address_number]);
                          break;
                    }
              }
              if(((rp->prev_hop_address[11] == '1') && (location < 0))
|| ((rp->prev_hop_address[11] == '2') && (location > 0))) {
                    printf("CONT:TOPORQT: I SEE THE PKT am %d from %d
COUNT %d\n", myaddr_, rp->prev_hop, ncount);
                    /*
                     * If it is the first packet just add it and update
the table of neighbors
                     */
                    if(rp->seqnumber == 1) {
                          for(i=0; i<max_seg_intersection; i++) {
                                if(strcmp(neighboring_controller[i],
rp->prev_cntrl_address1) == 0) {
                                      seen_controller_flag = SET;
                                      strcpy(neighboring_controller[i],
rp->prev_cntrl_address1);
```

```
                                        nseqnumber[i] = rp->seqnumber;
                                        upstream_node_controller[i] = rp-
>prev_hop;

strcpy(upstream_addr_controller[i], rp->prev_hop_address);
                                  }
                            }
                            if(seen_controller_flag != SET) {
                                  strcpy(neighboring_controller[ncount],
rp->prev_cntrl_address1);
                                  nseqnumber[ncount] = rp->seqnumber;
                                  upstream_node_controller[ncount] = rp-
>prev_hop;

strcpy(upstream_addr_controller[ncount], rp->prev_hop_address);
                                  ncount++;
                            }
                      }
                      /*
                       * If it is not the first packet then we have to
check if the packet was delivered by a  closer sensor this time and if
so update the table of neighbor
                       */
                      else {
                            for(i=0; i<max_seg_intersection; i++) {
                                  if(strcmp(neighboring_controller[i],
rp->prev_cntrl_address1) == 0) {
                                        if(rp->prev_hop_address[11] ==
'1') {
                                              int difference = strcmp(rp-
>prev_hop_address, upstream_addr_controller[i]);
                                              if (difference > 0) {

strcpy(my_controller_head, rp->prev_cntrl_address1);

strcpy(upstream_addr_controller[i], rp->prev_hop_address);

upstream_node_controller[i] = rp->prev_hop;
                                              }
                                        }
                                        if(rp->prev_hop_address[11] ==
'2') {
                                              int difference = strcmp(rp-
>prev_hop_address, upstream_addr_controller[i]);
                                              if (difference < 0) {

strcpy(my_controller_head, rp->prev_cntrl_address1);

strcpy(upstream_addr_controller[i], rp->prev_hop_address);

upstream_node_controller[i] = rp->prev_hop;
                                                    // dbags printf("I am
%d and up is %d \n", myaddr_, rp->prev_hop);
                                              }
```

```
                                        }
                                        controller_found_flag = SET;
                                        iph->daddr() =
upstream_node_controller[i];
                                }
                        }
                                /*
                                 * If that particular neighboring
cntroller has never sent me a packet before then I will store his entry
in the neighbor tabe
                                 */
                        if(controller_found_flag != SET) {
                                printf(" TWO: NH MI ME %d %d %d\n",
number_of_highways, max_seg_intersection, myaddr_);
                                strcpy(neighboring_controller[ncount],
rp->prev_cntrl_address1);
                                nseqnumber[ncount] = rp->seqnumber;
                                upstream_node_controller[ncount] = rp-
>prev_hop;

strcpy(upstream_addr_controller[ncount], rp->prev_hop_address);
                                printf("I AM ... GURUDEV %d %s seeing
pkt from %d %s COUNT %d\n", myaddr_, addr_proto[0],
upstream_node_controller[ncount], upstream_addr_controller[ncount],
ncount);
                                ncount++;
                        }
                }
                /*
                 * All the updates for neighbor tables are done and
we have to now send the RESPONSE packet
                 */
                printf("CONT:TOPORQT: ME %d COUNT %d  VALUES %d %d
%d %d \n", myaddr_, ncount, upstream_node_controller[0],
upstream_node_controller[1], upstream_node_controller[2],
upstream_node_controller[3]);
                rp->prev_hop = myaddr_;
                mh->purpose = TOPOLOGY_RESPONSE;
                for(i=0; i<max_seg_intersection; i++) {
                                /*
                                 * Check which address matches
the previous hop and send that address  in  the packet so that it will
propagate
                                 */
                        if(strncmp(addr_proto[i], rp-
>prev_hop_address,3) == 0) {
                                strcpy(rp->prev_hop_address,
addr_proto[i]);
                                rp->pointer_to_address = 0;
                                rp->phy_address[rp->pointer_to_address]
= myaddr_;
                                strcpy(rp->address_topology[rp-
>pointer_to_address], addr_proto[i]);
```

```
                                    node_ =
(MobileNode*)Node::get_node_by_address(myaddr_);
                                    node_->getLoc(&p5, &q5, &r5);
                                    rp->xcor[rp->pointer_to_address] = p5;
                                    rp->ycor[rp->pointer_to_address] = q5;
                              }
                        }
                        for(i=0; i<max_seg_intersection; i++) {
                              if(strcmp(rp->prev_cntrl_address1,
neighboring_controller[i]) == 0) {
                                    if(rp->seqnumber > nseqnumber[i]) {
                                          if(myaddr_ == 0)
                                          printf("CONT:TOPORQT: SENDING
RESPONSE ME %d %s\n", myaddr_, rp->address_topology[rp-
>pointer_to_address]);
                                          sendOutBCastPkt(p);
                                          nseqnumber[i] = rp->seqnumber;
                                    }
                              }
                        }
                  }
      }
      /*
       * This function is executed  if the packet type is TOPOLOGY
RESPONSE
       */
    if((mh->purpose == TOPOLOGY_RESPONSE) && (myaddr_ != rp-
>phy_address[0])) {
            printf("CONT:TOPORESP: FROM %d TO %d ME %d PREV %d RESP
COUNT %d\n", rp->phy_address[0], myaddr_, myaddr_, rp->prev_hop,
controller_array_count);
            int max_sensor_depth;
            max_sensor_depth = 0;
            for(i=0; i<nodeNum; i++) {
                  if(rp->address_topology[i][0] != 'I')
                        break;
                  else max_sensor_depth = i+1;
            }
            controller_array_found = 1;
                  /*
                   * Check and find whetehr you have received response
from this neighboring controller before
                   * If you have then just update the table entry and
if not then create a new entry
                   */
            for(i=0; i<max_seg_intersection; i++) {
                  if(strcmp(controller_local_topology[i][0], rp-
>address_topology[0]) == 0) {
                        controller_array_found = SET;
                        temp_i = i;
                break;
                  }
            }
```

```
            if(controller_array_found == SET && controller_array_count
< 4) {
                  /* UPDATE */
                  max_seg_depth[temp_i] = max_sensor_depth;
                  for(i=0; i< max_sensor_depth; i++) {
                        strcpy(controller_local_topology[temp_i][i],rp-
>address_topology[i]);
                        controller_local_topology_phy[temp_i][i] = rp-
>phy_address[i];
                        controller_xcor[temp_i][i] = rp->xcor[i];
                        controller_ycor[temp_i][i] = rp->ycor[i];
                  }
            }
            if(controller_array_found != SET && controller_array_count
< 4) {
                  /* ADD A NEW ENTRY */
                  max_seg_depth[controller_array_count] =
max_sensor_depth;
                  for(i=0; i< max_sensor_depth; i++) {

      strncpy(controller_local_topology[controller_array_count][i],rp-
>address_topology[i],15);

      controller_local_topology_phy[controller_array_count][i] = rp-
>phy_address[i];
                        controller_xcor[controller_array_count][i] =
rp->xcor[i];
                        controller_ycor[controller_array_count][i] =
rp->ycor[i];
                  }
                  controller_array_count++;
            }
            printf("CONT:TOPORESP: ME %d COUNT %d  VALUES %d %d %d %d
\n", myaddr_, ncount, upstream_node_controller[0],
upstream_node_controller[1], upstream_node_controller[2],
upstream_node_controller[3]);
            printf("CONT:TOPORESP: ME %d COUNT %d \n", myaddr_,
controller_array_count);

            for(j=0; j<controller_array_count; j++) {
                  for(i=max_seg_depth[j]; i>max_seg_depth[j] - 3; i--)
{
                        printf("%d ",
controller_local_topology_phy[j][i]);
                  }
                  printf(" %f \n", Scheduler::instance().clock());
            }
            for(i=0; i<ncount; i++) {
                  printf("%d ", upstream_node_controller[i]);
            }
            printf("COUNT %d \n", ncount);
      }
      /*
```

```
       * If the controller finds the node in its table then it can send
it to the next hop
       */
      if(mh->purpose == DESTINATION_DOWN || mh->purpose ==
DESTINATION_UP || mh->purpose == DESTINATION_DOWN_SEC || mh->purpose ==
DESTINATION_UP_SEC) {
            char dest[20];
            if(mh->purpose == DESTINATION_UP_SEC || mh->purpose ==
DESTINATION_DOWN_SEC)
                  strcpy(dest, rp->dummy_destination_haddress);
            else strcpy(dest, rp->destination_haddress);
            /*
            * If destination on either of segments of controller
            */
            printf("DESTINATION PACKET: Seeinf dest_down packet %s \n",
addr_proto[0]);
            for(i=0; i<2; i++) {
                  if((strncmp(addr_proto[i],dest,3)==0)) {
                        printf("DESTINATION PACKET: found match with
one address %d ... I am %s and %d sending to %s \n", i, addr_proto[i],
myaddr_, dest);
                if(strncmp(addr_proto[i],dest,8) < 0) {
                              for(j=0;j<4;j++) {
                                    if((strncmp(addr_proto[i],
upstream_addr_controller[j],3) == 0) && (strncmp(addr_proto[i],
upstream_addr_controller[j],8) < 0)) {
                                          printf("CONT:DESTUP \n");
                                          iph->daddr() =
upstream_node_controller[j];
                                          if(mh->purpose ==
DESTINATION_UP_SEC || mh->purpose == DESTINATION_DOWN_SEC)
                                                mh->purpose =
DESTINATION_UP_SEC;
                                          else mh->purpose =
DESTINATION_UP;
                                          sendOutBCastPkt(p);
                                          return;
                                    }
                              }
                        }
                        else {
                              for(j=0;j<4;j++) {
                                    if((strncmp(addr_proto[i],
upstream_addr_controller[j],3) == 0) && (strncmp(addr_proto[i],
upstream_addr_controller[j],8) > 0)) {
                                          printf("CONT:DESTUP2 \n");
                                          iph->daddr() =
upstream_node_controller[j];
                                          if(mh->purpose ==
DESTINATION_UP_SEC || mh->purpose == DESTINATION_DOWN_SEC)
                                                mh->purpose =
DESTINATION_UP_SEC;
                                          else mh->purpose =
DESTINATION_UP;
```

```
                                        sendOutBCastPkt(p);
                                        return;
                                  }
                            }
                      }
                }
          }
          /*
          * CHECK FOR ENTRIES ALREADY IN THE  TABLE
          */
          FILE *fk;
          char read_cont[20]; int prev_hop, seq_num, hops; int c;
          if(prte = table_->GetEntry(2, dest)) {
                bcopy(prte, &rte, sizeof(rte));
                //printf("DESTINATION PACKET : FOUND AN ENTRY ... OFF
YOU GO ... I am %s and %d sending to %s and %d\n",addr_proto[0],
myaddr_, dest, rte.prev_hop);
                iph->daddr() = rte.prev_hop;
                if(mh->purpose == DESTINATION_UP_SEC || mh->purpose
== DESTINATION_DOWN_SEC)
                      mh->purpose = DESTINATION_DOWN_SEC;
                else mh->purpose = DESTINATION_DOWN;
                sendOutBCastPkt(p);
                return;
          }
          printf("CHECKING MY POWERS %s %d \n", addr_proto[0],
myaddr_);
          int flag_routing;
          flag_routing = 2;
          for(int i = 0; i < 10; i++) {
                if(strncmp(rp->destination_haddress,
reachable_highways[i], 3) == 0) {
                      flag_routing = 1;
                      printf("I am able to go it %s %d %s \n",
addr_proto[0], myaddr_, gateway_address[i]);
                      strcpy(rp->dummy_destination_haddress,
gateway_address[i]);
                      mh->purpose = DESTINATION_DOWN_SEC;
                      break;
                }
          }
          if(flag_routing == 1) {
                if(prte = table_->GetEntry(2, rp-
>dummy_destination_haddress)) {
                      bcopy(prte, &rte, sizeof(rte));
                      iph->daddr() = rte.prev_hop;
                      sendOutBCastPkt(p);
                      return;
                }
          }
          /*
          * If no entry OR match found then route the packet to the
higher level...
          */
```

```
            if(prte = table_->GetEntry(2, head_addr_)) {
                  bcopy(prte, &rte, sizeof(rte));
                  iph->daddr() = rte.prev_hop;
                  mh->purpose = DESTINATION_DOWN_SEC;
                  strcpy(rp->dummy_destination_haddress, head_addr_);
                  sendOutBCastPkt(p);
                  return;
            }
      }
      /*
      * Sending out discovery message
      */
      if(mh->purpose == TOPOLOGY_DISCOVERY) {
            if(rp->version_val == HIERARCHICAL_HEAD_DISCOVERY &&
heard_head_flag != 1) {
                  Packet::free(p);
                  return;
            }
            double xcor, ycor, zcor;
        node_ = (MobileNode*)Node::get_node_by_address(myaddr_);
        node_->getLoc(&xcor, &ycor, &zcor);
            strcpy(rp->source_haddress, addr_proto[0]);
            rp->source_xcor = xcor;
            rp->source_ycor = ycor;
            rp->dest_xcor = destination_xcor;
            rp->dest_ycor = destination_ycor;
            rec_seqnumber  = rec_seqnumber + 1;
            printf("CONT:TOPODISC:I am %s starting discovery WITH %d at
%f send %d %d %d %d \n", addr_proto[0],rec_seqnumber,
Scheduler::instance().clock(), upstream_node_controller[0],
upstream_node_controller[1], upstream_node_controller[2],
upstream_node_controller[3]);
            for(i=0; i<ncount; i++) {
                  for(int k = 0; k < 2; k++) {
                        if(strcmp(addr_proto[k],"BULL") != 0) {
                              int temp = 0;
                              Packet *p1 = allocpkt ();
                              New_Packet(p, addr_proto[k], p1, 0, i,
head_addr_, rec_seqnumber);
                              /*
                               * Put addresses of reachable highways
                               */
                              struct hdr_snetrout_pkt* rp1 =
hdr_snetrout_pkt::access(p1);
                              if(rp1->version_val ==
HIERARCHICAL_HEAD_DISCOVERY) {
                                    for(int i = 0; i < 10; i++) {
                                          strncpy(rp1-
>reachable_address[i], reachable_highways[i], 3);
                                          printf(" <- %s ", rp1-
>reachable_address[i]);
                                    }
                              }
                              sendOutBCastPkt(p1);
```

```
                        }
                  }
            }
      }
      /*
      * Forward the Topology discovery message
      */
      if(mh->purpose == TOPOLOGY_DISCOVERY_PROP) {
            if(rp->version_val == HIERARCHICAL_HEAD_BCAST) {
                  int diff;
                  diff = difference(addr_proto[0], rp-
>source_haddress);
                  return_value = hierarchical_head_bcast(rp-
>source_xcor, rp->source_ycor, rp->source_haddress);
                  if((return_value == 1) && (diff == 1)) {
                        heard_head_flag = 1;
                        destination_xcor = rp->source_xcor;
                        destination_ycor = rp->source_ycor;
                        strcpy(head_addr_, rp->source_haddress);
                        printf("CONTROLLER:TOPODISCPROP: HHB ME %s with
%s %f %f \n", addr_proto[0], rp->source_haddress, destination_xcor,
destination_ycor);
                  }
                  else {
                        if(return_value !=1) {
                              Packet::free(p);
                              return ;
                        }
                  }
            }
            if(rp->version_val == HIERARCHICAL_HEAD_DISCOVERY) {
                  int i_;
                  printf("CONTROLLER:STARTING DISCOVERY AT %s FROM %s
FOR %s at %f\n", addr_proto[0], rp->source_haddress, rp-
>destination_haddress, Scheduler::instance().clock());
                  if((strcmp(addr_proto[0],rp->destination_haddress) ==
0) || (strcmp(addr_proto[1],rp->destination_haddress)== 0)) {
                        printf("GURUDEV FOUND IT: I am %s %d from %s
\n", addr_proto[0], myaddr_, rp->source_haddress);
                        /*
                         * The below part of code updates the reachable
highways in the array
                         */
                        int flag = 1;
                        for(int i = 0; i<10; i++) {
                              if(strncmp(rp->reachable_address[i],
"NUL", 3) == 0)
                              break;
                              for(i_ = 0; i_<10; i_++) {
                                    if(strncmp(reachable_highways[i_],
rp->reachable_address[i], 3) == 0) {
                                          flag = 2;
                                          break;
                                    }
```

```
                                    if(strncmp(reachable_highways[i_],
"NUL", 3) == 0) {
                                          flag = 1;
                                          break;
                                    }
                              }
                              if(flag == 1) {
                                    strncpy(reachable_highways[i_], rp-
>reachable_address[i], 3);
                                    strcpy(gateway_address[i_], rp-
>source_haddress);
                              }
                        }
                        flag = 1;
                        for(i_ = 0; i_<10; i_++) {
                              if(strncmp(reachable_highways[i_], rp-
>source_haddress, 3) == 0) {
                                    flag = 2;
                                    break;
                              }
                              if(strncmp(reachable_highways[i_], "NUL",
3) == 0) {
                                    flag = 1;
                        break;
                              }
                        }
                        if(flag == 1) {
                              strncpy(reachable_highways[i_], rp-
>source_haddress, 3);
                        }
                        for(int i = 0; i<10; i++) {
                              if(strncmp(reachable_highways[i], "NUL",
3) != 0)
                                    printf(" %s %d -->> %s : ",
reachable_highways[i], i, gateway_address[i]);
                              else break;
                        }
                  }
                  return_value = hierarchical_head_discovery(rp-
>source_xcor, rp->source_ycor, rp->dest_xcor, rp->dest_ycor);
            }
            printf("I am %s return value in the final func is %d \n",
addr_proto[0], return_value);
            if(return_value == 1) {
                  FILE *fk;
                  char read_cont[20]; int prev_hop, seq_num, hops; int
c;
                  printf("i saw discovery at %f at %s \n",
Scheduler::instance().clock(), addr_proto[0]);
                  if(prte = table_->GetEntry1(2, rp->source_haddress))
{
                        bcopy(prte, &rte, sizeof(rte));
                        printf("FOUND AN ENTRY for %s ... I am %s and
%d\n",rp->source_haddress, addr_proto[0], myaddr_);
```

```
                        if((rp->seqnumber > rte.sequencenumber) ||
((rp->seqnumber == rte.sequencenumber) && (rte.distance > rp->scope)))
{
                                strcpy(rte1.lname, rp->source_haddress);
                                rte1.sequencenumber = rp->seqnumber;
                                rte1.prev_hop = rp->prev_hop;
                                rte1.distance = rp->scope;
                                printf("Jus b4 ADDING %s %d %d \n",
rte1.lname, rte1.sequencenumber, rte1.prev_hop);
                                table_->AddEntry(rte1);
                                printf("i am addding %s \n",
addr_proto[0]);
                                for(i=0; i<4; i++) {
                                      Packet *p1 = allocpkt ();
                                      New_Packet(p, rp->source_haddress,
p1, rp->scope+1, i, rp->destination_haddress, rp->seqnumber);
                                      struct hdr_snetrout_pkt* rp1 =
hdr_snetrout_pkt::access(p1);
                                      for(int i = 0; i<10; i++) {
                                            strncpy(rp1-
>reachable_address[i], rp->reachable_address[i], 3);
                                      }
                                      sendOutBCastPkt(p1);
                                }
                        }
                  }
                  else {
                        strcpy(rte1.lname, rp->source_haddress);
                        rte1.sequencenumber = rp->seqnumber;
                        rte1.prev_hop = rp->prev_hop;
                        rte1.distance = rp->scope;
                        printf("jus b4 ADDING %s %d %d \n", rte1.lname,
rte1.sequencenumber, rte1.prev_hop);
                        table_->AddEntry(rte1);
                        for(i=0; i<4; i++) {
                              Packet *p1 = allocpkt ();
                              New_Packet(p, rp->source_haddress, p1,
rp->scope+1, i, rp->destination_haddress, rp->seqnumber);
                              struct hdr_snetrout_pkt* rp1 =
hdr_snetrout_pkt::access(p1);
                              for(int i = 0; i<10; i++) {
                                    strncpy(rp1->reachable_address[i],
rp->reachable_address[i], 3);
                              }
                              sendOutBCastPkt(p1);
                        }
                  }
            }
      }
return;
}
/*
 * This function creates new packet according to the passed arguments
 */
```

```
void SnetRout_Agent::New_Packet (Packet * p, char source[20], Packet *
p1, int scope, int count, char discovery_dest[20], int seqnumber_)
{
      struct hdr_mm* mh1 = hdr_mm::access(p1);
      struct hdr_snetrout_pkt* rp1 = hdr_snetrout_pkt::access(p1);
      struct hdr_snetrout_pkt* rp = hdr_snetrout_pkt::access(p);
      hdr_ip *iph1 = HDR_IP(p1);
      mh1->purpose = TOPOLOGY_DISCOVERY_PROP;
      if(rp->version_val == HIERARCHICAL_HEAD_DISCOVERY)
            strcpy(rp1->destination_haddress,discovery_dest);
      else
            strcpy(rp1->destination_haddress,"BROADCAST_MSG");
      iph1->daddr() = upstream_node_controller[count];
      rp1->seqnumber = seqnumber_;
      rp1->version_val = rp->version_val;
      rp1->source_xcor = rp->source_xcor;
      rp1->source_ycor = rp->source_ycor;
      rp1->dest_xcor = rp->dest_xcor;
      rp1->dest_ycor = rp->dest_ycor;
      rp1->scope = scope;
      strcpy(rp1->source_haddress,source);
      return;
}
/*
 * Calculate the difference in level of the packet originator and the
current node
 */
int SnetRout_Agent::difference(char address1[20], char address2[20])
{
      int first, second;
      switch(address1[9]) {
            case '1': first = 1; break;
            case '2': first = 2; break;
            case '3': first = 3; break;
            case '4': first = 4; break;
            case '5': first = 5; break;
      }
      switch(address2[9]) {
         case '1': second = 1; break;
         case '2': second = 2; break;
         case '3': second = 3; break;
         case '4': second = 4; break;
         case '5': second = 5; break;
    }
      return (second - first);
}

/*
 * Checking the hierarchical bcast limits
 */
int SnetRout_Agent::hierarchical_head_bcast(double xcor, double ycor,
char source_address[20])
{
      int xlim, ylim;
```

```
	if(source_address[9] == '3')
		{
			xlim = 5000;
			ylim = 800;
		}
	if(source_address[9] == '4')
		{
			xlim = 5000;
			ylim = 5000;
		}

	printf("he source is %f %f %s \n", xcor, ycor, source_address);
	double xcor2, ycor2, zcor2;
	node_ = (MobileNode*)Node::get_node_by_address(myaddr_);
	  node_->getLoc(&xcor2, &ycor2, &zcor2);
	int i_xcor, i_xcor2, i_ycor, i_ycor2;
	i_xcor = xcor; i_xcor2 = xcor2; i_ycor = ycor; i_ycor2 = ycor2;
	if(abs(i_xcor2 - i_xcor) < xlim && abs(i_ycor2 - i_ycor) < ylim)
		{
			return 1;
		}
	else return 0;
}

/*
 * This function tells whetehr the hierarchical head discovery message
should be forwarded by this node or not
 * Basically providing functionality like Location Aided Routing
 */
int SnetRout_Agent::hierarchical_head_discovery(double xcor, double
ycor, double xcor1, double ycor1)
{

	double xcor2,ycor2,zcor2;
	node_ = (MobileNode*)Node::get_node_by_address(myaddr_);
	node_->getLoc(&xcor2, &ycor2, &zcor2);
	printf("values are %f %f %f %f %f %f\n", xcor,xcor1,xcor2, ycor,
ycor1, ycor2);
	if(xcor1 > xcor)
		{
			xcor = xcor - 200;
			xcor1 = xcor1 + 200;
		}
	else
		{
			xcor = xcor + 200;
				xcor1 = xcor1 - 200;
		}
	  if(ycor1 > ycor)
			{
				ycor = ycor - 200;
				ycor1 = ycor1 + 200;
			}
	  else
```

```
                {
                        ycor = ycor + 200;
                        ycor1 = ycor1 - 200;
                }
      printf("values are %f %f %f %f %f %f\n", xcor,xcor1,xcor2, ycor,
ycor1, ycor2);
      if((((xcor2 < xcor) && (xcor2 > xcor1)) || ((xcor2 > xcor) &&
(xcor2 < xcor1)))  && (((ycor2 < ycor) && (ycor2 > ycor1)) || ((ycor2 >
ycor) && (ycor2 < ycor1))))
            {
                  printf("appropriate \n");
                  return 1;
            }
      else
            {
                  printf("out of range \n");
                  return 0;
            }
}



static class SnetRoutClass:public TclClass
{
  public:
  SnetRoutClass ():TclClass ("Agent/SnetRout") {}
  TclObject *create (int, const char *const *) {
    return (new SnetRout_Agent ());
  }
} class_snetrout;

SnetRout_Agent::SnetRout_Agent (): Agent (PT_MESSAGE), ll_queue (0),
seqno_ (0), myaddr_ (0), subnet_ (0), node_ (0), port_dmux_(0),
  periodic_callback_ (0), be_random_ (1), use_mac_ (0), verbose_ (1),
trace_wst_ (0), lasttup_ (-10), alpha_ (0.875),  wst0_ (6), perup_
(15),
  min_update_periods_ (3) // constants
{
  table_ = new snet_RoutingTable ();
  //  helper_ = new SnetRout_Helper (this);
  // trigger_handler = new SnetRoutTriggerHandler(this);
  controller_array_count = 0;
  bind_time ("wst0_", &wst0_);
  bind_time ("perup_", &perup_);
  bind ("use_mac_", &use_mac_);
  bind ("be_random_", &be_random_);
  bind ("alpha_", &alpha_);
  bind ("min_update_periods_", &min_update_periods_);
  bind ("verbose_", &verbose_);
  bind ("trace_wst_", &trace_wst_);
  //DEBUG
  for(int k=0; k<5; k++)
        strcpy(addr_proto[k],"BULL");
  for(int k=0; k<10; k++)
```

```
        strcpy(reachable_highways[k], "NUL");
  heard_head_flag = 0;
  address = 0;
}

void SnetRout_Agent::startUp()
{
}

int SnetRout_Agent::command (int argc, const char *const *argv)
{
  if (argc == 2) {
      if (strcmp (argv[1], "start-snetrout") == 0) {
            startUp();
            return (TCL_OK);
      }
    else if (strcmp (argv[1], "dumprtab") == 0) {
            Packet *p2 = allocpkt ();
            hdr_ip *iph2 = HDR_IP(p2);
            snet_rtable_ent *prte;
            printf ("Table Dump %d[%d]\n------------------------------
---\n",iph2->saddr(), iph2->sport());
            trace ("VTD %.5f %d:%d\n", Scheduler::instance ().clock
(),iph2->saddr(), iph2->sport());
            // Freeing a routing layer packet --> don't need to
            // call drop here.
            Packet::free (p2);
            for (table_->InitLoop (); (prte = table_->NextLoop ());)
              //   output_rte ("\t", prte, this);
            printf ("\n");
          return (TCL_OK);
      }
  }
  else if (argc == 3) {
        if(strcasecmp(argv[1], "delay-specs") == 0) {
            delay_spec_=atoi(argv[2]);
                  return TCL_OK;
            }
            else if(strcasecmp(argv[1], "Number_Highways") == 0) {
                  number_of_highways = atoi(argv[2]);
                  max_seg_intersection = 2 * number_of_highways;
                  printf("Number_of_Highway Maxima_Intersection ME %d
%d %d\n", number_of_highways, max_seg_intersection, myaddr_);
                  return TCL_OK;
            }
            else if(strcmp(argv[1], "type_exp") == 0) {
                  strcpy(type_exp_, argv[2]);
                  printf("SNETROUT: type %s \n", type_exp_);
                  return (TCL_OK);
            }
            else if(strcmp(argv[1], "rate-exp") == 0) {
                  rate_ = atoi(argv[2]);
                  printf("SNETROUT: rate %d \n", rate_);
                  return (TCL_OK);
```

```
            }
            else if(strcmp(argv[1], "run_exp") == 0) {
                  run_ = atoi(argv[2]);
                  printf("SNETROUT: run %d \n", run_);
                  return (TCL_OK);
            }
            else if(strcmp(argv[1], "head_addr") == 0) {
                  strcpy(head_addr_, argv[2]);
                  printf("SNETROUT: type %s \n", head_addr_);
                  return (TCL_OK);
            }
            if (strcasecmp (argv[1], "addr") == 0) {
                  myaddr_ = Address::instance().str2addr(argv[2]);
                  return TCL_OK;
            }
            TclObject *obj;
            if ((obj = TclObject::lookup (argv[2])) == 0) {
                  fprintf (stderr, "%s: %s lookup of %s failed\n",
__FILE__, argv[1],argv[2]);
                  return TCL_ERROR;
            }
            if (strcmp(argv[1], "attach-agent") == 0) {
                  agent1_ = (Agent*) TclObject::lookup(argv[2]);
                  return(TCL_OK);
            }
            if (strcasecmp (argv[1], "tracetarget") == 0) {
                  tracetarget = (Trace *) obj;
                  return TCL_OK;
            }
            else if (strcasecmp (argv[1], "node") == 0) {
                  node_ = (MobileNode*) obj;
                  return TCL_OK;
            }
            else if (strcasecmp (argv[1], "port-dmux") == 0) {
                  port_dmux_ = (NsObject *) obj;
                  return TCL_OK;
            }
            else if (strcasecmp (argv[1], "ll-queue") == 0) {
                  if (!(ll_queue = (PriQueue *) TclObject::lookup
(argv[2]))) {
                        fprintf (stderr, "SnetRout_Agent: ll-queue
lookup of %s failed\n", argv[2]);
                        return TCL_ERROR;
                  }
                  return TCL_OK;
            }
  }
  else if (argc == 4) {
            if(strcmp(argv[1], "dest-coordinates") == 0) {
                    destination_xcor = atoi(argv[2]);
                    destination_ycor = atoi(argv[3]);
                    return (TCL_OK);
            }
  }
```

```
  return (Agent::command (argc, argv));
}
```

A.1.4 TCL File

A.1.4.1 Main File "VII_sp.tcl"

```
# NS-2 TCL Script
# Yongchang Ma, Ph.D. Dissertation
# Clemson, SC 2007

#
# ======================================================================
# Default Script Options
# Basic configurations and model selection
#
# ======================================================================
set val(chan)           Channel/WirelessChannel    ;# channel type
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation
model
set val(netif)          Phy/WirelessPhy            ;# network interface
type
set val(mac)            Mac/802_11                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue
type
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(rp)             SnetRout                   ;# routing protocol
set val(adhocRouting)   AODV                       ;# ad hoc routing
protocol

set val(ifqlen)         100               ;# max # of packet in queue
set val(ps)             100               ;# packet size in bytes
set val(x)              6000              ;# X dimension of the
topography
set val(y)              6000              ;# Y dimension of the
topography
set val(stop)           1800.0            ;# simulation time
set val(fnnum)          20                ;# number of fixed nodes+1
set val(mnnum)          200               ;# number of mobile nodes
# calculate total number of nodes
set val(nn) [expr $val(fnnum)+$val(mnnum)]

set val(fnloc)             "./sp_fnLoc2.txt"    ;# locations of fixed
nodes
set val(fnadd)             "./sp_fnAdd2.txt"    ;# address of fixed
nodes
set val(fnFN)            "./sp_fnFN2.txt"       ;# filename of fixed
nodes
set val(mp)             "./nodeM"          ;# movement pattern of mobile
nodes
```

```
set opt(energymodel)    EnergyModel     ;
set opt(initialenergy)    5               ;# Initial energy in Joules

Phy/WirelessPhy set CSThresh_ 2.259e-11
Phy/WirelessPhy set RXThresh_ 3.652e-10
Phy/WirelessPhy set bandwidth_ 54e6

#
# ======================================================================
# Simulation Scripts
#
# ======================================================================

# Read parameters from input
set arg [lindex $argv 0]
set arg_1 [lindex $argv 1]
set arg_2 [lindex $argv 2]
set arg_3 [lindex $argv 3]
set arg_4 [lindex $argv 4]

# Initialize global variables
set ns_         [new Simulator]

# Set trace file and nam file
set tracefd     [open \\VII_trace\\temp_trace.tr w]
#set tracefd     [open
\\VII_trace\\VII_sp_Run$arg.Time$arg_2.Location$arg_3.Lane$arg_4.tr w]
set namtrace    [open VII_sp.nam w]
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

# Set up topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)

# Create god
create-god $val(nn)

# Set node configuration
$ns_ node-config -adhocRouting $val(rp) \
     -llType $val(ll) \
     -macType $val(mac) \
     -ifqType $val(ifq) \
     -ifqLen $val(ifqlen) \
     -antType $val(ant) \
     -propType $val(prop) \
     -phyType $val(netif) \
     -channel [new $val(chan)] \
     -topoInstance $topo \
     -agentTrace ON \
     -routerTrace OFF \
     -macTrace ON \
     -movementTrace OFF
```

```
# for all the nodes get the routing agent, queue agent and set the node
for {set i 0} {$i < $val(nn) } {incr i} {
    set node_($i) [$ns_ node]
    #$god_ new_node $node_($i)
    set ragent_($i) [$node_($i) get-ragent]
    set ifqq_($i) [$node_($i) get-ifq]
    $node_($i) random-motion 0;
}

# define color index
$ns_ color 0 red
$ns_ color 1 blue
$ns_ color 2 chocolate
$ns_ color 3 red
$ns_ color 4 brown
$ns_ color 5 tan
$ns_ color 6 gold
$ns_ color 7 black

for {set i 0} {$i < $val(nn)} {incr i} {
    # Set the UDP agent
    set udp_s($i) [new Agent/UDP/UDPsnet]
    $ns_ attach-agent $node_($i) $udp_s($i)
    $ragent_($i) attach-agent $udp_s($i)

    # Set packet size for UDP
    $udp_s($i) set packetSize_ $val(ps)

    # Set the appilcation agent
    set snet_s($i) [new Application/Snet]

    # Attach the application agent to the UDP agent
    $snet_s($i) attach-agent $udp_s($i)

    # Set the packet size for application agent
    $snet_s($i) set pktsize_ $val(ps)

    # This command is used to pass to the nodes application agent its
identity
    # Basically using this there is a varable in the C++ code of
snet.cc
    # that tells the node number
    $snet_s($i) set nodenumber_ $i

    $snet_s($i) set roundnumber_ $arg
}

# Define locations of fixed nodes
puts "Loading locations of controllers, sensors and repeaters..."
source $val(fnloc)

#  Define the initial locations and address of the mobile nodes
for {set i $val(fnnum)} {$i < $val(nn) } {incr i} {
      $node_($i) set X_  0.0
```

```
      $node_($i) set Y_  0.0
      $node_($i) set Z_ 0.0
      $node_($i) random-motion 0          ;# disable random motion
      $node_($i) color "blue"
      #puts "[expr $i%7]"
      $node_($i) shape "hexagon"
      $ns_ initial_node_pos $node_($i) 20
      $snet_s($i) addr IX1.0000.0.2
}

# Define address of fixed nodes
puts "Loading address of controllers, sensors and repeaters..."
source $val(fnadd)

# Define filename of fixed nodes
puts "Loading filename of controllers, sensors and repeaters..."
source $val(fnFN)

for {set i 0} {$i < $val(fnnum)} {incr i} {
      $node_($i) color red
      $node_($i) shape box
      $ns_ initial_node_pos $node_($i) 1

      $snet_s($i) time_duration 3 START 90
      $snet_s($i) time_duration 3 END 110
      $snet_s($i) time_duration 4 START 130
      $snet_s($i) time_duration 4 END 150
}

$ragent_(2) NUMBER_HIGHWAYS 4
$ragent_(19) NUMBER_HIGHWAYS 4

for {set i 0} {$i < $val(fnnum)} {incr i} {
      $ns_ at 0.0 "$snet_s($i) start"
      $ns_ at $val(stop).0 "$snet_s($i) stop"
}

# Tell nodes when the simulation ends
for {set i $val(fnnum)} {$i < $val(nn) } {incr i} {
    $ns_ at $val(stop).0 "$node_($i) reset";
}

#Define a 'finish' procedure
proc finish {} {
      global ns_ namtrace tracefd
      global simStart

      $ns_ flush-trace
      close $tracefd
      #Execute nam on the trace file
      #puts "running nam..."
      #exec ../nam-1.11/nam VII_sp.nam &
```

```
      set simEnd [clock seconds]           ;# end-time of the
simulation
      set execTime [expr $simEnd-$simStart]
      # display some statistics
      puts "Finishing ns... Execution time: $execTime seconds (End:
[clock format $simEnd -format {%d.%m.%y %H:%M:%S}])"
      exit 0                                    ;# ... and we're done
}

proc getthetime {} {
      set now [exec date]
      puts stdout "$now"
}

$ns_ at $val(stop).0001 "finish"

#puts $tracefd "M 0.0 nn $val(nn) x $val(x) y $val(y) rp
$val(adhocRouting)"
#puts $tracefd "M 0.0 sc $val(sc) cp $val(cp) seed $val(seed)"
#puts $tracefd "M 0.0 prop $val(prop) ant $val(ant)"

$ns_ at 0.0 "$ns_ set-animation-rate 150ms"
$ns_ run
```

A.1.4.2 Fixed Nodes Location File "sp_fnLoc.tcl"

```
$node_(0)     set     X_     0
$node_(0)     set     Y_     0
$node_(0)     set     Z_     0

$node_(1)     set     X_     1824.43
$node_(2)     set     X_     2018.98
$node_(1)     set     Y_     1320.73
$node_(2)     set     Y_     1334.76
$node_(1)     set     Z_     0
$node_(2)     set     Z_     0

$node_(3)     set     X_     2212.42
$node_(3)     set     Y_     1343.60
$node_(3)     set     Z_     0

$node_(4)     set     X_     2430.03
$node_(4)     set     Y_     1371.04
$node_(4)     set     Z_     0

$node_(5)     set     X_     2626.79
$node_(5)     set     Y_     1390.55
$node_(5)     set     Z_     0
```

```
$node_(6)    set    X_    2807.97
$node_(6)    set    Y_    1447.56
$node_(6)    set    Z_    0

$node_(7)    set    X_    2967.99
$node_(7)    set    Y_    1551.22
$node_(7)    set    Z_    0

$node_(8)    set    X_    3120.70
$node_(8)    set    Y_    1685.06
$node_(8)    set    Z_    0

$node_(9)    set    X_    3267.53
$node_(9)    set    Y_    1828.96
$node_(9)    set    Z_    0

$node_(10)   set    X_    3407.32
$node_(10)   set    Y_    1973.17
$node_(10)   set    Z_    0

$node_(11)   set    X_    3546.27
$node_(11)   set    Y_    2114.33
$node_(11)   set    Z_    0

$node_(12)   set    X_    3679.54
$node_(12)   set    Y_    2252.44
$node_(12)   set    Z_    0

$node_(13)   set    X_    3817.76
$node_(13)   set    Y_    2391.46
$node_(13)   set    Z_    0

$node_(14)   set    X_    3944.59
$node_(14)   set    Y_    2523.48
$node_(14)   set    Z_    0

$node_(15)   set    X_    4089.25
$node_(15)   set    Y_    2663.11
$node_(15)   set    Z_    0

$node_(16)   set    X_    4233.88
$node_(16)   set    Y_    2797.56
$node_(16)   set    Z_    0
```

```
$node_(17)    set    X_    4397.03
$node_(17)    set    Y_    2895.43
$node_(17)    set    Z_    0

$node_(18)    set    X_    4602.21
$node_(18)    set    Y_    2963.11
$node_(18)    set    Z_    0

$node_(19)    set    X_    4813.23
$node_(19)    set    Y_    3046.34
$node_(19)    set    Z_    0
```

A.1.4.3 Fixed Nodes Address File "sp_fnAdd.tcl"

```
$snet_s(0) addr IX1.0000.1.2
$snet_s(1) addr IX1.5095.1.2
$snet_s(2) addr IX1.5250.1.2
$snet_s(3) addr IX1.5400.1.2
$snet_s(4) addr IX1.5550.1.2
$snet_s(5) addr IX1.5700.1.2
$snet_s(6) addr IX1.5850.1.2
$snet_s(7) addr IX1.6007.1.2
$snet_s(8) addr IX1.6107.1.2
$snet_s(9) addr IX1.6207.1.2
$snet_s(10) addr IX1.6307.1.2
$snet_s(11) addr IX1.6407.1.2
$snet_s(12) addr IX1.6507.1.2
$snet_s(13) addr IX1.6625.1.2
$snet_s(14) addr IX1.6775.1.2
$snet_s(15) addr IX1.6900.1.2
$snet_s(16) addr IX1.7025.1.2
$snet_s(17) addr IX1.7175.1.2
$snet_s(18) addr IX1.7325.1.2
$snet_s(19) addr IX1.7450.2.2
```

A.2 Implementation of Integrated Simulation Platform in PARAMICS

The following sections present selected source codes for the implementation of integrated simulation platform in PARAMICS.

A.2.1 Plugin File for Traffic Condition Assessment

```
#include "c:\program Files\paramicsV5\programmer\include\programmer.h"
```

```
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <math.h>
#include <process.h>
#include <malloc.h>
#include <memory.h>
#include "svm.h"
#include "plugin.h"
#include "plugin_p.h"

int total_nodeNum=fixed_nodeNum+mobile_nodeNum;

typedef struct coordinate     coord;

static char *swi = "C:\\bang.txt";
static char *syn = "C:\\incident.txt";
static char *model_filename="C:\\queue4.scale.model";
static char *range_filename="C:\\range";
static char *nodeM_filename="C:\\nodeM";
static char *roundN_filename="C:\\VII_sim\\int_1line\\20\\roundnumber";
const int penetration=20;
const int demand =3200;
struct svm_model* model;
static char *enter_links[enter_linkNum]={"196:197"};
static char *exit_links[exit_linkNum]={"210:333","531:293"};
static float enterX[enter_linkNum]={1331747.00};
static float enterY[enter_linkNum]={12695591.00};

FILE *swi_file;
FILE *syn_file;
FILE *nodeM;
FILE *roundN;
coord *cur_loc;

static float inc_startTime=900;
static float rec_time=30;
static float left_x=1321840.00,down_y=12690320.00;
static char
*node[node_Num+2]={"196","197","332","210","294","292","531"};
static int node_LES[node_Num+2]={4121,4858,6898,8938,9214,10823,11960};
static char
*links[linkNum+2]={"196:197","197:332","332:210","210:294","294:292","2
92:531"};
static float
startNode_x[linkNum+2]={1331747.00,1332262.13,1333691.38,1335217.1,1335
306.38,1336683.25};
static float
startNode_y[linkNum+2]={12695591.00,12696118.00,12697574.00,12699136.0,
12699233.00,12700037.00};
static float
endNode_x[linkNum+2]={1332262.13,1333691.38,1335217.1,1335306.38,133668
3.25,1337784.25};
```

```
static float
endNode_y[linkNum+2]={12696118.00,12697574.00,12699136.0,12699233.00,12
700037.00,12700319.00};
static int arc[linkNum+2]={0,0,0,0,-1,0};
static float center_x[linkNum+2]={0,0,0,0,1337591.6,0};
static float center_y[linkNum+2]={0,0,0,0,12696928.0,0};
static float links_length[linkNum+2],Radius[linkNum+2];
static char *inc_link;
int counter=0;
int tagged=0;
int i,j,k,ab_cnt,i_t,printed;
double rep_ab[3][6];
float lnb0,lnb1,lnb2,check_time,check_pt;
float veh_stat[maxCounter][elem];
int veh_num[7];
float veh_time[7];
float rep_vehStatus_int=4;
int inc_distance;
struct svm_node *x;
struct veh_tt *vtt;
struct veh_tt *vtt_buffer;
struct road_stat *rst;
int max_nr_veh = 1000;
int max_nr_attr = 13;
errno_t err;
double lower=0.0,upper=1.0;
double *feature_max;
double *feature_min;
double *feature_max1;
double *feature_min1;
int max_index=12;
int max_index1=15;
int
in_cnt[7],out_cnt[7],vid,num_tt,vid_buffer,within[fixed_nodeNum+mobile_
nodeNum];
float avg_tt;
int warmup_time=57600+1200;
int startup_time=57600;
int clear;
int time_step=300,rec_nodeM_int=1;
float rdStatus[23][6][4];
int seg_length=4*40*3.28;
int laneBlock=1,inc_location;
Bool inc_occure1 = FALSE;
Bool inc_occure2 = FALSE;
Bool inc_occure3 = FALSE;
static int switch_startTime=300;
static int restartTime=1800;
static int Nveh_msg;


void qpx_NET_postOpen(void)
{
    FILE *fp=NULL;
```

```
	int idx,roundnumber_;
	double fmin, fmax;
	qps_GUI_printf("\n-------VII API--------\n");
	err = fopen_s(&roundN,roundN_filename,"r+");
	if(err!=0)  qps_GUI_printf("Cannot find file %s
\n",roundN_filename);
	else {
		fscanf_s(roundN,"%d",&roundnumber_);
		roundnumber_++;
		rewind(roundN);
		fprintf(roundN,"%d",roundnumber_);
	}
	if(roundN!=NULL) fclose(roundN);
	clear=(warmup_time-startup_time)/time_step;
	laneBlock=qpg_UTL_randomInteger(APIRNG_FREEWAYLANES,3)+1;
	laneBlock=2;
	inc_location=node_LES[0]+qpg_UTL_randomInteger(APIRNG_INCIDENT,(n
ode_LES[node_Num-1]-node_LES[0]));
	for(i=0; i<linkNum; i++) {
		if(inc_location<=node_LES[i+1]) {
			inc_link=links[i];
			inc_distance=(int)max(180,min((node_LES[i+1]-
inc_location),qpg_LNK_length(qpg_NET_link(inc_link))));
			break;
		}
	}
	qps_GUI_printf("round %d: %d lanes incident will occurre at %5.0f
on Link %s at distance %d feet from the end (LES:
%d);\n",roundnumber_++,laneBlock,inc_startTime,inc_link,inc_distance,in
c_location);
	err = fopen_s(&syn_file,syn,"w+");
	if(err!=0)  qps_GUI_printf("Cannot find file %s \n",syn);
	else
		fprintf(syn_file,"%d %d %d %d
",switch_startTime,(int)inc_startTime,inc_location,laneBlock);
	if(syn_file!=NULL) fclose(syn_file);

	if(vtt!=NULL) {qps_GUI_printf("vtt not NULL\n");free(vtt);}
	vtt = (struct veh_tt *) realloc(vtt,max_nr_veh*sizeof(struct
veh_tt));
	vtt_buffer=(struct veh_tt *) realloc(vtt_buffer,500*sizeof(struct
veh_tt));
	rst=(struct road_stat *) calloc(180,sizeof(struct road_stat));
	for(i=0; i<linkNum; i++) {
		links_length[i]=sqrt(pow((endNode_x[i]-
startNode_x[i]),2)+pow((endNode_y[i]-startNode_y[i]),2));
		Radius[i]=sqrt(pow((endNode_x[i]-
center_x[i]),2)+pow((endNode_y[i]-center_y[i]),2));
	}
	fopen_s(&nodeM,nodeM_filename,"w+");
	fclose(nodeM);
	x = (struct svm_node *) realloc(x,max_nr_attr*sizeof(struct
svm_node));
	if((model=svm_load_model(model_filename))==0)
```

```
	{
		qps_GUI_printf("can't open model file
%s\n",model_filename);
	}
	err = fopen_s(&fp,range_filename,"r");
	if(err!=0)
	{
		qps_GUI_printf("cannot find file %s \n",range_filename);
	}
	feature_max = (double *)malloc((max_index+1)* sizeof(double));
	feature_min = (double *)malloc((max_index+1)* sizeof(double));
	if (fgetc(fp) == 'x') {
		fscanf(fp, "%lf %lf\n", &lower, &upper);
		while(fscanf(fp,"%d %lf %lf\n",&idx,&fmin,&fmax)==3)
		{
			if(idx<=max_index)
			{
				feature_min[idx] = fmin;
				feature_max[idx] = fmax;
			}
		}
	}
	fclose(fp);
}
void qpx_VHC_release(VEHICLE* vehicle)
{
   struct veh_motion *vmo=NULL;
   vmo = calloc(1,sizeof(struct veh_motion));
   // check for a bad vehicle
   if(!vehicle) return;
   // store the data with the vehicle
   qps_VHC_userdata(vehicle, (VHC_USERDATA*) vmo);
}
void  qpx_VHC_transfer(VEHICLE* vehicle, LINK* link1, LINK* link2)
{
	int j;
	float speed;
	struct veh_motion *vmo=NULL;
	vmo=(struct veh_motion *) calloc(1,sizeof(struct veh_motion));

	vmo = (struct veh_motion *) qpg_VHC_userdata(vehicle);
	if (link2==qpg_NET_link(enter_links[0]) &&
qpg_VHC_type(vehicle)==3 && qpg_CFG_simulationTime()>switch_startTime)
{
		for(i=fixed_nodeNum;i<total_nodeNum;i++) {
			if(within[i]==0) {
				vmo->nodeID=i;
				vmo->startTime=qpg_CFG_simulationTime();
				for(j=0;j<rep_vehStatus_num;j++)
					vmo->speed[j]=qpg_VHC_speed(vehicle);
				within[i]=1;
				vmo->x=left_x+10.0;
				vmo->y=down_y+10.0;
				break;
```

```
                  }
            }
            qps_VHC_userdata(vehicle, (VHC_USERDATA*) vmo);
      }
      if ((link2==qpg_NET_link(exit_links[0]) ||
link2==qpg_NET_link(exit_links[1])) && qpg_VHC_type(vehicle)==3 && vmo-
>nodeID!=0 && qpg_CFG_simulationTime()>switch_startTime) {
            within[vmo->nodeID]=0;
            speed=sqrt(pow((down_y + 10 - vmo->y),2)+pow((left_x + 10 -
vmo->x),2)) / 0.5;
            vmo->x=left_x+10.00;
            vmo->y=down_y+10.00;
            qps_VHC_userdata(vehicle, (VHC_USERDATA*) vmo);
            fopen_s(&nodeM,nodeM_filename,"a+");
            fprintf(nodeM,"$ns_ at %5.1f \"$node_(%d) setdest %8.2f
%8.2f %8.2f\"\n",qpg_CFG_simulationTime()-0.5,vmo-
>nodeID,10.00/3.28,10.00/3.28,speed/3.28);
            fclose(nodeM);
      }
}

void  qpx_LNK_vehicleTimeStep(LINK* link, VEHICLE* vehicle)
{
      int j,idx,stucked=0;
      double scaled_value,v=0;

      /* Generate Incident */
      if(!inc_occure1 && link==qpg_NET_link(inc_link) &&
(3.28*qpg_VHC_distance(vehicle))<=(inc_distance+80) &&
qpg_CFG_simulationTime()>=inc_startTime && qpg_VHC_lane(vehicle)==1)
      {
            qps_VHC_stopped(vehicle,TRUE);
            inc_occure1=TRUE;
            inc_startTime=qpg_CFG_simulationTime();
            inc_location=inc_location+(int)(inc_distance-
3.28*qpg_VHC_distance(vehicle));
            qps_GUI_printf("\n \t Incident occured at: %5.0f.
\n",qpg_CFG_simulationTime());
            err = fopen_s(&syn_file,syn,"w");
            if(err!=0)  qps_GUI_printf("Cannot find file %s \n",syn );
            else
                  fprintf(syn_file,"%d %d %d %d %d
%d",switch_startTime,(int)inc_startTime,inc_location,laneBlock,penetrat
ion,demand);
            fclose(syn_file);
      }
      if (laneBlock>=2 && !inc_occure2 && link==qpg_NET_link(inc_link)
&& (3.28*qpg_VHC_distance(vehicle))<=(inc_distance+80) &&
qpg_CFG_simulationTime()>=inc_startTime && qpg_VHC_lane(vehicle)==2)
      {
            qps_VHC_stopped(vehicle,TRUE);
            inc_occure2=TRUE;
            qps_GUI_printf(" \t Lane 2 Blocked at: %5.0f.
\n",qpg_CFG_simulationTime());
```

```
	}
	if (laneBlock==3 && !inc_occure3 && link==qpg_NET_link(inc_link)
&& (3.28*qpg_VHC_distance(vehicle))<=(inc_distance+80) &&
qpg_CFG_simulationTime()>=inc_startTime && qpg_VHC_lane(vehicle)==3)
	{
		qps_VHC_stopped(vehicle,TRUE);
		inc_occure3=TRUE;
		qps_GUI_printf(" \t Lane 3 Blocked at: %5.0f.
\n",qpg_CFG_simulationTime());
	}

	if(qpg_VHC_type(vehicle)==3 &&
qpg_CFG_simulationTime()>switch_startTime) {
		struct veh_motion *vmo=NULL;
		float speed;
		vmo=(struct veh_motion *) calloc(1,sizeof(struct
veh_motion));
		vmo = (struct veh_motion *) qpg_VHC_userdata(vehicle);
		for(i=0;i<linkNum;i++) {
			if(link==qpg_NET_link(links[i]) && vmo->nodeID!=0) {
				if(fmod(qpg_CFG_simulationTime(),rec_nodeM_int)
== 0) {
					cur_loc =
find_coord(link,qpg_VHC_distance(vehicle),qpg_VHC_lane(vehicle));
					speed=sqrt(pow((cur_loc->y - vmo-
>y),2)+pow((cur_loc->x - vmo->x),2)) / rec_nodeM_int;
					fopen_s(&nodeM,nodeM_filename,"a+");
					fprintf(nodeM,"$ns_ at %5.1f \"$node_(%d)
setdest %8.2f %8.2f %8.2f\"\n",qpg_CFG_simulationTime()-
rec_nodeM_int,vmo->nodeID,(cur_loc->x-left_x)/3.28,(cur_loc->y-
down_y)/3.28,speed/3.28);
					fclose(nodeM);
					vmo->x=cur_loc->x;
					vmo->y=cur_loc->y;
					qps_VHC_userdata(vehicle, (VHC_USERDATA*)
vmo);
				}
				if(fmod((qpg_CFG_simulationTime() - vmo-
>startTime),rep_vehStatus_int) == 0 && qpg_CFG_simulationTime()>vmo-
>alertTime) {
					x = (struct svm_node *)
realloc(x,max_nr_attr*sizeof(struct svm_node));
					for(j=rep_vehStatus_num-1;j>=1;j--) {
						vmo->speed[j]=vmo->speed[j-1];
						vmo->laneChange[j]=vmo-
>laneChange[j-1];
					}
					vmo-
>laneChange[0]=(qpg_VHC_lane(vehicle)==vmo->lane ? 0 : 1);
					vmo->lane=qpg_VHC_lane(vehicle);
					vmo->speed[0]=qpg_VHC_speed(vehicle);
					idx=0;
					for(j=0;j<2*rep_vehStatus_num;j++) {
						if(j<rep_vehStatus_num) {
```

```
                                                scaled_value=lower + (upper-
lower) * (vmo->laneChange[j]-feature_min[j+1])
                                                      /(feature_max[j+1]-
feature_min[j+1]);
                                          }
                                          else {
                                                scaled_value=lower + (upper-
lower) * (vmo->speed[j-rep_vehStatus_num]-feature_min[j+1])
                                                      /(feature_max[j+1]-
feature_min[j+1]);
                                          }
                                          x[idx].index=j+1;
                                          x[idx].value=scaled_value;
                                          idx++;
                                    }
                                    x[idx].index=-1;
                                    v=svm_predict(model, x);
                                    vmo->alertTime=qpg_CFG_simulationTime();
                                    vmo-
>alertLocation=get_LES(qpg_LNK_nodeEnd(link))-
(int)(3.28*qpg_VHC_distance(vehicle));
                                    if((qpg_CFG_simulationTime()-vmo-
>startTime)>rec_nodeM_int) {

      fopen_s(&nodeM,nodeM_filename,"a+");
                                          fprintf(nodeM,"$ns_ at %9.5f
\"$snet_s(%d) veh_msg %d %5.2f
%d\"\n",qpg_CFG_simulationTime()+(float)qpg_UTL_randomInteger(APIRNG_MI
SC,1000)/100000,vmo->nodeID,(int)v,qpg_CFG_simulationTime(),vmo-
>alertLocation);
                                          Nveh_msg++;
                                          fclose(nodeM);
                                    }
                                    qps_VHC_userdata(vehicle, (VHC_USERDATA*)
vmo);
                              }
                        }
                  }
      }
}
void  qpx_NET_second(void)
{
      int syn_time,wait_time=0,j;
      float rdStatus_sum=0;
      char checkofy[11];

      if(qpg_CFG_simulationTime()>restartTime) {
            qps_GUI_printf("Restarting...\n");
            qps_GUI_simRestart();
      }
      if(qpg_CFG_simulationTime()>switch_startTime &&
fmod(qpg_CFG_simulationTime(),switch_int) == 0)
      {
            while(1)
```

```
            {
                  err = fopen_s(&swi_file,swi,"w+");
                  if(err != 0)      qps_GUI_printf("can not open file
%s, error number is: %d\n",swi,err);
                  else  {
                        fprintf_s(swi_file,"ns2
%d\n",(int)qpg_CFG_simulationTime());
                        //qps_GUI_printf("write ns2 at
%d\n",(int)qpg_CFG_simulationTime());
                        fclose(swi_file);
                        break;
                  }
            }

            while(1)
            {
                  qps_SIM_running(FALSE);
                  wait_time++;
                  err = fopen_s(&swi_file,swi,"r+");
                  if(err == 0)      {
                        fscanf_s(swi_file,"%s
%d",checkofy,11,&syn_time);
                        //qps_GUI_printf("switch value is: %s; syn_time
is: %d\n",checkofy,syn_time);
                        fclose(swi_file);
                        if(strncmp(checkofy,"para",3) == 0 &&
syn_time==(int)qpg_CFG_simulationTime())  {
                              qps_SIM_running(TRUE);
                              fopen_s(&nodeM,nodeM_filename,"w+");
                              fclose(nodeM);
                              break;
                        }
                        else if(strncmp(checkofy,"para",3) == 0 &&
syn_time<(int)qpg_CFG_simulationTime())   {
                              if(err = fopen_s(&swi_file,swi,"w+")==0)
                                    fprintf_s(swi_file,"ns2
%d\n",(int)qpg_CFG_simulationTime());
                              fclose(swi_file);
                        }
                        else if(strncmp(checkofy,"STOP",3) == 0)  {
                              //Sleep(1000);
                              qps_GUI_printf("Restarting...\n");
                              qps_GUI_simRestart();
                              //qps_SIM_running(TRUE);
                              //switch_startTime=999999;
                              break;
                        }
                  }
                  else Sleep(100);
                  if(wait_time>300000)    break;
            }
      }
}
```

```
static struct coordinate * find_coord(LINK* i_link, float Distance,int
Lane)
{
	struct coordinate *cur_coord;
	int i;
	cur_coord=(struct coordinate *) calloc(1,sizeof(struct
coordinate));
	Distance=3.28*Distance;
	for(i=0; i<linkNum; i++)
	{
		if(qpg_NET_link(links[i])==i_link)
		{
			cur_coord->linkID=i;
			if(arc[i]==0) {
				cur_coord->x=endNode_x[i]-(endNode_x[i]-
startNode_x[i])*Distance/links_length[i]+(12*Lane-6)*(endNode_y[i]-
startNode_y[i])/links_length[i];
				cur_coord->y=endNode_y[i]-(endNode_y[i]-
startNode_y[i])*Distance/links_length[i]-(12*Lane-6)*(endNode_x[i]-
startNode_x[i])/links_length[i];
			}
			else {
				cur_coord-
>x=center_x[i]+arc[i]*Radius[i]*cos(atan((endNode_y[i]-
center_y[i])/(endNode_x[i]-center_x[i]))+Distance/Radius[i])+(12*Lane-
6)*sin(atan((endNode_y[i]-center_y[i])/(endNode_x[i]-
center_x[i]))+Distance/Radius[i]);
				cur_coord-
>y=center_y[i]+arc[i]*Radius[i]*sin(atan((endNode_y[i]-
center_y[i])/(endNode_x[i]-center_x[i]))+Distance/Radius[i])-(12*Lane-
6)*cos(atan((endNode_y[i]-center_y[i])/(endNode_x[i]-
center_x[i]))+Distance/Radius[i]);
			}
			break;
		}
	}
	return cur_coord;
}
static int get_LES(NODE* i_node)
{
	int i;
	int i_LES = 0;
	for(i=0; i<node_Num; i++)
	{
		if(qpg_NET_node(node[i])==i_node)
		{
			i_LES = node_LES[i];
			break;
		}
	}
	return i_LES;
}
```

A.2.2 Plugin File for Travel Time Prediction

```
#include "c:\program Files\paramicsV5\programmer\include\programmer.h"
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <math.h>
#include <process.h>
#include <malloc.h>
#include <memory.h>
#include "svm.h"
#include "plugin.h"

#define rec_num 2
#define enter_linkNum 9
#define exit_linkNum 6
#define head 4
#define para 2
#define interval 6
#define maxCounter 200
#define elem head+para*interval

static char *rec_filename_pre = {"C:\\t"};
static char *rec_filename;

static char *roundN_filename = {"C:\\VII_sim\\tt\\G_C\\roundnumber"};
static char *model_filename="C:\\queue1.scale.model";
static char *range_filename="C:\\range";
struct svm_model* model;
static char
*enter_links[enter_linkNum]={"210:1049","436:1049","437:1273","1503:150
4","1506:186","441:442","447:1447","1400:1402","477:175"};
static char
*exit_links[exit_linkNum]={"200:1495","1298:1296","1330:1321","176:1380
","1393:1404","452:1419"};
struct svm_model* model1;

FILE *rec_file;
FILE *roundN;
static float inc_startTime=530;
static float rec_time=30;
int counter=0;
int tagged=0;
int i,j,k,ab_cnt,i_t,printed;
double rep_ab[3][6];
float lnb0,lnb1,lnb2,check_time,check_pt;
float veh_stat[maxCounter][elem];
int veh_num[7];
float veh_time[7];
float increment=4;
float inc_distance=1500;
struct svm_node *x;
struct svm_node *x1;
```

```
struct veh_tt *vtt;
struct veh_tt *vtt_buffer;
struct road_stat *rst;
int max_nr_veh = 1000;
int max_nr_attr = 13;
errno_t err;
double lower=0.0,upper=1.0;
double *feature_max;
double *feature_min;
double *feature_max1;
double *feature_min1;
int max_index=12;
int max_index1=15;
int in_cnt[7],out_cnt[7],vid,num_tt,vid_buffer;
float avg_tt;
int warmup_time=57600+1200;
int startup_time=57600;
int clear;
int time_step=120;
void qpx_NET_postOpen(void)
{
    FILE *fp=NULL;
      int idx,roundnumber_;
      double fmin, fmax;
      char round[4];
      qps_GUI_printf("\n-------VII APIs--------\n");
      clear=(warmup_time-startup_time)/time_step;
      if(vtt!=NULL) {qps_GUI_printf("vtt not NULL\n");free(vtt);}
      vtt = (struct veh_tt *) realloc(vtt,max_nr_veh*sizeof(struct
veh_tt));
      vtt_buffer=(struct veh_tt *) realloc(vtt_buffer,500*sizeof(struct
veh_tt));
      rst=(struct road_stat *) calloc(180,sizeof(struct road_stat));

      err = fopen_s(&roundN,roundN_filename,"r+");
      if(err!=0)  qps_GUI_printf("Cannot find file %s
\n",roundN_filename);
      else {
            fscanf_s(roundN,"%d",&roundnumber_);
            roundnumber_++;
            itoa (roundnumber_,round,10);
            rewind(roundN);
            fprintf(roundN,"%d",roundnumber_);
      }
      if(roundN!=NULL) fclose(roundN);
      rec_filename = (char *)malloc((strlen(rec_filename_pre) +
strlen(round) + 1)*sizeof(char));
      strcpy(rec_filename, rec_filename_pre);
      strcat(rec_filename, round);
}

void  qpx_VHC_transfer(VEHICLE* vehicle, LINK* link1, LINK* link2)
{
```

```
      if (link2==qpg_NET_link(enter_links[0]) &&
qpg_VHC_type(vehicle)==3) {
            in_cnt[0]++;
            vtt[vid].startTime=qpg_VHC_startTime(vehicle);
            vtt[vid].origin=qpg_VHC_origin(vehicle);
            vtt[vid].enterTime=(int)qpg_CFG_simulationTime();
            rst[((int)qpg_CFG_simulationTime()-startup_time)/time_step-
1].enter_num++;
            vid++;
            if(vid>=max_nr_veh) {
                  qps_GUI_printf("buffer\n");
                  for(i=0;i<vid;i++) {
                        if(vtt[i].startTime!=999999) {

      vtt_buffer[vid_buffer].startTime=vtt[i].startTime;

      vtt_buffer[vid_buffer].origin=vtt[i].origin;

      vtt_buffer[vid_buffer].enterTime=vtt[i].enterTime;
                              vid_buffer++;
                        }
                  }
                  vid=vid_buffer;
                  qps_GUI_printf("post_vid is %d\n",vid);
                  for(i=0;i<vid;i++) {
                        vtt[i].startTime=vtt_buffer[i].startTime;
                        vtt[i].origin=vtt_buffer[i].origin;
                        vtt[i].enterTime=vtt_buffer[i].enterTime;
                  }
                  vid_buffer=0;
            }
      }
      for(i=1;i<enter_linkNum;i++) {
            if(link1==qpg_NET_link(enter_links[i]) &&
qpg_VHC_type(vehicle)==3) in_cnt[0]++;
      }
      if (link1==qpg_NET_link(exit_links[exit_linkNum-1]) &&
qpg_VHC_type(vehicle)==3) {
            out_cnt[0]++;
            for(i=0;i<vid;i++) {
                  if(vtt[i].startTime==qpg_VHC_startTime(vehicle) &&
qpg_VHC_origin(vehicle)==vtt[i].origin) {
                        avg_tt+=qpg_CFG_simulationTime()-
vtt[i].enterTime;
                        num_tt++;
                        if((vtt[i].enterTime-
startup_time)/time_step>=1) {
                              rst[(vtt[i].enterTime-
startup_time)/time_step-1].leave_num++;
                              rst[(vtt[i].enterTime-
startup_time)/time_step-1].tt+=qpg_CFG_simulationTime()-
vtt[i].enterTime;
                              rst[(vtt[i].enterTime-
startup_time)/time_step-1].exit_num++;
```

```
                        }
                        if(rst[clear].leave_num==rst[clear].enter_num
&& rst[clear].enter_num!=0 && qpg_CFG_simulationTime()-startup_time-
clear*time_step>time_step && qpg_CFG_simulationTime()>=warmup_time) {
                              qps_GUI_printf("\t time step %d, tt
%5.2f, in_tra %d, col_tt
%5.2f\n",clear,rst[clear].tt/(float)rst[clear].exit_num,rst[clear].in_t
ra[0],rst[clear].col_tt[0]);
                              fopen_s(&rec_file,rec_filename,"a+");
                              fprintf(rec_file,"%5.2f
",rst[clear].tt/(float)rst[clear].exit_num);
                              for(j=0;j<1;j++)
{if(rst[clear].col_tt[j]!=0) fprintf(rec_file,"%d:%5.2f
",j+1,rst[clear].col_tt[j]);}
                              for(j=0;j<1;j++)
{if(rst[clear].in_tra[j]!=0) fprintf(rec_file,"%d:%d
",j+2,rst[clear].in_tra[j]);}
                              for(j=0;j<1;j++)
{if(rst[clear].enter_tra[j]!=0) fprintf(rec_file,"%d:%d
",j+3,rst[clear].enter_tra[j]);}

                              fprintf(rec_file,"\n");
                              fclose(rec_file);
                              clear++;
                        }
                        vtt[i].startTime=999999;
                        break;
                  }
            }
      }
      for(i=0;i<exit_linkNum-1;i++) {
            if(link2==qpg_NET_link(exit_links[i]) &&
qpg_VHC_type(vehicle)==3) {
                  out_cnt[0]++;
                  for(j=0;j<vid;j++) {
                        if(vtt[j].startTime==qpg_VHC_startTime(vehicle)
&& qpg_VHC_origin(vehicle)==vtt[j].origin) {
                              rst[(vtt[j].enterTime-
startup_time)/time_step-1].leave_num++;
                              vtt[j].startTime=999999;
                              break;
                        }
                  }
            }
      }
}
void  qpx_NET_second(void)
{
      if(fmod(qpg_CFG_simulationTime(),time_step) == 0)
      {
            for(i=5;i>0;i--) {
                  veh_num[i]=veh_num[i-1];
                  veh_time[i]=veh_time[i-1];
            }
```

```
            veh_num[0]+=in_cnt[0]-out_cnt[0];
            veh_time[0]=avg_tt/(float)(num_tt);
            for(i=0;i<6;i++) {
                  rst[((int)qpg_CFG_simulationTime()-
startup_time)/time_step].enter_tra[i]=in_cnt[i];
                  rst[((int)qpg_CFG_simulationTime()-
startup_time)/time_step].exit_tra[i]=out_cnt[i];
                  rst[((int)qpg_CFG_simulationTime()-
startup_time)/time_step].in_tra[i]=veh_num[i];
                  rst[((int)qpg_CFG_simulationTime()-
startup_time)/time_step].col_tt[i]=veh_time[i];
            }
            avg_tt=0;
            num_tt=0;
            for(i=5;i>0;i--) {
                  out_cnt[i]=out_cnt[i-1];
                  in_cnt[i]=in_cnt[i-1];
            }
            in_cnt[0]=0;
            out_cnt[0]=0;
            qps_GUI_printf("time step %d, veh_num is %d,veh_time is
%5.2f, vid is %d\n",((int)qpg_CFG_simulationTime()-
startup_time)/time_step,veh_num[0],veh_time[0],vid);
      }
}
```

REFERENCES

Abdulhai, B., and Ritchie, S.G. (1999). "Enhancing the universality and transferability of freeway incident detection using a Bayesian-based neural network." *Transportation Research, Part C*, 7(5), 261-280.

Adeli, H., and Samant, A. (2000). "An adaptive conjugate gradient neural network-wavelet model for traffic incident detection." *Computer-Aided Civil and Infrastructure Engineering,* 5(4), 251.260.

Antoniades, C. N., and Y. J. Stephanedes. (1996). "Single-Station Incident Detection Algorithm (SSID) for Sparsely Instrumented Freeway Sites." *Proc., International Conference on Applications of Advanced Technologies in Transportation Engineering*, New York, 218-221.

Ahmed, M. S., and Cook, A. R. (1977). "Analysis of freeway traffic time-series data using Box-Jenkins techniques." *Transportation Research Record*, *722*, Transportation Research Board, Washington D.C., 1-9.

Ahmed, M. S., and Cook, A. R. (1980). "Time series models for freeway incident detection." *Journal of Transportation Engineering*, 106(6), 731-745.

Ahmed, M. S., and Cook, A. R. (1982). "Application of time-series analysis techniques to freeway incident detection." *Transportation Research Board*, *841*, Transportation Research Board, Washington D.C., 19-21.

Avila, A., Korkmaz, G., Liu, Y., Teh, H., Ekici, E., Ozguner, F., Ozguner, U., Redmill, K., Takeshita, O., Tokuda, K., Hamaguchi, M., Nakabayashi, S. and Tsutsui, H. (2005). "A complete simulator architecture for inter-vehicle communication intersection warning systems." *Proc., IEEE Conf. on Intelligent Transportation Systems*, Vienna, Austria, 461-466.

Balke, K. N. (1993). *An evaluation of existing incident detection algorithms*. Research Report, FHWA/TX-93/1232-20, Texas Transportation Institute, the Texas A&M University System, College Station, TX.

Balke, K., Dudek, C.L., and Mountain, C.E. (1996). "Using probe-measured travel time to detect major freeway incidents in Houston, Texas." *Transportation Research Record*, *1554*, Transportation Research Board, Washington D.C., 213-220.

Bhavsar, P., Chowdhury, M. A., Sadek, A., Sarasua, W., and Ogle, J. (2007). "Decision support system for predicting traffic diversion impacts across transportation networks using support vector regression." *Transportation Research Board Annual Meeting* (CD-ROM), Washington D.C., 2007.

Biswas, S., Tatchikou, R., and Dion, F. (2006). "Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety." *IEEE Communications Magazine*, 44(1), 74-82.

Blosseville, J., Morin, J., and Lochegnies, P. (1993). "Video Image Processing Application: Automatic Incident Detection Freeways." *Proceedings of the Pacific Rim Trans Tech Conference,* 69-76.

Boser, B. E., Guyon, I., and Vapnik, V. (1992). "A training algorithm for optimal margin classiers." *Proc. the Fifth Annual Workshop on Computational Learning Theory*, New York, NY, 144-152.

Boxill, S. A., and Yu, L. (2000) "An evaluation of traffic simulator models for supporting ITS development." Center for Transportation Training and Research, Texas Southern University.

Cao, L., and Tay, F. E. H. (2001). "Financial forecasting using support vector machines." *Neural computing and applications*, 10(2), 184-192.

Cambridge Systematics and Texas Transportation Institute. (2005). *Traffic Congestion and Reliability: Trends and Advanced Strategies for Congestion Mitigation.* Final report prepared for FHWA, Washington, D.C.

Chang, C-C., and Lin, C-J. (2007) "LIBSVM: a library for support vector machines." <http://www.csie.ntu.edu.tw/~cjlin/libsvm>, (November 2007)

Chang, E. C. P., and Wang, S. H. (1994). "Improved freeway incident detection using fuzzy set theory." *Transportation Research Record*, *1453*, Transportation Research Board, Washington D.C., 75-82.

Chassiakos, A. P., and Stephanedes, Y. J. (1993). "Smoothing algorithms for incident detection." *Transportation Research Record*, *1394*, Transportation Research Board, Washington D.C., 8-16.

Cheu, R. L., Qi, H., and Lee., D. H. (2002). "Mobile Sensor and sample-based algorithm for freeway incident detection." *Transportation Research Record*, *1811*, Transportation Research Board, Washington D.C., 12-20.

Cheu, R. L., Srinivasan, D., and Teh, E. T. (2003). "Support vector machine models for freeway incident detection." *Proc., IEEE Conf. on Intelligent Transportation System*, 1, Shanghai, China, 238-243.

Chen, C. H., and Chang, G. L. (1993). "A dynamic real-time incident detection system for urban arterials—system architecture and preliminary results." *Proc., Conf. on Pacific Rim Transportation technology*, 1, Seattle, 98-104.

Cheu, R. L., and Ritchie, S. G. (1995). "Automated detection of lane-blocking freeway incidents using artificial neural networks." *Transportation Research, Part C*, 3(6), 371-388.

Choe, T., Skabardonis, A., and Varaiya, P. (2002). "Freeway performance measurement system (PeMS)," *Transportation Research Record*, *1811*, Transportation Research Board, Washington D.C., 67-75.

Choffnes, D. R., and Bustamante, F. E. (2005). "An Integrated Mobility and Traffic Model for Vehicular Wireless Networks." *Proc., 2nd ACM International Workshop on Vehicular Ad Hoc Networks (VANET)*, Cologne, Germany, 69-78.

Chowdhury, M., A. Sadek, Y. Ma, N. Kanhere and P. Bhavsar. (2006). "Applications of artificial intelligence paradigms to decision support to real-time traffic management." *Transportation Research Record*, *1968*, Transportation Research Board, Washington D.C., 92-98.

Chowdhury, M. A. and Sadek, A. W. (2003) *Fundamental of Intelligent Transportation System (ITS) Planning*, Artech House Publishers, U.S.

Chiu, Y. C., and Mahmassani, H. S. (2003). "Routing profile updating strategies for online hybrid dynamic traffic assignment operation." *Transportation Research Record*, *1857*, Transportation Research Board, Washington D.C., 39-47.

City of Cape Town, South Africa. (2005). Traffic signal services. <http://www.capetown.gov.za/atrams/>, (October 2007).

Clarke, S. M., Zaeh, M. F. and Griebsch, J. H. (2003). "Predicting haptic data with support vector regression for telepresence applications." *Design and application of hybrid intelligent systems*, 572-581.

Coifman, B. and Ramachandran, M. (2004). "Distributed surveillance on freeways with an emphasis on incident detection." *Proc., IEEE Conf on Intelligent Transportation Systems*, Washington, 773-778.

Coleri, S. and Varaiya, P. (2004). "PEDAMACS: Power efficient and delay aware medium access protocol for sensor networks." *California PATH working paper UCB-ITS-PWP-2004-6*.

Collins, J. F., Hopkins, C. M., and Martin, J. A. (1979). "Automatic incident detection—TRRL algorithms HIOCC and PATREG." *TRRL Supplementary Report*, 526, Crowthorne, Berkshire, U.K.

Cook, A. R., and Cleveland, D. E. (1974). "Detection of freeway capacity-reducing incidents by trafficstream measurements." *Transportation Research Record*, *495*, Transportation Research Board, Washington D.C., 1-11.

Cortes, C., and Vapnik, V. (1995). "Support-vector network." *Machine Learning*, 20,273–297.

Crabtree, J. D., and Stamatiadis, N. (2007). "Using dedicated short-range communication (DSRC) technonlogy for freeway incident detection: A performance assessment based on traffic simulation data." *Transportation Research Board Annual Meeting* (CD-ROM), Washington D.C.

Dia, H., and Rose, G. (1997). "Development and evaluation of neural network freeway incident detection models using field data." *Transportation Research Part C*, 5(5), 313-331.

Ding, A., Zhao, X., and Jiao, L. (2002). "Traffic flow time series prediction based on statistics learning theory." *Proc., 5th IEEE Conf. on International Intelligent* Transportation System, Singapore, 727-730.

Dudek, C. L., Messer, C.J. and Nuckles, N.B. (1974). "Incident detection on urban freeway." *Transportation Research Record*, *495*, Transportation Research Board, Washington D.C., 12-24.

Eichler, S., Benedikt, O., Schroth, C., and Timo, K. (2005). "Simulation of car-to-car messaging: Analyzing the impact on road traffic." *Proc., IEEE Computer Society's Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, MASCOTS*, Atlanta, 507-510.

Eichler, S., Schroth, C., Kosch, T., and Strassberger, M. (2006). "Strategies for context-adaptive message dissemination in vehicular ad hoc networks", *Proc., Third Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (IEEE Cat. No. 06EX1437)*, San Jose, 217-225.

Eichler, S., Schroth, C., Eberspacher, Jorg. (2006). "Car-to-car communication", University of St.Gallen - Alexandria Repository (Switzerland). <http://www.alexandria.unisg.ch/EXPORT/DL/31973.pdf> (November 2007)

Estrin, D., Girod, L., Pottie, G. and Srivastava, M. (2001). "Instrumenting the world with wireless sensor networks." *International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2001)*, Salt Lake City, Utah.

Ewing, T., Doss, E., Hanebutte, U., Canfield, T., Brown-VanHoozer, A. and Tentner, A. (1996). "Argonne Simulation Framework for Intelligent Transportation Systems." *ITS America the 6th annual meeting,* Houston, TX.

Fambro, D. B., and Ritch, G.P. (1979). *Automatic detection of freeway incidents during low volume conditions.* Report No. FHWA/TX-79/23-210-1, Texas Transportation Institute, Texas A&M University System, College Station, TX.

Fambro, D. B., and Ritch, G.P. (1980). "Evaluation of an algorithm for detecting urban freeway incidents during low-volume conditions." *Transportation Research Record*, *773*, Transportation Research Board, Washington D.C, 31-39.

Feron, E. (2003). "Distributed And Centralized Conflict Management Under Traffic Flow Management Constraints," Massachusetts Institute of Technology.

Federal Highway Administration (FHWA) Office of Safety. (2004). *Targeting Highway Fatalities.* <http://safety.fhwa.dot.gov/facts/stats2002/> (November 2007)

FHWA. (2005). VII Architecture and Functional Requirements Version 1.1. ITS Joint Program Office, US DOT.

Fitzgibbons, B., Fujimoto, R., Guensler, R., Hunter, M., Park, A. and Wu, H. (2004) "Distributed Simulation Test Bed for Intelligent Transportation Systems Design and Analysis." *Proc., Annual National Conference on Digital Government Research*, 1-2.

Forbes, G. J., and Hall, F. L. (1990). "The applicability of Catastrophe theory in modeling freeway traffic operations." *Transportation Research Part A*, (24)5, Cairns, Australia, 335-344.

Forbes, G. J. (1992). "Identifying incident congestion." *Journal of ITE,* Institute of Transportation Engineers, (62) 6, 17-22.

Friedman, J. (1996). *Another approach to polychotomous classification.* Technical report, Department of Statistics, Stanford University. <http://www-stat.stanford.edu/reports/friedman/poly.ps.Z.> (November 2007)

Fujimura, K., and Hasegawa, T. (2005). "Performance evaluation of the MAC protocol for integrated inter-vehicle and road to vehicle." *Proc., IEEE Conf on Intelligent Transportation Systems*, Vienna, Austria, 308-313.

Funt, B., and Xiong, W. (2004). "Estimating illumination chromaticity via support vector regression." *Proc., 12th Color Imaging Conference on Color Science, Systems & Applications*, AZ, 47-52.

Gall, A. I., and Hall, F. L. (1989). "Distinguishing between incident congestion and recurrent congestion: a proposed logic." *Transportation Research Record*, *1232*, Transportation Research Board, Washington D.C, 1-8.

Ghaman, R. S., Zhang, L., Mchale, G., and Stallard, C. (2003). "The role of traffic simulation in traffic signal control system development." *Proc., Conf. on IEEE Intelligent Transportation Systems*, 1, Shanghai, China, 872-877.

Hall, F. L., Shi, Y., and Atala, G. (1993). "On-line testing of the McMaster incident detection algorithm under recurrent congestion." *Transportation Research Record*, *1394*, Transportation Research Board, Washington D.C, 1-7.

Huisken, G., and Van Berkum, E. C. (2003). "A comparative analysis of short-range travel time prediction methods." *Transportation Research Board Annual Meeting* (CD-ROM), Washington, D.C.

Hsin, V. J. K., and Wang, P. T. R. (1992). "Modeling concepts for intelligent vehicle highway systems (IVHS) applications." *Proc., 24th Conference on Winter Simulation*, Arlington, 1201-1209.

Hsu, C-W., Chang, C-C., and Lin, C-J. (2007) "A Practical Guide to Support Vector Classification." <http://www.csie.ntu.edu.tw/~cjlin/guide.pdf>.

Innamaa, S. (2001). "Short-term prediction of highway travel time using MLP neural networks." *Proc., 8th World Congress on Intelligent Transport Systems* (CD-ROM), Sidney, Australia.

Innamaa, S. (2007). "Online Prediction of Travel Time: Experience from a Pilot Trial." *Transportation Planning and Technology*, 30(2), 271-278.

ITS America. (2005). "Primer on VII." <http://www.itsa.org/itsa/files/pdf/VIIPrimer.pdf.> (July 2007).

ITS America, "VII Resources," <http://www.itsa.org/Library/c197/ITS_Resources/Library.html> (October 2007).

Ivan, J. N., Schofer, J. L., Koppelman, F. S. and Massone, L. L. E. (1995). "Real-time data fusion for arterial street incident detection using neural networks." *Transportation Research Record*, *1497*, Transportation Research Board, Washington D.C, 27-35.

Ivan, J. N., and Chen, S.-R. (1997). "Incident detection using vehicle-based and fixed-location surveillance." *Journal of Transportation Engineering*, 123(3), 209-215.

Ivan, J. N. (1997). "Neural network representations for arterial street incident detection data fusion." *Transportation Research Part C*, 5(3), 245-254.

Ivan, J. N., and Sethi, V. (1998). "Data fusion of fixed detector and probe vehicle data for incident detection." *Journal of Computer-Aided Civil and Infrastructure Engineering*, (13)5, 329-337.

Jayakrishnan, R., and McNally, M. G. (2006) "A Distributed On-Line Database System for Transportation Managment Using Cooperating Roadside and In-Vehicle Communication Devices." <http://www.its.uci.edu/its/research/its.html> (November 2007).

JHK and Associates. (1993). Bay Area Traffic Operations System Incident Detection Algorithms Report.

Killat, M., Schmidt-Eisenlohr, F., Hartenstein. H. (2007). "Enabling Efficient and Accurate Large-Scale Simulations of vehicular ad hoc networks (VANETs) for Vehicular Traffic Management", University of Munich, Germany. <http://delivery.acm.org/10.1145/1290000/1287754/p29-killat.pdf?key1=1287754&key2=1438673911&coll=&dl=ACM&CFID=15151515&CFTOKEN=6184618> (November 2007)

Klein, L.A. (2001). *Sensor technologies and data requirement for ITS*. Artech House, Norwood, MA.

Knerr, S., Personnaz, L., and Dreyfus, G. (1990). *Single-layer learning revisited: a stepwise procedure for building and training a neural network.* Springer-Verlag, Berlin, Germany.

Kochhal, M., Schwiebert, L., and Gupta, S. (2003). "Role-based hierarchical self organization for wireless ad hoc sensor networks." *Proceedings of ACM Workshop on Wireless Sensor Networks and Applications (WSNA)*, San Diego, California.

Kreβel, U. (1999). "Pairwise classification and support vector machines." *Advances in Kernel Methods – Support Vector Learning*, 255-258, MIT Press, Cambridge, MA.

Kurfees, W., Collins, J. F., and Tanhaee-Motlagh, M. (1995) "A New Advanced Traffic Management System For The Memphis Area." *Compendium of Technical Papers on Institute of Transportation Engineers 65th Annual Meeting*, 286-289.

Lee, J. K., Lim, Y. H. and Chi, S. D. (2004). "Hierarchical Modeling and Simulation Environment for Intelligent Transportation Systems," *Simulation*, No. 80, 61-76.

Levin, M., and Krause, G. M. (1979). "Incident-detection algorithms, Part1: Off-Line evaluation." *Transportation Research Record, 722,* Transportation Research Board, Washington D.C, 49-64.

Lin, W. (2004). "A case study on Support Vector Machine versus Artificial Neural Networks." University of Pittsburgh.

Lin, C. K., and Chang, G. L. (1998). "Development of a fuzzy-expert system for incident detection and classification." *Journal of Mathematical and Computer Modeling*, 27(9-11), 9-25.

Mangharam, R., Weller, D. S., Stancil, D. D., Rajkumar, R. and Parikh, J. S. (2005) "GrooveSim: a topography-accurate simulator for geographic routing in vehicular networks." *Proc., Conf. on vehicular ad hoc networks (VANET)'05*, Cologne, Germany, 59-68.

Marc, T. M., Felix, S.E., and Hannes, H. (2006). "Effects of a realistic channel model on packet forwarding in vehicular ad hoc networks." *Proc. Conf. on IEEE Wireless Communications and Networking Conference (IEEE Cat. No. 06TH8876),* Las Vegas, 285-391.

Marc, T. M. (2007). "Inter-vehicle communications: Assessing information dissemination under Safety Constraints." *4th Annual Conference on Wireless Demand Network Systems and Services (WONS '07)* (CD-ROM), Oberguyrgl, Austria.

Maurer, J., Thomas, F., Thomas, S., and Werner, W. (2004)."A new Inter-Vehicle Communications (ivc) Channel model." *Proc., IEEE 60th Vehicular Technology Conference on Wireless Technologies for Global Security*, Los Angeles, 9-13.

Martin, P. T., Perrin, J., and Hansen, B. (2001). "Incident Detection Algorithm Evaluation." Prepared for Utah Department of Transportation.

McTrans at University of Florida. (2006) "CORSIM microscopic traffic simulation model." <http://mctrans.ce.ufl.edu/featured/TSIS/Version5/corsim.htm> (November 2007)

Michalopoulos, P. G. (1991). “Vehicle detection video through image processing: the autoscope system.” *IEEE Transactions on Vehicular Technology*, 40(1), 21-29.

Michalopoulos, P. G., Jacobson, R. D., Anderson, C. A. and DeBruycker, T. B. (1993).“Automatic incident detection through video image processing.” *Traffic Engineering and Control*, 34(2), 66-75.

Michalopoulos, P. G., Jacobson, R. D. and Anderson, C. A. (1993). “Field implementation and testing of a machine vision based incident detection system.” *Proc., Conf on Pacific Rim TransTech*, 1, Seattle, 69-76.

Michigan Department of Transportation. (2005). *VII Michigan Test Bed Program Concept of Operations*.

Mirchandani, P., and Head, L. (1998). “RHODES: a real-time traffic signal control system: architecture, algorithms, and analysis.” <www.sie.arizona.edu/ATLAS/docs/TRISTANIII.pdf> (October 2006).

Mirchandani, P. and Wang, F. Y. (2005). “RHODES to Intelligent Transportation Systems.” *IEEE Intelligent Systems*, 20(1), 10-15.

M.I.T. (2006). “A microscopic traffic simulator for evaluation of dynamic traffic management systems.” <http://web.mit.edu/its/mitsimlab.html> (November 2007).

Mouskos, K. C., Niver, E., Lee, S., Batz, T. and Dwyer, P. (1999). “Transportation operations coordinating committee system for managing incidents and traffic: evaluation of the incident detection 101 system.” *Transportation Research Record*, *1679*, Transportation Research Board, Washington D.C, 50-57.

Mussa, R. N. (1997). “Evaluation of driver-based freeway incident detection.” *Journal of ITE*, Institute of Transportation Engineers, 67(3), 33-40.

Mussa, R. N., and Upchurch, J. E. (1999). “Simulation assessment of incident detection by cellular phone call-in programs.” Transportation, 26(4), 399-416.

Mussa, R. N., and Upchurch, J. E. (2000). “Modeling incident detection using vehicle-to-roadside communication system.” *Journal of the Transportation Research Forum*. 39(4), 117-127.

National VII Coalition. “VII Coalition Homepage.” <http://www.vehicle-infrastructure.org> (October 2007).

Niver, E., Mouskos, K. C., Batz, T., and Dwyer, P. (2000). "Evaluation of the TRANSCOM's system for managing incidents and traffic (TRANSMIT)." *IEEE Transactions on Intelligent Transportation Systems*, 1(1), 15-31.

OPNET Technologies, Inc. (2006). "OPNET network simulation software." <http://www.opnet.com/> (November 2007).

Ozbay. K., and Pushkin, K. (1999). *Incident Management in Intelligent Transportation Systems.* Artech House: Boston, MA.

Park, D. J., and Rilett, L. R. (1998). "Forecasting multiple-period freeway link travel times using modular neural networks." *Transportation Research Record*. *1617*, Transportation Research Board, Washington, D.C., 163–170.

Park, D. J., Rilett, L. R., and Han, G. (1999). "Spectral Basis Neural Networks for Real-Time Travel Time Forecasting." *Journal of Transportation Engineering*, 125(6), 515–523.

Parkany, E., and Bernstein, D. (1995). "Design of incident detection algorithms using vehicle-toroadside communication sensors." *Transportation Research Record*, 1494, Transportation Research Board, Washington D.C, 67-74.

Payne, H. J., and Tignor, S. C. (1978). "Freeway Incident Detection Algorithms Based on Decision Trees with States." *Transportation Research Record, 682,* Transportation Research Board, Washington D.C, 30-37.

Partners for Advanced Transit and Highways (PATH). (2007). "VII California", <http://www.path.berkeley.edu/VIICalifornia> (October 2007).

Petty, K. R., and Mahoney, W. P. III. (2007). Weather applications and products enabled through vehicle infrastructure integration (VII) feasibility and concept development study. Publication FHWA-HOP-07-084, FHWA, U.S. Department of Transportation.

Persaud, B. N., and Hall, F. (1989). "Catastrophe theory andpatterns in 30-second freeway traffic data. Implications for incident detection." *Transportation Research, Part A,* 2, 103-113.

Persaud, B. N., Hall, F. L., and Hall, L. M. (1990). "Congestion identification aspects of the McMaster incident detection algorithm." *Transportation Research Record*, *1287*, Transportation Research Board, Washington D.C, 167-175.

Papageorgiou, M., Diakaki, C., Dinopoulou, V., Kotsialos, A., and Wang, Y. (2003). "Review of road traffic control strategies." *Proc., the IEEE Conf. on Electronic Textiles: A Platform for Pervasive Computing*, 91(12), 2043-2065.

Qi, H., Cheu, R. L., and Lee, D. H. (2002). "Freeway Incident Detection Using Kinematics Data from Probe Vehicles." *Proc., 9th World Congress on Intelligent Transport Systems* (CD-ROM).

Quadstone Limited. (2005). "Paramics V5.0 Modeler User's Guide," Edinburgh, United Kingdom.

Ritchie, S. G., and Cheu, R. L. (1993). "Simulation of freeway incident detection using artificial neural networks." *Transportation Research Part C*, 1(3), 203-217.

Road and Traffic Authority, New South Wales, Australia. (2006). "Sydney coordinated sdaptive traffic system (SCATS)." <http://www.rta.nsw.gov.au/trafficinformation/trafficfacilities/scats> (October 2007).

Samant, A., and Adeli, H. (2000). "Feature extraction for traffic incident detection using wavelet transform and linear discriminant analysis." *Computer-Aided Civil and Infrastructure Engineering*, 15(4), 241-250.

Sarle, W. S. (1997). "Neural Network FAQ - Periodic posting to the Usenet newsgroup comp.ai.neural-nets." <ftp://ftp.sas.com/pub/neural/FAQ.html>. (November 2007)

SAS Institute, Inc. (2005). *SAS Language: Reference*, *Version 8,* 1st Ed., SAS Institute Inc., Cary, NC.

Sato, K., Saisho, K., and Fukuda, A. (1999) "A spatial-temporal resource allocation protocol (STRAP) with mobility specification: simulation and performance evaluation," *Proc., 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems* (*MSWiM)*, Seattle, 87-94.

Scalable Network Technologies. (2006). "QualNet network simulation software." <http://www.scalable-networks.com/> (November 2006).

Schroth, C., Dotzer, F., Kosch, T., Ostermaier, B., and Strassberger, M. (2006). "Simulating the traffic effects of vehicle-to-vehicle messaging systems", *Proc., 5th International Conference on ITS Telecommunications*, 4.

Schmidt-Eisenlohr, F., Torrent-Moreno, M., Mittag, J., and Hartenstein, H. (2007). "Simulation platform for inter-vehicle communications and analysis of periodic information exchange." *Proc., 4th Annual Conf. on Wireless on Demand Network Systems and Services(WONS),* Obergurgl, Austria, 50-58.

Sermons, M. W., and F. S. Koppelman. (1996). "Use of vehicle positioning data for arterial incident detection." *Transportation Research, Part C*, 4(2), 87-96.

Sethi, V., Bhandari, N., Koppelman, F. S., and Schofer, J. L. (1995). "Arterial incident detection using fixed detector and probe vehicle data." *Transportation Research Part C*, 3(2), 99-112.

Sewell, M. (2005), online resource at <http://www.svms.org/introduction.html> (November 2007)

Siemens. (2006). "Principles of SCOOT." <http://www.itssiemens.com/en/t_nav224.html> (October 2006).

Smola, A., and Scholkopf, B. (1998). A tutorial on support vector regression. NeuroCOLT2 Technical Report Series, NC2-TR-1998-030.

Skabardonis, A., Chavala, T.C., and Rydzewski, D. (1998). "The I-880 field experiment: effectiveness of incident detection using cellular phones." PATH Report UCB-ITS-PRR-98-1, Institute of Transportation Studies, University of California at Berkeley, CA.

Sobeih, A., Chen, W. P., Hou, J. C., Kung, L. C., Li, N., Lim, H., Tyan, H. Y. and Zhang, H. (2005) "J-Sim: A Simulation and Emulation Environment for Wireless Sensor Networks." *Proc., of the Annual Simulation Symposium (ANSS 2005),* Urbana, 104-109.

SpeedInfo Inc. (2005). "SpeedInfo deploys real time traffic sensor network for SFO Bay area." <http://www.speedinfo.com/downloads/WC_press_release_101005.pdf> (October 2006).

Stephanedes, Y. J., and Chassiakos, A. P. (1993a). "Application of filtering techniques for incident detection." *Journal of Transportation Engineering*, 119(1), 13-26.

Stephanedes, Y. J., and Chassiakos, A. P. (1993b). "Freeway incident detection through filtering." *Transportation Research Part C*, 1(3), 219-233.

Stephanedes, Y. J., Chassiakos, A. P., and Michalopoulos, P.G. (1992). "Comparative performance evaluation of incident detection algorithms." *Transportation Research Record*, 1360, Transportation Research Board, Washington D.C, 50-57.

Stephanedes, Y. J., and Liu, X. (1995). "Artificial neural networks for freeway incident detection." *Transportation Research Record*, *1494*, Transportation Research Board, Washington D.C, 91-97.

Stitson, M. O., Weston, J. O. E., Gammerman, A., Vovk, V., and Vapnik, V. (1996) *Theory of Support Vector Machines*. Technical Report CSD-TR-96-17.

Somers, F. (1996) "HYBRID: Unifying centralized and distributed network management using intelligent agents," *Proc., IEEE Conf. on Network Operations and Management Symposium*, Kyoto, Japan, 34-43**.**

Subramanian, L. and Katz, R. H. (2000) "An architecture for building self-configurable systems." *Proc., IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing,* Boston, MA.

Sun, Z., Bebis, G., and Miller, R. (2002). "On-road vehicle detection using gabor filters and support vector machines." *Proc., IEEE 14th International Conf. on Digital Signal Processing*, 2, Santorini, Greece, 1019-1022.

Sukthankar, R., Hancock, J., Pomerleau, D. and Thorpe, C. (1996). "A Simulation and Design System for Tactical Driving Algorithms." *Proc., Conf. on AI, Simulation and Planning in High Autonomy Systems* (CD-ROM)*,* La Jolla.

Skabardonis, A., Petty, K.F. and Varaiya, P.P. (1999). "Los Angeles I-10 field experiment: Incident patterns." *Transportation Research Record*, 1683, Transportation Research Board, Washington D.C, 22-30.

Tanikella, H., Smith, B., Zhang, G., Park, B. and Scherer, W. (2007). "Simulating vehicle-infrastructure-integration-enabled operations applications: Traffic monitoring case study." *Transportation Research Board Annual Meeting* (CD-ROM), Washington D.C.

Tanka, W. A., and Piotrowicz. G. (2007). "Vehicle infrastructure integration: More than a very intriguing idea." *Journal of ITE*, 29-32.

Tokuyama, H. (1996). Intelligent Transportation Systems in Japan. <http://www.tfhrc.gov/pubrds/fall96/p96au41.htm> (November 2006).

Torrent-Moreno, M. (2007). "Inter-Vehicle Communications: Assessing Information Dissemination under Safety Constraints." *Proc., 4th Annual IEEE/IFIP Conference on Wireless On Demand Network Systems and Services (WONS)*, Obergurgl, Austria.

Trafficware. (2006). Synchro Studio, <http://www.trafficware.com/> (October 2006).

Tsai, J., and Case, E.R. (1979). "Development of freeway incident detection algorithms by using pattern recognition techniques." *Transportation Research Record*, *722*, Transportation Research Board, Washington D.C, 113-116.

Tyco Integrated Systems. (2006). Traffic management – SCATS. <www.traffic-tech.com/pdf/scatsbrochure.pdf> (October 2006).

University of California, Berkeley. (2006). “VII California Demonstrates Success: Finalist in ITSA "Best of" Research and Innovation.” *Intellimotion*, 12(1).

Van Lint, J. W. C. (2006). “Reliable Real-Time Framework for Short-Term Freeway Travel Time Prediction.” *Journal of Transportation Engineering*, 132(12), 921-932.

Vanschoenwinkel, B., and Manderick, B. (2006) “Context-sensitive Kernel Functions: A Comparison Between Different Context Weights.” Lecture Notes in Computer Science, 3930, 861-870.

Vapnik, V. (1982). *Estimation of Dependences Based on Empirical Data,* Springer-Verlag, New York.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory,* Springer, New York.

Vanajakshi, L., and Rilett, L. R. (2004). “A comparison of the performance of artificial neural networks and support vector machines for the prediction of traffic speed. *Proc., IEEE Conf. on Intelligent Vehicles Symposium*, Parma, Italy, 194-199.

Versavel, J. (2000). “Sparing lives, saving time: a unified approach to automatic incident detection.” *Traffic Technology International,* Issue: The 2000 International Review of Advanced Traffic Management, 189.

Versavel, J., and Boucke, B. (1998). “Video for Traffic Data and Incident Detection for Traficon.” *Applications of Advanced Technologies in Transport*. 33-40.

Walters, C. H., Wiles, P. B., and Cooner, S. A. (1999). “Incident detection primarily by cellular phones—an evaluation of a system for Dallas, Texas.” *78th TRB Annual Meeting* (CD-ROM), Transportation Research Board, Washington D.C.

Westman, M., Litjens, R., and Linnartz, J. P. (1996). “Integration of probe vehicle and induction loop data—estimation of travel times and automatic incident detection.” PATH Research Report UCB-ITS-PRR-96-13, Institute of Transportation Studies, University of California, Berkeley, CA.

Willsky, A. S., Chow, E.Y., Gershwin, S. B., Greene, C. S., Houpt, P. and Kurkjian, A. L. (1980). “Dynamic model-based techniques for the detection of incidents on freeways.” *IEEE Transactions on Automatic Control*, 25(3), 347-360.

Wang, K.C., Chowdhury, M. A., and Fries, R. (2005). “Real-time Traffic Monitoring and Automated Response with Wireless Sensor Networks,” *12th World Congress on Intelligent Transport Systems,* (CD-ROM).

Wu, C. H., Ho, J. M., and Lee, D.T. (2004). "Travel-time prediction with support vector regression." *IEEE Transactions on Intelligent Transportation Systems*, 5(4), 276-281.

Xie, C., and Parkany, E. (2002). "Use of driver-based data for incident detection." *Proc., 7th International Conference on Application of Advanced Technologies in Transportation Engineering*, Cambridge, 143-150.

Xu, H., and Barth, M. (2004). "A transmission-interval and power-level modulation methodology for optimizing inter-vehicle communications." *Proc., Conf. on Vehicular Ad Hoc Networks (VANET)'04*, Philadelphia, 97-98.

Yin, J., ElBatt, T., Yeung, G., Ryu, B., Habermas, S., Krishnan, H., and Talty T. (2004). "Performance evaluation of safety applications over DSRC vehicular ad hoc networks." *Proc., Conf. on Vehicular Ad Hoc Networks (VANET)'04*, Philadelphia, 1-9.

Zang, Y., Stibor, L., Orfanos, G., Guo, S., and Reumerman, H. J. (2005). "An error model for inter-vehicle communications in highway scenarios at 5.9GHz." *Proc., Conf. on PE-WASUN'05*, Montreal, Canada, 49-56.

Zeng, X., Bagrodia, R., and Gerla, M. (1998). "GloMoSim: A library for parallel simulation of large-scale wireless networks." *Proc., Workshop on Parallel and Distributed Simulation*, Banff, Canada, 154-161.

Zhang, X., and Rice, J. A. (2003). "Short-term travel time prediction." *Transportation. Research, Part C: Emergency Technology*, 11C (3–4), 187–210.