Title
Technologies for Mobile ITS Applications and Safer Driving

Author
Manasseh, Christian Georges

Publication Date
2010

Peer reviewed|Thesis/dissertation

Technologies for Mobile ITS Applications and Safer Driving

by

Christian Georges Manasseh

A dissertation submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ENGINEERING – CIVIL AND ENVIRONMENTAL ENGINEERING

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Raja Sengupta, Chair

Professor Samer Madanat

Professor Pravin Varaiya

Fall 2010

Abstract

Technologies for Mobile ITS Applications and Safer Driving

by

Christian Georges Manasseh

Doctor of Philosophy in Engineering – Civil and Environmental Engineering

University of California, Berkeley

Professor Raja Sengupta, Chair

This thesis presents a user application for safer driving. The technology to enable such an application is discussed and evaluated for the more general use in ITS mobile applications. Two main problems present themselves when building user-centric ITS mobile applications: the first has to do with knowing the state of the environment variables as captured by relevant sensing hardware, and the second has to do with relaying the right amount of information to the user to achieve the purpose of the application. For the first, we present a *middleware* architecture that abstracts sensor data into an HTTP interface to allow data consumers to discover and bind to data producers. For the second we present *user preference adaptive services* that would enable the application to learn from past user experience and adapt its user interaction to meet user preferences. Both research efforts are combined to field test a Smartphone traffic safety application system that is enabled by the *middleware* and that leverages the *user preference adaptive services* to enhance user experience and improve driver safety.

The *middleware* is implemented and tested on three types of traffic mobility and safety applications to measure its performance. The performance measurements show the middleware to be efficient enough for road safety and congestion relief applications by limiting the overhead on the system to the order of 100 msec.

A field test consisting of 9 drivers in the San Francisco Bay Area is conducted using a Smartphone traffic safety application built on the middleware. Results from the field test show that driver safety on the highway can be improved through a soft-safety warning of slow traffic 1-mile ahead on the driver's route. Data from the field test is then fed to a couple of *user preference adaptive services* to improve user experience of the Smartphone application. In the first service we construct a Support Vector Machine learning engine for each driver and predict to 75% accuracy their classification of favorable vs. non-favorable alerts. In the second service, we construct a Decision Tree model for the driver's previous destinations and predict with 96% accuracy their destination given their current location on the road, time of day, and day of week.

CONTENTS

List of Figures iii
Acknowledgments vi
Chapter 1: Introduction 1
1.1 Middleware to Enhance Communications for ITS Mobile Applications 1
1.1.1 Impact of Middleware on ITS Mobile Applications 1
1.2 Smartphone-Based Driver Safety Application 2
1.2.1 Impact of Smartphones on ITS Mobile Applications 2
1.3 User Preference Adaptive Services 3
1.3.1 Impact of Machine Learning Techniques on User Experience 4
Chapter 2: Middleware to Enhance Mobile Communications for Road Safety and Traffic Mobility Applications 5
2.1 Introduction 5
2.2 Middleware Design 7
2.2.1 Component Discovery 8
2.2.2 The Broker 8
2.2.3 SOAP-based SOA 10
2.2.4 HTTP Interface 10
2.2.5 The Middleware Modes of Connection 11
2.3 Middleware Evaluation 11
2.3.1 Hardware and Middleware Setup 12
2.3.2 Applications Description 14
2.3.3 Message Content for Road Safety and Traffic Mobility Applications 15
2.3.4 Enhancements to In-Vehicle Setup 15
2.4 Performance Measures 16
2.4.1 End-to-end Test of Middleware 20
2.5 Conclusion 21
2.6 Acknowledgments 21
Chapter 3: Smartphone Safety Application and the Effects on Driver Behavior 22
3.1 Introduction 22
3.2 Literature Review 23

3.3 System Architecture 24
3.3.1 Cellular Phone 24
3.3.2 NT Server 28
3.3.3 Third Party Services 28
3.4 Experiment Design 29
3.4.1 Validating Smartphone GPS Speed Measurements 29
3.4.2 Validating Similar Traffic Conditions For GPS Traces Used in Analysis 29
3.5 Change in Driver Behavior Due to Slow Traffic Alerts 31
3.6 Is the change in driver behavior Persistent? 34
3.7 Conclusion 36
3.8 Acknowledgement 36
Chapter 4: Learning User Alert Setting Preferences 37
4.1 Introduction 37
4.2 Background 37
4.3 Support Vector Machine Classifiers 41
4.4 The Networked Traveler Safety Application SVM 42
4.5 Results 44
4.6 Acknowledgments 45
Chapter 5: Destination Learning 46
5.1 Introduction 46
5.2 Related Work 46
5.3 Decision Trees 47
5.3.1 Entropy and Information Gain 50
5.4 The Networked Traveler Destination Prediction Algorithm 51
5.4.1 GPS Trace Destination 54
5.5 Results 56
5.6 Conclusion 56
Bibliography 57

LIST OF FIGURES

Figure 2.1. Middleware architecture showing the various producers and consumers of data and the role of the broker, SOAP protocols and HTTP 7

Figure 2.2. Test-bed setup used to test and evaluate the middleware 13

Figure 2.3. Screenshots from the Traffic Management Application (left) and in-vehicle safety application (right) showing traffic-light phase information and nearby vehicle data. Both applications are web-based and utilize Google Maps to display data 14

Figure 2.4. SOAP message to register a car producer service that provides all vehicle properties (AllProperties) with the broker 15

Figure 2.5. Setup used to measure middleware performance 16

Figure 2.6. Determining hardware threshold. (a) Failed transactions start when hardware limitations are met. (b) Increasing the number of client requests does not affect r once the hardware threshold is met 18

Figure 2.7. Variation of transactions/s handled by the broker as threshold client requests are issued..... 19

Figure 2.8. The variations in r as the experiment progresses. Plot shows, minimum, average, and maximum values for r between minutes 5 and 10 of the experiment 20

Figure 3.1. The NT Client warning display screen; the sentence “Slow traffic ahead” is voiced once while the above screen is displayed for 45 seconds. 24

Figure 3.2 Networked Traveler System Architecture. 25

Figure 3.3 Setup used to detect if the Smartphone is close to slow traffic. 26

Figure 3.4 NT Client processes. 27

Figure 3.5 Distribution of times GPS traces are selected for the analysis. Data used for analysis is retrieved from trips that occurred during similar traffic conditions occurring on weekdays Tue – Fri. The time divisions shown comply with the HOV activation times in that location. 30

Figure 3.6 Raw GPS speed plots from GPS traces that passed through the location and satisfied the time constraints. Different colors represent different GPS traces. 31

Figure 3.7 Space Mean Speed plots of GPS traces collected around location. Upper plot shows average values of speed in the cases when alerts were issued vs. cases when no alerts were issued. Lower plot shows the average speed with the standard deviation interval as dashed lines. 32

Figure 3.8 Speed profile plots vs. time for the traces being analyzed. 33

Figure 3.9 Percentage of dips in speed profiles vs. weeks in experiment. Different colors represent different users. 35

Figure 3.10 Histogram of percent of users experiencing a lasting effect of the change in behavior 35

Figure 4.1. Survey responses to the question: "How often, if ever, did FCW give you a warning that was false?" [7] show a large spread of how 96 users perceived a "false warning" 38

Figure 4.2. Survey responses to the question: "How annoying was the FCW alert?"[7] show that users are spread in their decision on how to classify the level of annoyance of an alert 38

Figure 4.3. Survey responses to the question: "Select the statement which best describes how often you changed the FCW alert timing adjustment (select one)?" [7] 39

Figure 4.4. The user interface of the NT Smartphone application allowed us to capture, in real-time, the driver's feedback to the alert by tapping on the thumbs-up or thumbs-down buttons ... 40

Figure 4.5. Histogram showing how users perceived alerts 40

Figure 4.6. Graph depicting a hypothetical 2-dimensional hyper-plane separation 41

Figure 4.7. Accuracy of SVM in relation to data collected 45

Figure 5.1. Example of a Decision Tree 48

Figure 5.2. Different views of the data from a single driver..... 53

Figure 5.3. Trip and destination data for a single driver..... 54

Figure 5.4. Decision Tree Modeling Process in RapidMiner® 55

ACKNOWLEDGMENTS

Delivering the work required for graduating as a PhD student from UC Berkeley is nothing that a single individual could achieve on their own. Throughout the six years I spent at Berkeley, I have received great advice, assistance and support from several individuals that surrounded me. I would like to express my sincere gratitude towards their presence in my life during this period and I apologize for missing out some names.

First and foremost, every member of my dissertation committee has contributed in one way or the other to my successfully completing this effort. Raja has been with me on this road since the very beginning, we have had several discussions around my research as well as other matters. These discussions helped form my scientific acumen and helped advance my research efforts. I had learned several things from Raja though-out my stay at Berkeley, including performing good quality research that leads to practical results and industry acceptance. If this thesis's results make any impact, it is because of Raja's constant insistence that a PhD student at Berkeley needs to deliver a working product and not merely a theoretical proof. Before being on my dissertation committee, both Samer and Pravin had provided me with guidance and advice regarding my academic track and research during my first year at Berkeley. This advice helped me shape the choices I made with regards to my research focus. Throughout the years they have both been a source of reality check and validation of my efforts.

Aside from my thesis committee, several people at Berkeley including the staff at California PATH have contributed to this thesis one way or the other. I include their acknowledgments in the respective chapters of this thesis where their assistance greatly influenced the quality of the results. In particular I would like to express special thanks to Prof. Adib Kanafani and Jim Misener, both have provided their beyond-expected support to my research at times when I needed it.

Finally, my thanks go to my family members, especially my wife Zeina who has supported my decision to get the PhD and at times of despair has acted as a source of energy to keep me on track. I also would like to thank my parents, George and Rita, who have always instilled the value of education in me and my siblings and have supported our efforts in any possible way to achieve the best of our God-granted abilities.

For all of you,

Thank you!

Christian Manasseh

CHAPTER 1: INTRODUCTION

This thesis presents a user application for safer driving. The technology to enable such an application is discussed and evaluated for the more general use in ITS mobile applications. Two main problems present themselves when building user-centric ITS mobile applications: the first has to do with knowing the state of the environment variables as captured by relevant sensing hardware, and the second has to do with relaying the right amount of information to the user to achieve the purpose of the application. For the first, we present a *middleware* architecture that abstracts sensor data into an HTTP interface to allow data consumers to discover and bind to data producers. For the second we present *user preference adaptive services* that would enable the application to learn from past user experience and adapt its user interaction to meet user preferences. Both research efforts are combined to field test a Smartphone traffic safety application system that is enabled by the *middleware* and that leverages the *user preference adaptive services* to enhance user experience and improve driver safety.

1.1 Middleware to Enhance Communications for ITS Mobile Applications

Middleware has emerged as an important architectural component in supporting distributed applications. The role of middleware is to present a unified programming model to application writers and to mask out problems of heterogeneity and distribution. Our work in this field is motivated by the convergence of the embedded sensor and mobile communication revolutions in the automobile industry. The vehicle fleet is morphing into a vast mobile sensor fleet. In the second chapter of this thesis, we introduce a middleware architecture and implementation that address the needs of a distributed system of mobile sensors comprised of vehicles and intersections producing traffic related data for traffic safety and operations. The middleware addresses the technical challenges of locating and binding to data producers in real-time conditions. A middleware broker that provides a querying language and HTTP-based binding interface proves to alleviate those challenges. The proposed middleware architecture in Chapter 2: utilizes a broker and was implemented for three applications to address: traffic management, intersection safety and vehicle-to-vehicle safety. The performance of the implementation is quantified in terms of the overhead incurred from using the middleware in addition to testing the real-time behavior of the system. The performance measurements show the middleware to be efficient enough for the road safety and congestion relief applications by limiting the overhead on the system to the order of 100 msec.

1.1.1 Impact of Middleware on ITS Mobile Applications

In the field of Intelligent Transportation Systems, the concept of vehicle to infrastructure integration has been evolving as a new paradigm for surface transportation. Under the direction of IntelliDrive, a US DOT program, the concept of integrating the vehicle with the infrastructure takes on a spirit of cooperation between industry (automakers, suppliers, communication service providers) and government (federal, state, and local). IntelliDrive presents the opportunity to advance surface transportation safety, mobility, and sustainability through the use of ITS technology to connect vehicles together and to the infrastructure.

In 2006, the IntelliDrive program – operating at that time under the name of the Vehicle Infrastructure Integration (VII) Program – released an architectural document [1] outlining how the various components of VII in the US would integrate. The architecture was based on On-Board Units (OBU) and Road-Side Units (RSU) communicating with each other via a centralized message switch board. The architecture required communication over Dedicated Short Range Communication (DSRC) spectrum for all of its components and applications. Since then, IntelliDrive has relaxed the requirement of DSRC, except for active safety applications, and has introduced other means of communication [2] such as WiFi, WiMAX, 3G and 4G based on availability. The new talked-about architecture; although open to various communication protocols, and possible layers between the software and hardware, does not leverage any of the existing middleware technologies on the market. Existing middleware technology such as that provided by IBM, Microsoft, BEA, and Oracle, and that has shown to work in several other sectors of the economy such as finance, manufacturing, healthcare, etc. could be used for ITS applications. Our work in this field uses existing middleware components such as those in [3], [4] & [5]. These middleware are built to be used in stationary environments and under-perform in real-time dynamic situations [5] [6]; we improve on them to be used in ITS mobile applications. Not only does this provide an expandable and robust architecture for building ITS mobile applications, but it also offers a faster route-to-market by utilizing and integrating already available technologies that do not require new infrastructure deployments. This middleware-based architecture also allowed us to build the mobile safety application presented in Chapter 3 of this thesis with no major infrastructure or vehicle modifications.

1.2 Smartphone-Based Driver Safety Application

The middleware described above transforms data producers, such as road-side sensors, in-pavement loop detectors, GPS sensors in vehicles and in Smartphones, to HTTP accessible services. These services can be integrated on various hardware platforms including desktops, OBU's, RSU's, and Smartphones. We choose for our implementation a Smartphone-based system with the purpose of improving driver safety. Chapter 3 in the thesis is dedicated to describing the design of the system and its impact on driver behavior as measured in a field test of 9 users on the highways of the San Francisco Bay Area. The field test shows that warnings given to users on slow traffic conditions 1-mile downstream on their road causes them to drive safer by experiencing a smoother speed profile on that section of the highway. This finding contributed to further testing of a purely vehicle based implementation of the same application. This test is currently underway. The next sub-section highlights the impact of having a Smartphone deliver such a result.

1.2.1 Impact of Smartphones on ITS Mobile Applications

The US DOT and the automaker industry have worked on integrating safety applications into the vehicle and road-side infrastructure. The VII architecture [1], championed by IntelliDrive, places safety applications – in particular soft-safety applications – among the list of applications to be deployed in Day 1. This list includes applications that provide the traveler with information regarding the performance of the trip, including travel times, incident alerts, road closures, work zones, non-recurring congestion, and civil emergency notifications.

Since 2006, various automakers have been working on embedding safety applications into the dashboard of the vehicle. In the last two years, car manufacturers (Volvo, Audi, BMW,

Mercedes, Ford, etc.) have started to release embedded sensors in their vehicles providing the driver with warnings on objects in their blind spots, and close proximity to forward or side-based objects. These applications are isolated systems that are disconnected from the infrastructure and provide a vehicle-centric sensing hardware to detect what is in the vicinity of the subject vehicle. Similar to the automaker's in-vehicle applications, there were earlier research efforts expended on vehicle-embedded safety applications such as the Forward Collision Warning (FCW) performed by the ACAS project [7].

The VII Architecture document presented by IntelliDrive provides a more "connected" approach to soft-safety applications. The IntelliDrive approach to safety applications requires the application to converse with other vehicles on the road and more importantly with the infrastructure sensors. IntelliDrive differentiates between those applications that require real-time communication for active safety applications and those that require a more soft form of real-time communication allowing for soft-safety applications to exist. It is the latter that we refer to here. The solution presented by IntelliDrive for soft safety applications relies on the availability of special On-Board Units (OBU's) that are manufactured for the purpose of communicating with the proposed VII infrastructure (which includes DSRC).

In both the above approaches to delivering soft-safety applications, we see a hurdle for fast deployment. Proliferation of soft-safety applications will be hindered by the rate of turning around the vehicle fleet and by the ability of the government to rollout new infrastructure. This motivated us to consider alternative approaches to delivering soft-safety applications to the driver using the technology that is already available in the market without massive investments in infrastructure and with no alteration to the vehicle. It is this motivation that drove us to using Smartphones as a delivery mechanism of soft-safety applications to the driver. Smartphones are now powerful computing online devices, with location and acceleration sensors. Moreover, they are within the reach of a large percentage of the population; 72.5 million Smartphone users in the US as of the end of 2009 [8].

The work presented in this thesis, uses the Smartphone as a user interface to deliver soft-safety warnings to the driver. We address the various challenges involved in handling resources on the Smartphone while keeping it a phone, delivering information to the driver without distracting the driver, and presenting the driver with an easy-to-use interface that runs quietly in the background and only issues alerts when the situation on the road warrants an alert.

1.3 User Preference Adaptive Services

One of the biggest challenges in delivering consumer services on a small platform device such as a Smartphone or in-vehicle dashboard is user adoption; [9] & [10] state that less than 20% of users who download Smartphone applications use their applications and those that use it, use it on average 1.2 times a day. Users are of different ages, sexes, and psychological background with different perception to information presented on small screen devices such as Smartphones.

In Chapter 4 we show that a Support Vector Machine (SVM) classifier can be applied to the Smartphone-based soft-safety application presented in Chapter 3 to learn from past user feedback with regards to alerts they have received. An implementation of the SVM classifier predicts the classification of the alert as favorable vs. non-favorable with an average of 75% accuracy for all 9 users in the field test.

Chapter 5 presents a decision tree model to predict the destination of the driver at any point in their route. A model is built for each driver from their previous trips. An algorithm for detecting destinations from Smartphone GPS traces as captured by the Smartphone-based soft-safety application is also presented in this chapter. Using Decision Trees for each driver, we are able to predict each driver's destination at any point in their trip to 96% accuracy using only GPS traces as input.

1.3.1 Impact of Machine Learning Techniques on User Experience

In delivering an ITS mobile application using small devices, two main concerns related to user experience present themselves: one is the nuisance attributed to high false alarms; the other is the inability of the user to key-in large amounts of information (such as an address) on a small-screen device. While the first challenge has been addressed by others before us; their approach to the problem was to improve on the algorithms producing the alarm to eliminate false alarms. In Chapter 5 we show that there is no on-size-fits-all approach to this problem and that individuals – as surveyed in other studies – have shown individual preference to how they perceive alarms from ITS mobile applications. Regarding the second challenge, no attempts have been made to address the difficulty of entering information on a small form-factor device in ITS mobile applications.

We introduce machine learning as a way to address both challenges. Creating a user profile that is unique to each particular driver, allows us to repress false alarms as perceived by each driver. Furthermore, by learning the driver's habits of driving and including that in their user profile, we can infer where they might be going offering them a selection of destinations rather than requiring them to key in full addresses. Both user-preference learning services reduce the frustration of the driver with the system and enhance the user experience.

CHAPTER 2: MIDDLEWARE TO ENHANCE MOBILE COMMUNICATIONS FOR ROAD SAFETY AND TRAFFIC MOBILITY APPLICATIONS

Middleware has emerged as an important architectural component in supporting distributed applications. The role of middleware is to present a unified programming model to application writers and to mask out problems of heterogeneity and distribution. The research presented in this chapter is motivated by the convergence of the embedded sensor and mobile communication revolutions in the automobile industry. The vehicle fleet is morphing into a vast mobile sensor fleet. In this chapter, we provide a middleware architecture and implementation that addresses the needs of a distributed system of mobile sensors comprised of vehicles and intersections producing traffic related data for traffic safety and operations. We discuss the technical challenges that the middleware addresses and describe a prototype implementation. Traffic management, intersection safety and vehicle-to-vehicle safety applications are three applications described and implemented on the middleware. We conclude this chapter by conducting performance measures that relate to the cost of overhead incurred from using the middleware. The measurements show the presented middleware is efficient for the road safety and congestion relief applications presented in this chapter.

2.1 Introduction

The vehicle fleet is morphing into a vast mobile sensor fleet. The cars of today carry Global Navigation Satellite System (GNSS) Sensors and cellular modems, making them ready sources of traffic data, potentially useful to traffic management departments trying to squeeze more out of the existing pavement. The cars of tomorrow may know the blood pressure of their occupants, or measure precipitation, illumination, pavement condition, atmospheric pollutants, etc.

We present a middleware technology to connect the producers of data, i.e., the vehicles and their occupants, with the many potential consumers of the data. A parent may value an alert on her cell phone whenever the son exceeds the speed limit driving her car. The insurance company may be willing to provide the driver with a discount on premium if provided with speed and route data by the driver. Roadway operators see Global Positioning System (GPS) data from cars as a potential substitute or complement to an expensive sensing infrastructure currently operated to measure traffic flow. Therefore the purpose of our research is to develop a technology able to connect the myriad of consumers, some of whom we foresee and others that we do not, with the many kinds of sensor data that will be produced by the vehicles of tomorrow.

The United States Department of Transportation (US DOT)'s Vehicle Infrastructure Integration (VII) [1],[11], the EU's Cooperative Vehicle Infrastructure Systems (CVIS) program [12],[13], and Japan's SmartWay project [14], are three major programs seeking solutions to this producer-consumer inter-connection problem. All three programs aim to proliferate new safety and congestion relief applications enabled by vehicle-borne sensors and wireless modems. However, the VII and SmartWay programs have also tied these applications to a particular communication

technology, namely Dedicated Short Range Communication (DSRC) [15]. By contrast, we believe application programming should be agnostic to communication or sensing hardware. For example, a safety or congestion relief application should be programmed to work over multiple communication media such as DSRC, cellular, Wi-Fi, etc., or any GPS sensor regardless of whether it is mounted in the vehicle, a cell phone carried by the driver into the vehicle, or in a Portable Navigation Device (PND) mounted on the dashboard. The European CVIS program shares this philosophy, calling its technology-agnostic architecture Continuous Air interface for Long and Medium range communication (CALM) [16]. While CALM encompasses several communication technologies including DSRC, cellular, Wi-Fi, Wi-Max, and infra-red (IR), its architecture forces a tight coupling between the CVIS application and the communication technology to be used. This forces the application to change with evolving communication infrastructure and changes to the sensing hardware. To this end, we describe the design and implementation of a layer of indirection, *the middleware*, enabling applications deployed to survive the ongoing evolution of their underlying sensing or communications.

The structure of this chapter is as follows. The section titled Middleware Design describes our middleware. We have adopted a SOAP [17] based Service-Oriented Architecture (SOA). This has the advantage of being an open standard, with tools from capable vendors such as IBM, Sun Microsystems, and Microsoft, and many services are already deployed on the Internet, that can be leveraged to deploy safety or congestion relief applications economically and rapidly. We show this by leveraging Google Maps. Unlike, other middleware like Jini or Corba, the SOA allows application components to synchronize loosely [18][19], which helps program applications to be compatible with different communication and sensing services with different performance characteristics. We then describe the testing and evaluation of the middleware by using it to program three applications with three different architectures. These are:

- A traffic management application involving a component on the Internet estimating the speed on a road, by collecting GPS data from vehicles on the road,
- An intersection safety application involving a component in a vehicle collecting the state of the traffic light from a component living in the traffic light,
- A vehicle-to-vehicle safety application involving a component in a vehicle obtaining the motion data of a vehicle in front of it.

The three applications cover the three kinds of distributed architectures required for safety and congestion relief applications, i.e., local area vehicle-to-vehicle, local area roadside-to-vehicle, and wide area Traffic Management Center (TMC)-to-vehicle. Thus our principal technical challenge has been to transform SOA, as conceived for connecting static Internet based components [19], into a middleware connecting mobile, wirelessly communicating components. This transformation is enabled by the new Broker component, described in the Middleware Design section.

Finally, the flexibility gained by use of the middleware comes at a price. We present results quantifying this price in the section titled Performance Measures. The extra middleware layer between application component and sensor, adds latency. We quantify this latency and discuss it in the context of the overall latencies tolerated by different safety and congestion relief applications. The Conclusion summarizes our findings and discusses important extensions currently underway.

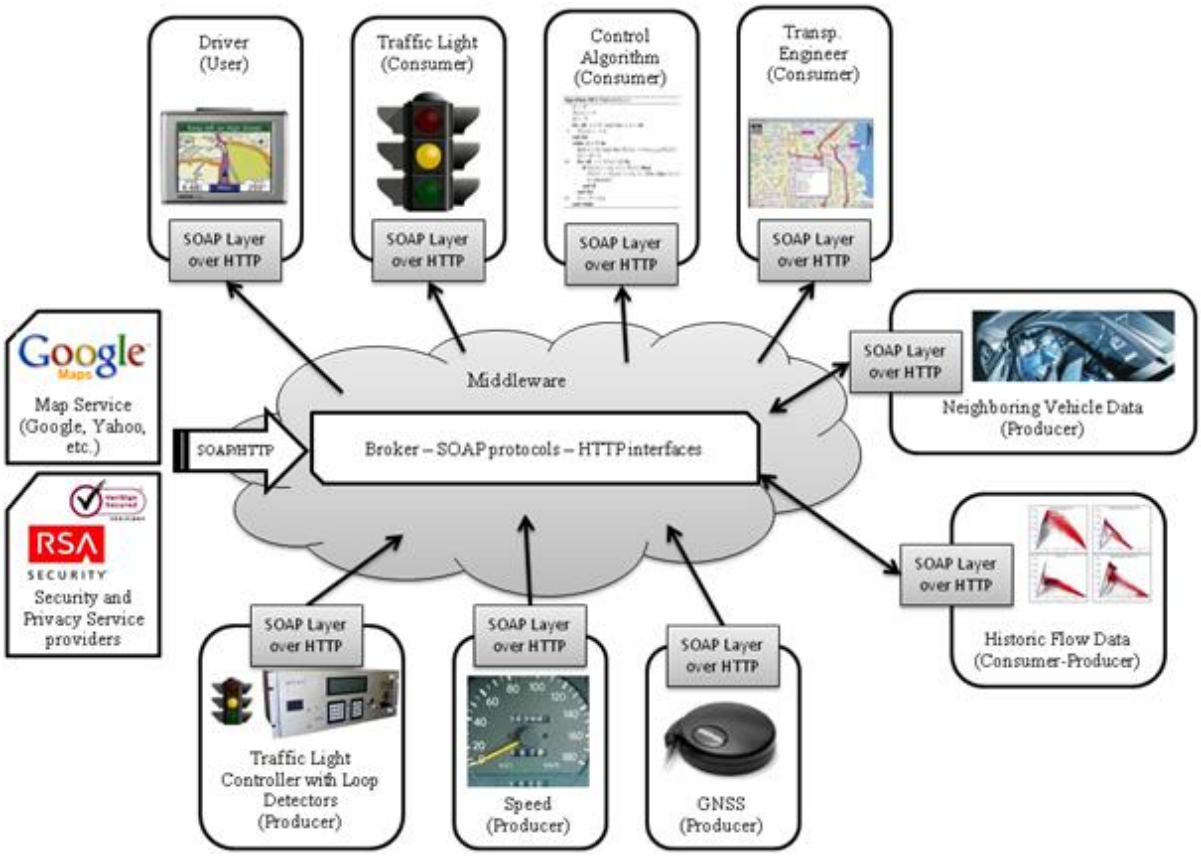

Figure 2.1. Middleware architecture showing the various producers and consumers of data and the role of the broker, SOAP protocols and HTTP

2.2 Middleware Design

The middleware provides a layer of indirection, enabling applications deployed to survive the ongoing evolution of their underlying sensing or communication technology. As a result, application programming becomes agnostic to communication or sensing hardware. The middleware layer consists of three main components (Figure 2.1) that address the complexity of the producer-consumer inter-connection problem in traffic safety and congestion relief applications:

- The Broker: The broker is used to initiate the connection between the application and the sensing hardware over the appropriate communication medium. The broker also solves the sensor discovery problem, discussed in Section 2.2.1, by providing a geo-spatial query interface allowing the data consumer to locate the sensors of choice.
- SOAP-based SOA: SOAP allows application components to synchronize loosely, which makes applications compatible with different communication and sensing services. This reduces the complexities of the heterogeneous communication layer and data sources to a simple standardized "services" interface allowing the application developer to consume whatever data is generated from the sensors.

- HTTP Interface: Sensing services are exposed to the outside world via a web URL allowing applications to consume their data over HTTP thus simplifying the development effort of connecting across various Application Programming Interfaces (API)s. As a result, the application developer can treat the intricate details of the communication standards that reach the sensors (producers) as black boxes interfaced through the HTTP protocol.

2.2.1 Component Discovery

The principal technical challenge in designing the middleware has been to transform SOA, as a concept conceived for connecting static Internet based components, into a middleware connecting mobile, wirelessly communicating components. In a static Internet setting, communicating devices are well located and identified.

Locating devices on the Internet is not a new problem and several solutions have been presented [20][21]. The challenge remains when attempting to locate a "moving" device in the network. The term "moving" is used to define a device that moves from one network address space (subnet) to another on the Internet. Devices in the same subnet can be located [22], devices across subnets can be located [23], but devices moving from subnet A to subnet B face the challenge of being located by devices in subnet C [24]. In transportation settings, subnets are defined around the road-side unit and form between in-vehicle units connecting to a single road-side unit. As vehicles move on the road, they might encounter several road-side units. The time that a vehicle stays connected to a single road-side unit is small compared to the time that a node spends connected in static Internet networks. A vehicle traveling at 90 km/h passing near a road-side unit with a wireless communication range of 200 m (typical of Wi-Fi routers used in static wireless networks) would have a maximum time-span of 8 s to be in the subnet of that road-side unit. After 8 s, it might be off the network for some time until it encounters another road-side unit of a different subnet for another maximum time-span of 8 s.

The other side of the sensor discovery problem is in identifying the mobile component to be reached. In distributed networks, this problem is referred to as the routing problem and is tightly related to the location problem. Communicating nodes in a static network are identified by their address [25] which is in many cases a static unique identifier for that particular node. As communicating nodes start moving inside the same subnet, a local Domain Name Server (DNS) would work as an address directory keeping track of the current address for each node [25]. When communicating across subnets (such as the Internet), a global DNS is used. DNS addresses the routing problem in static communicating networks where address changes are rare and address updates can take up to 24-48 hrs in large networks [24][25]. A vehicle traveling at 90 km/h for 50 km may traverse 4 or 5 road-side units and get 4 or 5 different addresses in about half an hour. A more robust solution than global DNS needs to be utilized.

2.2.2 The Broker

In order to solve the location and identification problem, we introduce the broker component as a principle element of the middleware. In simple terms, the broker acts as the entity that introduces a consumer of data to a producer of data and provides the location and choice of communication medium over which the two communicate. Consider a Bay Bridge Speed Monitoring Application that averages the speed of westbound vehicles on the Bay Bridge and returns the result to the user (a human or another application). In this example the Bay Bridge Speed Monitoring

Application is the "consumer". It first queries the broker for a list of vehicles on the westbound side of the Bay Bridge. The broker returns the result set of all vehicles that have registered with a GPS location on the Bay Bridge and a heading that is Westbound. The vehicles that satisfy the query are the "producers". Each entry in the broker query result set consists of the location of the vehicle and the communication technology to reach it – this could be a URL over Wi-Fi or an IP address and port number over DSRC, etc. The main assumption in this example is that the broker location is known for vehicles to register with it and for applications to query it. The broker in this case can be stationary at a TMC or other public entity. In a different scenario, a parent may value an alert on her cell phone whenever her son exceeds the speed limit. In this case the broker can be part of the cell phone application requiring the vehicle to register with it whenever it comes online over DSRC, Wi-Fi, or cellular. The broker can also share registered producer services with other brokers of known public locations. For example, insurance companies willing to provide drivers with a discount on premium, if they provide them with speed and route data, might query several brokers installed at several geographically distributed locations. Brokers can also be part of the in-vehicle setup in which vehicle-to-vehicle communication would rely on to register and connect with data producing services in the vehicle neighborhood. The mobile nature of the broker and the flexibility by which services can register and query it are critical to the middleware layer. Broker-to-broker communication protocols by which brokers share and update each other's information is another important research topic not addressed here. For the purposes of this chapter, a simple replication model is assumed between multiple brokers. See [26] for more details.

In order to satisfy the above requirements, the broker is designed to act as a service in a SOA by which data consuming applications connect to it over SOAP, get the location data they need, disconnect and then connect directly to the data producing service. The following public methods are exposed as part of the broker:

- GetRegisteredServices: returns an XML list of all services registered with the broker.
- GetServices(GPS Area): returns an XML list of all services registered in a certain GPS square area.
- GetServiceProperties(Service ID): returns an XML document containing all the data properties provided by a certain producer service. This method does a direct relay to the producer service's public methods and returns the result to the consuming service.
- RegisterService(ServiceInfo): the producer service information passed as an XML string to this method registers the service with the broker. Minimum service information constitute of the GPS location and a URL.
- SaveRegistrationInfo: dumps all registered services information to a local file on the broker server.
- SetTrace: enables debugging and tracing levels for the broker service.
- UnRegisterService(ServiceID): forces a de-registration of a certain service.

In addition to the public methods, the broker has internal operations that keep track of which services are registered and which have gone out of range and automatically un-registers them. Broker-to-broker communication and synchronization is handled by replicating the data across all brokers in the range of a certain broker service. Other methods are being developed as part of this research effort to improve the management of registered services in the broker and broker-to-broker communication; please refer to [26] for more details.

2.2.3 SOAP-based SOA

The mobile components in a traffic safety or congestion relief application need to communicate over various types of wireless media. Several media have been proposed among which is DSRC [15], WiFi, Cellular Data Plans, etc. While the current efforts of VII, CVIS and SmartWay tend to build different components for the different wireless communication media, the middleware described here would enable the building of components that work over different wireless communication media. A principal element in enabling this to happen is the choice of the interface layer between the components. The problem of defining an interface that is agnostic to the communication layer has been addressed in Corba and SOAP. In Corba the solution is aimed at fixed nodes in a static network. SOAP presents a more loosely coupled interface between components allowing them to be mobile. SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment [17]. Our middleware adopts a SOAP-based Service-Oriented Architecture. It is XML-based that consists of three parts:

- An envelope that defines a framework for describing what is in a message and how to process it.
- A set of encoding rules for expressing instances of application-defined data types; in this thesis we borrow those rules adopted for the Microsoft .Net Framework.
- A convention for representing remote procedure calls and responses.

The choice of SOAP as an information exchange protocol allows for the "consumer" and "producer" services to describe their interfaces regardless of the network protocol used to communicate. This achieves the application-communication technology detachment which the middleware aims for. Furthermore, SOAP has the advantage of being an open standard, with tools from capable vendors such as IBM, Microsoft, Sun Microsystems and others. Many SOAP-based services already deployed on the Internet, can be leveraged to deploy safety or congestion relief applications economically and rapidly. We show this by leveraging Google Maps over Wi-Fi and DSRC without any modifications to the code at Google. Similarly other services such as data encryption, authentication, and caching already exist as SOAP-based services on the Internet and can be easily incorporated into the SOA middleware. SOAP is also adopted in the design of the broker. The methods of the broker listed in the previous section are accessible over SOAP making the communication to the broker possible over different communication technologies.

2.2.4 HTTP Interface

The middleware exposes the SOAP-based services as web-based services allowing "consumer" and "producer" services to converse over HTTP. The parent that is interested in her son's driving habits can access his vehicle sensors through her cell phone or desktop browser by pointing to her son's vehicle service HTTP URL. Using HTTP allows the parent to further subscribe to the data stream from her son's vehicle using HTTP-based technologies such as Really Simple Syndication (RSS) [27] and ATOM [28] which come pre-installed in several Internet browsers. HTTP also allows the connection between the vehicle and the browser to be encrypted using publicly available technologies for HTTP encryption such as TLS (HTTPS) [29]. Having a web-authentication service blocking access to the vehicle's data except by authorized consuming services or users can be easily incorporated as a service in the middleware.

By adopting HTTP as the interface by which services in the middleware can be reached, we reduce the reliance on proprietary Application Programming Interfaces (APIs) that tie the service to a certain communication technology. A sensing hardware API would be enveloped with a SOAP protocol and given an HTTP interface before it becomes a service that can register with the middleware broker (See Figure 2.1). In-vehicle GNSS sensors, traffic-light controllers and vehicle Controller Area Network (CAN)'s are converted to web services before they are registered with the broker. Application developers or consuming services only interface with the HTTP URL and the SOAP message of the web service to converse with the sensor, without having to go into the details of the GNSS sensor's API, the traffic-light controller's interface or the vehicle's CAN standard.

2.2.5 The Middleware Modes of Connection

The middleware, through SOAP and HTTP allows for two modes of producer-to-consumer service connectivity:

Asynchronous connectivity: After getting the location and means of connection from the broker, consumer services can connect to the producer services in an asynchronous manner. Failure or timeouts at one or more of the producer services can happen without crashing or delaying the rest of the consuming application processes. In an asynchronous mode, the consumer service process is independent of the data delivery from its producer service(s). Each consumer service wishing to launch an asynchronous connection to one or more producer services will spawn a set of "listeners" that will contain the identifier for each producer service. The listeners will be active in the memory of the consumer service process for a certain timeout period during which the process will continue doing its work [30]. Once a listener captures a result from the producer service it will activate a sub-process at the consumer service to handle the captured data and end the listener session.

Streaming connectivity: After getting the location and means of connection from the broker, consumer services can subscribe to data streams from producer services. Such a connection can, for example, provide a consumer service with data elements at a certain frequency (every 5 s) for a certain period of time (for 30 min) from a producer service. This data stream is provided through a RSS 2.0 channel or an Atom 1.0 feed. Both data streaming standards are built on top of HTTP and have been adopted in the middleware. In RSS data streams, updates as frequent as every minute can be maintained, whereas in Atom feeds, no minimum frequency constraints are set, allowing more real-time behavior of the streaming data.

2.3 Middleware Evaluation

To test and evaluate the middleware, we surveyed the several application scenarios proposed by the USDOT's VII and the EU's CVIS programs. Both programs have a myriad of application scenarios covering safety and non-safety applications [1][13]. The USDOT's VII program lists 43 such scenarios of which 33 are safety related. These scenarios can be summed in three kinds of distributed architectures: local area vehicle-to-vehicle, local area roadside-to-vehicle, and wide area TMC-to-vehicle. The following three applications, covering the three kinds of distributed architectures, are designed to test and evaluate our middleware:

- A traffic management application involving a service on the Internet estimating the speed on a road, by collecting GPS data from vehicles on the road. The user interface would be

a website visible through a browser displaying vehicles moving on a map provided by the Google Maps service. Clicking on any of the vehicles on the map would stop the data collection and display the details of the sensor reading from the clicked vehicle. The speed at which this vehicle is driving will be among the data displayed.

- An intersection safety application involving a service in a vehicle collecting the state of the traffic light from a service living in the traffic light. The user interface would be a browser based application viewed on the laptop inside the vehicle showing the approaching traffic light on a map provided by the Google Maps service. The user can click on the traffic light and read the time-to-green in seconds as well as see the current phase status of the traffic light.
- A vehicle-vehicle safety application involving a service in a vehicle obtaining the motion data of a vehicle in front of it. The user interface will be a browser based application running on the laptop inside one vehicle displaying the other vehicle on a map provided by the Google Maps service. The user in the first car can click on the second car to obtain sensor data on that car. Sensor data includes: GPS position, speed, wiper status (on or off) as read from the vehicle CAN.

The three applications above are implemented in such a way to evaluate the middleware and show that:

- Different types of vehicle sensors (GNSS, Speed, and CAN) by different vehicle manufacturers (American Ford and European VW) can be made into web services that can be consumed similarly by different consuming applications
- The same "consumer" application (Traffic Management) can communicate over Wi-Fi in one vehicle and over DSRC in another vehicle without having to re-code the application
- Different "consumer" applications (Traffic Management and Intersection Safety), each written in a different programming framework, Microsoft.Net on Windows vs. Java on Windows, are able to consume a web service written in Java on Linux at the same time
- The publicly available service of Google Maps can be seamlessly integrated into the middleware without Google changing their code to deliver maps to the middleware applications

2.3.1 Hardware and Middleware Setup

A Ford mini-van and VW Golf are used to implement the above applications. Each vehicle was equipped with a laptop. Each laptop has two Wi-Fi radio cards and a DENSO DSRC radio connected through the Ethernet port. Cellular radios are being investigated for the second release of the prototype and are not part of this deployment due to the security constraints imposed by cellular providers on certain network ports. Those security constraints enable one-way session initiations from the cellular to a remote web server but limit the session initiation on the cellular phone itself. This prevents us from hosting a SOAP and HTTP web service on the cellular end. We are investigating the use of a "push" service to be installed on the cell phone as a means of bypassing this limitation and will be implemented in the second release of this middleware. Our current middleware prototype relies on Wi-Fi and DSRC to be available in the field. USB-enabled GPS sensors and Bluetooth-enabled vehicle CAN connectors were used to connect the sensors to the laptops in the vehicles. For each sensor a service was written using Microsoft .NET to convert the sensor interface into a SOAP-based service accessible via HTTP. The service was then hosted on a light-weight web server on the laptop. A third laptop was installed

at the traffic light that connected to the 2070 traffic light controller at the UC Berkeley Richmond Field Station Test-bed Intersection. This laptop had a Wi-Fi radio, a DSRC radio and a connection to the Internet backbone via Ethernet. A service was written on that laptop to convert the 2070 controller readings into a SOAP-based service. A desktop located remotely in an office and connected to the Internet was also used as part of the evaluation of the middleware. In addition to the SOAP services written to convert the sensor readings to web services, the broker was installed in one of the vehicles and on the desktop in the office. Figure 2.2 shows a schematic representation of the setup.

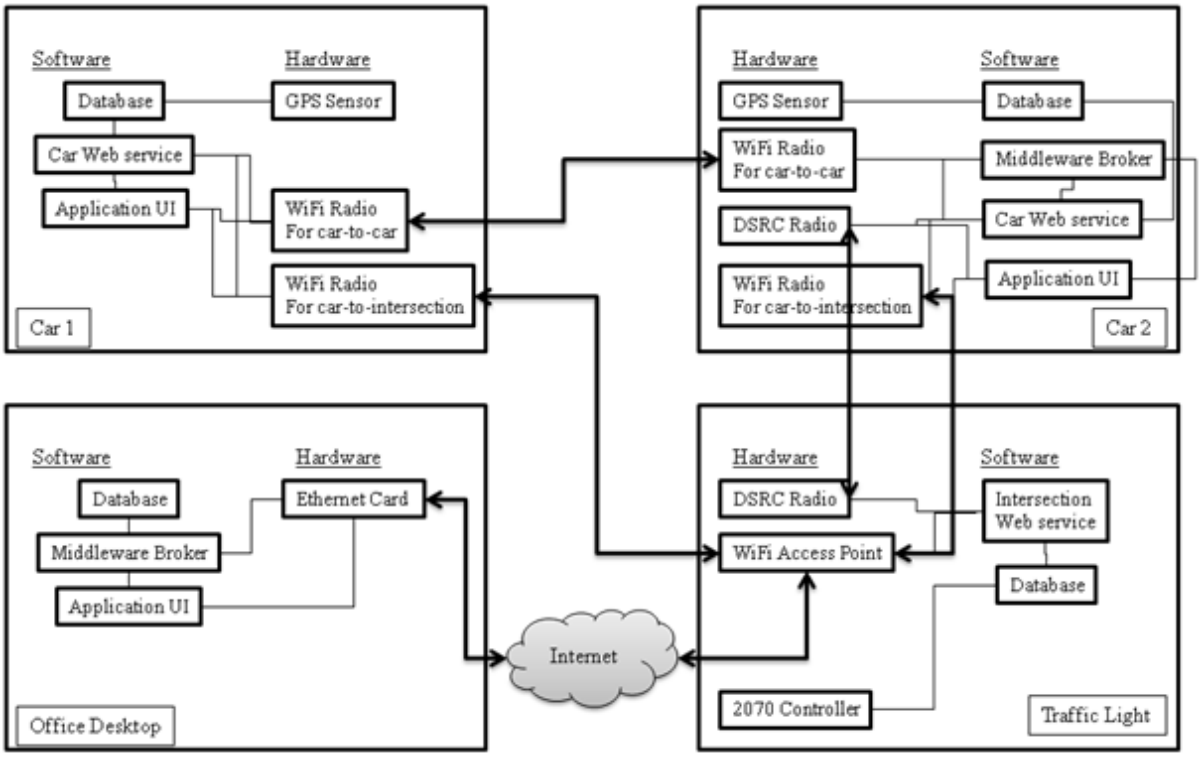

Figure 2.2. Test-bed setup used to test and evaluate the middleware

In-vehicle "registration" services would check for online connectivity on the Wi-Fi or DSRC channels. If detected, the services would register the vehicle URL and GPS location with the broker. In the current setup the vehicle would have to register with one broker and the other broker would receive the data through replication. As vehicles drive in and out of range of Wi-Fi or DSRC, the "registration" service automatically registers and de-registers from the brokers forcing an update on their location and identity in the broker. The traffic light controller does the same and registers with the broker. Since the traffic light is stationary and always connected, its registration with the broker persists until it loses connection due to failure in the Ethernet backhaul at the traffic light cabinet. When this happens, the traffic light controller can still register with any broker that is in a nearby vehicle in-range of Wi-Fi or DSRC and offer its "producer" service to those vehicles.

2.3.2 Applications Description

The three applications listed above are coded in different languages: C#, VB.Net, and Java. The traffic management application and the vehicle-to-vehicle safety application are written as web applications utilizing the Google Maps service to display vehicles and traffic light data on the map. The intersection safety application is programmed as a windows application providing safety alert messages on one DSRC channel and commercial advertisement messages on another DSRC channel to the driver. Figure 2.2 shows the complete setup used to evaluate the middleware.

The traffic management application (Figure 2.3) is installed at the office desktop to monitor vehicles in the field as they drive in and out of range. Google Maps is used to display the location and information provided by the vehicles that are in range. The traffic management application connects to the vehicles by querying the broker to determine which vehicles are online (registered) and what URL to use to connect to their services.

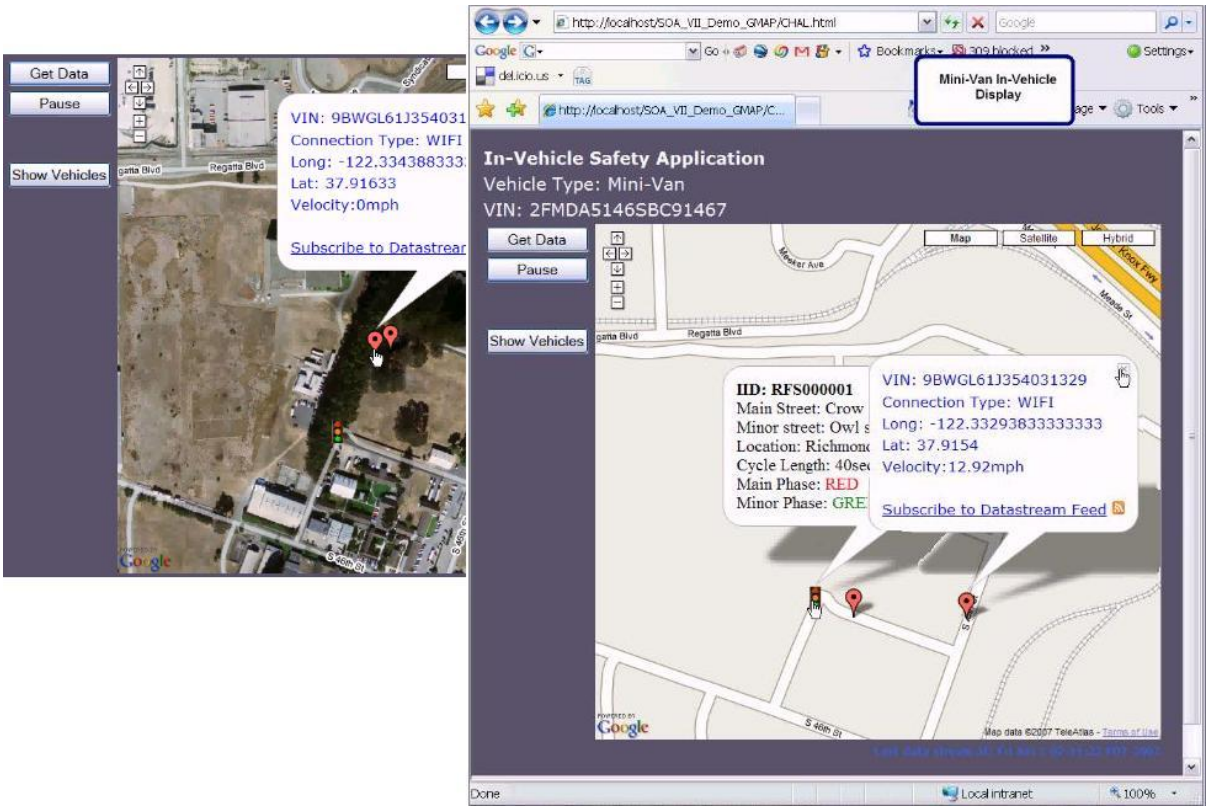

Figure 2.3. Screenshots from the Traffic Management Application (left) and in-vehicle safety application (right) showing traffic-light phase information and nearby vehicle data. Both applications are web-based and utilize Google Maps to display data

The intersection safety windows alert application is installed on the vehicle laptops and queries the closest broker for registered services. This application is only interested in traffic light services, so it connects to the traffic light services provided by the broker. It establishes a data stream with the traffic light and shows the safety and non-safety (commercial) alerts to the driver in the vehicle.

The vehicle-to-vehicle application (Figure 2.3) is installed on the laptops inside the vehicles. This application, similar to the traffic management application utilizes Google Maps to display nearby vehicle and traffic light data. The application queries the broker that is in one of the vehicles or the broker at the office, depending on which is online and in its subnet, and obtains the list of services in its vicinity. As the vehicle approaches a traffic light, the application starts providing traffic light phase information (number of seconds to the next green or red) on the Google Map. The application simultaneously displays the nearby vehicle on the map with its current speed.

2.3.3 Message Content for Road Safety and Traffic Mobility Applications

The applications, described in the previous section, exchange data for two main reasons: to communicate with the broker and to communicate with the data producers once the broker brokered a connection. The message format for those two types of communication is different and so is their content.

When communicating with the broker, the application or the data producing service utilizes the generic SOAP protocol as defined in Section 2.2.3. Figure 2.4 shows a sample SOAP message used to register a car web service with the broker. After the broker establishes the connection between the producer and consumer of data, the message format changes. SOAP is still used as the packaging protocol; however the content of the SOAP message implements the SAE J2735 DSRC Message Set Dictionary Standard [31] that is defined by the US DOT for message exchange in Traffic applications. Other message exchange standards for particular traffic applications can be implemented in the SOAP package.

Figure 2.4. SOAP message to register a car producer service that provides all vehicle properties (AllProperties) with the broker

2.3.4 Enhancements to In-Vehicle Setup

The current setup depends on laptops existing inside the cars. Our next step is to reduce the footprint of the physical hardware in the car by providing a small hardware device. The device that is being prototyped includes a CPU and flash drive, 3 radios (WiFi, DSRC, and Bluetooth), and 4 USB connectors to connect GPS and other sensors. The device operates from the cigarette

lighter of the car and has no screen or visual interaction capability. The intention is to utilize existing screens in the car such as the screen of a PND or cellular phone with the driver. Our testing has been on cellular phone connected to the in-vehicle device using Bluetooth. The user would be able to open the above described application in the cell phone browser and get all the information that the application holds. By utilizing a personal device such as the cellular phone, we would be able to provide a more personalized experience to the user regarding the subscription to information from the middleware. [32] presents the benefits and our methods of personalizing user experience for traffic applications.

2.4 Performance Measures

The introduction of a middleware layer into the system affects the performance of the overall system. The performance hit is the delay incurred by using the middleware to setup the connection between two services. The performance of the traffic management application presented in the Middleware Evaluation section is not affected by the middleware when it is displaying the vehicle data on the map since the middleware is not part of the connection between the vehicle and the application after the initial connection is established. It is only when the application is trying to query the broker and establish the initial connection that the traffic management application performance is affected by the middleware.

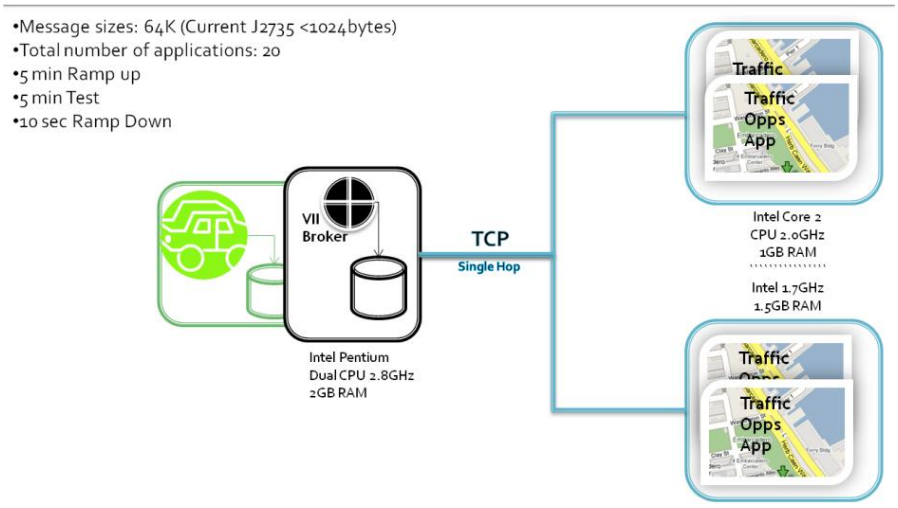

Figure 2.5. Setup used to measure middleware performance

In order to measure the middleware overhead, we perform a modified setup of the components. The modifications aim at accomplishing two main tasks: one, to isolate our measurements from any network overhead that might arise from network latency and congestion; two, to increase the load on the broker to reach saturation stress levels allowing us to capture the overhead cost of a stressed broker. The setup in Figure 2.5 is used, in which a single vehicle sensor, converted to a web service, registered with a broker on the same machine as the vehicle sensor. This eliminated any network overhead between the sensor and the broker. Two other machines were utilized as virtual application creators that used Oracle's Application Testing Suite (a web stress testing tool) to simulate the existence of about 10 clients consuming the web service and running

simultaneously for a 10+ min duration (the first 5 min of the test were ignored for warming up the services, the second 5 min were used for performance measures and the last 10 s were ignored for ramp down tasks). The two machines and the broker are connected over a TCP link with one hop. In this setup the roundtrip transaction time, *r*, from the application consuming the web service consists of the following time elements:

$$r = t_1 + s_1 + b + s_2 + t_2 \qquad (1)$$

where:

t_1 and t_2 are the times spent in TCP to connect to and from the broker

s_1 and s_2 are the times spent to serialize and de-serialize the SOAP message

To figure out the conditions under which the broker (in the given hardware setup) is stressed we ran several experiments each done with a varying number of clients making requests to the broker. By varying the number of clients we vary the number of transactions on the broker. The experiments varied the transactions between 500 and 16,000 transactions in the active 5 minutes of the test. Figure 2.6 (a) shows that once you reach a certain number of transactions, the broker starts timing out. This is what we call the hardware threshold since its cause is due to the limitation in the hardware resources available on the broker server machine. Hardware limitation as assumed to be solved by adding more processing power (CPU's, RAM) or by load balancing the broker server across several server machines. The hardware threshold in our experiments is around 2520 transactions in the active 5 minutes of the experiment. Figure 2.6 (b) shows that by increasing the number of transactions beyond the threshold, the transaction processing time stays around the 120 ms. For the remainder of this section, we focus our analysis around the hardware threshold condition.

Figure 2.7 shows an average 8.4 transactions/s throughout the duration of the experiment. Looking at the time it took to process each of the 2520 transactions in more detail, we notice the following: average transaction time is 120 ms; minimum transaction time is 116 ms and maximum is 130 ms. Figure 2.8 shows the variation of transaction time in seconds throughout the active 5 minutes of the experiment. The standard deviation was about 10 ms throughout the experiment.

In analyzing the components that make up the 120 ms roundtrip transaction time we come up with the following:

- $t_1 + t_2$ averaged 1.3—2.6 ms. This number would vary based on network traffic, number of hops and wireless signal strength.
- $s_1 + s_2$ averaged at 47 ms. This number is close to the average published by Microsoft and Sun Microsystems of 50 ms.
- *b* averaged at 71 ms.

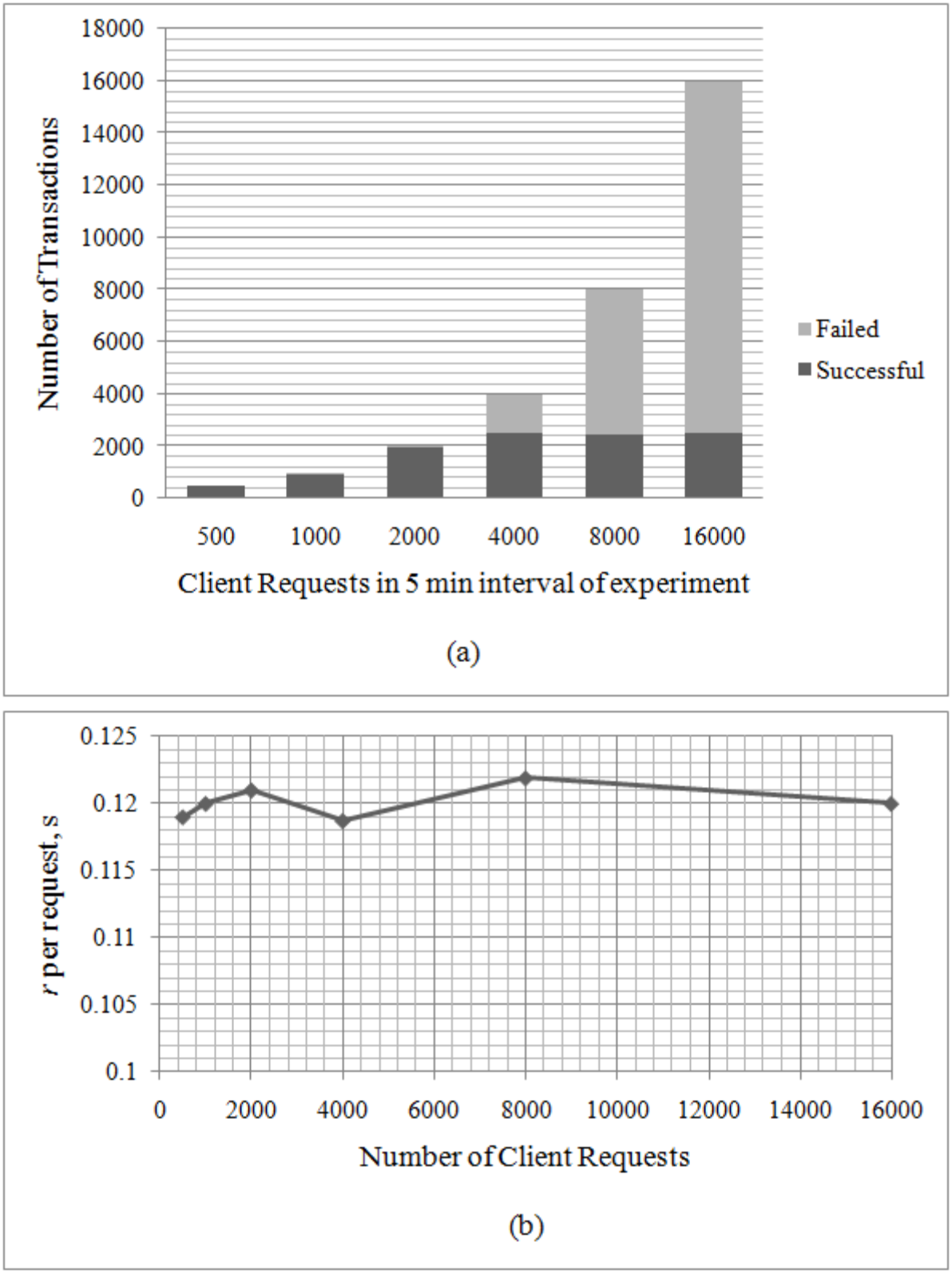

Figure 2.6. Determining hardware threshold. (a) Failed transactions start when hardware limitations are met. (b) Increasing the number of client requests does not affect r once the hardware threshold is met

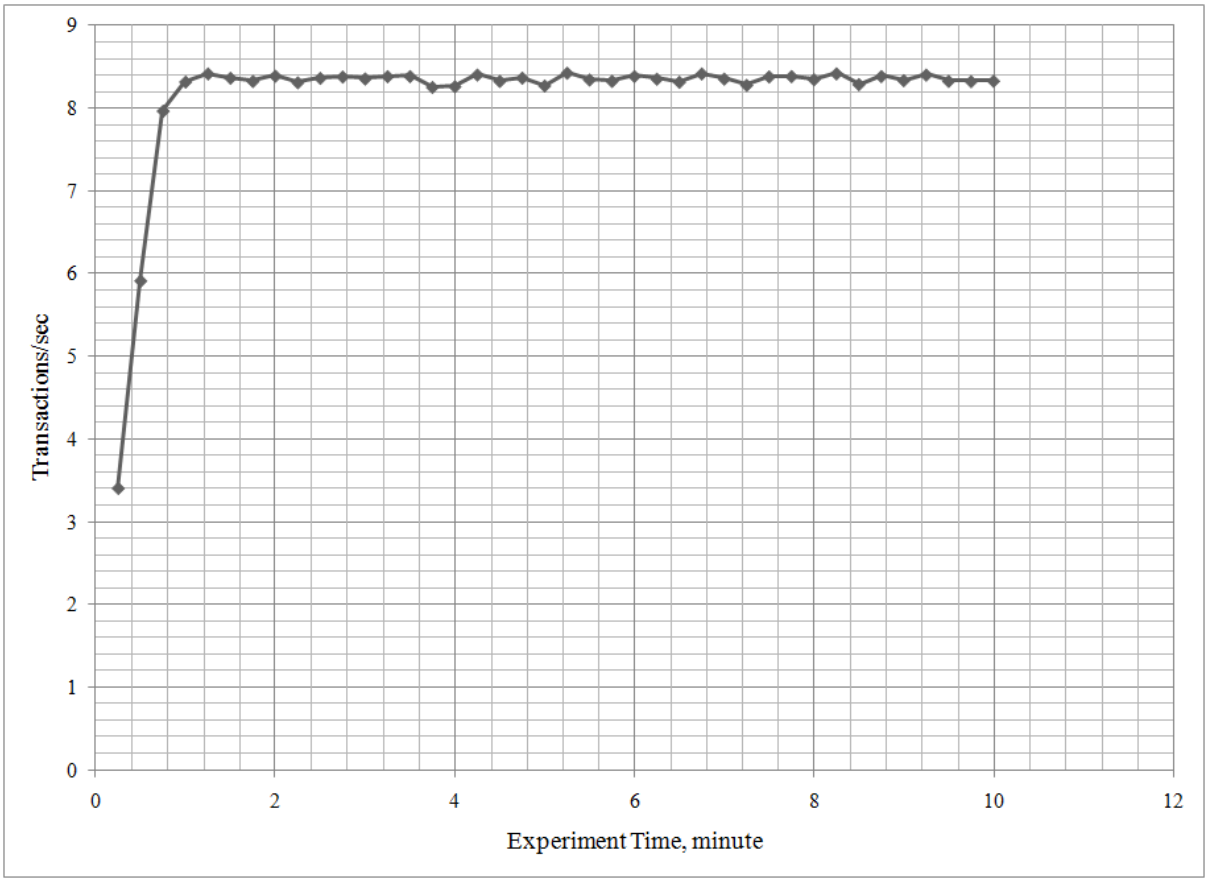

Figure 2.7. Variation of transactions/s handled by the broker as threshold client requests are issued

The 120 ms average round-trip transaction time can be considered as the cost of using the middleware. This number can only make sense when compared to how long a mobile service in the middleware can remain registered with the broker. For a stationary service, this overhead is encountered at the initial exchange of information between producer and consumer services and does not hinder overall application performance. Whereas with mobile "producer" services, a large middleware overhead can prevent certain "producer" services from ever getting "consumed". In order to establish the baseline of how long a mobile service can remain registered and online with a broker, we look at the various wireless communication media ranges (Table 1) and the various speeds at which a mobile consumed service might travel. Wi-Fi is the communication technology with the smallest range of 200 m. Vehicles travelling at speeds of 108 km/h (30 m/s) would take about 6.7 s to fall out of range of the Wi-Fi radio. Given that the middleware overhead is an order of magnitude smaller than the smallest amount of time spent online by a fast traveling producer service, we can safely state that the performance of road safety and congestion relief applications as evaluated in this chapter will not be hindered by the use of the middleware.

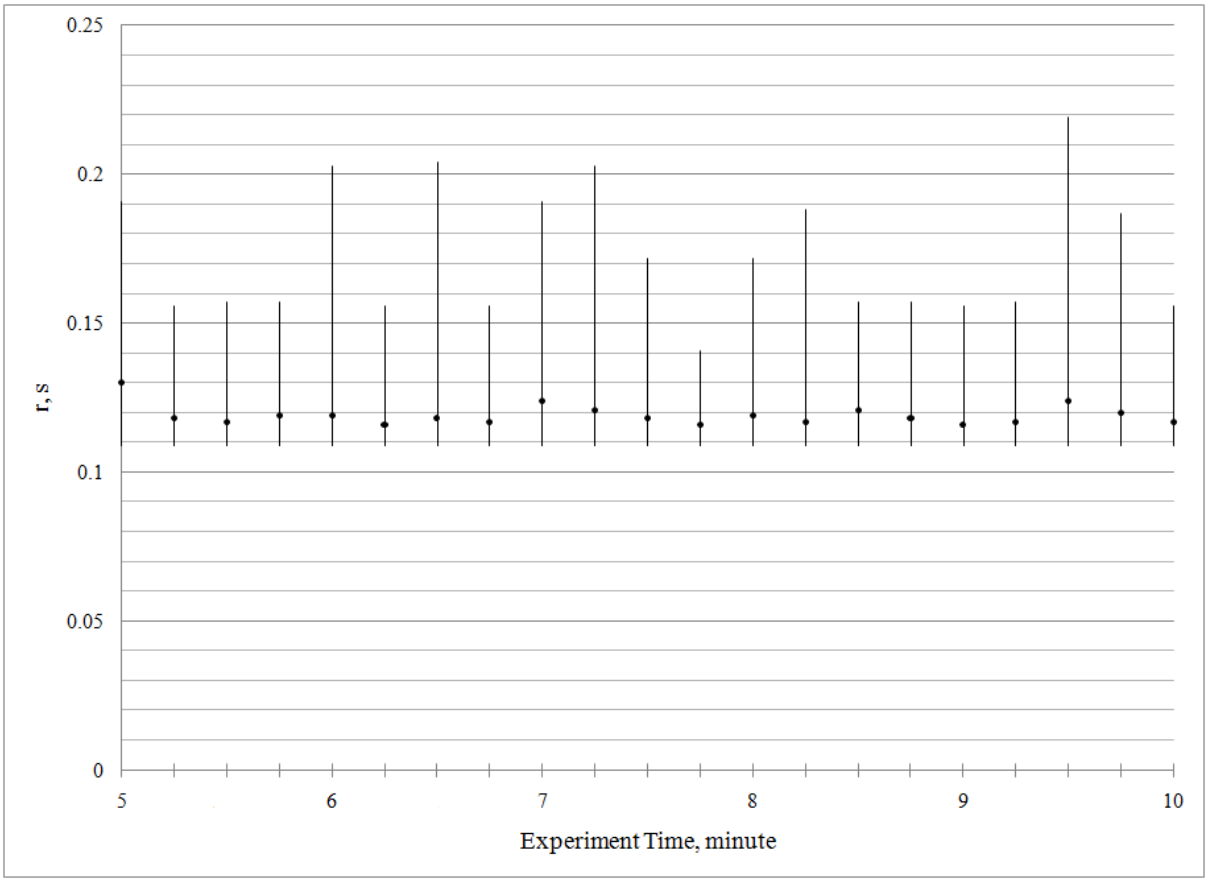

Figure 2.8. The variations in r as the experiment progresses. Plot shows, minimum, average, and maximum values for r between minutes 5 and 10 of the experiment

Table 1. Communication ranges for various wireless communication technologies. (Sources [15] and [33])

Communication	Average Range
Wi-Max	1000 m up to 40,000 m
Wi-Fi	150 m indoor, 300m outdoor, average 200 m
DSRC	1000 m
Cellular	Thousands of meters

2.4.1 End-to-end Test of Middleware

In this section we present the measurements of *r* as encountered in the prototype set up. Wireless communication delays are factored into these measurements and provide a more real-life scenario to compare the cost of the middleware. In this setup, the clients are created on a wireless network communicating over WiFi to a backhaul line traversing more than 1-hop to reach the

broker. The services producing data are also located in a WiFi network that is more than 1-hop away from the broker. In the active 5 minutes of the experiment, only 1398 transactions were processed from the client side. No timeouts due to server overload occurred, so the server was not over-loaded. The *r* for the transactions ranged from 148 ms to 352 ms with an average of 216 ms and a standard deviation of 7 ms. When looking at the components of *r* in equation (1), we notice that the TCP communication time from and to the broker ($t_1 + t_2$) is close to 100 ms. While the numbers presented in this section fall within the acceptable range for road safety and congestion relief applications, we would like to caution the reader that these numbers would vary widely by nature of application, quality and availability of wireless communication, and number of clients in the wireless subnet. The work of the DSRC committee is to reduce delays encountered in the wireless communication medium for traffic applications. By supporting DSRC standards in the middleware, applications written to consume services from the middleware would benefit from these efforts once they are commercialized.

2.5 Conclusion

The implementation presented in this chapter has proved the main objectives that it was set out to deliver. First it reduced the complexities of the heterogeneous communication layer and data sources to a SOAP standardized services interface allowing the application developer to consume whatever data is generated from the sensors. In our implementation we choose web services as the middleware interface. Second, it solved mobile components discovery problems by providing a geo-spatial query interface allowing the data consumer to locate and connect to the sensors of choice by providing a services broker. And third, it offered seamless HTTP connectivity between the application (data consumer) and the sensors (data producer) by hiding the intricate details of the several communication standards from the application developer. As a result, the application developer's task is reduced to just connecting to the middleware services. This simplification imposes an overhead to the connectivity process between components which is measured and compared to applications for which this middleware is intended to be used. These measurements show that the order of magnitude of this overhead is small compared to the applications that use the middleware.

Our efforts described in this chapter will enable us to implement road safety and traffic mobility applications on the cell-phone platform leveraging the middleware presented here. Enabling communication over cellular data plans alleviates the need for infrastructure deployment of WiFi or DSRC hotspots. While we have shown that deploying a traffic safety or mobility solution on the middleware reduces some of the current business challenges involved in deploying an infrastructure solution. This solution still poses a proliferation and user adoption problem. Data producers are required for data consumers to benefit from the service. By having the application platform reside on cell phones, which are in the possession of a large number of drivers, data producers (drivers) can be more easily attained. In this case the challenge would be shifted to creating attractive applications for the drivers and providing user-friendly services. This challenge is partly tackled in the remaining chapters of this thesis.

2.6 Acknowledgments

We would like to acknowledge the effort of the California Partners for Advanced Transit and Highways (PATH) for the use of their facilities and hardware to conduct the tests and deployment described in this chapter.

CHAPTER 3: SMARTPHONE SAFETY APPLICATION AND THE EFFECTS ON DRIVER BEHAVIOR

The effect on driver behavior of using in-vehicle applications which provide soft-safety warnings is unknown. In this chapter we present a Smartphone-based application that delivers a warning to the driver regarding slow traffic ahead s/he might encounter 60 seconds downstream of their travel. We present the findings of releasing the application to nine drivers in the San Francisco Bay Area and collecting data on their driving patterns for 8 weeks using the Smartphone GPS sensors. We provide an analysis of the individual user traces for cases before and after the alerts were activated, and we deduce that a statistically significant change in behavior, where drivers brake less abruptly, is due to alerts received from our Smartphone application. We define an objective measure of "brake less abruptly", and using this measure we analyze the sustainability of this change in behavior throughout the 8 weeks of the experiment. We deduce that the change in behavior undergoes a fluctuation in the first 3 weeks of the experiment and becomes stable after the third week of using the application and persists till the end of the experiment duration. We conclude the chapter by conjecturing the potential effects on safety and traffic flow, given further research and deployment.

3.1 Introduction

Smartphones connected to the Internet through broadband cellular networks provide an intriguing option for delivering timely awareness to drivers, where awareness may be operationally bounded by "more than traffic conditions" and "less than collision warning"; we therefore generally define such awareness as the driver knowledge of safety-relevant traffic conditions in a horizon extending approximately 10 to 60 seconds ahead of his or her path. Smartphones are interesting and likely conveyers of this type of awareness, since nowadays they are responsible for delivering community-based applications ranging from social networking applications such as Facebook and Twitter to information dissemination applications covering sports, weather and traffic data. Indeed, the widespread proliferation of Smartphones with internet connectivity in vehicles provides a fertile environment for several ITS applications to serve the driver by enhancing traffic mobility and safety with minimal infrastructure investments [34].

The effect of using in-vehicle applications to provide safety information on driver behavior is still unknown. We present a Smartphone-based application that delivers a warning to the driver regarding slow traffic s/he might encounter 60 seconds ahead on their road. We release the application to nine drivers in the San Francisco Bay Area and collect data on their driving patterns for 8 weeks using the Smartphone sensors. We provide an analysis of the individual user traces for cases before and after the alerts are activated and deduce that a statistically significant change in behavior is warranted to the Smartphone application. We objectively define then describe such behavior in terms of speed and acceleration response to a Smartphone alert. We further analyze the sustainability of this change in behavior throughout the 8 weeks of the experiment. We deduce that the change in behavior undergoes a fluctuation in the first 3 weeks

of the experiment and becomes sustainable and stable after the third week of using the application.

This chapter starts by providing a literature review on related work that brings understanding to the change in behavior due to driver informational and warning systems. It then presents the architecture of our system, the experiment setup and the analysis of the results. It concludes with the future work related to this research effort, and with that, some intriguing possibilities of the safety and mobility benefits that could be understood given this further research.

3.2 Literature Review

The field of soft safety applications in the transportation domain covers a vast array of applications including real-time information to the driver on incidents, work zones, non-recurring traffic congestions and several safety applications such as forward collision warning, curve over-speed warning, and collision avoidance systems. The study of the effect of safety applications on driver behavior has been well documented. Examples include: forward collision warning [35], curve overspeed warning [36], and collision avoidance systems [37]. The part of ITS mobile applications that deals with soft safety, on the other hand, has not been well researched. In particular, the effect of real-time information regarding traffic a few miles downstream on driver behavior has not been studied. Few papers examine the effect of these types of soft safety applications on driver behavior. The work referenced in [38] presents a simulator test with 51 participants to evaluate driver behavior and user acceptance of Infrastructure –to-Vehicle (I2V) warning systems. The results show that I2V systems influence drivers in a positive manner with direct influence on road safety is observed through reduced driving speeds before dangerous situations occur.

The authors of [39] show that drivers drive with safer distances to vehicles in-front when they receive combined advisory/critical warnings compared to critical warnings alone where critical warning warn of an eminent collision compared to an advisory slow-down or change lane messages . Participants also report that the advisory warning system helped them notice potential dangers in the traffic environment. Similar to [38],[39] derive their conclusions from simulator experiments.

The SAFESPOT European project [40][41] presents a deployment of a soft safety application inside the vehicle. Applications within the SAFESPOT project include extension of visibility through foggy areas as well as static information on road properties such as speed limits and work zones. While this is close to the proposed Smartphone application presented in this chapter, there has not been any attempt by SAFESPOT to address the effect of those applications on driver behavior; their focus has been on the effectiveness of the application architecture and the application functionality.

From what others have done by the use of simulators [38], [39] or in real deployments [40], we expect a positive effect on driver behavior due to the use of in-vehicle soft safety applications. However, none of the previous contributors to this field have attempted a field experiment to assess the empirical benefits. The contributions of this chapter lie in conducting a field experiment to analyze the effect of such applications on driver behavior and the system architecture used to conduct the experiment. Our reliance on the Smartphone to issue soft safety

warnings to the driver and the use of Smartphone sensors to collect driving pattern data are also new and presage a medium and application which may become ubiquitous.

3.3 System Architecture

We present the Networked Traveler system which consists of a Smartphone application to test the hypothesis of whether Smartphone-issued alerts could affect driver behavior. The application focuses on a "Slow Traffic Ahead" soft safety warning alert issued to the drivers as they approach traffic conditions reported to be slower than their current speed as measured by the Smartphone GPS sensor. The alert is issued via the Smartphone speaker and screen; the driver is given the opportunity to react to the alert, either by pressing a "thumbs up" or "thumbs down" button on the screen indicating if the alert was useful or not, or by not pressing anything and letting the alert display fade out after a few seconds (Figure 3.1).

Figure 3.1. The NT Client warning display screen; the sentence "Slow traffic ahead" is voiced once while the above screen is displayed for 45 seconds.

The system architecture consists of three tiers: Cellular Phone, Networked Traveler Server, and Third-Party Services (Figure 3.2).

3.3.1 Cellular Phone

The Cellular Phone component encompasses all the software and hardware that would run on the Smartphone. The platform used for this project was the Windows Mobile 6.1 Smartphone operating system running on any phone that satisfied the following minimum conditions: capable of getting touch screen input, had a GPS sensor, and able to connect to the Internet via a data plan on any non-specific service provider using EDGE GPRS or better. The data collected in this study was over HTC Touch, HTC Touch Diamond, LG, Palm Treo and Samsung Smartphones operating over AT&T's cellular service. Software code was written and installed on the Smartphone to perform the required computations on the device. This code is referred to as the Networked Traveler (NT) Client. The NT Client code was the same version across all different hardware platforms used in this project.

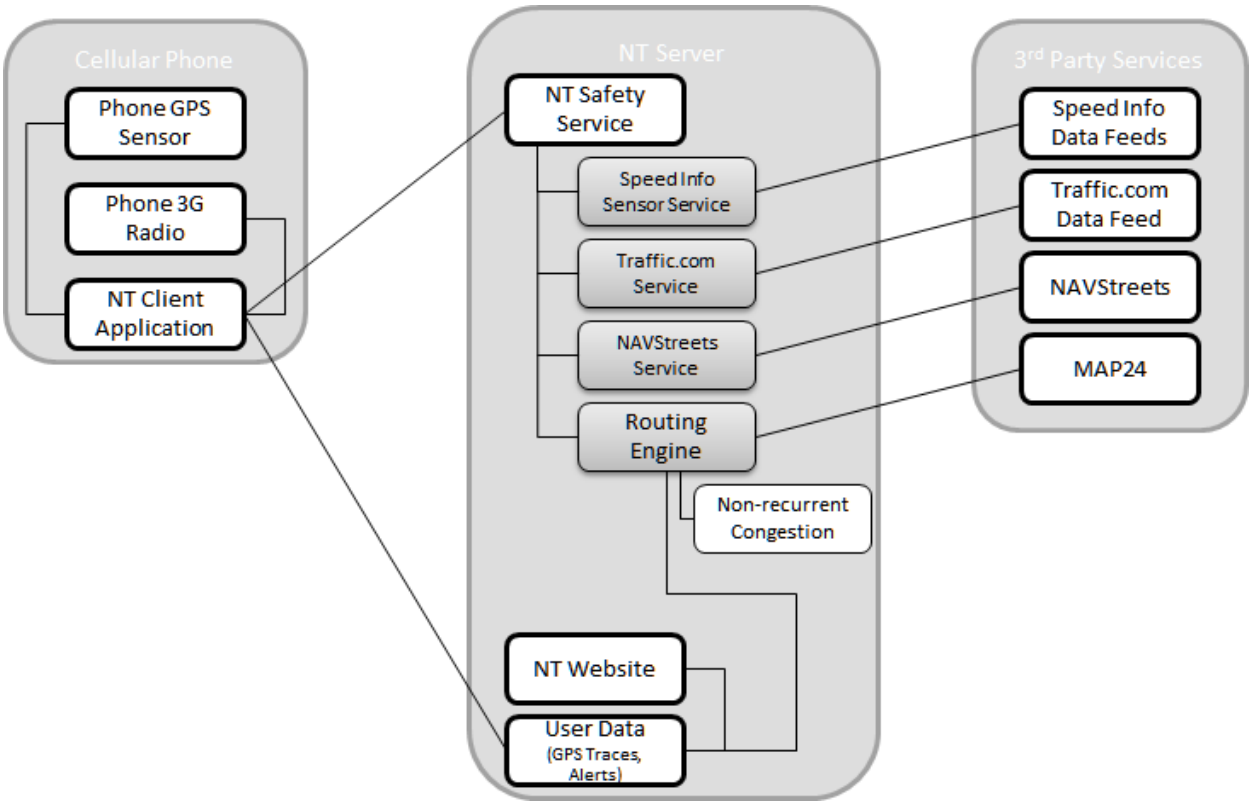

Figure 3.2 Networked Traveler System Architecture.

Three main processes govern the functionality of the NT Client. Below we present the details of those processes. The following definitions will be used in conjunction with Figure 3.3:

- GPS Speed of Smartphone: S_S
- Speed reported by road sensor at sensor location: S_T
- GPS Heading of Smartphone: H_S
- Trigger Point is a point upstream from the sensor location chosen depending on the geometry of the road to give ample time for the driver to react before reaching the speed sensor location: TP
- Heading of TP is the heading of the road at the TP: H_{TP}
- Distance between Smartphone and TP location: d_L
- Threshold Minimum Speed, the minimum speed difference between the Smartphone GPS and the Sensor speeds above which an alert might be warranted: ΔS_{min}
- Threshold Maximum Speed, the maximum speed at which traffic is considered to be slow at the TP: S_{max}
- Threshold distance is the margin of error when matching the Smartphone GPS location with the TP location: Δd_{min}
- Threshold heading is the margin of error when matching the Smartphone GPS heading with the TP heading: ΔH_{min}

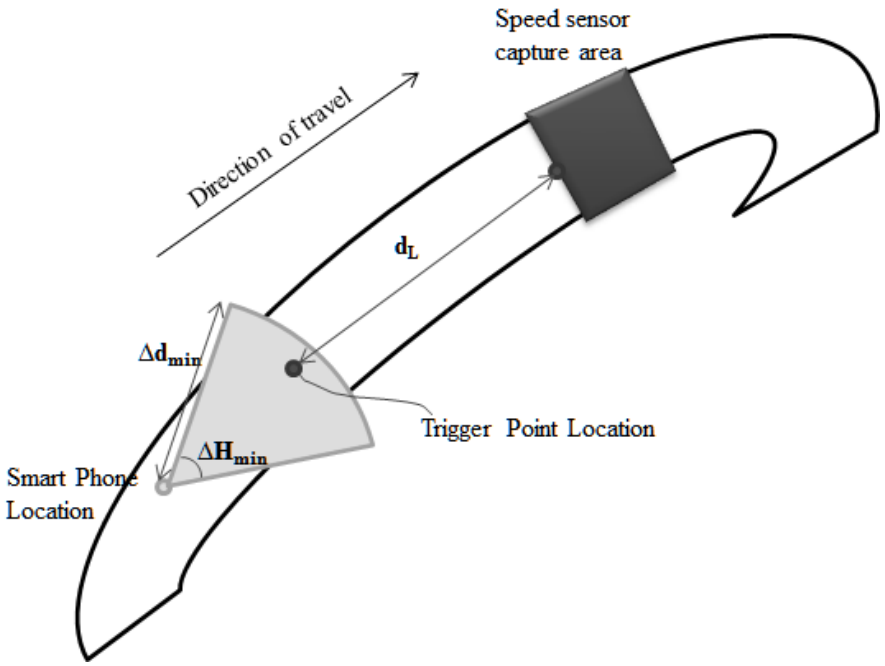

Figure 3.3 Setup used to detect if the Smartphone is close to slow traffic.

Three following processes run on the client (Figure 3.4):

(1) Get Current GPS
(2) Upload GPS Traces and Get Alert Data
(3) Check Alert Proximity and Display Alert

The "Get Current GPS" process is a threaded process that runs on the client to query the GPS sensor on the smart phone and record values of GPS Longitude, Latitude, Speed, Heading, Number of Satellites and Altitude. This process runs at a determined tunable time setting, T_G, and records the GPS values locally on the Smartphone.

The "Upload GPS Traces and Get Alert Data" process runs at another determined tunable time setting, T_A. Each time this process runs, it checks if data connectivity to the server exists and uploads all locally cached GPS values to the NT Server's web service. The web service responds by returning all potential alert situations within the vicinity of the last GPS coordinate.

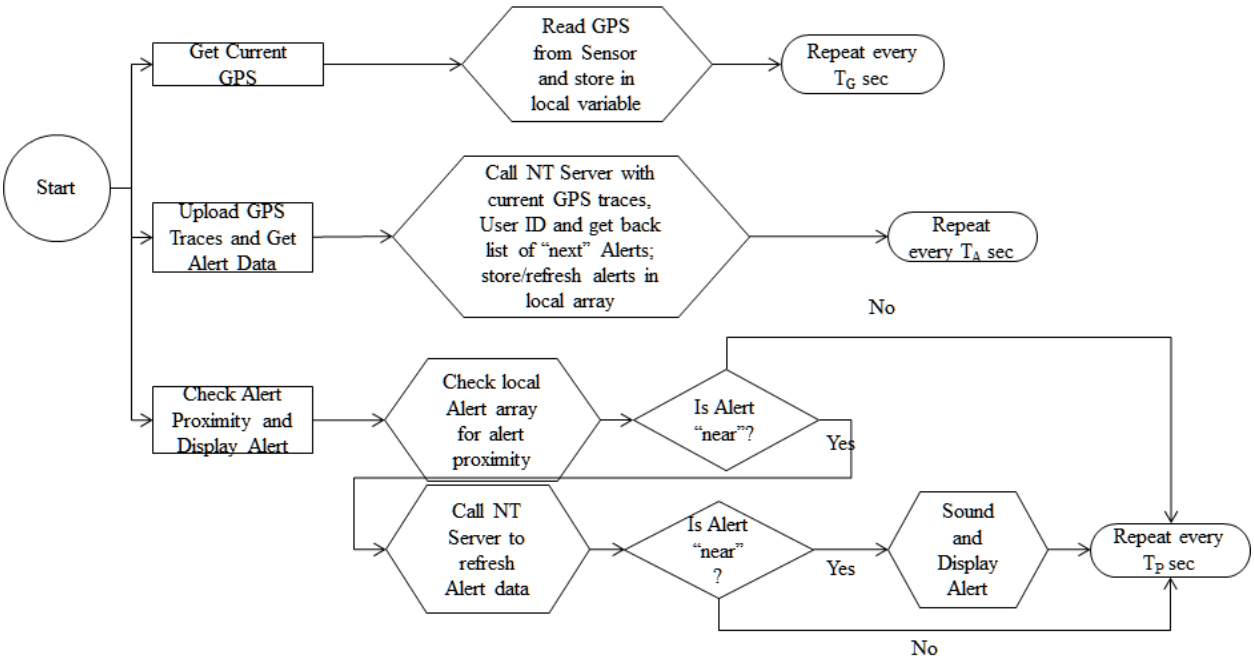

Figure 3.4 NT Client processes.

The "Check Alert Proximity and Display Alert" process is also a threaded process that runs periodically at a determined tunable time setting, T_P. Each time it runs, it gets the most recent GPS values output from the "Get Current GPS" process and compares them with the list of alerts cached by the "Upload GPS Trace and Get Alert Data" process. If the Smartphone is found to be within "proximity" of an alert event, an alert is displayed on the Smartphone device and an alert audio notification is played on the audio output terminal of the device.

Proximity to an alert event is satisfied when:

$$S_S - S_T > \Delta S_{min}$$

$$H_S - H_{TP} < \Delta H_{min}$$

$$d_L < \Delta d_{min}$$

$$S_S < S_{max}$$

The threshold values: ΔS_{min}, ΔH_{min}, Δd_{min}, and S_{max} and the processes cycle lengths T_G, T_A, and T_P were determined by calibrating the system through several drives conducted with the Smartphones near TP's of known locations and traffic conditions. As a result of the calibration, the following were adopted:

$T_G = 1$ sec

$T_A = 20$ sec

T_P = 1 sec

ΔS_{min} = 15 mph (24.14 km/hr)

S_{max} = 50 mph (80.47 km/hr)

ΔH_{min} = 45 degrees

Δd_{min} = 1000 ft (304.8 m)

The TP location for each sensor was set to be 1 mi (1.61 km) upstream from the sensor location; hence, d_L = 1 mi (1.61 km).

3.3.2 NT Server

The NT Server is hosted in the PATH Project offices at UC Berkeley's Richmond Field Station, the off-site facility used for this project. The NT Server performs the following functions:

(1) Hosts the NetworkedTraveler.org website (http://www.networkedtraveler.org)
(2) Aggregates road sensor data from different sources
(3) Captures and archives user GPS traces and alert feedback when submitted by the NT Client

The Networked Traveler website allows the experiment subjects to register with the experiment, download the latest NT Client, and view and comment on their received alerts.

The server also acts as the aggregator of all the data and does several of the computation intensive calculations that would otherwise be performed on the client. Data sources that feed to the server send road sensor speed measurement information at varying time intervals and confidences. The server captures the data from the different sources and fuses close-by sensor readings from different sources into a single speed reading for that location. Each source sends a confidence value of its speed measurement which the server users to prioritize speed readings for nearby locations. The server provides a service-oriented architecture web-services interface as described in Chapter 2: for the NT Client to authenticate and retrieve this data whenever needed. Each NT Client device would submit an HTTP post to the NT Server with its current location and get an HTTP response from the server with a list of the nearby trigger points and their current speed measurements. The NT Client would then cache this list and run its proximity checks.

Each NT Client, when submitting its request to the NT Server, can upload a GPS trace as opposed to a single location. If a GPS trace is sent to the NT Server, the NT Server would archive that for that particular device. The NT Server also offers a web-service by which the NT Client can upload a list of its alerts to be archived. This data archive is used in the analysis presented in this chapter.

3.3.3 Third Party Services

The Third Party Services utilized in this experiment provide the NT Server with web-services feeds into their interfaces. SpeedInfo provides speed data on several of the roads in the San Francisco Bay Area through the use of side-fire radar installed at various stations in the freeway network. Each radar sensor reports the average speed of cars at its point location. For this

experiment, we used over 500 SpeedInfo sensors. NAVTEQ Traffic and NAVStreet jointly provide the NT Server with a web service that reports speeds on several links in Northern California. Each NAVStreet link reports an average speed on the link with a confidence value based on different sources of information used to estimate the average speed. The NT Server filters out the speeds with confidence values less than 90%. Over 2000 NAVStreet links are used in this experiment. Map24 provides the NT Server with a web-service that allows it to map-match the current GPS location of the Smartphone with nearby sensors from SpeedInfo and NAVTEQ.

3.4 Experiment Design

An experiment using the application and system architecture presented in this chapter is design to answer the following two questions:

(1) Is there an effect on driver behavior due to the situational alerts issued by the Smartphone?
(2) If there is a change in driver behavior, is this change persistent over time or does is experience a novelty effect and fade away with time?

Nine users are recruited as part of the experiment. Users are asked to keep to their regular driving routes while placing the NT Client Smartphone in their cars. Data is collected for a period of eight weeks. During the first two weeks of data collection, the alerting mechanism on the Smartphone is disabled to construct a baseline for that driver. The alerting mechanism is turned on for the remainder of the experiment. An analysis of the speed profiles for each user around the trigger point locations is performed for cases when an alert is issued versus when no alerts were issued. In order for the results of this comparison to be attributed to the "Slow Traffic Ahead" alert, two assumptions are made: (a) the Smartphone speed measurements reflect the actual speed of the car, (b) similar traffic conditions exist in the vicinity of the driver for cases when alerts were issued, compared to cases when no alerts are issued. The following two sub-sections address validates that those assumptions are true for the cases we analyzed.

3.4.1 Validating Smartphone GPS Speed Measurements

A validation of the speed being reported from the GPS sensor in the Smartphone is done to ensure the accuracy of the speed values collected from the NT Client. For this validation we develop a speed logger application on the Smartphone to record the GPS speed on the phone. We also run a laptop software application that measures the actual speed of the car by connecting to the On-Board Diagnostic (OBD-II) connector of the vehicle. The two sources of speed measurements are then compared for various speed ranges: 50 mph (80.47 km/hr), 60 mph (96.56 km/hr), and 65 mph (104.61 km/hr) lasting 2 minutes each. Table 2 reports the results.

The results show that for the two types of Smartphones used in this experiment, speed values are close to the actual speed of the vehicle.

3.4.2 Validating Similar Traffic Conditions For GPS Traces Used in Analysis

In order to show that the Smartphone application produced an effect in driver behavior we compare speed profiles of drivers before and after alerts are issued at the same location. We do this at times when the similar traffic conditions existed for the two cases.

Table 2 Results of Validating Smartphone Speed Measurements.

Car Speed Odometer Reading [actual speed measured from OBD-II]	LG 540X	HTC Touch Cruise
50 mph (80.47 km/hr) [48.2 mph (77.57 km/hr)]	48.75 +/- 0.10 mph (78.46 +/- 0.16 km/hr)	48.67 +/- 0.08 mph (78.33 +/- 0.13 km/hr)
60 mph (96.56 km/hr) [56.5 mph (90.93 km/hr)]	56.8 +/- 1.85 mph (91.41 +/- 2.98 km/hr)	57.10 +/- 0.07 mph (91.89 +/- 0.11 km/hr)
65 mph (104.61 km/hr) [65.25 mph (105.01 km/hr)]	63.16 +/- 0.55 mph (101.65 +/- 0.89 km/hr)	64.84 +/- 0.20 mph (104.35 +/- 0.32 km/hr)

Figure 3.5 shows a plot of the times of day that driver traces are used for the analysis. The red dots reflect the times of GPS traces with alerts. The blue dots reflect the times of GPS traces with no alerts. Additionally, the size of the dot scales with the number of traces that fall within that time slot. The plot shows that the times of day and the days of week during which alert versus no alert traces are similar. This would lead us to assume that traffic conditions were similar during the times the traces were selected. The analysis was limited to week-days and the time spots between 9am-2pm and 2pm-7pm.

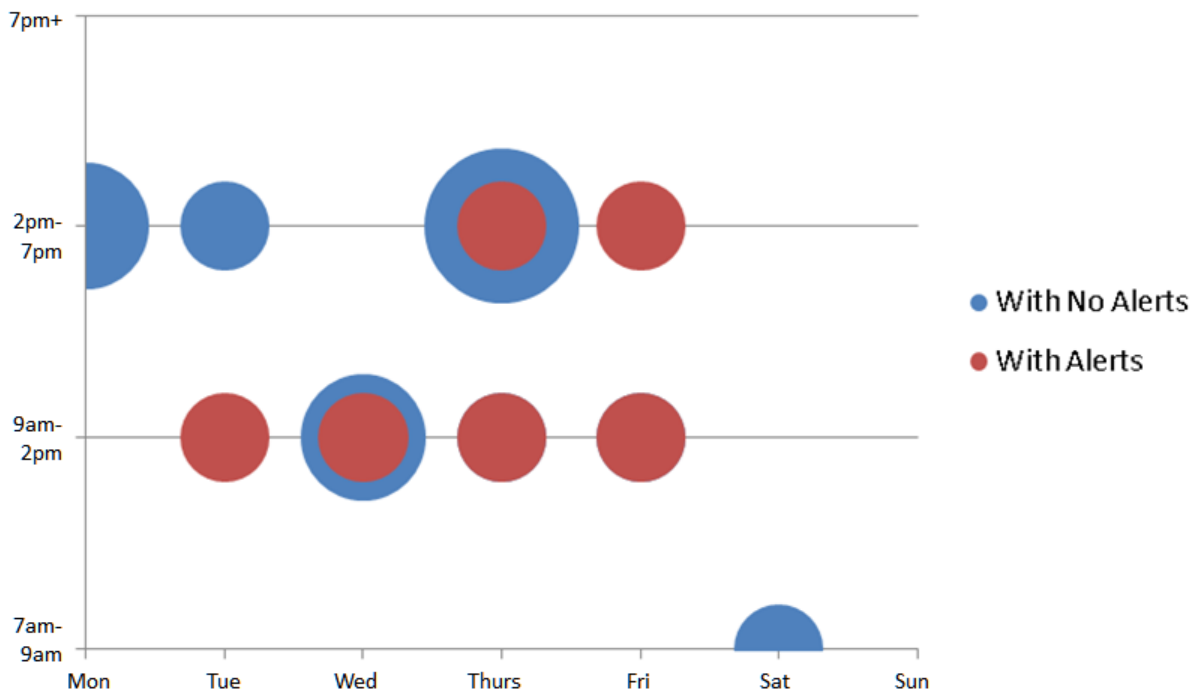

Figure 3.5 Distribution of times GPS traces are selected for the analysis. Data used for analysis is retrieved from trips that occurred during similar traffic conditions occurring on weekdays Tue – Fri. The time divisions shown comply with the HOV activation times in that location.

3.5 Change in Driver Behavior Due to Slow Traffic Alerts

In order to address the question of whether there is a change in driver behavior due to the alert, we focus the analysis on a single location and collect data on all users and their GPS traces that passed through that location. The location chosen for this analysis is on the I-580 South Bound/ I-80 West Bound highway near Berkeley, CA.

The location speed sensor is located at the following GPS coordinates 37.881018,-122.30813. The sensor type is a solar-powered radar provided by SpeedInfo. Two trigger points were established 1 mi (1.61 km) up-stream from the sensor, one on I-80 and the other on I-580.

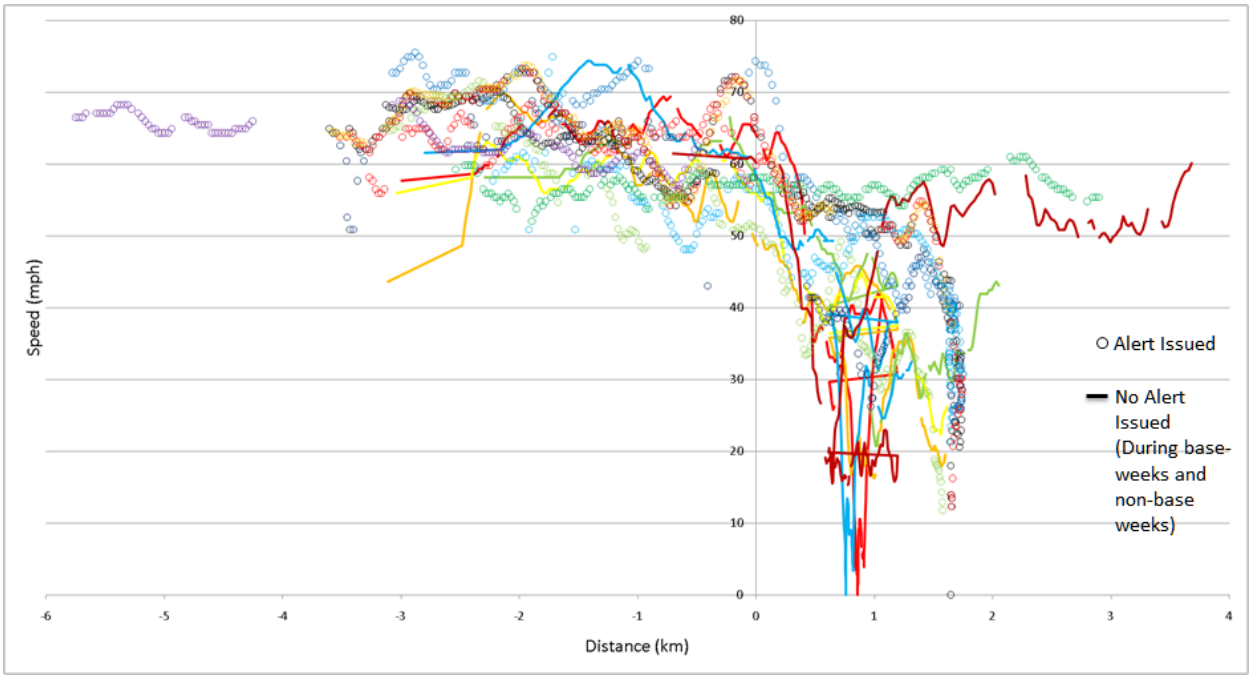

Figure 3.6 Raw GPS speed plots from GPS traces that passed through the location and satisfied the time constraints. Different colors represent different GPS traces.

We collect all GPS traces for all users around that location. We then filter the traces to include only those that occur during similar travel conditions. This resulted in 17 total traces from the 9 users in the study which we plot in Figure 3.6. That figure shows speed values measured every 1 sec from the Smartphone GPS sensor as a function of distance driven. Traces that included alerts are drawn as circles, and traces that did not include alerts are drawn as solid lines. The zero mark on the x-axis is the sensor location.

We compute a space-mean speed every 0.1 km (0.06 mi) for all traces that belong to the same group: traces with alerts vs. traces without alerts, and we produce the plot shown in Figure 3.7 for space-mean speed vs. distance for each group. Figure 3.7 shows a difference between the two groups of traces. Traces with no alerts experience a "dip" in the speed profile. The driver seems to rapidly decrease the speed of the car when they notice the slow traffic resulting in a drop of 30 mph (48.28 km/hr) in the speed of the car. As the driver approaches the slow traffic event – that is usually moving at a higher speed than the reduced speed of the driver, the driver increases the

speed by 10-15 mph (16.09 – 24.14 km/hr) to flow with the traffic. Traces with alerts start a gradual decrease about 2 km (1.24 mi) upstream from the slow traffic event to arrive at a 15 mph (24.14 km/hr) slower driving traffic. We characterize traces with no alerts as traces with a "dip" vs. traces with alerts as traces with no dips.

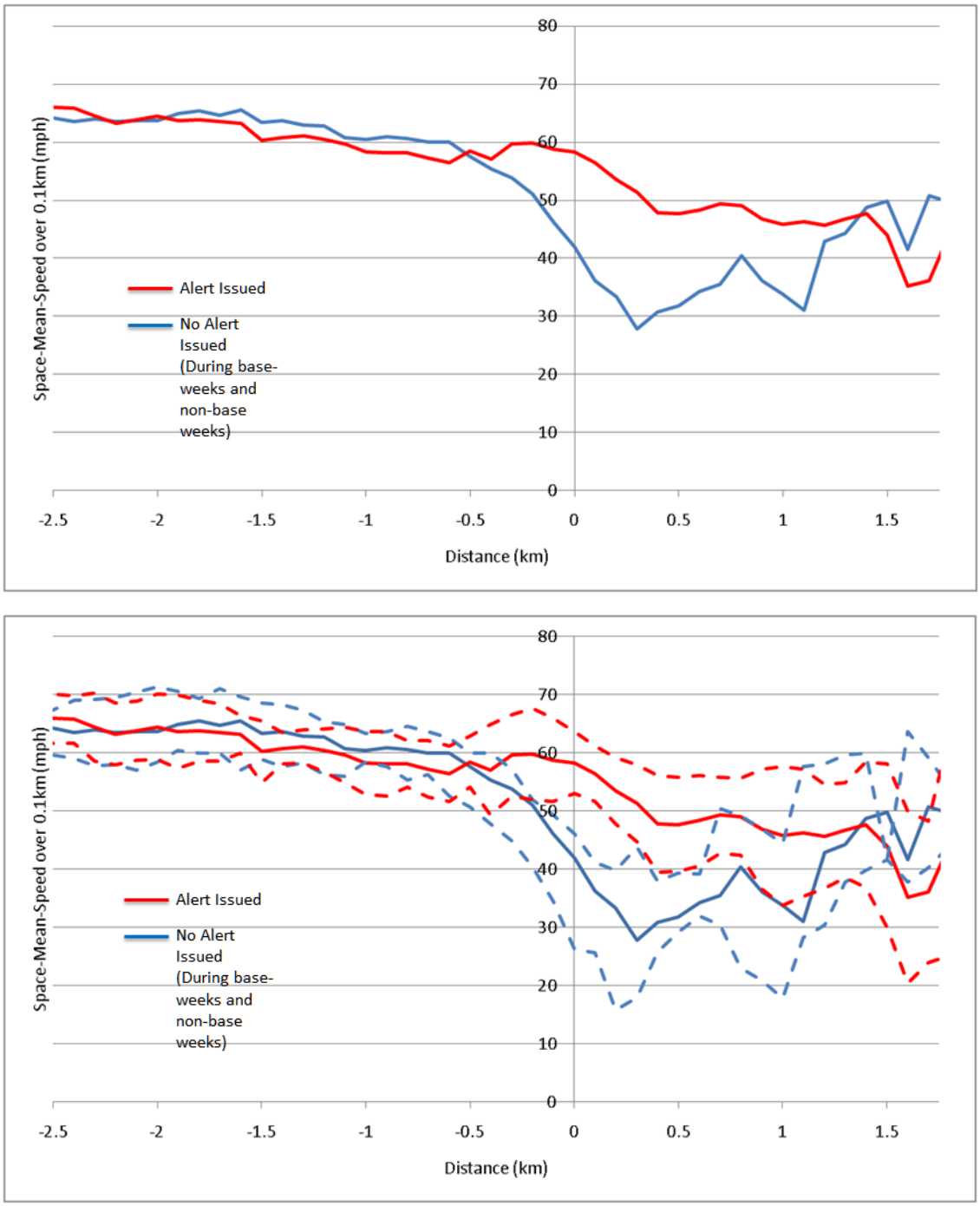

Figure 3.7 Space Mean Speed plots of GPS traces collected around location. Upper plot shows average values of speed in the cases when alerts were issued vs. cases when no alerts were issued. Lower plot shows the average speed with the standard deviation interval as dashed lines.

Looking at the individual traces in Figure 3.6 we notice the difference between traces with dips and those with no dips. Producing a similar plot of individual speed profiles vs. time shows a clearer distinction between the two types of traces (Figure 3.8).

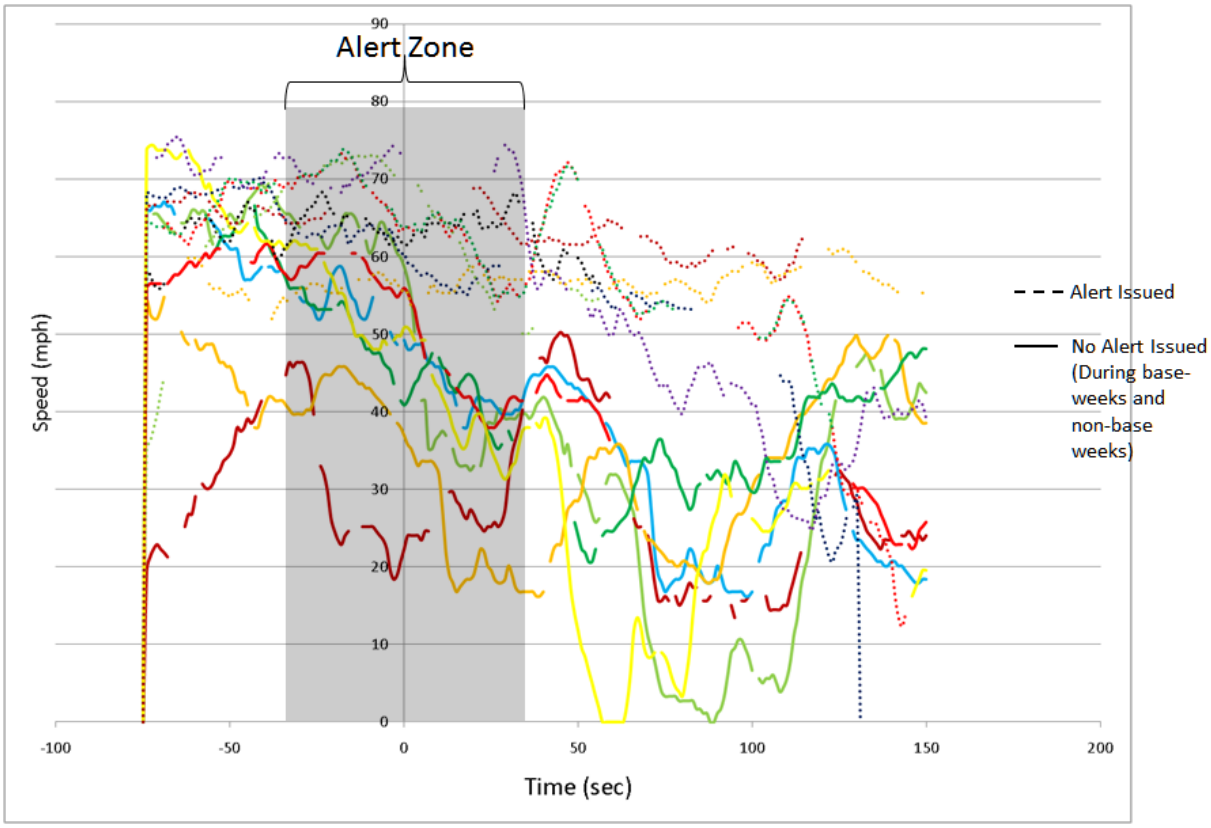

Figure 3.8 Speed profile plots vs. time for the traces being analyzed.

The standard deviation of the speed profile has been used in the field of transportation safety to quantify the smoothness of the speed profile especially as it relates to the probability of getting into an accident [42][43]. We use the standard deviation as a measure of smoothness to compare between the two trace types. We calculate the standard deviation of each trace and then compare the means of the standard deviation for each class of traces. We use the standard error of the mean to determine the statistical significance of the difference between the two trace groups.

Table 3 Quantitative Analysis Results of the Effect of Alerts on Driver Speed Profiles.

	Standard Deviation Analysis	
Class Type	**Mean of Standard Deviation taken across all traces in class type**	**90% confidence interval**
Traces with no alerts	16.45 mph (26.47 km/hr)	3.07 mph (4.94 km/hr)
Traces with alerts	10.13 mph (16.3 km/hr)	2.66 mph (4.28 km/hr)

Table 3 shows a 38% smoother speed profiles for traces with alerts when compared to traces with no alerts.

3.6 Is the change in driver behavior Persistent?

The second part of the analysis focuses on understanding the lasting effect of the change in behavior identified in the previous section. In this part we look across all the 9 users involved in the experiment and analyze their GPS traces for the duration of the experiment. We look at the percentage of "dips" each user has for each week of the experiment and compare that with previous weeks. A reduction in the percentage of dips reflects smoother speed profiles.

A trace is classified as a trace with a dip if 1Hz speed measurements, V(.), as taken by the GPS sensor in the Smartphone, satisfy the following criteria:

$\exists \{V(t_1), V(t_2), V(t_3)\}$ such that

$t_3 > t_2 > t_1$

and $V(t_1) - V(t_2) > 30$ mph (48.28 km/hr)

and $V(t_3) - V(t_2) >= 10$ mph (16.09 km/hr)

and $t_3 - t_1 <= 225$ sec

Figure 3.9 shows for each user the percentage of dips decreases as the experiment progresses in time. The x-axis shows the number of weeks since the alert mechanism was turned on. Some users have missing data between weeks 7 and 10 since they stopped using the application while driving in those weeks. Therefore, we limit our analysis to the first 6 weeks after the alert was turned on. We also observe a sharp drop in the percentage of dips for each user in the first week the alert was turned on. We later see that the percentage of dips does increase but not to previous values.

Figure 3.10 shows that 50% of users sustained 30-45% fewer dips in their profile after using the NT Client for 6 weeks. About 25% of users sustained more than 45% reduction in dips.

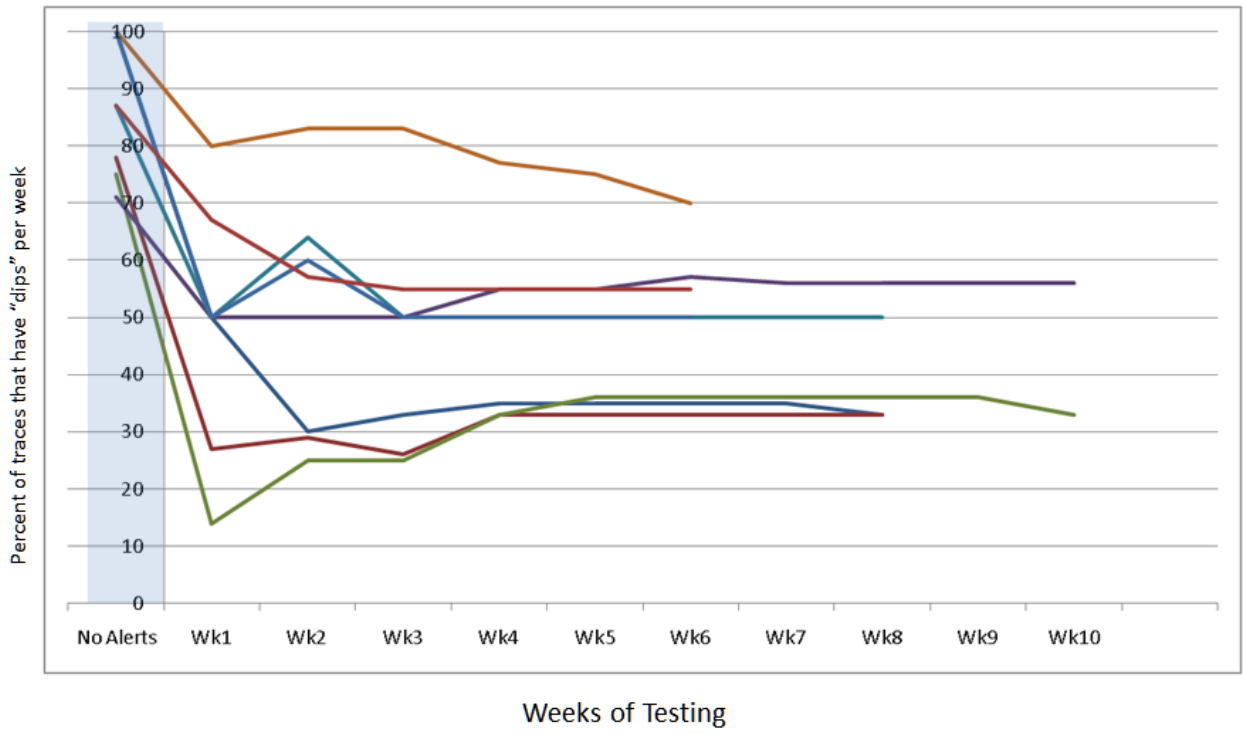

Figure 3.9 Percentage of dips in speed profiles vs. weeks in experiment. Different colors represent different users.

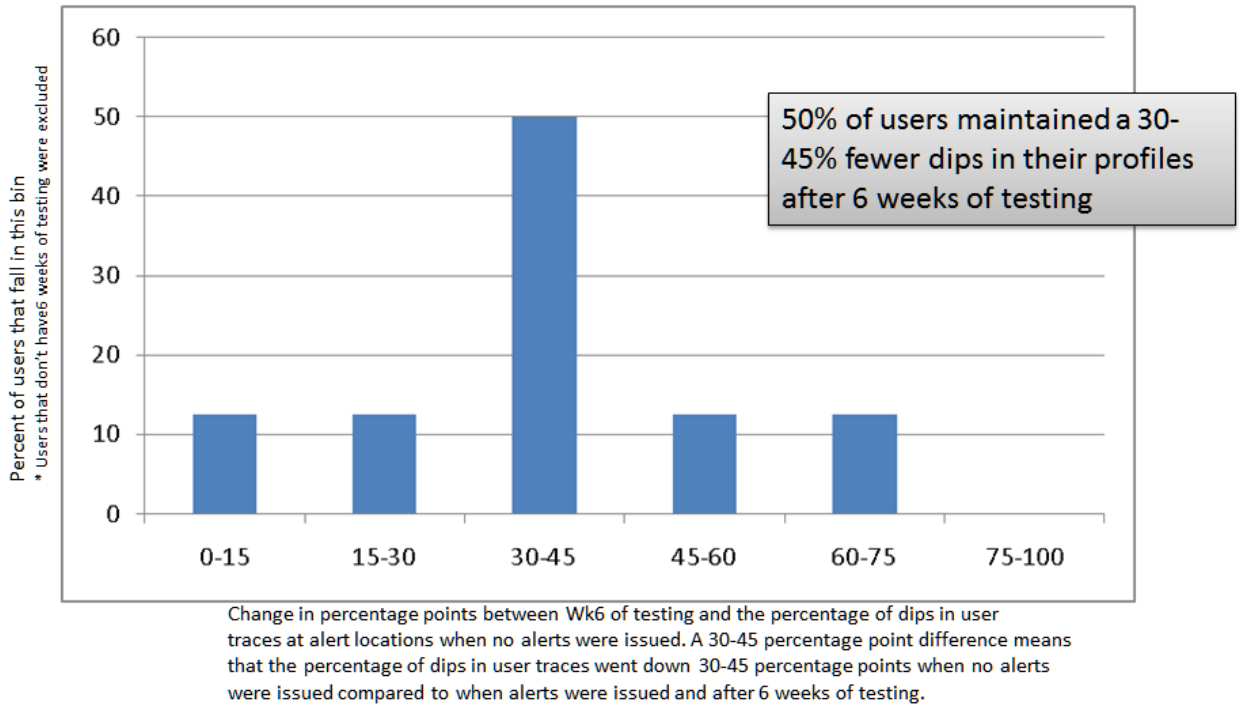

Figure 3.10 Histogram of percent of users experiencing a lasting effect of the change in behavior.

3.7 Conclusion

We presented a system of Smartphones and existing road-side sensors to issue soft safety alerts for divers on slow traffic conditions they might encounter 60 sec ahead on their road. We analyzed the behavior of the driver in cases when no alerts were issued compared to cases when alerts were issued. We found that drivers drive with a smoother speed profile when they are given warnings of slow traffic ahead. Furthermore, we show that the smoother speed profile driving pattern persists while using the application beyond 6 weeks.

Our analysis in this chapter focused on one location and with data collected at the frequency of 1 data point per second. We believe that a higher resolution of data might help explain in more detail the behavior of the driver surrounding the alert time and location. This will also help in identifying the causes of the change in behavior. Our future direction is to instrument vehicles with high resolution location and motion sensors to detect the changes in driver behavior around the alert conditions. We also plan on recruiting a larger set of participants to improve on the confidence of the change in behavior detected.

The hypotheses for future work beyond the addressing of the aforementioned areas of data quantity, quality and confidence are two-fold, and if true, quite interesting: we conjecture that if driver performance results in slower approach to end of queues on freeways, this may result in safer driving; we conjecture also, that given wide adoption, the effect of such alert may dampen or slow the propagation of shock waves, and with that offer benefit in not only safety but in freeway flow and capacity. We hasten to add that significant fundamental research, on-road verification and large-scale simulation or observational work would need to be accomplished, but we submit that the research we describe and the advent and proliferation of GPS-enabled geo-location services – and Smartphones – may portend such benefits and warrant further investigation.

3.8 Acknowledgement

The work presented in this chapter was funded by the California DOT (Caltrans) contract 65A0343. We wish to thank Caltrans for allowing us to use their right-of-way to conduct our experiments, the San Francisco Metropolitan Transportation Commission (MTC) for providing us with data sources for traffic conditions. We also wish to thank NAVTEQ and SpeedInfo for the San Francisco Bay Area data feeds they supplied during this project. Last but not least, we wish to thank the staff members and development engineers of the University of California PATH Program for their hard work in implementing components of the Networked Travel application used in this research.

CHAPTER 4: LEARNING USER ALERT SETTING PREFERENCES

The use of mobile devices to deliver traveler information such as soft safety warnings and navigation guidance is on the rise. However, an optimal experience – and even more widespread use – may be impeded by the lack of personalized, user-tailored applications. The issue is that users react differently to traffic information, and information perceived useful by one user may be considered as nuisance by another. We provide evidence of this from a pilot field test performed on a soft safety application that alerts the user of approaching slow traffic 1.6 km ahead. We furthermore propose a machine learning algorithm based on a support vector machine that segregates alerts into "favorable" vs. "nuisance" based on user feedback and user GPS trace analysis. An implementation of the support vector machine can detect 75% of the type of alerts before they are issued to the user.

4.1 Introduction

Smartphones connected to the Internet through broadband cellular networks provide an intriguing option for delivering timely soft safety warning information to drivers, where soft safety warnings may be operationally bounded by "more than traffic conditions" and "less than collision warning" or roughly safety-relevant traffic conditions approximately 10 to 60 seconds ahead. Applications that assist the traveler in multi-modal transportation choices, situation awareness on the road, and navigation guidance have all been ported to mobile devices such as Smartphones and personal navigation devices. Yet, user adoption to such applications has been low. In this chapter, we address one of the factors which may impede even wider adoption of mobile device-based applications: the lack of personalized user experience. Users react differently to traffic information, and information perceived to be useful by one user may be considered as nuisance by another. Data from the Automotive Collision Avoidance System (ACAS) [7] and the preliminary data collected by our project presented provide evidence that a one-size-fits-all approach does not work in presenting soft safety alerts to drivers. By using a statistical classifier, the Support Vector Machine (SVM), we show that soft safety alert presented to the driver via a Smartphone can be categorized as "favorable" or as a "nuisance" by learning from the driver's past experience with the application. In this chapter, we present a background of user perception to in-vehicle alerts, followed by a brief description of the SVM. We then present our Smartphone-based nine-user experiment which alerted drivers of slow traffic 1.6 km ahead on their route. This application was used to collect data to train the SVM for each user. Finally, we present the results of running the SVM classifier on the post-processed data collected from nine users for a duration of 6-8 weeks.

4.2 Background

The Automotive Collision Avoidance System (ACAS) Field Operational Test (FOT) performed in 2005 by the University of Michigan Transportation Research Institute (UMTRI) and General Motors (GM) [7] revealed a large standard deviation of drivers' perception of a false Forward

Collision Warning (FCW) across the 96 participants (Figure 4.1). The ACAS researchers also identified a significant difference across age and gender. Furthermore, when the users were asked to locate the annoyance level associated with the false FCW alerts, answers varied widely, indicating that different users perceive annoyance differently (Figure 4.2).

During the ACAS FOT, users were provided with a knob on the steering wheel to adjust the sensitivity of the FCW algorithm.

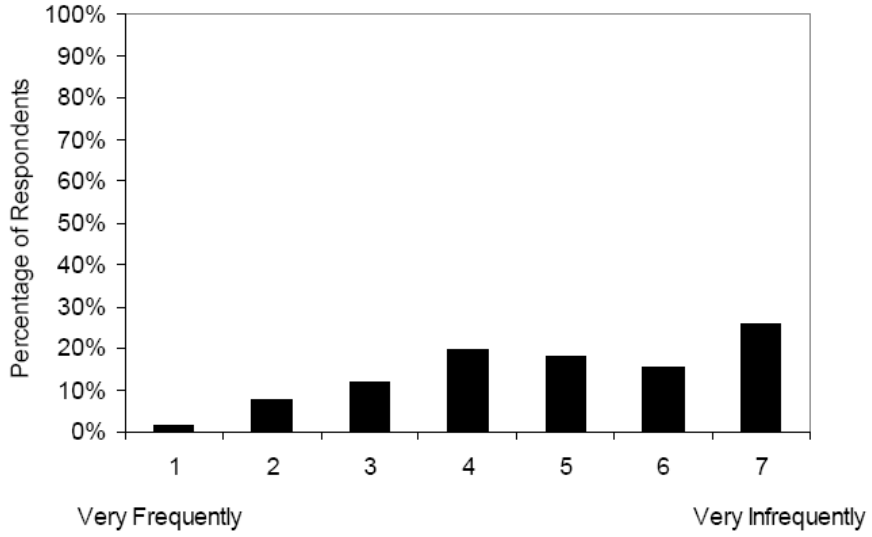

Figure 4.1. Survey responses to the question: "How often, if ever, did FCW give you a warning that was false?" [7] show a large spread of how 96 users perceived a "false warning"

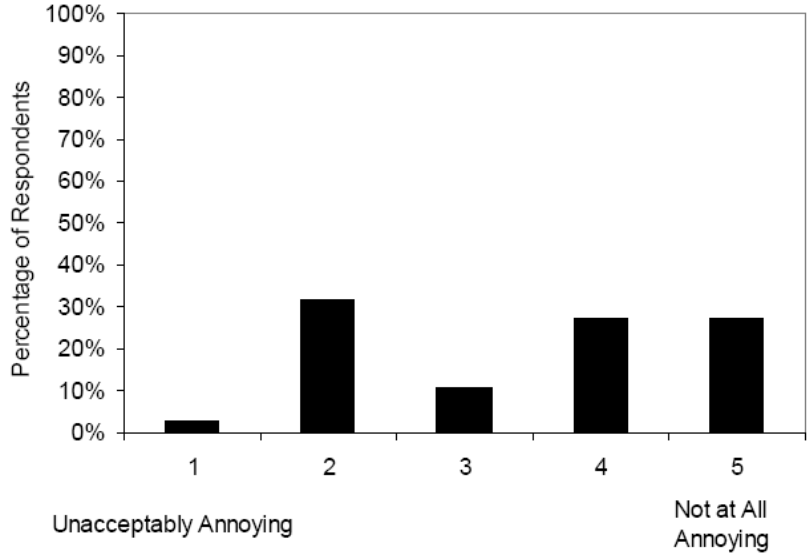

Figure 4.2. Survey responses to the question: "How annoying was the FCW alert?"[7] show that users are spread in their decision on how to classify the level of annoyance of an alert

Figure 4.3 shows that a majority of the users changed the setting every week, indicating that identifying the proper setting for the algorithm varies, not just across users, but also across setting conditions in which the FCW is issued. Such setting conditions could be time of day, day of week, location, etc. The ACAS researchers report setting conditions such as road type, lighting, and traffic density as conditions that might affect driver behavior in general.

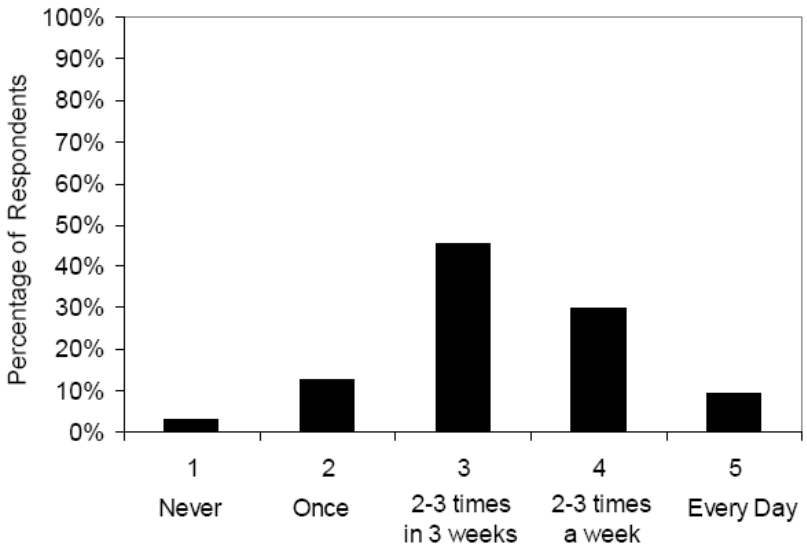

Figure 4.3. Survey responses to the question: "Select the statement which best describes how often you changed the FCW alert timing adjustment (select one)?" [7]

The Networked Traveler application presented in Chapter 3:, and used to collect data for the analysis in this chapter, issues in-vehicle warnings to the driver through a smartphone application that displays and voices the alerts. In conducting the Networked Traveler system test in 2009, 30 unique user ID's for a duration of 29 days used the application. During that time, 392 alerts were issued to drivers. Feedback on the alerts were captured by two methods: (a) via the Smartphone application through a touchscreen that enabled the driver to tap on a "thumbs up" for a "favorable" alert and a "thumbs down" for a nuisance alert, (b) from web-based surveys to allow users to individually rate every alert they received during the test.

Figure 4.4. The user interface of the NT Smartphone application allowed us to capture, in real-time, the driver's feedback to the alert by tapping on the thumbs-up or thumbs-down buttons

The same algorithm parameters for issuing the alert were used across all the users. The feedback that was captured from the phone showed a wide spread on how users perceived alerts (Figure 4.5).

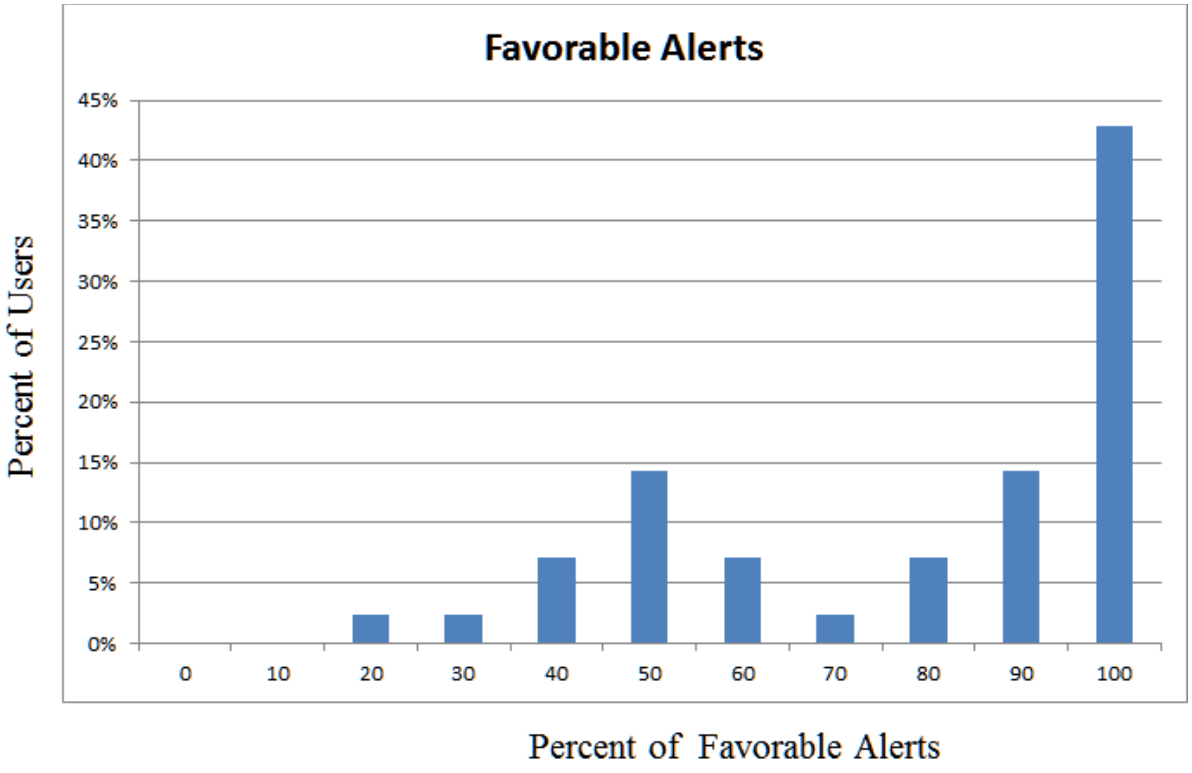

Figure 4.5. Histogram showing how users perceived alerts

The data from both the ACAS project and our system test data from the Networked Traveler experiment show that when an in-vehicle warning system is deployed, users perceive information from the system in different ways. A “one-size-fits-all” approach on how and when

information is displayed to the user might render the information a nuisance to several drivers, although physical properties warranting an alert may be correct, that is, even when the algorithm as conceived functioned properly. This motivated us to build a system that displays information in a personalized manner to the driver. The personalization we seek is not graphical and layout-related, but is that personalization to what constitutes a 'car ahead' hazard, which depends on frequency and timing of information. One driver might want to hear alerts at a certain distance from the event, and that might be different by time of day, day of week, road condition and location. Those settings might not be the same for another driver. For this purpose, we apply statistical methods to enable us to learn driver preferences as they relate to setting conditions. We collect data for a particular driver over a period of time – called the "training period" – where the application trains itself on that particular driver's habits. We then model this record of driver habits into a SVM classifier that recognizes the "favorable" vs. nuisance alert-setting conditions. Once trained, the SVM classifier can be used to filter and in a perfect world, eliminate, the nuisance alerts heard by the driver. Before we describe our SVM classifier, we present a brief introduction to SVM classifiers.

4.3 Support Vector Machine Classifiers

A Support Vector Machine (SVM) is a statistical classifier that is trained to classify its input data as belonging to one of two groups. The training is done by presenting the SVM with a set of training examples, each marked as belonging to one of those two groups. The SVM then learns the hyper-plane that optimally separates those two groups. The SVM-learned model of the hyper-plane can then be used to classify new unmarked data [44].

Assuming a linear separation between the two groups of data (Figure 4.6) and given:

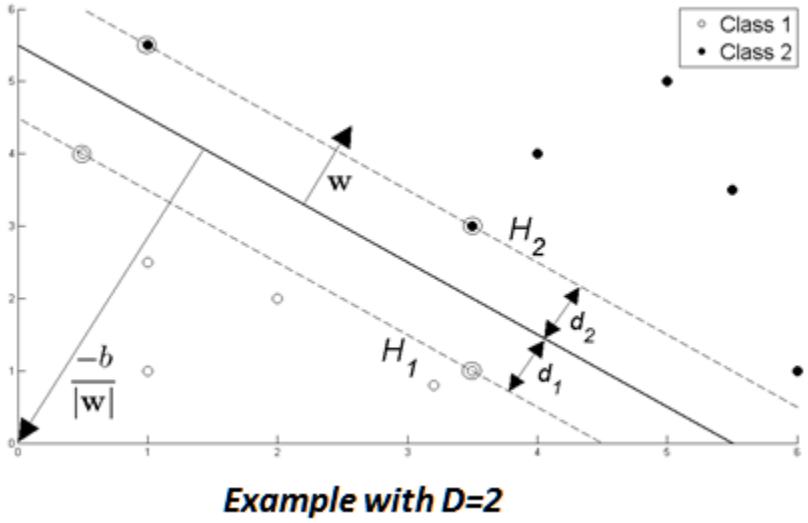

Figure 4.6. Graph depicting a hypothetical 2-dimensional hyper-plane separation

$$\{x_i, y_i\} \ \ where \ \ i = 1 \ldots L, \qquad y_i \in \{-1,1\}, \qquad x \in \Re^D$$

Where L is the total number of data points collected and -1, 1 are the two groups into which the data can be classified. We then construct a hyper-plane such that:

$$w.x + b = 0$$

w is normal to the hyperplane

$\frac{b}{\|w\|}$ *is the perpendicular distance from the hyperplane to the origin*

The SVM solves the problem of finding *w* and *b* such that the hyper-plane is equidistant from the two classes. This can be translated into an optimization problem that solves the following objective:

$$maximize \ \frac{1}{\|w\|} \ or \ minimize \ \|w\|$$

Once solved, the decision of whether the data belongs to group -1 or 1 becomes a function of:

$$D(x') = \ sign(w.x' + b)$$

If D(x') is positive then that data point belongs to class 1; otherwise, the data belongs to class -1. The above formulation can be generalized to the nonlinear separation of the two classes and also to the case of multiple dimensions by optimizing the following objective function:

$$\min \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L}\xi_i \ \ s.t. \ y_i(x_i.w + b) - 1 + \xi_i \geq 0 \ \ \forall i$$

where C is a parameter that can be set to calibrate the SVM for the nonlinear separation of the classes.

4.4 The Networked Traveler Safety Application SVM

The system described in Chapter 3: was given to nine drivers to use for a period ranging between 6 to 8 weeks. During this timeframe the users' Smartphone GPS traces were captured at the rate of 1Hz, and alerts were issued based on the proximity and alerting algorithms defined in Chapter 3:. For each alert, the following parameters were recorded:

- GPS longitude and latitude of the location the alert was issued
- GPS longitude and latitude of the location of the event that triggered the alert
- Speed at which the car was driving when the alert was issued
- Speed reported at the location of the event by the road-side sensor at that location
- Time spent on the road since the application was turned on
- Day of week the alert was issued
- Time of day the alert was issued

- Feedback to the alert as captured by the Smartphone (-1= thumbs down; 0=no tapping; +1=thumbs up).

The above alert data was captured for every user. The feedback information was translated to -1/+1 by converting all 0 values to +1 reflecting that if the user did not tap the alert thumbs down button then the user was not annoyed by the alert. One out of the 9 users had all the alerts classified as the same class of +1, so that user was dropped from the analysis. The remainder 8 users had a total 334 alerts. For each user, we then divided the alerts into two groups: training and testing groups. The *testing alerts* are alerts that were collected in the last two weeks of testing for each user. The training alerts were all other alerts. Table 4 shows the number of alerts in each group for each user.

Table 4. User alert counts as collected from the NT Smartphone application.

User	Training Alerts	Testing Alerts
1	32	13
2	44	15
3	42	8
4	23	31
5	14	18
6	26	19
7	15	13
8	17	4
9	-	-

A SVM was then set up to run for each user's training alert set with the following parameters:

- Speed at which the car was driving when the alert was issued
- Speed reported at the location of the event by the road-side sensor at that location
- Time spent on the road since the application was turned on
- Day of week the alert was issued
- Time of day the alert was issued

Bootstrapping [45] was performed to increase the number of alerts for each user by a multiple of 10. Cross-validation [46] was performed for each user to determine the optimum value of C for the SVM. The SVM was then trained for each user to achieve a 100% accuracy in detecting the Training Alert classification. This produced an SVM model for each user that can now be used to classify new alert as "favorable" or "nuisance".

4.5 Results

The testing set of alerts for each user was then run using the SVM model that was developed for each user. This yielded varying accuracies of detection for each user. Results are presented in Table 5.

Based on the 8 users we have profiled using the SVM, we can predict false alerts with 75% +/- 18% to 90% confidence (alpha = 0.1). This variation is due to the amount of data we have on each user.

Figure 4.7 shows the trend of improved accuracy in the SVM prediction engine as more data is available on the driver.

Table 5. Results of running the SVM on test data

User	Test Data Accuracy of SVM	Squared Error in Test Data Detection
1	100%	-
2	93.33%	0.06
3	62.5%	0.236
4	100%	-
5	50%	0.241
6	42.11%	0.254
7	53.85%	0.253
8	100%	0.086
9	-	-

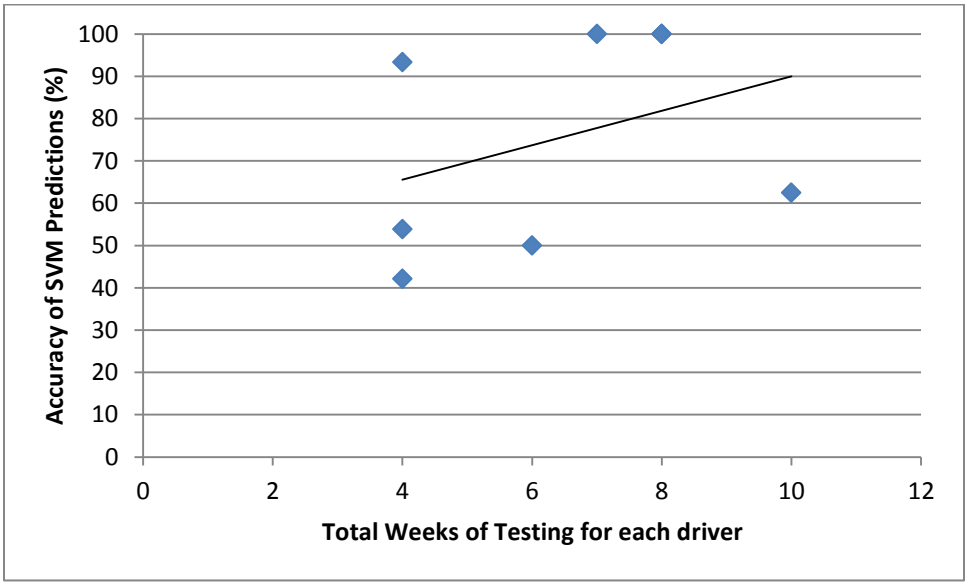

Figure 4.7. Accuracy of SVM in relation to data collected

4.6 Acknowledgments

The work presented in this chapter is funded by the California DOT (Caltrans) contract 65A0343. We wish to thank Caltrans for allowing us to use their right-of-way to conduct our experiments, the San Francisco Metropolitan Transportation Commission (MTC) for providing us with data sources for traffic conditions. We also wish to thank NAVTEQ and SpeedInfo for the San Francisco Bay Area data feeds they supplied during this project. Last but not least, we wish to thank the staff members and development engineers of the University of California PATH Program for their hard work in implementing components of the Networked Travel application used in this research.

CHAPTER 5: DESTINATION LEARNING

5.1 Introduction

The use of Smartphones for travel information and transportation related applications is suffering from lack of personalization and user acceptance. Transportation-related applications that are being deployed adopt a miniaturization approach to soliciting user input for information they need to deliver results [47]. They adopt similar methodologies used for desktop applications with large screen interfaces to the small screens available on smartphones. As a result, users are faced with the challenge of typing on small screens with small keyboards. Furthermore, several of those applications lack the intelligence that is expected of them. The Smartphone is a great source of information about its owner. It knows about the user's movements, schedule, contacts, favorites etc. Why can't transportation applications use that information to infer some of the user required input to assist the user in their moving around? In this Chapter we present a machine learning algorithm that infers the user's destination from previous GPS traces collected from the user's Smartphone. The proposed algorithm enables traffic information and route navigation applications to route the user to their destination without requiring the user to enter the destination manually. The ability to predict user destination could also be used in applications that provide soft safety warnings on road closures, traffic incidents, and congestion conditions to seamlessly alert the user of such events without bothering the user with destination input on every trip.

5.2 Related Work

Current and recent efforts of predicting driver destination have been treated as part of the problem of predicting driver routes. Several of the papers we surveyed treat the destination as another point on the route and attempt to predict the whole route as part of predicting the destination. We hypothesize that predicting the destination by itself would be an easier task yielding better results with less data sources. This hypothesis is motivated by the fact that at any given time and location, a driver has more route options to take than destination options. The prediction of destination covers a smaller space and hence should be easier to attain. The following papers are part of the related work in this field, their approach and results are summarized and compared with our approach and results.

The WhereNext project [48][49] uses GPS traces collected from several drivers' smartphones to infer possible trajectories that drivers might take in a city like Milan. The learning algorithms adopted by [49] depend on the movement of all available objects in a certain area instead of on the individual history of an object. They produce prediction trees that contain the spacio-temporal properties that have emerged from the data and they define matching methods that strictly depend on the properties of such movements. Limited by how far and how long the trips are, their algorithm fails to predict far-away destinations and long trip destinations. They achieve an accuracy prediction close to 80% for 20% of the predicted trajectories and 40% for all trajectories. Morzy in [50] uses data mining techniques based on geo-grids of varying sizes to

predict a moving object's trajectory. The work presented in [50] achieves 80% accuracy on simulator-generated data.

Simmons et al [51] use a Hidden Markov Model (HMM) to define the possible routes on a driver's trajectory. Infused with the map restrictions, they are able to predict with 98% accuracy the route and destination of the driver. However, by relaxing the map restrictions, they achieve 72% accuracy in their results which were evaluated using one month data for one user.

Lamb and Thiebaux [52] offer a method to predict vehicle location using a Kalman Filter to predict the next position and then use an HMM to force possibilities on a map. Their work presents a design of a system with no actual verification on a data set.

Letchner et al [53] profile each driver with a ratio that depicts the driver's efficiency in choosing the fastest route when compared to other drivers. They then use a HMM to snap the predicted routes on the map. Using this technique they achieve 46.6 % accuracy in their prediction.

All the related work that we reviewed couples the problems of solving for destination prediction with route prediction. In this Chapter, our attempt is to de-couple the two problems by providing a method for predicting the driver's destination irrespective of the route they are taking. We argue that solving the destination problem will enable a simpler solution for the route prediction problem in later work. The problem of predicting the driver destination is shown to be a simple decision tree classifier problem. Data collected from the Networked Traveler Field Test across 10 users driving for 6-8 weeks equipped with GPS readers on their Smartphones reveal that a regular person visits at most less than 20 different locations during the test period. By running a decision tree classifier on the collected data, one can classify the destination of the trip for a particular user based on the day of the week, the time of day, the current position and the position the user was at in the last 5 minutes.

The layout of this Chapter is as follows: we first introduce Decision Tree classifiers, then present the data collected on the 10 subjects of the Networked Traveler experiment. We include with that a simple algorithm to identify destinations in GPS traces collected from phones. Results are presented last showing the accuracy and confidence of the predictions.

5.3 Decision Trees

Decision trees [55] are powerful and popular tools for classification and prediction. The attractiveness of decision trees is due to the fact that decision trees represent rules. Rules can readily be expressed so that humans can understand them or even directly used in a database access language like SQL so that records falling into a particular category may be retrieved.

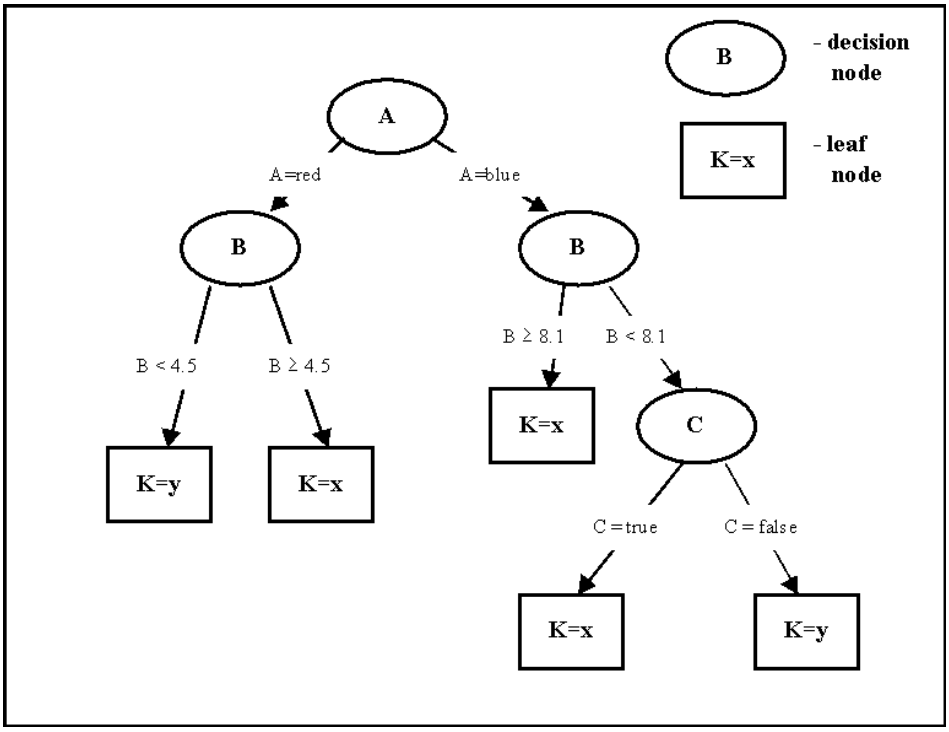

Figure 5.1. Example of a Decision Tree

There are a variety of algorithms for building decision trees that share the desirable quality of interpretability. Most algorithms define the decision tree as a classifier in the form of a tree structure (see Figure 5.1), where each node is either:

- a leaf node - indicates the value of the target attribute (class) of examples, or
- a decision node - specifies some test to be carried out on a single attribute-value, with one branch and sub-tree for each possible outcome of the test.

A decision tree can be used to classify an example by starting at the root of the tree and moving through it until a leaf node, which provides the classification of the instance.

Decision tree induction is a typical inductive approach to learn knowledge on classification. The key requirements to do mining with decision trees are:

- Attribute-value description: object or case must be expressible in terms of a fixed collection of properties or attributes. This means that we need to discretize continuous attributes, or this must have been provided in the algorithm.
- Predefined classes (target attribute values): The categories to which examples are to be assigned must have been established beforehand (supervised data).
- Discrete classes: A case does or does not belong to a particular class, and there must be more cases than classes.
- Sufficient data: Usually hundreds or even thousands of training cases.

Most algorithms that have been developed for learning decision trees are variations on a core algorithm that employs a top-down, greedy search through the space of possible decision trees. Decision tree programs construct a decision tree T from a set of training cases. Quinlan [55] originally developed ID3, the core of every algorithm that constructs decision trees. ID3 is based on the Concept Learning System (CLS) algorithm.

---

function ID3

Input: (R: a set of non-target attributes,

C: the target attribute,

S: a training set) returns a decision tree;

begin

If S is empty, return a single node with

value Failure;

If S consists of records all with the same

value for the target attribute,

return a single leaf node with that value;

If R is empty, then return a single node

with the value of the most frequent of the

values of the target attribute that are

found in records of S; [in that case

there may be be errors, examples

that will be improperly classified];

Let A be the attribute with largest

Gain(A,S) among attributes in R;

Let {aj| j=1,2, .., m} be the values of

attribute A;

Let {Sj| j=1,2, .., m} be the subsets of

S consisting respectively of records

with value aj for A;

Return a tree with root labeled A and arcs

labeled a1, a2, .., am going respectively

to the trees (ID3(R-{A}, C, S1), ID3(R-{A}, C, S2),

.....,ID3(R-{A}, C, Sm);

Recursively apply ID3 to subsets {Sj| j=1,2, .., m}

until they are empty

end

ID3 searches through the attributes of the training instances and extracts the attribute that best separates the given examples. If the attribute perfectly classifies the training sets then ID3 stops; otherwise it recursively operates on the m (where m = number of possible values of an attribute) partitioned subsets to get their "best" attribute. The algorithm uses a greedy search, that is, it picks the best attribute and never looks back to reconsider earlier choices. Note that ID3 may misclassify data.

The central focus of the decision tree growing algorithm is selecting which attribute to test at each node in the tree. For the selection of the attribute with the most inhomogeneous class distribution the algorithm uses the concept of entropy.

5.3.1 Entropy and Information Gain

The estimation criterion in the decision tree algorithm is the selection of an attribute to test at each decision node in the tree. The goal is to select the attribute that is most useful for classifying examples. A good quantitative measure of the worth of an attribute is a statistical property called information gain that measures how well a given attribute separates the training examples according to their target classification. This measure is used to select among the candidate attributes at each step while growing the tree.

In order to define information gain precisely, we need to define a measure commonly used in information theory, called entropy, that characterizes the (im)purity of an arbitrary collection of examples. Given a set S, containing only positive and negative examples of some target concept (a 2 class problem), the entropy of set S relative to this simple, binary classification is defined as:

$$Entropy(S) = - p_p log2\ p_p - p_n log2\ p_n \qquad (1)$$

where $p_p$is the proportion of positive examples in S and p_n is the proportion of negative examples in S. In all calculations involving entropy we define 0log0 to be 0. Notice that the entropy is 0 if all members of S belong to the same class. Note the entropy is 1 (at its maximum!) when the collection contains an equal number of positive and negative examples. If the collection contains unequal numbers of positive and negative examples, the entropy is between 0 and 1.

If the target attribute takes on c different values, then the entropy of S relative to this c-wise classification is defined as

$$Entropy(S) = \sum_{i=1}^{c} -p_i log_2 p_i \qquad (2)$$

where pi is the proportion of S belonging to class i. Note that if the target attribute can take on c possible values, the maximum possible entropy is $log_2 c$.

Given entropy as a measure of the impurity in a collection of training examples, we can now define a measure of the effectiveness of an attribute in classifying the training data. The measure we will use, called information gain, is simply the expected reduction in entropy caused by partitioning the examples according to this attribute. More precisely, the information gain, Gain (S, A) of an attribute A, relative to a collection of examples S, is defined as

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \qquad (3)$$

where Values(A) is the set of all possible values for attribute A, and S_v is the subset of S for which attribute A has value v. Note the first term in the equation for Gain is just the entropy of the original collection S and the second term is the expected value of the entropy after S is partitioned using attribute A. The expected entropy described by this second term is simply the sum of the entropies of each subset S_v, weighted by the fraction of examples $|S_v|/|S|$ that belong to S_v. Gain (S,A) is therefore the expected reduction in entropy caused by knowing the value of attribute A. Put another way, Gain(S,A) is the information provided about the target attribute value, given the value of some other attribute A.

The process of selecting a new attribute and partitioning the training examples is now repeated for each non-terminal descendant node, this time using only the training examples associated with that node. Attributes that have been incorporated higher in the tree are excluded, so that any given attribute can appear at most once along any path through the tree. This process continues for each new leaf node until either of two conditions is met:

1. every attribute has already been included along this path through the tree, or
2. the training examples associated with this leaf node all have the same target attribute value (i.e., their entropy is zero).

5.4 The Networked Traveler Destination Prediction Algorithm

The system described in Chapter 3: was given to 10 drivers to use for a period ranging between 6 to 8 weeks. Drivers were asked to drive naturally on their regular trips throughout the experiment. Drivers were also asked to turn on the application at the beginning of every trip and turn off the application at the end of the trip. The Smartphone application logged the following parameters of interest at a 1Hz frequency:

- GPS longitude and latitude of the location of the Smartphone
- Speed at which the Smartphone is moving
- GPS timestamp of the recorded measurement

The following table (Table 6) shows the number of GPS trace points collected for each user throughout the experiment period.

Table 6. Experiment Users Data Summary

User	Number of Trips	GPS Trace Points
1	43	91,133
2	14	40,405
3	22	58,206
4	27	79,603
5	13	47,390
6	49	103,987
7	13	31,440
8	42	125,693
9	27	49,405
10	19	44,076

A total of 125,693 data points were collected for a single driver for a period of 8 weeks. The plots in Figure 5.2 show the data for a single driver. The destination, as defined in the section "GPS Trace Destination" of this Chapter is used to count the number of destinations in Figure 5.3 for a single driver.

The plots show that there is no uniform distribution across day of week or hour of the day; in fact, both those parameters seem to affect the destination of the driver as we shall demonstrate in this Chapter.

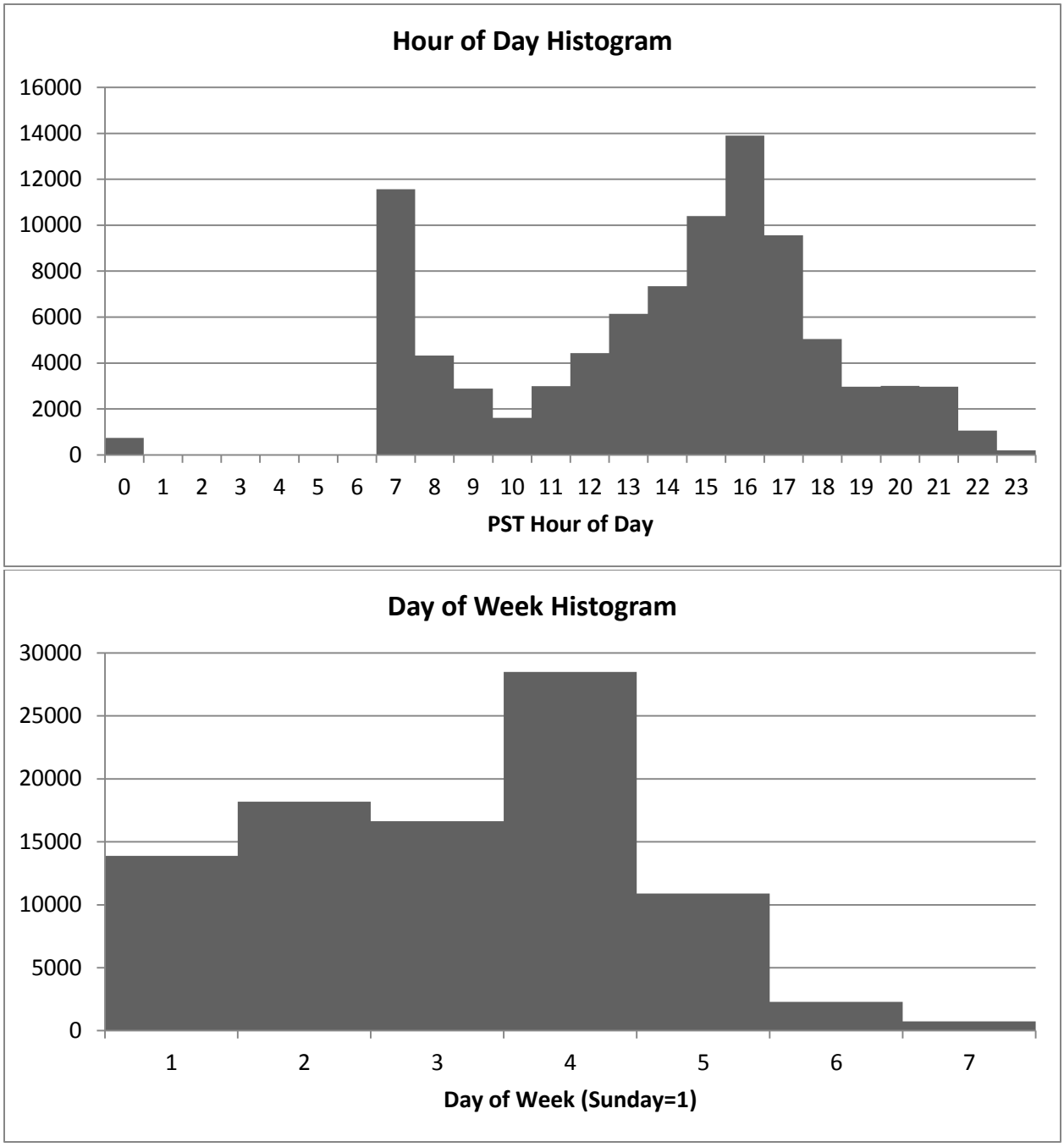

Figure 5.2. Different views of the data from a single driver.

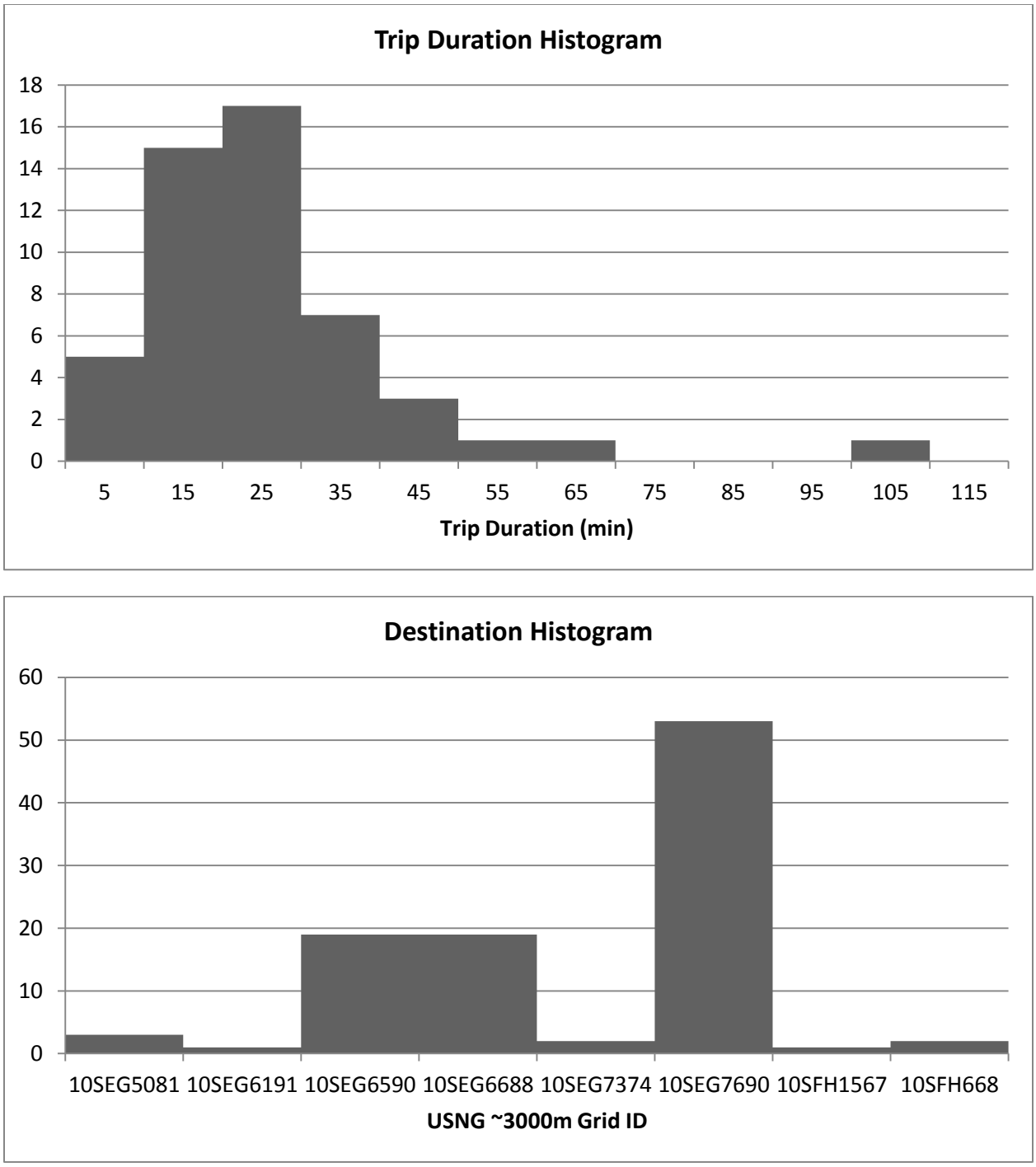

Figure 5.3. Trip and destination data for a single driver

5.4.1 GPS Trace Destination

The first step in predicting the driver destination is to define the destination as it relates to the GPS traces. The following mathematical representation of a destination will be used in the analysis of this Chapter.

GPS Speed is modeled as a time series:

$$V = \{V_t : t \in T\},\ V_t > 0 \qquad (4)$$

T: epoch time in sec

We define the set of times, t, at which trips end (destination) as S_D. Then

$$\forall t_e : V_{t_e} \leq 7kph\ and\ \sum_{i=e+1}^{e+300} V_{t_i} = 0 \rightarrow t_e \in S_D \qquad (5)$$

t_e then represents the times at which a destination in the GPS trace is reached.

Decision Tree Classification for Predicting Destination

Using the above data for each user, we create a training set comprising of 90% of the data captured for that user during the experiment. The training data set is made up of the following attributes:

- Current Position as defined within USGS 1000m grids
- Position Previous 5min as defined with USGS 1000m grids
- Time of Day in hours
- Day of Week (Sunday being 1)
- Destination as calculated from the definition of destination in this Chapter and mapped to a 1000m USGS grid

The Destination is treated as labeled data for the training set. The training set is then boosted up to 10 times and fed to a decision tree classifier. The classifier is configured to prune the tree to take out any noise fitting and to limit the maximum depth to 20 levels. No map configuration data is fed into the decision tree and no route information is provided beyond the current position of the driver.

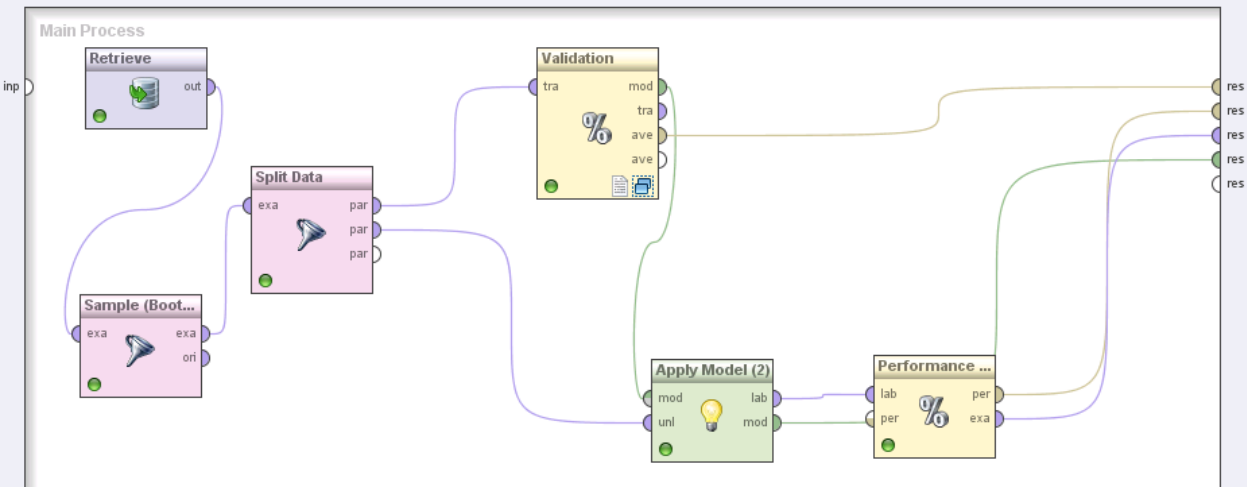

Figure 5.4. Decision Tree Modeling Process in RapidMiner®

The decision tree is programmed into RapidMiner® to optimize the classification accuracy of the training set. Once the model is created for a particular user, the model is then applied on the 10%

test set to assess the accuracy. The following table shows the results on the test sets for all the drivers:

Table 7. Prediction Results

User	Number of Trips	Training Sample Size	Accuracy of Prediction
1	43	82,020	95%
2	14	36,365	94%
3	22	52,385	96%
4	27	71,643	97%
5	13	42,651	99%
6	49	93,588	90%
7	13	28,296	96%
8	42	113,124	97%
9	27	44,465	96%
10	19	39,668	97%

5.5 Results

Accuracy is defined as:

$$accuracy = \frac{number\ of\ true\ positives + number\ of\ true\ negatives}{number\ of\ true\ positves + false\ positives + false\ negatives + true\ negatives} \quad (6)$$

The results show in Table 7 show an average accuracy of 96% on the test data for each driver. In fact, using the T-Statistics of the sample, we can say that with 95% confidence, the average accuracy is 96 +/- 1.72%.

5.6 Conclusion

We have shown that predicting the destination of a driver can be achieved with high accuracy and confidence from just GPS trace data. While others have treated the problem of destination prediction as part of route prediction, requiring more data sources such as map data, user route trends and GPS traces, they have failed to achieve this level of accuracy. We conjecture that similar performance can be achieved in predicting the routes now that the destination is known. This sets the stage for our future work in this area.

BIBLIOGRAPHY

[1] ITS U.S. Joint Program Office: 'VII Architecture and Functional Requirements Version 1.1', Federal Highway Administration, US Department of Transportation, Washington, D.C., 2005

[2] Bertini, R.: 'Transforming Transportation Through Connectivity', Presentation at the 17th ITS World Congress, Busan, Korea, October 2010.

[3] Raj, G.: 'A Detailed Comparison of CORBA, DCOM and Java/RMI', http://my.execpc.com/~gopalan/misc/compare.html, 1998

[4] Guttman, E.: 'Service Location Protocol: Automatic Discovery of IP Network Services', IEEE Internet Computing, 1999, 3, (4), pp. 71-80.

[5] Kempf, J. and St.Pierre, P.: 'Service Location Protocol for Enterprise Networks' (Wiley, 1999)

[6] Shaikh, A., Tewari, R. and Agarwal, M.: 'On the effectiveness of domain name systems', IEEE INFOCOM. Anchorage, Alaska, 2001

[7] University of Michigan Transportation Research Institute and General Motors. Automotive Collision Avoidance System Field Operational Test – Methodology and Results Appendices. DOT HS 809 901, National Highway Traffic Safety Administration, Washington , D.C., 2005.

[8] comScore "Reports December 2009 U.S. Mobile Subscriber Market Share," Reston, VA Feb. 8, 2010

[9] Malik, Om: 'iPhone App Downloads Are Up. What About Their Usage?', http://gigaom.com/2008/08/10/iphone-app-downloads-are-up-what-about-their-usage/, 2008

[10] Yardley, G. 'AppStore Secrets: What We've Learned From 30,000,000 Downloads', PinchMedia presentation by the CEO (http://www.slideshare.net/pinchmedia/iphone-appstore-secrets-pinch-media?type=powerpoint), 2009

[11] ITS US DOT: 'VII Concepts of Operations - ITS', Federal Highway Administration, US Department of Transportation, Washington, D.C., 2006

[12] Mietzner, R.: CVIS - Cooperative Vehicle-Infrastructure Systems. 'COM Safety' , Issue 3. July 2007

[13] http://www.cvisproject.org, accessed September 2007

[14] Setsou, H.: Smartway project towards the next generation ITS. 'COM Safety' , Issue 3. July 2007

[15] The CAMP Vehicle Safety Communication Consortium: 'Vehicle Safety Communication Project Task 3 Final Report: Identify Intelligent Vehicle Safety Applications Enabled by DSRC', National Highway Traffic Safety Administration, US DOT, Washington, D.C. 2005

[16] http://www.calm.hu, accessed September 2007

[17] http://www.w3.org/TR/soap, accessed September 2007

[18] Raj, G.: 'A Detailed Comparison of CORBA, DCOM and Java/RMI', http://my.execpc.com/~gopalan/misc/compare.html, 1998

[19] Erl, T.: 'Service-Oriented Architecture (SOA): Concepts, Technology, and Design.' (Prentice Hall, 2005)

[20] Guttman, E.: 'Service Location Protocol: Automatic Discovery of IP Network Services', IEEE Internet Computing, 1999, 3, (4), pp. 71-80.

[21] Kempf, J. and St.Pierre, P.: 'Service Location Protocol for Enterprise Networks' (Wiley, 1999)

[22] Jain, R., Raleigh,T., Graff, C. and Bereschinsky, M.: 'Mobile Internet Access and QoS Guarantees Using Mobile IP and RSVP with Location Registers', IEEE International Conference on Communications, 1998, 3, pp. 1690-1695

[23] Zahariadis, T.B., Vaxevanakis, K.G., Tsantilas, C.P. and Zervos, N.A.: 'Global Roaming in Next Generation Networks', IEEE Communications Magazine, 2002, 40 (2), pp. 145-151

[24] Tangmunarunkit, H., Govindan, T., Shenker, S. and Estrin, D.: 'The impact of routing policy in Internet paths', IEEE INFOCOM, Anchorage, Alaska, 2002

[25] Shaikh, A., Tewari, R. and Agarwal, M.: 'On the effectiveness of domain name systems', IEEE INFOCOM. Anchorage, Alaska, 2001

[26] Manasseh, C. and Sengupta, R.: 'Middleware for Cooperative Vehicle Infrastructure Systems', California Partners for Advanced Transit and Highways, Berkeley, 2007

[27] RSS2.0: 'Really Simple Syndication', 2007

[28] RFC 5023: 'The Atom Publishing Protocol', 2007

[29] RFC 4366: 'Transport Layer Security', 2008

[30] Manasseh, C. and Sengupta, R.: 'Middleware: Leveraging Mobile Communications for Road Safety and Congestion Relief', Proc. ITS World Congress, New York, NY, November 2008

[31] SAE J2735: 'Dedicated Short Range Message Set (DSRC) Dictionary', 2006

[32] Manasseh, C., Ahern, K. and Sengupta, R.: 'The Connected Traveler: using location and personalization on mobile devices to improve transportation', CHI LocWeb 2009, Boston, April 2009

[33] FHWA: 'Telecommunications Handbook for Transportation Professionals: The Basics of Telecommunications', Federal Highway Association, 2004

[34] Sengupta, R., and Y. P. Fallah. The Rise of the Mobile Internet: What does it mean for Transportation. Presented at *National Workshop for Research on High-Confidence Transportation Cyber-Physical Systems: Automotive, Aviation & Rail*, Washington, D.C., 2008.

[35] Kiefer, R.J., D. J. LeBlanc, and C. A. Flannagan. Developing an Inverse Time-To-Collision Crash Alert Timing Approach Based on Drivers' Last-Second Braking and Steering Judgments. In Accident Analysis & Prevention, Vol. 37, Issue 2, 2005, pp. 295-303.

[36] Brunetti-Sayer, T., J.R. Sayer, and M.H. Devonshire. Assessment of a Driver Interface for Lateral Drift and Curve Speed Warning Systems: Mixes Results for Auditory and Haptic Warnings. In Third Int.Driving Symp. Human Factors in Driver Assessment, Training, and Vehicle Design, University of Iowa Public Policy Center, 2005, pp. 218-224.

[37] Lee, J., D.V. McGehee, T.A. Dingus, and T. Wilson. Collision Avoidance Behavior of Unalerted Drivers Using a Front-to-Rear-End Collision Warning Display on the Iowa Driving Simulator. In Journal Transportation Research Record: Journal of the Transportation Research Board, No. 1573, Transportation Research Board of the National Academies, Washington, D.C., 1997, pp. 1-7.

[38] Böhm, M., S. Fuchs, R. Pfliegl, and R. Kölbl. Driver Behavior and User Acceptance of Cooperative Systems Based on Infrastructure-to-Vehicle Communication. In Journal Transportation Research Record: Journal of the Transportation Research Board, No. 2129, Transportation Research Board of the National Academies, Washington, D.C., 2009, pp. 136-144.

[39] Lindgren, A., A. Angelelli, P.A. Mendoza, and F. Chen. Driver Behaviour When Using an Integrated Advisory Warning Display for Advanced Driver Assistance Systems. IET Intell. Transp. Syst., Vol. 3, No. 4, 2009, pp. 390-399.

[40] Cosenza, S., P. Lytrivis, and F. Ahlers. SAFESPOT Integrated Project – IST-4-026963-IP Delliverable SP1 SAFEPROBE – Data Fusion Specifications. SAFESPOT Cooperative Systems for Road Safety, 2008.

[41] Lytrivis, P., G. Vafeiadis, M. Bimpas, A. Amditis, and C. Zott, Cooperative Situation Refinement for Vehicular Safety Applications: The SAFESPOT Approach. Presented at 16th World Congress on ITS, Stockholm, Sweden, 2009.

[42] Zheng, Z., S. Ahn, and C. Monsers. Impact of Traffic Oscillations on Freeway Crash Occurrences. Accident Analysis and Prevention, Vol. 42, 2010, pp. 626-636.

[43] Malyshkina, N. and F.L. Mannering. Analysis of Effect of Speed-Limit Increases on Accidents Causation and Injury Severity. Presented at 87th Annual Meeting of the Transportation Research Board, Washington, D.C., 2008.

[44] B. Schölkopf, Burges, C. J. C., and Smola, A.J. Advances in Kernel Methods: Support Vector Learning, MIT Press, Cambridge, MA, 1999.

[45] Efron, B. "Censored data and the bootstrap", Journal of the American Statistical Association, Vol.76, Issue. 374, pp. 312-319, 1981.

[46] Kohavi, R. "A study of cross-validation and bootstrap for accuracy estimation and model selection". In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, Vol. 2, Issue. 12, pp. 1137–1143, 1995.

[47] Krause, A., Smailagic, A., Siewiorek, D.P., "Context-Aware Mobile Computing: Learning Context-Dependent Personal Preferences from a Wearable Sensor Array," IEEE Transactions on Mobile Computing, vol. 5, no. 2, pp. 113-127, Feb. 2006

[48] Nanni, M., Trasarti, R., Renso, C., Giannotti, F., and Pedreschi, D. 2010. Advanced knowledge discovery on movement data with the GeoPKDD system. In Proceedings of the 13th international Conference on Extending Database Technology

[49] Monreale,A., Pinelli, F., Trasarti, R., and Giannotti, F. 2009. WhereNext: a location predictor on trajectory pattern mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (KDD '09). ACM, New York, NY, USA

[50] Morzy, M. Mining frequent trajectories of moving objects for location prediction. MLDM, volume 4571 of *LNCS*, pages 667–680. Springer, 2007.

[51] Simmons, R.; Browning, B.; Yilu Zhang; Sadekar, V.; , "Learning to Predict Driver Route and Destination Intent," *Intelligent Transportation Systems Conference, 2006. ITSC '06. IEEE* , vol., no., pp.127-132, 17-20 Sept. 2006

[52] P. Lamb and S. Thiebaux, "Avoiding explicit map-matching in vehicle location," in The 6th ITS World Congress (ITS-99), 1999.

[53] J. Letchner, J. Krumm, and E. Horvitz, "Trip router with individualized preferences (trip): Incorporating personalization into route planning," in Eighteenth Conference on Innovative Applications of Artificial Intelligence (IAAI-06), 2006.

[54] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and Regression Trees. Wadsworth, Belmont, CA, 1984.

[55] Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn. 1, 1 (Mar. 1986), 81-106.