

Prepared by

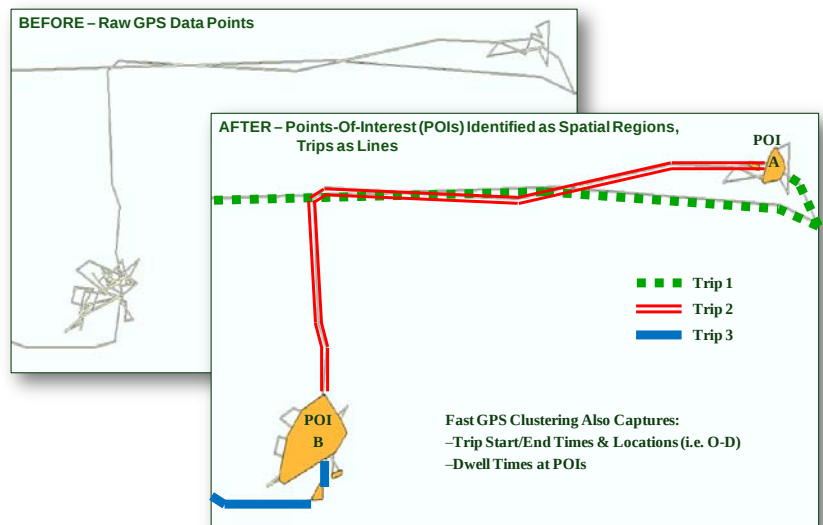
**National Center
for Transit Research**



Dynamic Travel Information Personalized and Delivered to Your Cell Phone

March 2011

Final Report



Funded by

**Florida Department
of Transportation**



FDOT BDK85 TWO 977-14

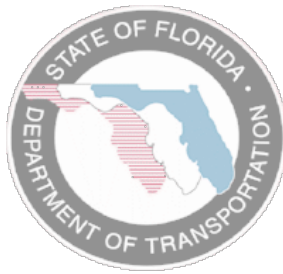
DISCLAIMER

The opinions, findings, and conclusions expressed in this publication are those of the authors and not necessarily those of the State of Florida Department of Transportation.

Dynamic Travel Information Personalized and Delivered to Your Cell Phone

FDOT BDk85 Task Work Order #977-14

Prepared for:



Florida Department of Transportation
Amy Datz, Project Manager

Prepared by:



Sean J. Barbeau, Research Associate
Nevine Georggi, Senior Research Associate
Philip Winters, Transportation Demand Management Program Director
USF Center for Urban Transportation Research

Final Report

March 2011

METRIC CONVERSION TABLE

SYMBOL	WHEN YOU KNOW	MULTIPLY BY	TO FIND	SYMBOL
LENGTH				
in	inches	25.4	millimeters	mm
ft	feet	0.305	meters	m
yd	yards	0.914	meters	m
mi	miles	1.61	kilometers	km
VOLUME				
fl oz	fluid ounces	29.57	milliliters	mL
gal	gallons	3.785	liters	L
ft³	cubic feet	0.028	cubic meters	m ³
yd³	cubic yards	0.765	cubic meters	m ³
NOTE: volumes greater than 1000 L shall be shown in m ³				
MASS				
oz	ounces	28.35	grams	g
lb	pounds	0.454	kilograms	kg
T	short tons (2000 lb)	0.907	megagrams (or "metric ton")	Mg (or "t")
TEMPERATURE (exact degrees)				
°F	Fahrenheit	5 (F-32)/9 or (F-32)/1.8	Celsius	°C

1. Report No. USF 21177804	2. Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle Dynamic Travel Information Personalized and Delivered to Your Cell Phone		5. Report Date January 2011	
		6. Performing Organization Code	
7. Author(s) Sean J. Barbeau, Nevine Labib Georggi, and Phil Winters		8. Performing Organization Report No.	
9. Performing Organization Name and Address National Center for Transit Research Center for Urban Transportation Research , University of South Florida 4202 East Fowler Avenue, CUT100, Tampa, FL 33620-5375		10. Work Unit No. (TRAIS)	
		11. Contract or Grant No. BDK85 TWO 977-14	
12. Sponsoring Agency Name and Address Florida Department of Transportation 605 Suwannee Street, MS 30 Tallahassee, FL 32399-0450		13. Type of Report and Period Covered DRAFT Final	
		14. Sponsoring Agency Code	
15. Supplementary Notes FDOT Project Manager - Amy Datz, State Transit Environmental Planner			
16. Abstract The policy of FDOT is to use "the Florida Advance Traveler Information System as the primary method to disseminate timely and important travel information to the public so that the public can make informed decisions regarding their travel plans." Travelers can subscribe to alerts for specific road segments, reasons for the alert, and days of week. However, well-defined "push" alerts can also subject the user to a high number of text messages or emails. Researchers found that a single subscription to 2 highways in Tampa Bay resulted in 6851 emails and many more text messages, sent to a single user, with an average of more than 16 emails per Friday, over 1.4 years. These messages also may be irrelevant to the user based on his real-time location (e.g., not outside the coverage area) and next intended destination. In addition, alerts received while the traveler is driving increase the risks of distracted driving. Overwhelming the travelers with irrelevant information will likely decrease its value and, therefore, its ability to influence travel behavior. If such systems automatically "pull" the information and deliver it intelligently at pertinent time and place, the usefulness of the information would have an impact on the user's travel behavior; and that is the premise of this research. With this information, people can make more informed decisions about their travel options such as seeking an alternate route, changing a departure time or changing mode. This project explored the development of technology that delivers dynamic, personalized traffic alerts only when the alert is relevant to the user's real-time location or predicted next destination and departure time. This system also sought to incorporate real-time transit information based on nearest transit stop. The approach was to use TRAC-IT, a software architecture supporting simultaneous travel behavior data collection and real-time location-based services for Global Positioning System (GPS)-enabled mobile phones, as the basis for developing several software applications to deliver predictive messaging. A Fast GPS Clustering algorithm was developed to identify Points-of-Interest (POIs) that users frequently visit. The Trip Segmentation algorithm developed by the research team separates raw GPS data into trips from POI to another. Destination and Departure Time Predictions module uses history of POIs and trip information to predict the user's next destination and time of departure to deliver timely pre-trip information. Path Prediction estimates the actual path a user will take to their next destination and identifies any active incidents they may encounter. Finally, real-time transit information from Hillsborough Area Regional Transit and real-time traffic incidents from the FL511 Application Programming Interface (API) were integrated with TRAC-IT to demonstrate the feasibility of personalized services using data from real-world traveler information systems. A prototype mobile application for the Android (Google, Inc.) platform was also implemented to demonstrate a method of delivering traffic alerts to a cell phone only when the user is traveling below a threshold speed. The message can also be delivered in audio format using the Android Text-to-Speech API so the traveler is not required to read while driving. Using predictive technology and based on real-time and historical travel patterns, this prototype demonstrated that providing personalized traveler information that has the potential to affect trip-making decisions is indeed feasible. Additional research needs were identified before full-scale deployment. Given the current capabilities of cell phones and expectation of more advancement in the industry, this promising software can potentially offer the traveling public the means of adapting to current traffic or weather situations and foster more use of public transportation.			
17. Key Word Global Positioning System, GPS, travel survey, mobile phone, cellular phone, 511, advanced traveler information systems, and ATIS		18. Distribution Statement Available to the public through the National Technical Information Service (NTIS), 5285 Port Royal Road, Springfield, VA 22161, (703) 487-4650, http://www.ntis.gov/ , and through the NCTR website at http://www.nctr.usf.edu/ .	
19. Security Classif. (of this report) unclassified	20. Security Classif. (of this page) unclassified	21. No. of Pages 89	22. Price

ACKNOWLEDGEMENTS

This report was prepared by the National Center for Transit Research (NCTR) at the Center for Urban Transportation Research (CUTR) at the University of South Florida (USF) through the sponsorship of the Florida Department of Transportation (FDOT) and the U.S. Department of Transportation.

University of South Florida (USF) CUTR Project Team:

- Sean Barbeau, Co-Principal Investigator and Research Associate, CUTR
- Nevine Labib Georggi, Co-Principal Investigator and Senior Research Associate, CUTR
- Philip L. Winters, Co-Principal Investigator and TDM Program Director, CUTR
- Miguel Labrador, PhD, Department of Computer Science & Engineering, USF
- Rafael Perez, PhD, Department of Computer Science & Engineering, USF

FDOT Project Manager:

- Amy Datz, State Transit Environmental Planner, FDOT

USF Computer Science and Engineering Student Team:

- Isaac Taylor, Graduate Research Assistant
- Narin Persad-Maharaj , Undergraduate Research Assistant
- Theodore Larkins, Research Experience for Undergraduates (REU) Student
- Richard Meana, Research Experience for Undergraduates (REU) Student

Narin Persad-Maharaj is the primary author of the “Fast Clustering of Global Navigation Satellite System Data and Trip Segmentation” portion of the report.

The research team thanks the Sprint-Nextel Application Developer Program for the donation of cellular service which was used in testing devices as part of this project.

Various technologies discussed in the report are 2011 patent pending by USF.

EXECUTIVE SUMMARY

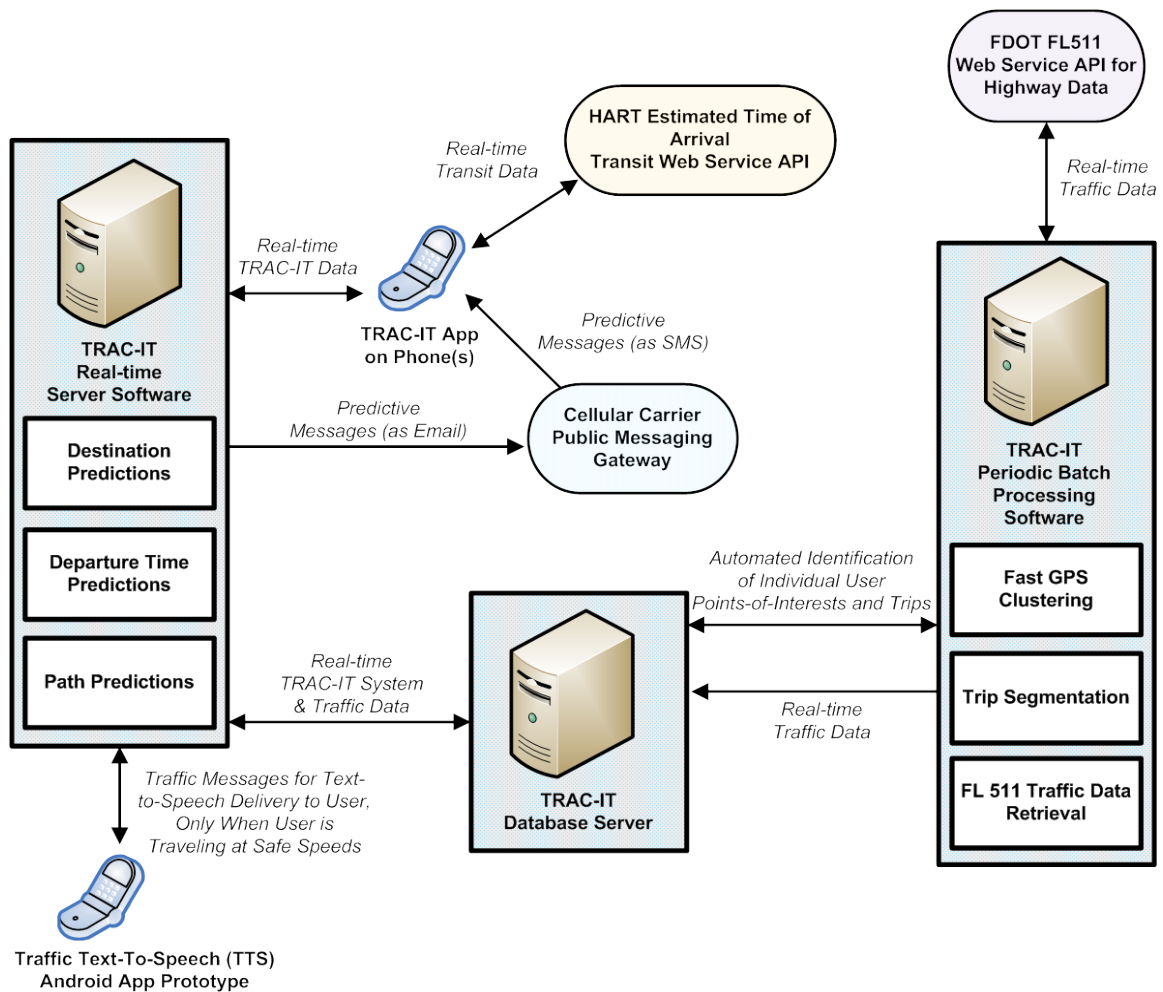
Advanced Traveler Information Systems (ATIS) are evolving in an effort to keep pace with modern communication networks and mobile devices such as cell phones. 511 systems in many states, including Florida, now support the delivery of traveler alerts via emails and text messages as part of traffic alert subscriptions. While the “push” delivery of the alerts reaches users faster than “pull” information obtained by actively requesting it, “push” alerts subject the traveler to a deluge of information that may not be relevant to the user, depending on context.

Advanced 511 “push” systems can provide the user a method of reducing the scope of alerts to specific pre-scheduled times and roads which define user commute patterns, but fixed-schedule systems can have difficulty providing relevant alerts to users that do not have travel behavior that can be represented as tightly-constrained commutes. This difficulty manifests primarily in a large number of false positive alerts (i.e., alerts are delivered to the user even if the incident is not relevant to the user’s current location, since the user is subscribed to the road but not currently in the process of traveling near the road) and false negative alerts (i.e., the user does not receive a traffic alert that is relevant to their current location since their subscription does not cover nearby roads or current times). The number of email messages, sent to a single user from subscriptions to two highways in the Tampa Bay area, I-75 and I-275, was observed during this project. The number of alerts reached an average of 13 email messages (and many more text messages) per day of the week, with an average of 17 emails on Fridays. Over 60 emails were received by the user on September 7, 2010. A large number of messages that are irrelevant can result in the user discarding and/or ignoring the messages and eventually unsubscribing from the service. The number of false positive and false negative alerts delivered to the user can be reduced by understanding more about the real-time context of the user, specifically the real-time location and the next intended destination.

This project explored dynamic, personalized travel information for GPS-enabled cell phones by building on past research sponsored by the National Center for Transit Research and the Florida Department of Transportation which created TRAC-IT, a software architecture supporting simultaneous travel behavior data collection for all modes of transportation and real-time location-based services for GPS-enabled mobile phones. ES Figure 1 is a high-level view of the architecture used to develop this prototype. TRAC-IT utilizes the Location-Aware Information System Client (LAISYC) framework, which enables the efficient use of GPS and wireless communication as part of a real-time location information system. The TRAC-IT system features a TRAC-IT mobile phone application which can capture real-time user positions as frequently as once per second on any mode of transportation (e.g., car, public transportation, biking, walking) for opt-in travel behavior surveys and archive this data for later analysis. Since travel behavior data is collected in real-time, TRAC-IT has the potential to provide real-time location-based services, such as travel information or traffic alerts, to the user in order to give them an incentive for participating in a survey. TRAC-IT also includes technology for a Personal Travel Coach, which provides post-trip information to users that suggests travel behavior changes, such as trip chaining or carpooling, which might improve the efficiency of the user’s travel while reducing energy use, pollution, and congestion.

TRAC-IT’s capabilities for real-time location-based services were enhanced in this project by integrating and advancing Path Prediction technology, as well as connecting TRAC-IT to real-world sources of both real-time traffic and public transportation information. Path Prediction uses the real-time location of the user, along with their personal travel route history, to predict the path they will travel in the immediate future. For example, if a user is leaving their work location and typically travels to home from work, Path Prediction will utilize the real-time knowledge of their presence at work and their past trips traveling from work to home to predict this path as a probable immediate path that the user may take. Past versions of Path Prediction relied

solely on spatial queries based on the user's real-time GPS position and the intersection of this position with past paths to predict possible immediate paths. While that method gave predictions when the user was moving, it did not provide predictions while the user was standing still (e.g., while the user is at work during the day). In order to determine future paths while the user is standing still for long periods of time, additional spatial-data mining of the user's travel behavior must be performed to extract additional patterns from their historical data. A Fast GPS Clustering algorithm was created that performs spatial-data mining on raw GPS data in order to identify and process Points-of-Interest (POIs) that the user frequently visits, as well as trips, which are defined as the spatial and temporal properties that define movement from one POI to another. A basic probabilistic engine using a Naïve Bayes classifier (i.e., an independent feature model) was also created to predict both the destination for the user's next trip while they are standing still for a long period of time at a POI, as well as an estimated departure time from the user's current location (i.e., POI). Destination and Departure Time prediction can be used to detect incidents that lie in the path the user typically takes from their current location to their next predicted destination, and the information delivered to the user, within a half hour (or other length of time) of their departure time. By having timely pre-trip information delivered to the user that is tailored to their personal travel behavior, the user is able to make decisions about whether to change, delay, or cancel their planned trip based on real-time traffic conditions. Travelers can also use this pre-trip information to make timely decisions on the use of alternative transportation modes such as public transportation.



ES Figure 1 - High-level Architecture of the Prototype Technology Developed under this Project

Once the TRAC-IT system has predicted the immediate future travel behavior of a user, it must have access to real-time data sources to provide real-time travel information to users. Software was created which integrated TRAC-IT with the Florida 511 (FL511) system through a web service Application Programming Interface (API) to provide real-time incident information for Florida's highways to travelers as related to their predicted path. This project also demonstrates integration of real-time multimodal travel information into a single mobile application through the processing of the Hillsborough Area Regional Transit (HART) estimated time-of-arrival API. By connecting to HART's system, TRAC-IT can now provide the estimated time-of-arrival for the stop closest to the traveler's position without the user having to enter any information into the phone.

This project also developed a prototype software application for Android (Google, Inc.) smartphones, Traffic Text-To-Speech (Traffic TTS), that can deliver traffic information only when the user is traveling below a speed threshold or has stopped moving, and can speak traffic information to the traveler using TTS APIs so the user does not have to look at the phone, both without affecting the primary functionality of the mobile phone or the ability to receive other text messages. Therefore, Departments of Transportation (DOTs) can provide traffic incident alerts in a manner that does not interfere with the traveler's normal use of the cell phone, but still provides traveler information in a responsible manner.

Many observations made throughout the course of this project are discussed in this report. First, DOTs should be encouraged to provide travel information to the general public as openly as possible, and allow personalized customization for the number and types of alerts that a user will receive. The existing My Florida 511 service is an excellent first step towards more personalized traffic information in the state of Florida. The creation of Application Programming Interfaces (APIs), such as the Florida Department of Transportation 511 Feed API, is an excellent way to provide travel information to third parties that can create new types of services based on this information without any cost to the DOT. It would not have been possible to test the technology created as part of this project without access to some type of API which provides programmatic access to real-time traffic information.

Advanced personalization (sometimes called hyper-personalization) of messages will become even more critical as DOTs begin monitoring arterial roads in real-time in addition to highways. If the number of alerts based on two interstates in the Tampa Bay area reached an average of 13 email messages (and many more text messages) per day of the week, with an average of 17 emails on Fridays, the number of alerts generated from interstates and arterials will reach extreme numbers very quickly. In addition, manually subscribing to arterial roads is expected to become more difficult and time-consuming for users because travelers may use different arterial roads. By further evolving personalization technologies, DOTs can further filter the deluge of real-time traffic information to travelers and ensure that users are only getting information that is directly relevant to them. One such technology as described in this report is the hyper-personalization techniques based on the user's current location and personal travel behavior history. Hyper-personalization can reduce the number of irrelevant alerts sent to a user, helping to retain users and preventing travel alerts from becoming another form of "digital noise." The Path Prediction technology does not require users to manually create subscription lists, as it builds a list of roads frequently traveled by each user, which is then used to generate personal traffic alerts based on the traveler's real-time location and their most-likely immediate travel behavior.

Departments of Transportation and researchers should examine how many emails and text messages are sent to each user per each incident, as too many messages can quickly result in message fatigue and may result in fewer users as a result of cancelled subscriptions. DOTs and researchers may be able to draw upon studies from the field of marketing and advertising to determine what a "reasonable" frequency of traffic alerts would be before users begin ignoring messages. For example, the current design used by the FL511 system of

sending messages out when an incident is created, updated, and cleared should be revisited to see if there is an alternate design methodology which could result in fewer messages per incident. For example, a traffic alert message could be given an estimated time-to-live, or expected duration, in the initial alert sent to the user, such as:

This alert is due to a new traffic incident: [1/6/2011 9:24:12 AM]
Region Tampa Bay
County Hillsborough
Highway I-275
Type Incidents
Severity intermediate
Reported At 1/6/2011 9:22:43 AM
Description Crash in Hillsborough on I-275 south ramp to Exit 39 Memorial Hwy, off-ramp right lane blocked. Last updated at 09:22:59AM.
Expected duration: 20 minutes

Then, additional emails could be sent only if the actual duration extends beyond the expected duration or a new expected duration is estimated. This technique would reduce the number of messages sent to the traveler by removing some of the “cleared” alerts for the incidents that have an estimated duration that was predicted correctly. Users could also be allowed to choose to receive message summary digests at given intervals (e.g., 15 minutes or 1 hour) which would contain all changes to incident statuses that occurred during this time period, instead of the current design which pushes a message out for every alert, even if they are seconds apart. Allowing users to specify the maximum number of alerts they wish to receive per hour or day would also reduce the number of messages sent to users.

DOTs and researchers should also consider the integration of real-time transit information when they are constructing state-wide traveler information systems. Currently, no real-time transit information appears in data available from the FL511 API system. To achieve real-time location-based multimodal traveler information within a single mobile application, the research team integrated data from a separate API maintained by Hillsborough County Area Transit (HART) which provides estimated arrival times for each HART bus stop. HART, and other transit information, could be integrated into the Florida 511 system in order to provide multimodal traveler information via a single state-wide API.

For the prototype development, this research project focused on accessing the FL511 traffic information resources. Testing the prototype with different real-time traveler information resources that have more comprehensive coverage on the traffic network will be a next step to enable the application with more useful information for travelers who may not use highways.

Future research should focus on the deployment of the Traffic TTS mobile application to users. This deployment should include integration with information that is received from the Florida 511 API, so that users may opt-out of travel behavior data collection if desired but still receive the benefits of the delivery of traffic messages via spoken voice below a certain speed. If Path Prediction provides a valuable service of highly relevant alerts, Path Prediction may be sufficient incentive for some users to participate in TRAC-IT data collection travel surveys without any monetary incentive from the travel behavior surveyor.

The Google Android platform was chosen for the evolution of the TRAC-IT mobile application to smartphones, since it is available on many different devices from different manufacturers as well as all major U.S. cellular carriers, including AT&T, Verizon, Sprint, Nextel, and T-Mobile. Android adequately supports location-based services and applications running in the background. The Traffic Text-To-Speech (Traffic TTS) application, which is partially based on the existing TRAC-IT Java Micro Edition (Java ME) mobile application,

delivers alerts to the user only when they are traveling at speeds below a threshold, and reads the message aloud to the user using a TTS API. Traffic TTS was created on the Android platform and successfully tested on several devices from different cellular carriers and device manufacturers. Android is truly an open development platform, allowing developers to use location-based services based on technology such as integrated Global Positioning System (GPS) without needing the permission of the wireless carrier, as was common with the Java ME platform.

Battery life is also likely to be even more of an issue on smartphones than on lower-end feature phones, so future research should examine the impact of Traffic TTS and TRAC-IT on smartphones. In future research, TRAC-IT could also be extended to other smartphone platforms such as iPhone's iOS and RIM's Blackberry OS as well to reach additional smartphone users. Since Blackberry utilizes Java ME, much of the existing TRAC-IT mobile software could be reused. For all these mobile applications, the same server-side TRAC-IT system code could be reused.

Future research should also examine the scalability of the Path Prediction system. Since numerous spatial queries are executed in a short amount of time, the capabilities of commercial databases to support such queries should be investigated. The accuracy of the Naïve Bayes method implemented in this paper for Destination and Departure time predictions should also be examined and compared against other more sophisticated probabilistic methods such as Markov models.

When testing Path Prediction as part of this project, challenges were encountered when executing the Path Prediction software within Glassfish Java Application server v2.1, PostGIS database server based on PostgreSQL v8.3, and PostGIS Java Database Connectivity (JDBC) drivers v1.5.0. Glassfish occasionally exhausted the database connection pool when executing spatial queries, which indicates that either the application or underlying software is not correctly releasing database connections. Future work should examine the use of newer versions of Glassfish to see if the problem is resolved, or consider utilizing SQL Server 2008 for spatial operations in place of PostGIS. While PostGIS is a free open-source spatial database server, it may not have the widespread use of SQL Server and therefore SQL Server may have a more stable platform for these specific spatial operations.

This project concludes that predictive, personal travel services are indeed feasible given today's cell phone technology and also that additional research to expand the work in the above-mentioned areas is needed.

TABLE OF CONTENTS

Executive Summary.....	vii
Chapter One.....	1
Introduction	1
Research Background	1
Background on Traveler Information Service	2
Project Overview	7
Chapter Two.....	9
Accessing real-Time Travel Information from My Florida 511.....	9
Current State-of-the-Art for the FL511 System	9
Chapter Three	17
Development of Software to Expand the Capabilities of FL511	17
FL511 Application Programming Interface.....	17
Retrieving Traffic Data from the F511 API.....	21
Retrieving Real-time Estimated Time of Arrival data for Public Transportation.....	35
Chapter Four	36
Fast Clustering of Global Navigation Satellite System Data and Trip Segmentation	36
Relationship Between Clustering, Path Prediction, and TRAC-IT.....	36
Related Works	38
The Fast Clustering Approach	39
Trip Segmentation.....	46
Fast Clustering and Trip Segmentation Results	46
Fast Clustering Conclusion.....	49
Chapter Five	50
Predicting User Destinations and Times of Departures.....	50
Using Trips, Points-of-Interest, and Naïve Bayes	50
Chapter Six	54
TRAC-IT Architecture Integration with Path Prediction, FL511 API, and Transit Information	54
Architecture Overview.....	54
Bus Stop Database	54
Sending Traffic Alerts to Phones.....	56
Proof-Of-Concept Walkthrough	57
Challenges	59
Chapter Seven	61
Traffic Text-To-Speech: A Prototype for Reducing Distracted Driving.....	61
Background on Text-To-Speech.....	61
Methodology for Developing Traffic TTS.....	62
UDP Receiver Application.....	64
Prototype Testing and Findings	65

Chapter Eight	68
Conclusions and Recommendations.....	68
Future Research.....	69
References	72

LIST OF FIGURES

ES Figure 1 - High-level Architecture of the Prototype Technology Developed under this Project	viii
Figure 1 – Beat the Traffic ®.....	3
Figure 2 - INRIX Crowd Sourcing Architecture	3
Figure 3 – Sample Camera Views from Mass511.com (Provided By Sendza).....	4
Figure 4 – Mass511 Sign-Up Screen	5
Figure 5 – FL511 Covered Roads.....	6
Figure 6 - Sample Travel Behavior Data Collected with TRAC-IT	7
Figure 7 – Florida Statewide 511 Website	9
Figure 8 – Sample Alert Sent to a My Florida 511 User.....	10
Figure 9 – Hourly Average of Mail Alerts from My Florida 511 for I-75 and I-275 in Tampa	13
Figure 10 – Average Number of Mail Alerts for Days of the Week from My Florida 511 for I-75 and I-275 in Tampa	13
Figure 11 – Number of Mail Alerts per Month from My Florida 511 for I-75 and I-275 in Tampa.....	14
Figure 12 – Highest Number of Mail Alerts in One Day from My Florida 511 for I-75 and I-275 in Tampa.....	15
Figure 13 – Architecture of FL511 Third Party API.....	20
Figure 14 – Picture Retrieved from A URL Provided by the FL511 API for I-275 at Jefferson St. on 1/4/2011 at 4:37 p.m.	21
Figure 15 – Request for Miami-Dade Floodgate Information	22
Figure 16 – XML Response Retrieved from FL511 API Showing Miami-Dade Floodgate Information	22
Figure 17 – XML-Formatted Text Sent to FL511 API to Request Event Information for Miami-Dade County.....	23
Figure 18 – XML-Formatted Response from FL511 API Showing Miami-Dade Event Information	24
Figure 19 – Database Schema that Holds Data Retrieved from FL511 Feed	27
Figure 20 – Tools such as Netbeans Simplify Implementation of Software that Communicates with FL511 API.....	28
Figure 21 – Netbeans Wizard – The WSDL of the FL511 API is entered here	29
Figure 22 – Netbeans-Generated Java Classes for the FL511 API	30
Figure 23 – Top 20 Counties with FL511 Events	31
Figure 24 – Florida Events by Hour of Day	31
Figure 25 – Hillsborough County Events by Hour of Day	32
Figure 26 – Hillsborough County Interstate Events by Hour of Day	32
Figure 27 – Number of Daily Events by Date	32
Figure 28 – Hillsborough County Events by Date	33
Figure 29 – Hillsborough County Interstate by Date	33
Figure 30 – Average Number of Events for Florida per Weekday.....	34
Figure 31 – System Architecture Showing USF, HART, and FL511 API Interaction.....	35
Figure 32 – POIs Identified Using the Fast Clustering Algorithm to Segment GPS Data into Trips	38

Figure 33 - Fast GPS Clustering Improves Performance over Traditional Hierarchical Clustering by Only Comparing Each Point to Others within its "Neighborhood"	39
Figure 34 - A Cluster is Assigned a Map of its Elements for Quick Access via an Enumerated List.....	41
Figure 35 - Fast GPS Clustering References the Element at the Root Index to Properly Merge Two Disjoint Clusters	45
Figure 36 - Raw GPS Data Before and After Processing for Two POIs.....	47
Figure 37 – Sample of Pseudo-POIs.....	48
Figure 38 - Destination Prediction Flow Chart.....	51
Figure 39 - Hour of Departure Prediction Flow Chart	52
Figure 40- TRAC-IT Architecture Supporting Path Prediction Service	55
Figure 41 - TRAC-IT Server Issues Message to Phone Using the Public Messaging Gateway	56
Figure 42 – Example Output from Destination and Departure Time Prediction Engine.....	57
Figure 43 - Path Predictions for a User’s Path in Real-Time.....	58
Figure 44 - TRAC-IT Web Application Software Log and Screenshot of Text Message	59
Figure 45 – Initial Login Screen for Traffic TTS.....	63
Figure 46 – Original Choice between UDP Receiver and TRAC-IT Android Functionality	63
Figure 47 – Original Interface for TRAC-IT Android Application	64
Figure 48 – Original Interface for UDP Receiver Application	65
Figure 49 – Final Traffic TTS Interface, with TRAC-IT Functionality	67

LIST OF TABLES

Table 1 – Traffic Event Description	19
Table 2 – Event Location Information	20
Table 3 – Clustering Algorithm Example	40
Table 4 - Fast GPS Clustering Performs Significantly Faster than Traditional Hierarchical Clustering.....	47
Table 5 - Public Messaging Gateways for Carriers.....	56
Table 6 – Estimated Time of Arrival Information Provided to TRAC-IT Mobile Phone Users	60

LIST OF ACRONYMS

API	Application Programming Interface
APK	Android Packet
ATIS	Advanced Traveler Information Systems
AVL	Automatic Vehicle Locator
CDC	Connected Device Configuration
CLDC	Connected Limited Device Configuration
CPU	Central Processing Unit
ETA	Estimated Time of Arrival
GNSS	Global Navigation Satellite Systems
GPS	Global Positioning System
GTFS	General Transit Feed Specification
GUI	Graphic User Interface
JAD	Java Application Descriptor
JAR	Java Archive
Java ME	Java Micro Edition
Java SE	Java Standard Edition
JDBC	Java Database Connectivity
JNI	Java Native Interface
JSR	Java Specification Request
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
LAISYC	Location-Aware Information System Client
MIDP	Mobile Information Device Profile
MSA	Mobile Service Architecture
OSGi	Open Services Gateway Initiative
OTA	Over-the-Air
POI	Points-of-Interest
REST	Representative State Transfer
SMS	Short Message Service
SOAP	Simple Object Access Protocol
TTS	Text-To-Speech
UDP	User Datagram Protocol
WGS	World Geodetic System

CHAPTER ONE

INTRODUCTION

RESEARCH BACKGROUND

Advanced Traveler Information Systems (ATIS) communicate essential information about travel conditions to the public. ATIS devices include variable message signs, 511 systems (including transit information), highway advisory radio, and websites that collect, process, and disseminate information. Real-time bus or train information systems have been implemented to facilitate transit usage through both perceived and actual reductions in passenger waiting time. Using information from transit Automatic Vehicle Locator (AVL) systems coupled with relative location of the traveler can help increase rider satisfaction with the transit system. For example, gaining access to the bus's location information before traveling to the bus stop can help to reduce the wait time. The guidance for developing and deploying real-time traveler information systems in transit published by the Federal Transit Administration emphasizes that [1],

Providing high quality information on transit status information has [been] shown to improve customer service, which in turn can lead to increased ridership....This real-time information enables existing and potential riders to make better pre-trip and en-route decisions. Providing real-time information has additional benefits to the transit agency: improving customer service; increasing customer satisfaction and convenience; maintaining or increasing ridership; and improving visibility of transit in the community.

Some state-of-the-art ATIS services deliver alerts via emails or text messages to mobile devices upon subscription by user. These alerts are disseminated to static subscription lists not filtered based on the user's travel history or real-time location. As a result, users will receive many irrelevant mass alerts that are not pertinent to their current location or immediate intended travel. It is likely that some users would be overwhelmed when their mobile devices are inundated by irrelevant travel information alerts and may eventually opt-out of subscriptions. The timeliness of the alert in relationship to where the commuter or the rider is affects trip-making decisions, such as postponing the start time of the trip or eliminating the trip to avoid congestion or bad weather.

This project developed a system that provides directed, personalized dynamic information to a traveler's mobile device based on his exact location determined by Global Positioning System (GPS). The timeliness of this project is heightened by the Federal Highway Administration's notice of proposed rulemaking of January 14, 2009 (23 Code of Federal Regulations Part 511) that proposes minimum parameters and requirements for states to make available and share traffic and travel conditions information via real-time information programs. It states that [2],

Real-time information programs are proposed to be established so that States easily can exchange information on the real-time operational status of the transportation network with other States and with the private sector, value-added information market. This cooperation and sharing of information could stimulate the dissemination of traffic and travel conditions that include Web or wireless access to route-specific travel time and toll

information; route planning assistance using historical records of congestion by time of day; and communications technologies that gather traffic and incident-related data from a sample of vehicles traveling on a roadway and then publishing that information to travelers via mobile phones, personal digital assistants (PDAs), in-car units, or dynamic message signs.

A research study funded by the Florida Department of Transportation (FDOT) through the National Center for Transit Research (NCTR) at the Center for Urban Transportation Research (CUTR) at the University of South Florida (USF) pioneered a travel behavior survey tool, referred to as TRAC-IT, which is software that can be installed on GPS-enabled mobile phones [3]. Several components of TRAC-IT, including location-aware information system architecture and a Path Prediction prototype, can be employed to create an application that provides highly-directed, real-time, multi-modal traffic alerts via mobile devices to the public. As part of the initial TRAC-IT project, a Path Prediction algorithm was developed and tested in a lab environment and successfully simulated the delivery of personalized real-time alerts to travelers by all modes based on their real-time GPS location and a set of sample traffic incident data. The algorithm is able to predict each user's intended route and destination based on the user's own historical travel data collected using the TRAC-IT mobile phone application, thereby reducing the number of irrelevant alerts provided to the user.

BACKGROUND ON TRAVELER INFORMATION SERVICE

There are many ways to collect and disseminate traffic information. There are also many types of agencies involved in the process including private, public, and public-private partnerships.

PRIVATELY-PROVIDED TRAVELER INFORMATION SYSTEMS

On the private side, examples of providers include INRIX, NAVTEQ, and AirSage. These private companies provide broad coverage to major metropolitan areas in the U.S. An example shown in Figure 1 is that of Beat the Traffic®, powered by INRIX and disseminated in Central Florida through a local TV station website as part of its current news coverage of events, traffic, weather, etc. There are mobile applications and subscription services to these private companies that are offered to the public to access traffic information. Signing up for INRIX TRAFFIC! through a mobile application immediately makes the subscriber a member of a crowd-sourced traffic network that is constantly gathering real-time traffic information on roads nationwide via cell phones, cameras, and probe vehicles [4].

Figure 2 is the INRIX schematic for crowd sourcing by mobile phones and probe vehicles. It shows different sources for gathering the traffic information that relies on involving the traveling public's vehicles and mobile devices as probes.

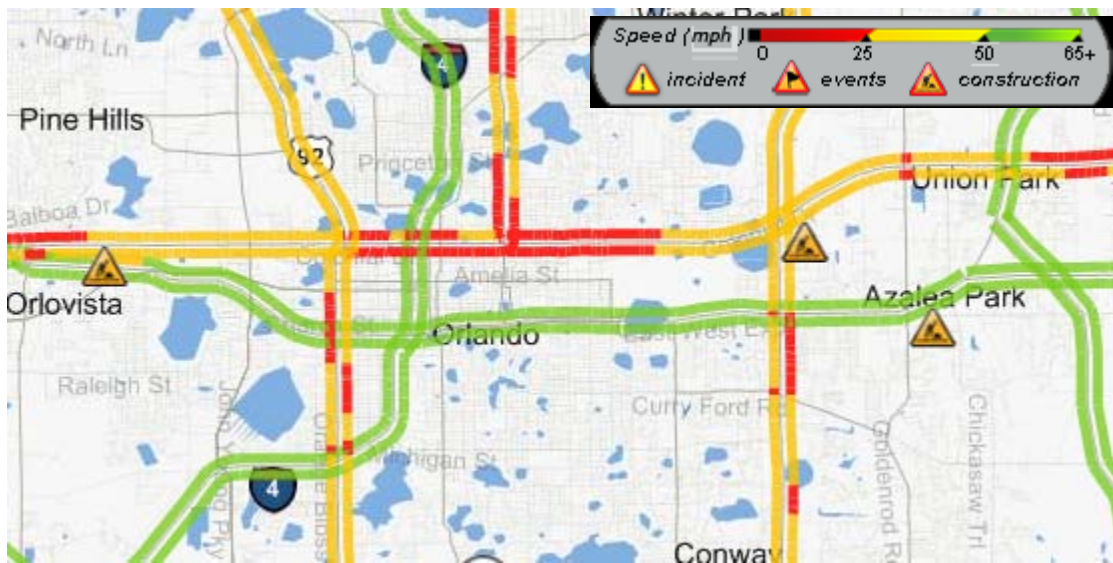


Figure 1 – Beat the Traffic

Source: <http://www.wftv.com/traffic/index.html>



Figure 2 - INRIX Crowd Sourcing Architecture

Source: <http://www.inrixtraffic.com/>

PUBLIC-PRIVATE TRAVELER INFORMATION

Another example is Mass DOT's partnership with Sendza, a growing Massachusetts communications software company, to make 511 traveler information service available as of May 2010. Sendza does not charge the

state for the service, instead relying on a sponsored-message model. Sendza uses raw data from INRIX. Figure 3 shows camera locations in and around the Boston area.

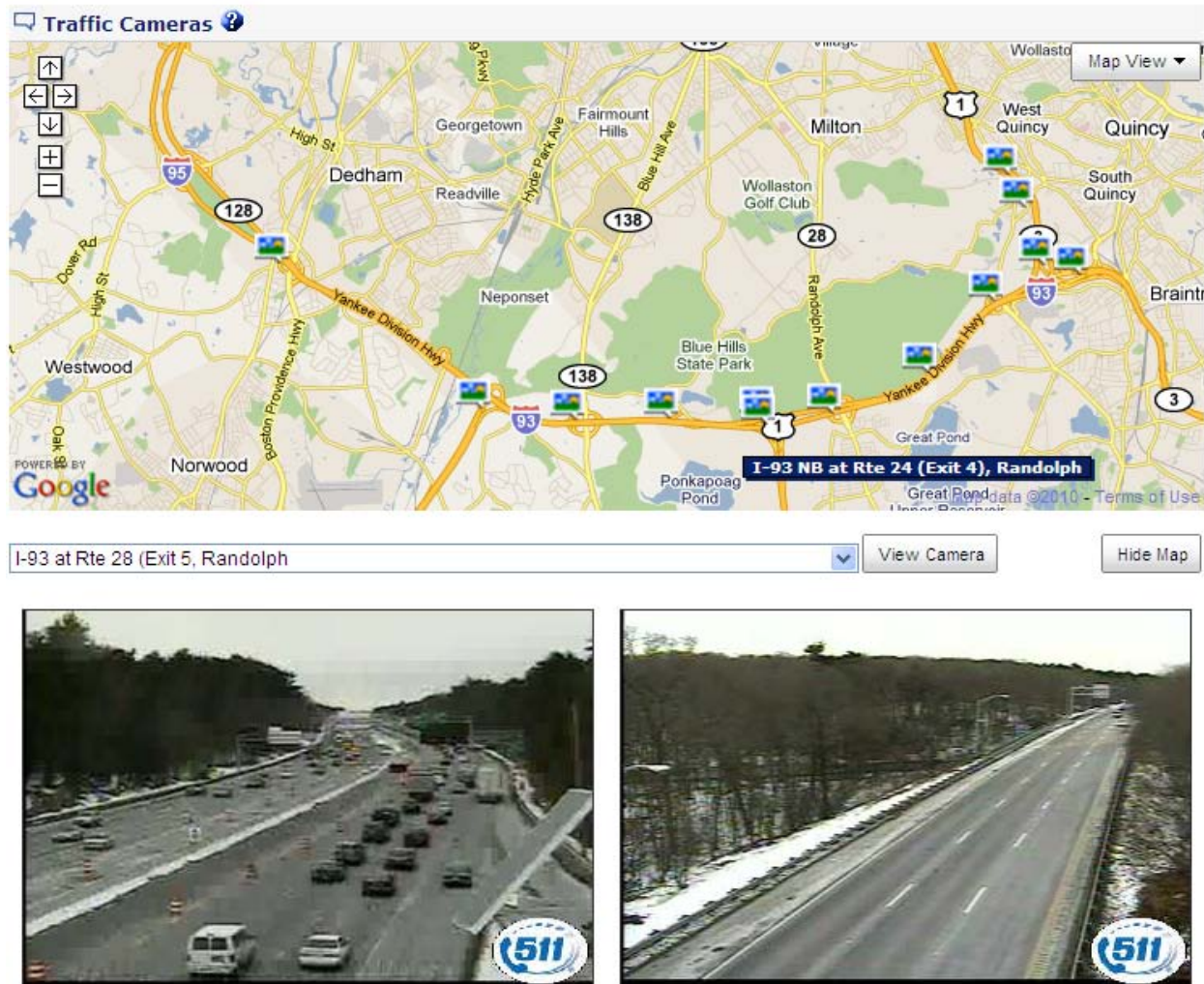
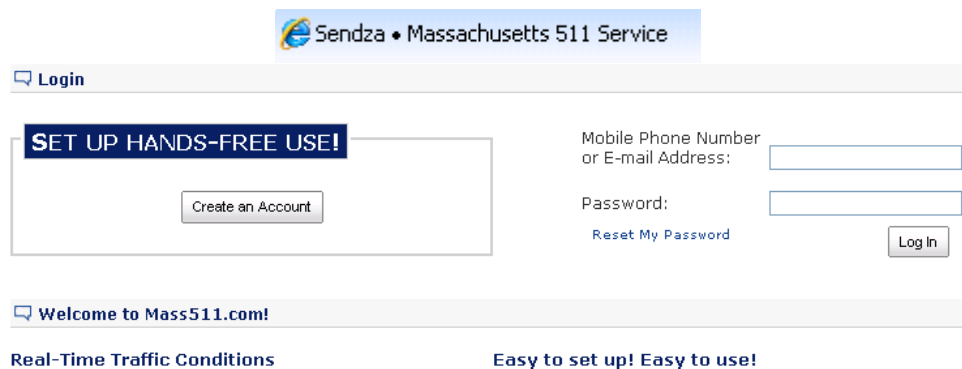


Figure 3 – Sample Camera Views from Mass511.com
(Provided By Sendza)

Source: <http://www.mass511.com/preferences.sws>

Figure 4 is an example of the sign-up screen for subscribing to the service via Mass511 Webpage.



The screenshot shows the Mass511 sign-up interface. At the top, there is a header with the Sendza logo and the text "Massachusetts 511 Service". Below this is a "Login" link. The main content area is divided into two sections. On the left, there is a box titled "SET UP HANDS-FREE USE!" with a "Create an Account" button. On the right, there are input fields for "Mobile Phone Number or E-mail Address:" and "Password:", followed by a "Reset My Password" link and a "Log In" button. Below the main content area, there is a "Welcome to Mass511.com!" message. At the bottom, there are two links: "Real-Time Traffic Conditions" and "Easy to set up! Easy to use!".

Figure 4 – Mass511 Sign-Up Screen

Source: <http://www.mass511.com/preferences.sws>

PUBLICLY-PROVIDED TRAVELER INFORMATION SYSTEMS

On the public side, 511 has become synonymous with real-time traveler information systems. Coverage and content of the information provided varies by state DOTs and local transportation agencies. For Florida, 511 is provided through a recently-centralized statewide system that is an integration of previously-deployed 511 in six Florida regions. The goals for combining the regional systems into one were to provide an integrated, seamless travel information resource and to improve customer service by offering a consistent user interface and bilingual information throughout the state. The data comes from sensors in and alongside the road, traffic surveillance cameras, and reports from Road Rangers, Florida Highway Patrol, and local law enforcement. Figure 5 shows roadways covered by 511 services in Florida. The system provides information such as commuter travel times, construction, lane closures, crashes, congestion, and severe weather affecting traffic. Callers can access traffic, transit, travel times, airports, seaports, or request a specific roadway, city, or county [5].

For the prototype development, this research project focused on accessing the FL511 traffic information resources. Testing the prototype with different resources that have more comprehensive coverage on the traffic network will be a next step to make the application more useful for more people who may not use highways for their travel.



Figure 5 – FL511 Covered Roads

Source: <http://www.fl511.com/pdf/English/511%20Covered%20Roads.pdf>

PROJECT OVERVIEW

This project explored dynamic, personalized travel information for GPS-enabled cell phones by expanding previous research sponsored by NCTR and FDOT, which created TRAC-IT, a software architecture supporting simultaneous travel behavior data collection and real-time location-based services for GPS-enabled mobile phones [6]. TRAC-IT is built using the Location-Aware Information System Client (LAISYC) framework, which enables the efficient use of GPS and wireless communication as part of a real-time location information system [7]. LAISYC framework allows TRAC-IT to capture GPS fixes on a frequent basis (e.g., every second) while the user is moving, but dynamically vary GPS refresh rate and data communication frequency to avoid having a severe negative impact on device battery life.

The TRAC-IT system features a TRAC-IT mobile phone application that can capture real-time user positions as frequently as once per second on any mode of transportation (e.g., car, public transportation, biking, walking) for opt-in travel behavior surveys and archive this data for later analysis (Figure 6). Since travel behavior data is collected in real-time, TRAC-IT can also provide real-time location-based services, such as travel information or traffic alerts, to users in order to give them an incentive for participating in a survey.

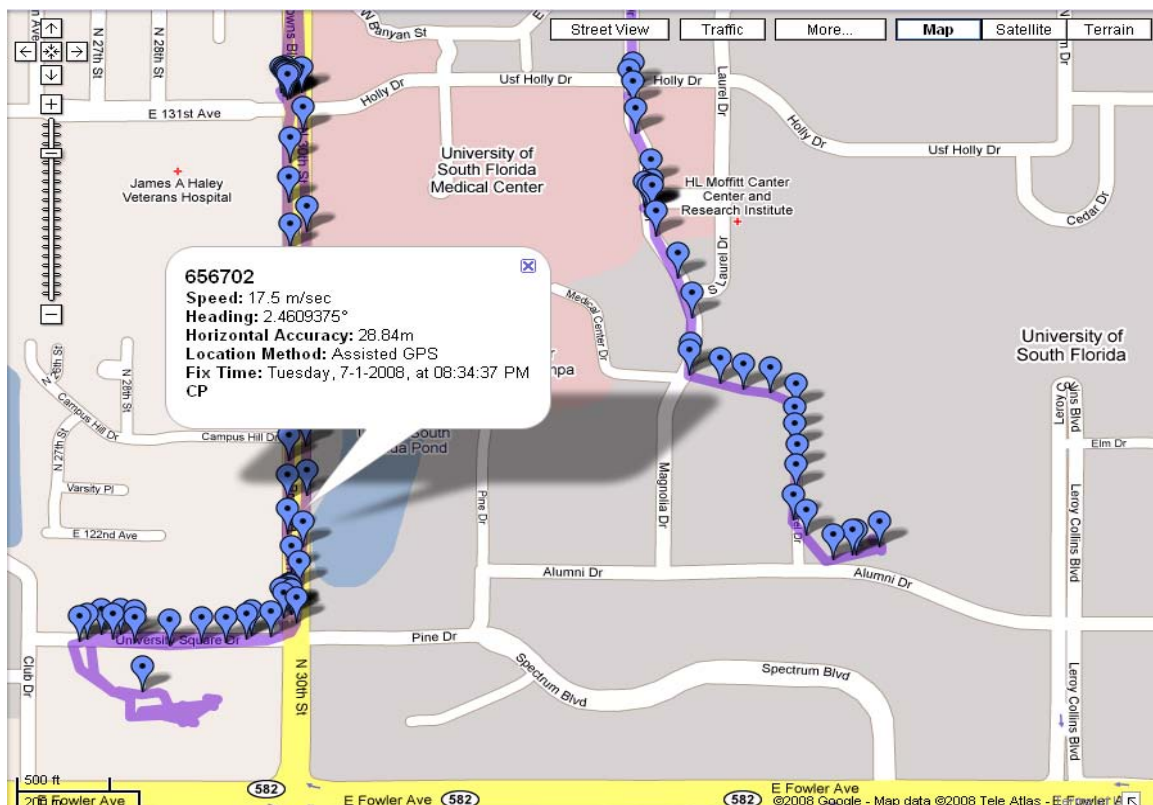


Figure 6 - Sample Travel Behavior Data Collected with TRAC-IT

TRAC-IT also includes technology for a Personal Travel Coach, which provides post-trip information to users that suggests travel behavior changes, such as trip chaining or carpooling, that might improve the efficiency of the user's travel while reducing energy use, congestion, and pollution [8].

This project enhances the TRAC-IT system by integrating and enhancing Path Prediction technology, as well as connecting TRAC-IT to real-world sources of both real-time traffic and public transportation information. The connection of Path Prediction to real-world traveler information was accomplished via an Application

Programming Interface (API) for the Florida 511 system. This integration provides real-time incident information for Florida's highways to travelers as related to their predicted path. This method provides an advantage over existing subscription-based text message and email services, as it provides information which is hyper-personalized and filtered so that users receive only information that is highly relevant to their real-time and predicted path. Real-time multimodal travel information in a single mobile application was achieved by integrating the Hillsborough Area Regional Transit (HART) estimated arrival time API. Previous research developed an application that displays estimated arrival time in the context of real-time transit navigation [9]. By connecting to HART's system, TRAC-IT can now provide the estimated arrival time for the stop closest to the traveler's position without the user entering any information into the phone. Information about the FL511 API is discussed in Chapter Two of this report while software development for integration with the FL511 API as well as the HART API is discussed in Chapter Three.

Path Prediction uses the real-time location of users, along with their personal past travel behavior, to predict the path they will travel in the immediate future. For example, if a user is leaving his work location and typically travels to home from work, Path Prediction will use the real-time knowledge of his presence at work and his past trips traveling from work to home to predict this path as a possible immediate path that the user may take. Past versions of Path Prediction relied solely on spatial queries based on the user's real-time GPS position and the intersection of this position with past paths to predict possible immediate paths. While that method gave predictions while the user was moving, it did not provide predictions while the user was not actively traveling (e.g., while the user is at work at a particular location). To determine future paths while the user is standing still for long periods of time, additional spatial-data mining of the user's travel behavior must be performed [10].

This project enhances Path Prediction by creating a Fast Clustering algorithm for GPS data that performs spatial-data mining in order to identify and process Points-of-Interest (POIs) that the user frequently visits, as well as trips, which are defined as the spatial and temporal properties that define movement from POI to another. Fast GPS Clustering is discussed in Chapter Four.

A basic probabilistic engine using a Naïve Bayes classifier (i.e., an independent feature model) was created to use the POI data produced by the Fast GPS Clustering algorithm. This probabilistic engine predicts both the destination for a user's next trip in real-time while standing still at a POI, as well as an estimated departure time from the user's current location. Destination and Departure Time prediction algorithms are discussed in Chapter Five. Chapter Six discusses the integration of the technologies discussed in Chapters Two, Three, Four and Five to create a dynamic travel information system that personalizes traffic alerts and delivers these messages to a traveler's cell phone.

This project also demonstrated a prototype mobile software application for Android (Google, Inc.) phones, Traffic Text-To-Speech (Traffic TTS), that can deliver traffic information only when the user is traveling below a speed threshold or has stopped moving, and can speak traffic information to the traveler using TTS APIs so the user does not have to look at the phone, both without affecting the primary functionality of the mobile phone or ability to receive other text messages. Therefore, DOTs can provide traffic incident alerts in a manner that does not interfere with the traveler's normal use of the cell phone, but still provides traveler information in a responsible manner. The creation of Traffic TTS is discussed in Chapter Seven. Chapter Eight concludes the report and provides recommendations for future research. Finally, an Addendum to this report provides a reference for software developers wanting to make a transition from the Java Micro Edition (Java ME) platform to Android. This research informed the creation of the Traffic TTS mobile application.

CHAPTER TWO

ACCESSING REAL-TIME TRAVEL INFORMATION FROM MY FLORIDA 511

CURRENT STATE-OF-THE-ART FOR THE FL511 SYSTEM

In addition to telephone services accessible through dialing the number “511,” the Florida 511 (FL511) system also provides a website at <http://www.fl511.com/> that provides users with a variety of information, including incidents in different Florida regions, estimated travel times on major highways, and snapshots from cameras mounted along the road. The homepage is shown in Figure 7.

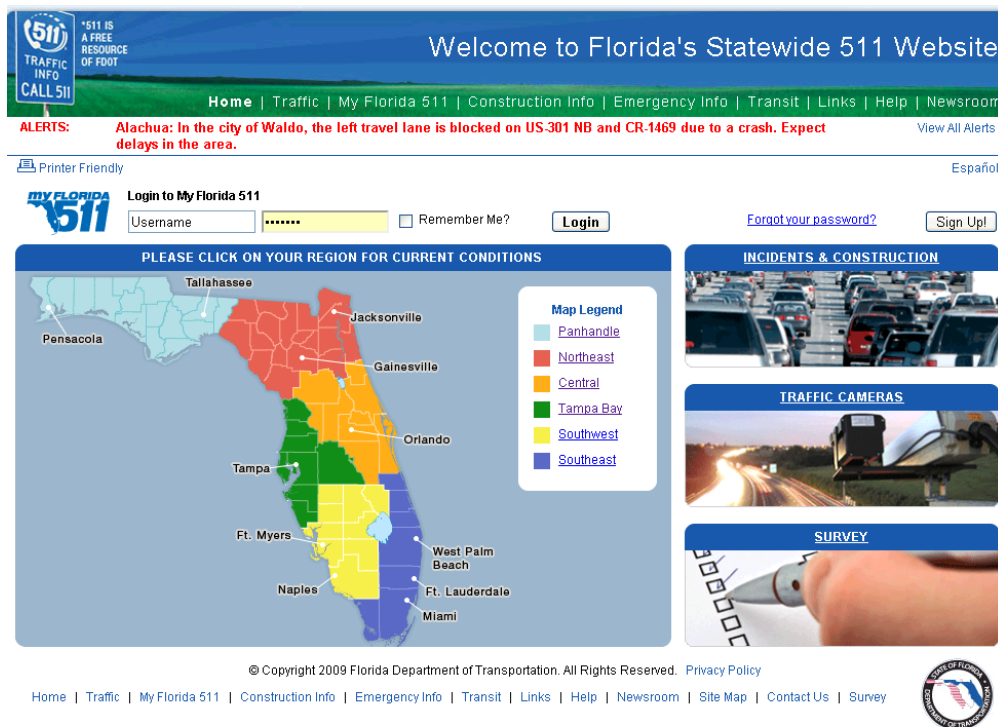


Figure 7 – Florida Statewide 511 Website

The website also includes a “My Florida 511” section, which allows the user to create a profile and sign up for specific roadways and routes at specific travel times to receive a phone call, text message, or email generated by the My Florida 511 system whenever an incident occurs on those routes within the designed area and timeframe. Figure 8 shows this alert page for a particular user’s profile.

Welcome to Florida's Statewide 511 Website

Home | Traffic | **My Florida 511** | Construction Info | Emergency Info | Transit | Links | Help | Newsroom

ALERTS: Duval: Expect full closure of the Hart Bridge Monday through Thursday nights, from 7pm to 6am, and Friday nights, beginning at 7pm, through Monday mornings, at 6am. [View All Alerts](#)

Printer Friendly Welcome, [User] User Home Page | Edit Profile | Logout [Español](#)

Modify Alert

Region & Roadway Alert

Region:

Highway:

Route Alert

Route:

Floodgate Alert

Region:

* Minimum Severity:

* Event Types: ☒ Congestion ☒ Incidents ☒ Construction ☒ Severe Weather ☐ Other

* Days Of The Week: ☒ Monday ☒ Tuesday ☒ Wednesday ☒ Thursday ☒ Friday ☒ Saturday ☒ Sunday

* Begin Time Of Day: * End Time Of Day:

Delivery Options

☒ Send by Email to Primary Email Address ([Redacted])

☒ Send by Text Message to Primary Phone Number (813 [Redacted])

☐ Send by Telephone Call to Primary Phone Number (813 [Redacted])

[Delete Alert](#) [Update Alert](#)

© Copyright 2009 Florida Department of Transportation. All Rights Reserved. [Privacy Policy](#)

Home | Traffic | My Florida 511 | Construction Info | Emergency Info | Transit | Links | Help | Newsroom | Site Map | Contact Us | Survey

Figure 8 – Sample Alert Sent to a My Florida 511 User

These subscription capabilities enable “push” communication, so that an alert is pushed to a user when it happens in real-time, much like a subscription to a discussion listserv or RSS feed. Push communication ensures that users receive information as soon as it happens, versus a “pull” approach in which users must initiate an action to receive information. An example of a “pull” approach is the general FL511 website, since the user must initiate the action of viewing the webpage. Since the user must initiate the “pull” action, a long time can pass between when an incident is identified and when the user retrieves the information. Enabling push communication of traffic alerts can be a powerful tool, since users are immediately notified of incidents. There is virtually no delay between information publication and receipt by the user. Additionally, since subscriptions for My Florida 511 allow the specification of certain highways, alerts will only be received for highways of the user’s choosing.

Following is an example of text sent in an email for an event:

Subject: RE: My Florida 511 Traffic Alert
This alert is due to a new traffic incident: [1/6/2011 6:58:39 PM]

Region Tampa Bay
County Sarasota
Highway I-75
Type Incidents
Severity major

Reported At 1/6/2011 6:44:56 PM

Description Crash in Sarasota on I-75 south at Exit 210 Fruitville Rd, 2 Left Lanes blocked. Last updated at 06:58:10PM.

Following is the same alert as it appears in a text message:

Crash on I-75 SB at Exit 210 - Fruitville Rd two Left Lanes blocked Last updated today at 6:58pm

The event description is shortened for the text message, as the full description will not fit within a standard Short Message Service (SMS) length.

THE CHALLENGE OF EFFECTIVE DISSEMINATION OF REAL-TIME TRAFFIC INFORMATION

While the push of information to a user based on subscription lists, such as email and text message alerts created using the My Florida 511 website, ensure that the user immediately receives information when it is published, push notifications can create a new challenge: information overload. When user requests information about a specific event, such as viewing the Tampa Bay portion of the website, the system does not have to guess what information is relevant to the user; the system simply responds with information on Tampa Bay when the user clicks on the “Tampa Bay” link. However, when information is pushed to a user via a subscription, the user is not actively requesting information at the time the subscription-based alert is received. Instead, the subscription-based alert is sent to the user based on a request that the user made at a previous time to receive certain types of alerts. The delay of time between when a user subscribes to a “push” information service and when information is actually received from the service can result in varying expectations on the number and frequency of alerts that will be delivered to the user. In a worst case scenario, the service might send so many irrelevant push messages that the user might discontinue the service. This would decrease the likelihood that the user would change behavior (e.g. reroute, delay trip, etc.). In this situation, the messages become “noise” that the user at first discards and then ignores completely.

My Florida 511 provides several filters that allow users to restrict the types of alerts they will receive in an attempt to reduce the number of messages, as shown in Figure 8 above. For example, users can elect to receive various types and severities of alerts and can specify the highways for which information is requested and the days of the week and time periods to receive the information. These filters prevent the user from receiving irrelevant information. For example, a user in Tampa could subscribe only to messages for highways in the Tampa Bay area, such as I-75 and I-275, during the hours of 8:30 a.m. to 8 p.m. For commuters, who have a very well-defined commute in terms of the time spent on the road (e.g. , 8:00 a.m. to 8:30 a.m. and 5:00 p.m. to 5:30 p.m. Monday-Friday) and very specific route, a “Route” alert can be created that narrows the selected travel area to specific mile markers on a highway. This ensures that the user will not receive traffic alerts, for example, for the Orlando area, only information regarding I-75 and I-275 in Tampa has been requested.

While these filters are a significant step towards reducing information overload for users, there are often cases where user travel behavior does not fit neatly into commute behavior that always occurs at specific times on specific days of the week. For example, if a user drives from one location to another within a city for meetings, but to various locations on different days of the week, this behavior will be very difficult, if not impossible, to tightly capture in a My Florida 511 subscription that covers only the time the user is actually on the road. Additionally, if a user has a regular work schedule but chooses to run errands on different days either before or after work, the tightly-bound subscription to commute behavior will not cover this travel and,

therefore, will not alert the user of incidents. Or, if a user wants to receive traffic alerts on a weekend, the exact day and time of travel, as well as destination, will likely vary from one weekend to another.

For users with travel behavior that does not fit into precise fixed time blocks and routes, a potential solution to receive traffic alerts using the existing My Florida 511 system is for the user to relax the time requirements to capture the majority of the day, such as 8:00 a.m. to 8:00 p.m., and subscribe to the highways that he most frequently travels, such as I-75 and I-275 in Tampa. Users will still miss alerts on other roads they travel or if they are driving earlier than 8:00 a.m. or later than 8:00 p.m., but, at first glance, it appears to be a reasonable approach to receive traffic information using My Florida 511 subscriptions. Another possibility would be to increase the number of roads for which the user will receive alerts, so he does not miss a critical incident.

Unfortunately, users that begin to expand subscription time or roads to encompass additional travel beyond simple regular commutes can be subjected to a deluge of information. The danger of ill-defined fixed subscriptions is that the user will miss critical alerts if he is on the road at a time outside the narrow subscription window, but the user will also receive many messages that are not relevant if a too broad subscription window is provided that encompasses additional times or roads not traveled at a fixed time and day of week.

To evaluate the number of alerts that were being issued for travel behavior that do not fall within well-defined commute trips, a subscription to My Florida 511 alerts via email and text message for a single user was established with the following details:

- Region & Roadway: Tampa Bay, I-275; Mon, Tue, Wed, Thu, Fri, Sat, Sun; 08:30 a.m. to 08:00 p.m.
- Region & Roadway: Tampa Bay, I-75; Mon, Tue, Wed, Thu, Fri, Sat, Sun; 08:00 a.m. to 08:00 p.m.

All emails sent by My Florida 511 were stored in a folder for evaluation at a later date. The following several graphs show statistics for the number of messages received from July 15, 2009, to December 31, 2010, from the My Florida 511 system for the single registered user. A total of 6,851 emails were received during the observed time period. The number of text messages received far exceeds this number, as is discussed later, but is not easily quantified over a long time period.

Figure 9 shows that email alerts tended to peak in the mid-afternoon to late evening, with an average of approximately 4-5 alerts during the evening rush hour of 5:00 p.m. to 8:00 p.m. Fridays typically had the largest number of email alerts, averaging 17 alerts, as shown in Figure 10.

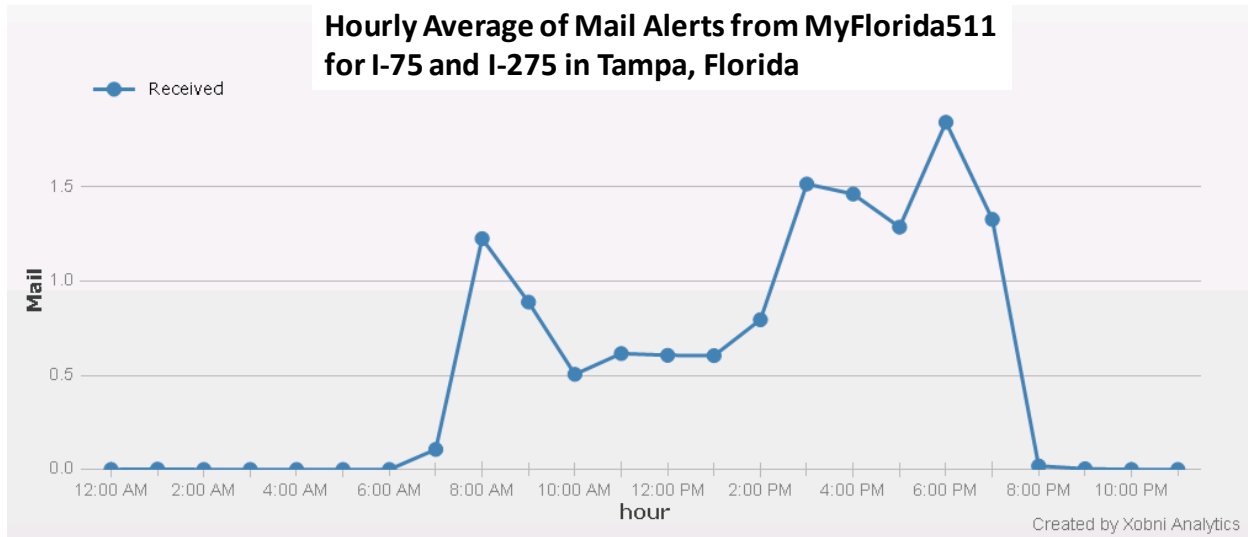


Figure 9 – Hourly Average of Mail Alerts from My Florida 511 for I-75 and I-275 in Tampa

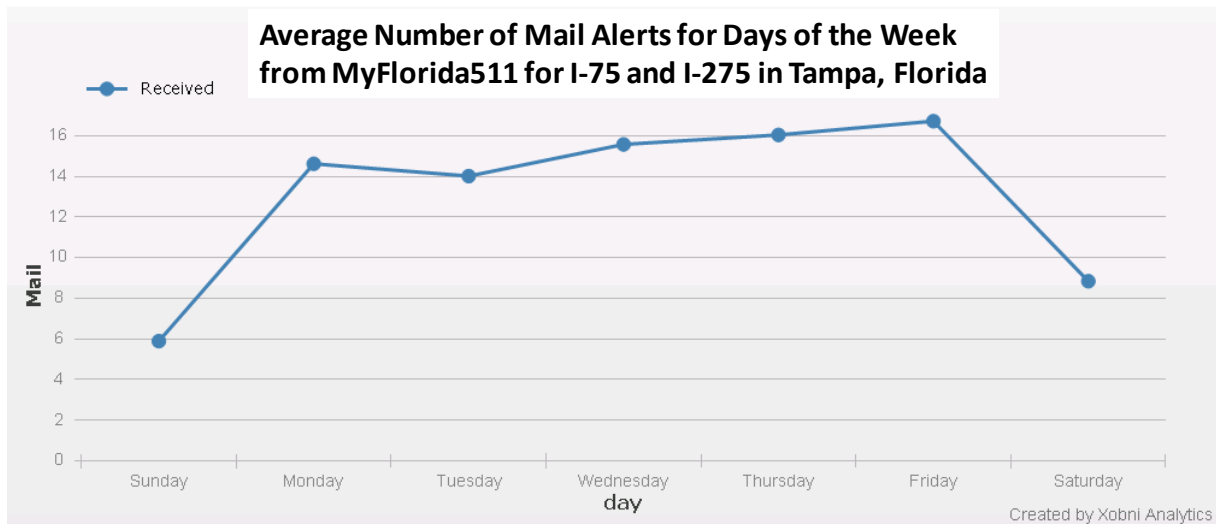


Figure 10 – Average Number of Mail Alerts for Days of the Week from My Florida 511 for I-75 and I-275 in Tampa

There is a continuing trend over time towards a larger number of messages being sent from My Florida 511 for the same subscriptions. Figure 11 shows the number of emails sent per month from July 15, 2009, to December 31, 2010. According to the FDOT Intelligent Transportation Systems Office, this increase is likely due to several factors. First, FDOT District 7 is actively expanding their ITS so that they are covering more roadways. Second, the FDOT ITS office has strongly encouraged FDOT Districts to update the information posted at least every 30 minutes so messages are not out of date. Finally, the Districts have become more comfortable with the operation of the ITS equipment and are more efficient with their reporting activities and can therefore produce more messages in less time. All of these developments contribute towards an increasing trend of alert messages.

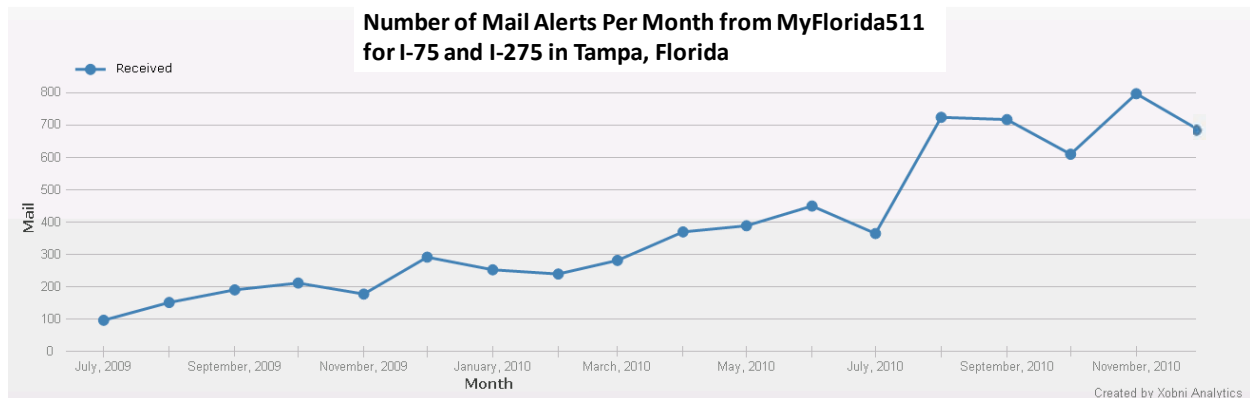


Figure 11 – Number of Mail Alerts per Month from My Florida 511 for I-75 and I-275 in Tampa

Several additional factors can cause a user to receive an extremely large number of email and text messages from the My Florida 511 system. One significant factor is that several emails may be sent to users for the same event. Typically, an email is sent when an event occurs, each time it is updated with new information, and when it is cleared from the system. This results in a minimum of two emails for each event, with often many more emails occurring per event.

Mobile phone users that subscribe to text messages for each event are subjected to an even greater possibility of information overload. During the observed time period, the number of text messages sent to a user far exceeded the number of emails. For example, on January 4, 2010, a total of 25 emails were sent by My Florida 511, but 41 text messages were sent, nearly double the amount of emails.

One reason for a greater number of text messages than emails is that at the beginning of a subscription period (e.g., 8:00 a.m.), My Florida 511 sends a single email containing a “List of Alerts” for the monitored roads that shows all currently active incidents. After this initial email, all subsequent emails until the end of the subscription period (e.g., 8:00 p.m.) are sent as single emails. However, since text messages have a limited number of characters, each event included in the “List of Alerts” is sent to a cell phone in a separate text message. Therefore, if the “List of Alerts” email includes seven active incidents at 8:00 a.m. when the subscription window starts, the user will receive seven text messages – one for each incident – in a very short amount of time.

Another reason for an extremely large number of text messages is that, frequently, the body of the text alert message is greater than the character limit of the text message. As a result, while most of the alert text appears in the first text message, the remaining text spills over into a second text message. Since the messages may not arrive simultaneously, each message has an impact on the user in terms of the effort required to read the text message. For example, a cell phone user may hear the text message notification sound for his phone when the first message arrives, open the phone to view and then delete the alert, and then return the phone to his pocket, only to receive another text notification with the remainder of the message. The remainder of the message may not always be relevant, which can further add to user frustration. An example of text spilling from one text message to another is shown below, with two text messages for a single alert issued on January 5, 2011:

First text message:

Traffic congestion on I-275 NB from before Howard Frankland N Terminus to Exit 41 DL Mabry H Last updated today at 4:17

Second text message contents:

pm

According to the FDOT ITS Office, this spillover is a result of the length of facility names from the District database. Since the database cannot be easily modified without creating ripple affects through the system, the spillover issue is not likely to be resolved in the near future.

As a result of the above issues, the sheer number of text messages received on a mobile phone within one day for travel behavior that does not easily fall within tightly constrained times and roads can be staggering. Figure 12 shows that 60+ emails were received by a single user on September 7, 2010. Since the number of text messages exceeds the number of emails, the average number of text messages received by this user exceeded five text messages per hour. If additional real-time travel information is made available for arterial roadways and users begin to subscribe to these roads as well, the noise wall of irrelevant information will become even more deafening.

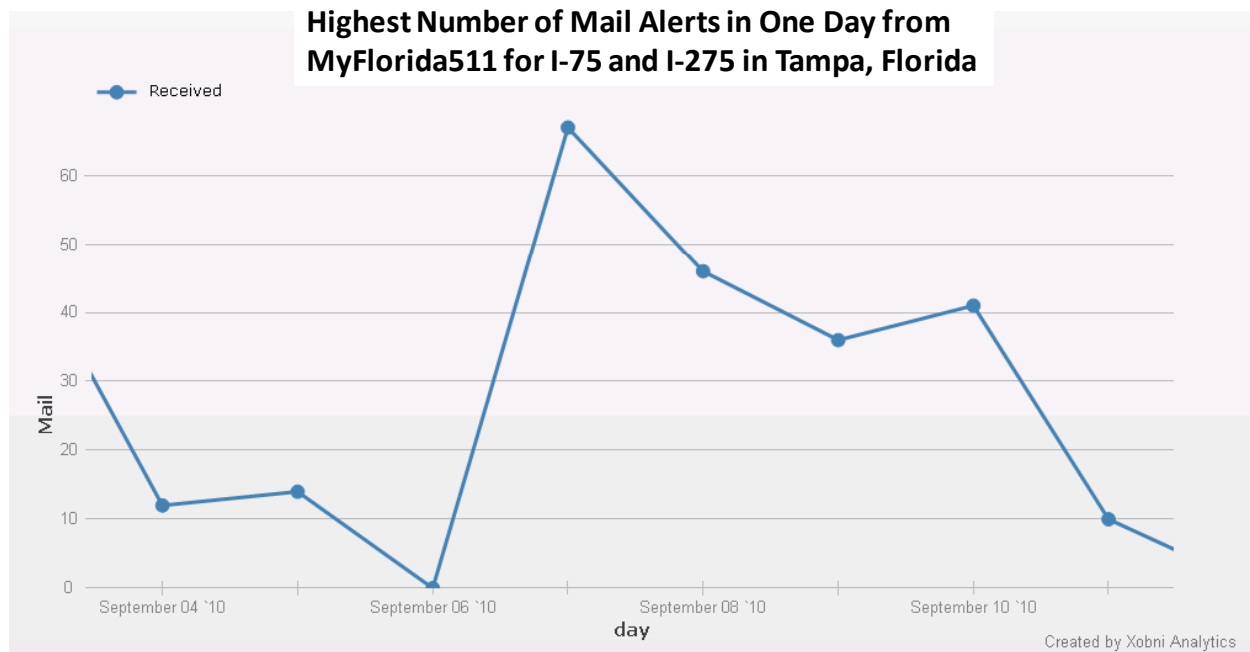


Figure 12 – Highest Number of Mail Alerts in One Day from My Florida 511 for I-75 and I-275 in Tampa

While the existing My Florida 511 service for traffic text messages makes a good initial effort to filter irrelevant text messages through personalized subscriptions, it is limited by the lack of knowledge of the context of each mobile user. This project focuses on creating several technologies based around GPS data from mobile applications that aid in further filtering out irrelevant messages through knowledge of the real-

time location of the user, as well as predictions of his immediate travel behavior for his next destination. Fortunately, My Florida 511 contains an API, which allows external systems (e.g., USF) to programmatically retrieve traffic event information through software for further processing and customized delivery to the user.

The following section details the My Florida 511 API, as well as the technology developed to access and process traffic event information.

CHAPTER THREE

DEVELOPMENT OF SOFTWARE TO EXPAND THE CAPABILITIES OF FL511

FL511 APPLICATION PROGRAMMING INTERFACE

The FL511 website is the primary electronic public-facing travel information portal for FDOT. It presents information that is collected and aggregated within the Florida SunGuide system, which is used primarily by transportation operations personnel. Inside the FL511 SunGuide system, the various traffic management centers around the state aggregate and communicate data to a centralized system using a derivation of the Society of Automotive Engineers (SAE) J2540 standard. Each message contains multiple codes, many defined by the SAE J2540 standard and some created by FDOT for specific messages not defined by the standard.

Once these data are aggregated to a centralized data store, they are made available to the general public via the Florida FL511 website. CellTrust (<http://www.celltrust.com/Products/SMS-Gateway/CellTrust-2WaySMS-Gateway.html>) is the company that issues the SMS (text message) alerts to cell phones on behalf of the FL511 website subscriptions, and it also retrieves data through tight integration with the SunGuide system.¹

While the FL511 website provides a number of useful services to the general public related to traffic information, including subscription-based “push” travel alerts, its main purpose is to provide information to individuals who visit the website. As with any general information portal, there are limitations to the number of different features that can be implemented due to constrained resources during the design and site implementation. Additionally, implementation of new third party services that interact directly with the FL511 SunGuide system via the SAE standard would be costly and, since SunGuide is used for critical transportation operations, limited to select individuals.

The system engineers behind the FL511 system realized that the travel information data within the SunGuide system could be used for many other purposes and services beyond the initial FL511 website if it were exposed to third parties in a design familiar to Web application developers. FDOT, in coordination with their consultant, implemented a “centralized XML data feed,” also referred to as an API, for the FL511 system in early 2010. This FL511 API exposes the same data presented in the FL511 website in a format that can easily be consumed by third party information systems. The FL511 API allows other parties to build upon the FL511 features currently provided to the general public and use the same travel information data source without requiring costly integration with the Florida SunGuide system.

FDOT provided the research team with access to the FL511 API hosted at <http://datafeed.fl511.com/>. The Web feed exposes real-time travel information to systems outside of the FL511 SunGuide system, such as an application running on servers at USF, via HTTP requests and responses. This interaction can be formatted in a Representative State Transfer (REST) format using HTTP calls directly or can be implemented using the XML-based Simple Object Access Protocol (SOAP). Figure 13 shows this portion of the system architecture for the interactions between USF servers and the FL511 API.

¹ The full design of the Florida511 system implementation is available online:
http://www.floridaitis.com/PDFs/TWO29-511/FLATIS_System_Design-V1-5-090408.pdf

Table 1 and

Table 2 show a sample of the SAE codes and descriptions for different incident types and locations. These descriptions are chained to form the messages that are available to third party applications via the FL511 API.

Table 1 – Traffic Event Description

1. EVENT DESCRIPTION (FDOT MOD SAE CODE GROUP 1)	
FDOT Mod SAE Code	Description
1494	Evacuation
205	Accident involving hazardous materials
2205	Crash involving hazardous materials
203	Multi-vehicle accident
2203	Multi-vehicle crash
93	Overturned tractor-trailer
30	Accident involving a tractor-trailer
2030	Crash involving a tractor-trailer
335	Accident involving bus
2335	Crash involving bus
204	Accident involving truck
2338	Crash involving truck
201	Accident
2978	Crash
2082	Emergency Road Construction
47	Disabled tractor-trailer
346	Disabled bus
212	Disabled truck
211	Disabled vehicle
213	Vehicle on fire
2081	Abandoned vehicle
95	Planned construction
61	Object on roadway
136	Traffic congestion
707	Bridge maintenance operations
907	Flooding
1706	Emergency vehicles
2048	Off Ramp Backup
2049	Police Activity
2085	Interagency coordination
2096	Weather event due to rain
2097	Weather event due to fog
2098	Weather event due to flooding
2074	Weather Event
2099	Special Event
214	Incident
2084	Unknown event

Table 2 – Event Location Information

2. EVENT LOCATION INFORMATION					
PARTIAL LIST					
2a. County	2b. Road	2c. Direction	2d. Distance from	2e. Relationship to exit	2f. Reference Point
Examples:	Examples:	Examples:	Example:	Examples:	Examples:
• Orange • Polk	• I-4 • I-95	• Eastbound • Westbound • Northbound • Southbound	• 5 miles	• At • Before • Beyond • Ramp to • Ramp from	• Exit 78 • International Drive • Exit 220A

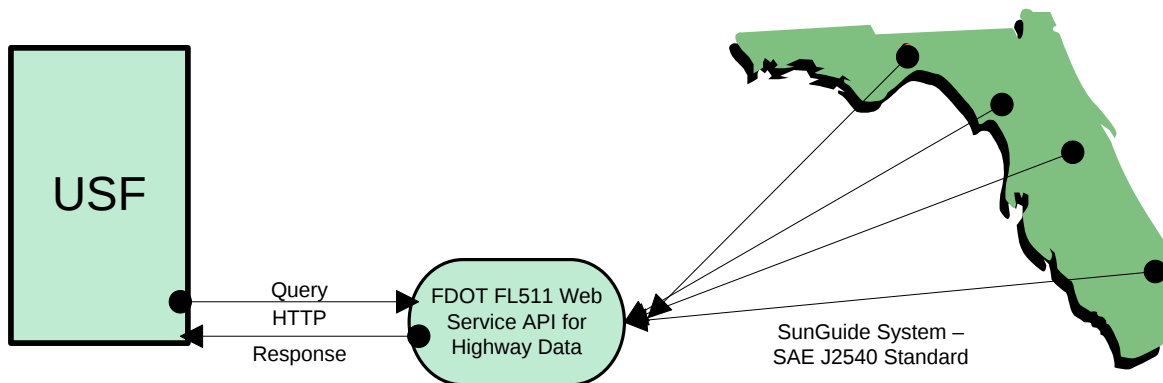


Figure 13 – Architecture of FL511 Third Party API

When accessing the FL511 API, credentials (username and password) must be provided as input as well as the county for which data should be retrieved. The FL511 API provides access to several categories of data:

- Events – Traffic incidents, construction/repair work, and lane blockage information for either a point or a stretch of road.
- Floodgates – Countywide events such as severe weather alerts or extended road closures.
- Message Boards – Dynamic message board messages for the given county.
- Travel Time Links – Real-time info about traffic speeds at a point
- Travel Times – Estimated travel times for a road segment, calculated based on the travel time links for that segment.
- Cameras – Web addresses of pictures taken by highway cameras, which can be up to 15 minutes old (see Figure 14).



Figure 14 – Picture Retrieved from A URL Provided by the FL511 API for I-275 at Jefferson St. on 1/4/2011 at 4:37 p.m.

RETRIEVING TRAFFIC DATA FROM THE F511 API

The software developed under this project to interface with the FL511 API is a Java Standard Edition (Java SE) application. Since the software is running on a desktop computer and battery life and communication bandwidth are not critical issues, the Java software was created to access the FL511 API using the SOAP protocol. When the FL511 API webservice receives a properly formatted request, it responds with an XML formatted response containing all of the objects of the requested type for the county indicated in the request. Each object in a given response contains all of the fields for the given object. For example, events contain fields for the timestamp, latitude and longitude (in micro degrees for the World Geodetic System (WGS) 84 datum), description, roadway, severity, and other attributes related to the event. Any field in a response that has no information is left blank, indicating a null value in that field. Since this project targeted events that could cause delays for travelers, the Java software was designed to retrieve information from the FL511 Web feed regarding traffic events and floodgate messages. Extreme changes in travel time on a highway generate event messages, so travel times did not need to be directly monitored. The following are examples for requests and responses formatted in XML to retrieve information about Floodgate messages.

Floodgate Request

```

extensible Markup Language
<?xml
  version="1.0"
  ?>
<S:Envelope
  xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
  <S:Body>
    <ObtainFloodgateData
      xmlns="http://www.FL511.com/webservices">
      <username>
        USF
      </username>
      <password>
        [REDACTED]
      </password>
      <county>
        Miami-Dade
      </county>
    </ObtainFloodgateData>
  </S:Body>
</S:Envelope>

```

Figure 15 – Request for Miami-Dade Floodgate Information

Floodgate Response

```

extensible Markup Language
<?xml
  version="1.0"
  encoding="utf-8"
  ?>
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <ObtainFloodgateDataResponse
      xmlns="http://www.FL511.com/webservices">
      <ObtainFloodgateDataResult>
        <Floodgates>
          <Floodgate>
            <Timestamp>
              09/30/2010 09:10:29 AM
            </Timestamp>
            <ID>
              2.4.87.000.0000.3
            </ID>
            <Center>
              Statewide
            </Center>
            <County>
              Miami-Dade
            </County>
            <Highway/>
            <Severity>
              unknown
            </Severity>
            <Entity/>
            <Message_En>
              <Message_Text>
                one SB and one NB lane closed on SW 87 Ave/Galloway Rd between SR-878 and Kendall Dr due to construction. Motorists should expect delays;
              </Message_Text>
              <Message_Speech/>
            </Message_En>
            <Message_Es>
              <Message_Text>
                La carretera SW 87 Ave/Galloway Rd entre SR-878 y Kendall Dr tiene un carril cerrado en ambas direcciones debido a construccion. Espere
              </Message_Text>
              <Message_Speech/>
            </Message_Es>
          </Floodgate>
        </Floodgates>
      </ObtainFloodgateDataResult>
    </ObtainFloodgateDataResponse>
  </soap:Body>
</soap:Envelope>

```

Figure 16 – XML Response Retrieved from FL511 API
Showing Miami-Dade Floodgate Information

Event Request

```
<?xml version="1.0" ?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ObtainEventData xmlns="http://www.FL511.com/webservices">
      <username>
        USF
      </username>
      <password>
        [REDACTED]
      </password>
      <county>
        Miami-Dade
      </county>
    </ObtainEventData>
  </S:Body>
</S:Envelope>
```

Figure 17 – XML-Formatted Text Sent to FL511 API to Request Event Information for Miami-Dade County

Event Response

```

<?xml
  version="1.0"
  encoding="utf-8"
  ?>
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <ObtainEventDataResponse
      xmlns="http://www.FL511.com/webservices">
      <ObtainEventDataResult>
        <Events>
          <Event>
            <Timestamp>
              10/01/2010 11:41:33 AM
            </Timestamp>
            <Description_En>
              Planned construction in Miami-Dade on I-75 north ramp to Exit 2 NW 138 St/Graham Dairy Rd, off-ramp left lane blocked. Last updated at 11:41:19AM.
            </Description_En>
            <Description_Es>
              Construcci\303\263n planeada en Miami-Dade en I-75 norte rampa en direcci\303\263n salida 2 NW 138 St/Graham Dairy Rd, carril izquierdo de la rampa de
            </Description_Es>
            <ID>
              259450
            </ID>
            <Center>
            </Center>
            <Type>
              current
            </Type>
            <Severity>
              unknown
            </Severity>
            <Reported_At>
              10/01/2010 12:15:07 AM
            </Reported_At>
            <Data_Last_Updated_At>
              10/01/2010 11:41:19 AM
            </Data_Last_Updated_At>
            <Primary_Location>
              <Name>
                NW 138 St/Graham Dairy Rd
              </Name>
              <County>
                Miami-Dade
              </County>
              <Highway>
                I-75
              </Highway>
              <Direction>
                n
              </Direction>
              <Exit>
                2
              </Exit>
              <Offset_Type>
                rampTo
              </Offset_Type>
              <Cross_Street>
                NW 138 St/Graham Dairy Rd
              </Cross_Street>
              <Latitude>
                25898555
              </Latitude>
              <Longitude>
                -80341927
              </Longitude>
            </Primary_Location>
            <Secondary_Location/>
          </Event>
          <Event>
          <Event>
          </Events>
        </ObtainEventDataResult>
      </ObtainEventDataResponse>
    </soap:Body>
  </soap:Envelope>

```

Figure 18 – XML-Formatted Response from FL511 API
Showing Miami-Dade Event Information

The Java application was set to retrieve a list of all active events and floodgates for the entire state of Florida. SOAP libraries in Java handle the encapsulation of the request into the proper XML format and parse the XML response from the FL511 API into Java objects that define events or floodgate messages. The event and floodgate data are then stored in the USF SQL Server database, with spatial information being contained in a PostGIS spatial database. A sample of the information retrieved from the Java objects and output in human-readable form to the console is below.

```

-----
FLOOD GATE
ID: 2.4.87.000.0000.3 TimeStamp: 09/30/2010 09:10:29 A.M.
Center: Statewide
Highway: , Entity:
Message text: One SB and one NB Lane closed on SW 87 Ave/Galloway Rd between
SR-878 and Kendall Dr due to construction. Motorists should expect delays and
seek alternate route.
Message type: null
Severity: Unknown
-----

```

```

-----
EVENT
Event 259450 Time stamp 10/01/2010 11:41:33 AM
Primary Location: 25898555, -80341927
County: Miami-Dade
Cross Street: NW 138 St/Graham Dairy Rd
Direction: n
Exit: 2
Highway: I-75
offset type: rampTo
Name: NW 138 St/Graham Dairy Rd
Secondary Location: null, null
County: null
Cross Street: null
Direction: null
Exit: null
Highway: null
offset type: null
Name: null
Description: Planned construction in Miami-Dade on I-75 north ramp to Exit 2
NW 138 St/Graham Dairy Rd, off-ramp Left
Lane blocked. Last updated at 11:41:19AM.
Severity: unknown
Last update: 10/01/2010 11:41:19 AM
Center: District 6
type: current Reported at: 10/01/2010 12:15:07 AM
-----

```

The event contains a variety of information about the location of the event, including the latitude and longitude. Each event and floodgate message is given a unique numerical ID and stored in a database table. The county information is stored in a separate table so that multiple events/floodgates can be joined to the same county, and vice versa. The flood gate data contains no location information since the message is for the entire county. A variety of sample events and floodgate messages appear below.

EVENT EXAMPLES

Hillsborough 2010-09-20 02:00:17.713 2010-09-20 01:18:44.000 intermediate current
Planned construction in Hillsborough on I-275 north ramp to Exit 39A Kennedy Blvd, off-ramp right lane blocked. Last updated at 01:18:44AM.

Hillsborough 2010-09-20 08:32:26.030 2010-09-20 08:07:49.000 minor current
Traffic congestion in Hillsborough on I-275 south from Exit 52 Fletcher Ave to beyond Exit 39 Memorial Hwy. Last updated at 08:07:49AM.

Hillsborough 2010-09-20 08:32:26.077 2010-09-20 08:09:14.000 minor current

Traffic congestion in Hillsborough on I-75 south from before Exit 270 Bruce B Downs Blvd to beyond Exit 265 Fowler Av. Last updated at 08:09:14AM.

Hillsborough 2010-09-20 13:34:01.780 2010-09-20 13:15:22.000 intermediate current
Multi-vehicle crash in Orange on US-441 north at Sand Lake Rd, left lane blocked. Last updated at 01:15:22PM.

FLOODGATE EXAMPLES

2010-09-15 02:00:25.430 Hart Expressway Minor 2010-09-15 01:01:21.000 1 2.2.72.000.0011.1
Expect full closure of the Hart Bridge Monday through Thursday evenings from 7pm to 6am and Saturday morning at 6am thru Monday morning at 6am.

2010-09-10 18:05:00.867 SR 8 Minor 2010-09-10 17:51:31.000 0 2.2.72.000.0127.1 295
Southbound ramp to 10 Eastbound is CLOSED due to a vehicle crash.

2010-09-17 15:04:24.310 Unknown 2010-09-17 14:58:02.0001 2.4.87.000.0000.3
One SB and one NB lane closed on SW 87 Ave/Galloway Rd between SR-878 and Kendall Dr from August 6th to September 30th due to construction.
Motorists should expect delays and seek alternate route.

2010-09-16 11:03:03.083 I-95 Major 2010-09-16 10:36:53.000 0 2.4.93.000.0022.1
Northbound lanes of I-95 are closed at Boynton Beach Blvd due to a vehicle crash please seek an alternate route

2010-09-16 14:34:09.900 US 98/SR 30 Minor 2010-09-16 14:26:17.000 0 2.3.58.000.0165.1
On US 98 and SR 87 in Navaree, there is a roadblock on the westbound lanes of US 98. Proceed with caution.

The USF Java application queries the FL511 feed periodically to see if any of the current events have been resolved and removed from the feed. The frequency of updates can be adjusted to any frequency, from every 30 minutes to nearly real-time updates every 10 to 20 seconds. Once in the database, the new events are marked as active, and old events no longer in the feed are marked as inactive. Each event is also inserted as a point shape in the PostGIS spatial database for use with the path prediction algorithm. If predicted paths intersect with an active incident, the user will be sent a text message containing the details on the incident.

One limitation of the current FL511 system is that most detected congestion events related to real-time speeds occur on Interstates. For example, in the Tampa Bay area the only roads monitored by automated speed sensors are I-75, I-4, and I-275. Events and congestion occurring on arterial roadways must be manually entered into the FL511 system by control stations. To be of use to a broader segment of users, the system would eventually require the inclusion of information about local conditions on main streets in urban areas as well as the currently included state and federal highway information. While not the focus of this project, in the future the TRAC-IT system could effectively provide “probe” GPS speeds from users in order to feed a traffic speed estimation system, such as that maintained by INRIX.

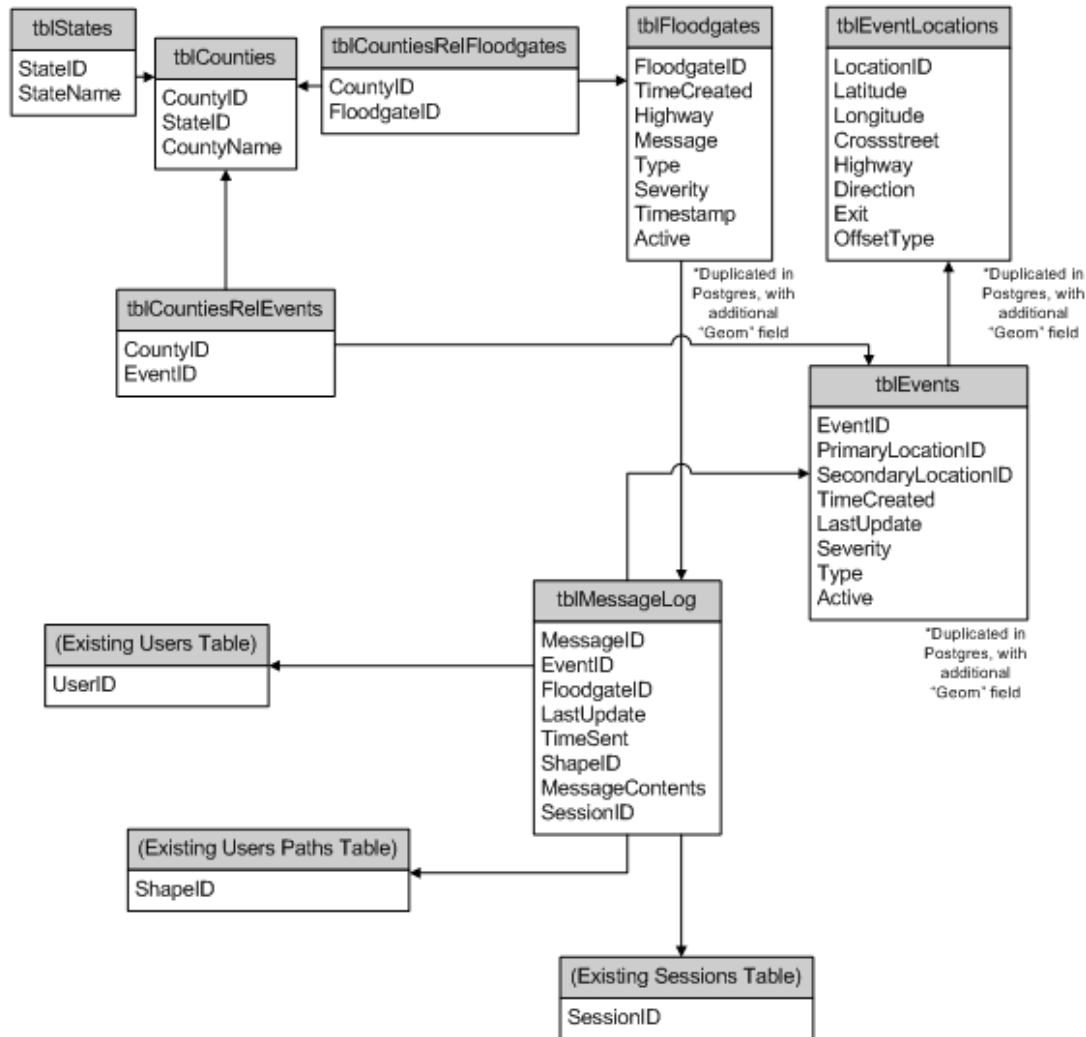


Figure 19 – Database Schema that Holds Data Retrieved from FL511 Feed

To create an early version of the software that would be developed to access the FDOT 511 API, the research team used tools available in the Netbeans integrated development environment (IDE). Once Netbeans has given the Web addresss to the WSDL file hosted on the FL511 webservice, it automatically creates the necessary Java code and objects to both send requests to and retrieve responses from the FL511 API. The following steps were used to create this webservice client using the Netbeans IDE:

1) A new webservice client is created within a Java application (Figure 20).

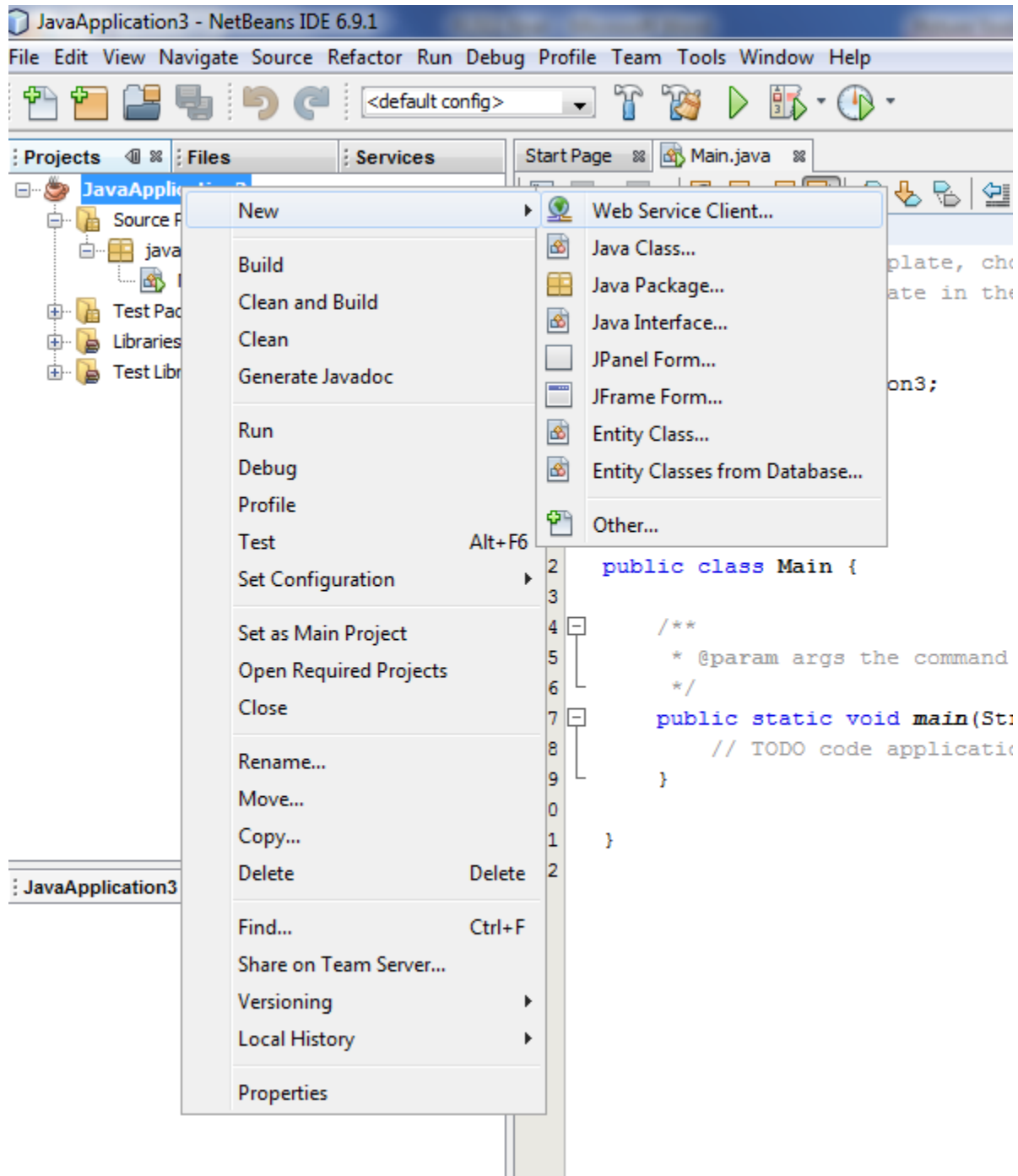
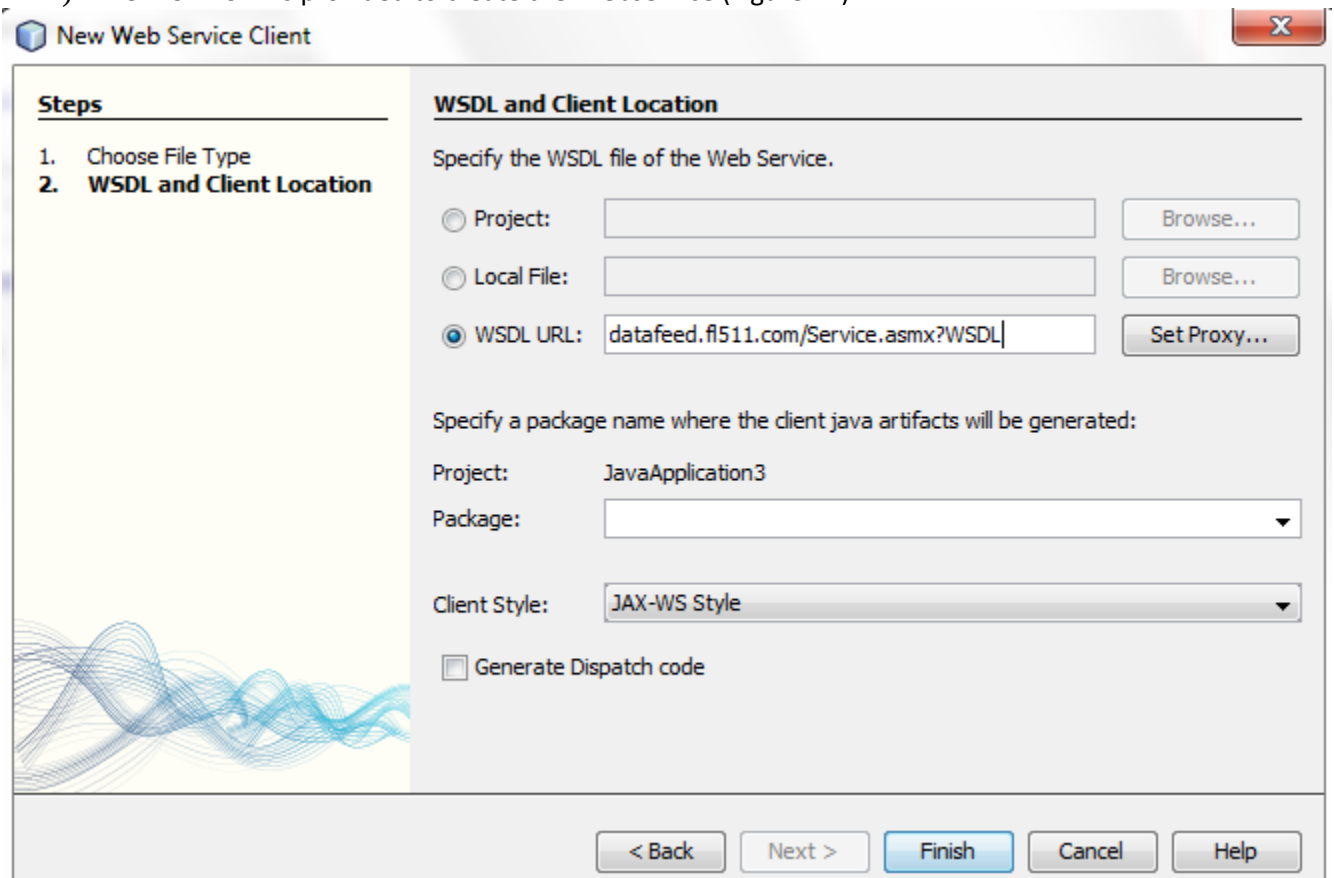


Figure 20 – Tools such as Netbeans Simplify Implementation of Software that Communicates with FL511 API

2) The WSDL URL is provided to create the webservice (Figure 21):



The image shows a NetBeans 'New Web Service Client' wizard dialog. On the left, a 'Steps' pane lists two steps: '1. Choose File Type' and '2. WSDL and Client Location', with the second step being the active one. The main area is titled 'WSDL and Client Location' and contains the following fields and controls:

- A heading: 'Specify the WSDL file of the Web Service.'
- Three radio buttons for WSDL source: 'Project:', 'Local File:', and 'WSDL URL:'. The 'WSDL URL:' option is selected.
- Input fields for 'Project:', 'Local File:', and 'WSDL URL:'. The 'WSDL URL:' field contains the text 'datafeed.fl511.com/Service.asmx?WSDL'.
- 'Browse...' buttons next to the 'Project:' and 'Local File:' fields.
- A 'Set Proxy...' button next to the 'WSDL URL:' field.
- A heading: 'Specify a package name where the client java artifacts will be generated:'
- A 'Project:' label followed by a text field containing 'JavaApplication3'.
- A 'Package:' label followed by an empty dropdown menu.
- A 'Client Style:' label followed by a dropdown menu showing 'JAX-WS Style'.
- A checkbox labeled 'Generate Dispatch code' which is currently unchecked.

At the bottom of the dialog are five buttons: '< Back', 'Next >', 'Finish' (highlighted in blue), 'Cancel', and 'Help'.

Figure 21 – Netbeans Wizard – The WSDL of the FL511 API is entered here

- 3) Once the webservice client wizard is completed, it generates Java source code containing the needed methods and objects to access the FL511 API (Figure 22).

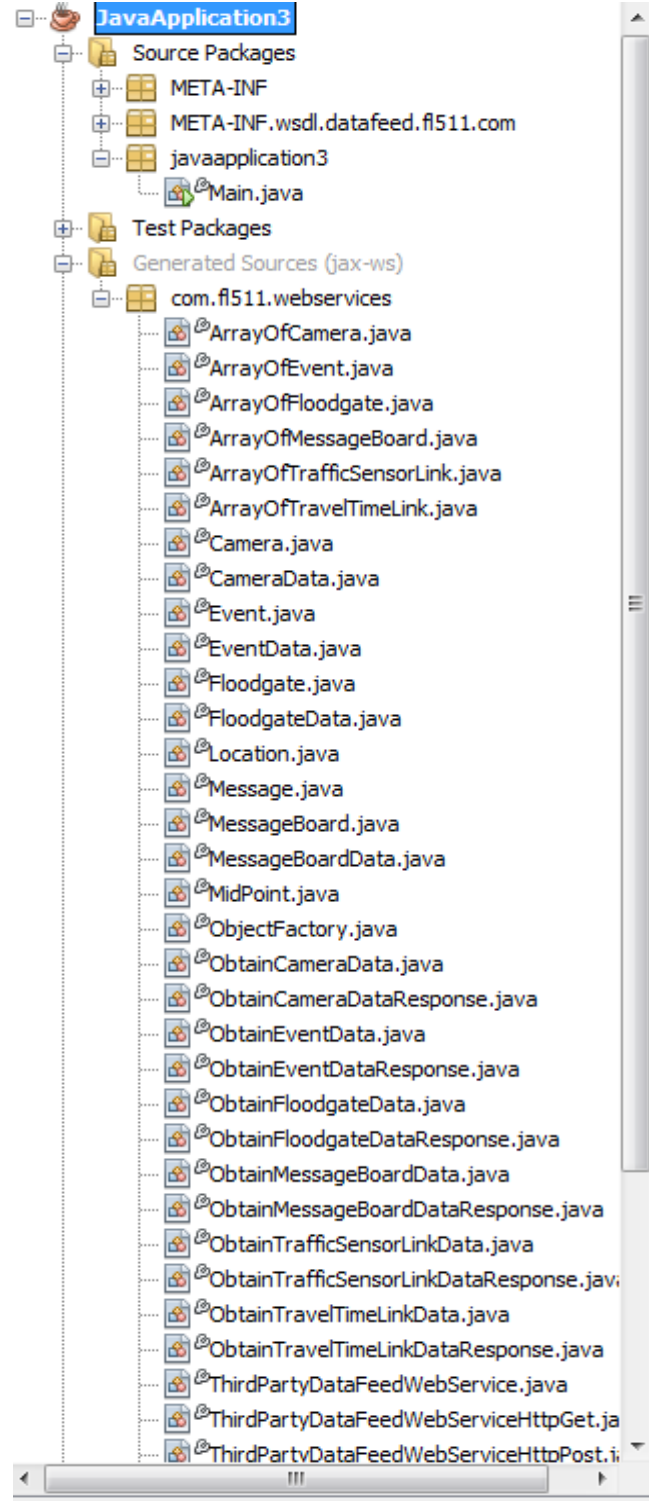


Figure 22 – Netbeans-Generated Java Classes for the FL511 API

STATISTICS FOR SAMPLE EVENT DATA RETRIEVED FROM THE FL511 API

Once the Java software client for the FL511 API was created, it was modified to retrieve and store information about events in a database. The data show several trends for data retrieved from the FL511 Feed from 9/13/2010 to 10/17/2010. First, as shown in Figure 23, the number of events for the top five counties is more than that for the rest of the counties combined. Also, spikes tend to occur in events during rush-hour traffic, as is shown in Figure 24 and Figure 25. Figure 26 and

Figure 27 show that events from all of Hillsborough County, and events from just the interstates I-75, I-275, and I-4 in Hillsborough County mirror each other closely, with the exception of rush-hour periods where some events are manually reported on non-interstate roads. Figure 30 shows that the numbers of events drops significantly on the weekends, being nearly as low on Saturday as on Sunday. These trends of information retrieved from the feed closely follow the information observed in the alerts which are issued by My Florida 511 subscription alerts and reported on in an earlier section of this report.

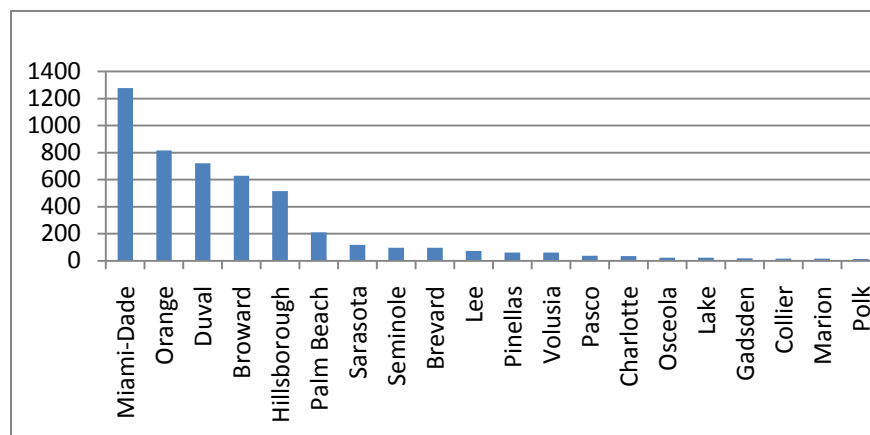


Figure 23 – Top 20 Counties with FL511 Events

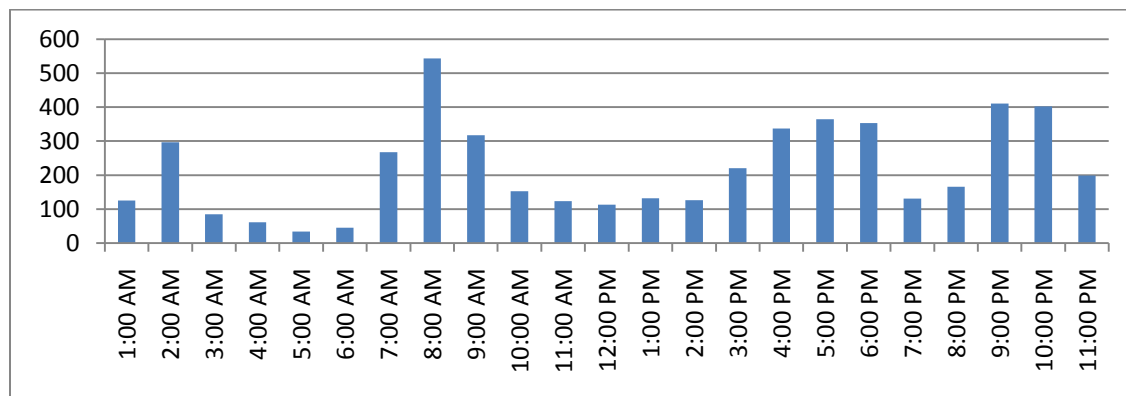


Figure 24 – Florida Events by Hour of Day

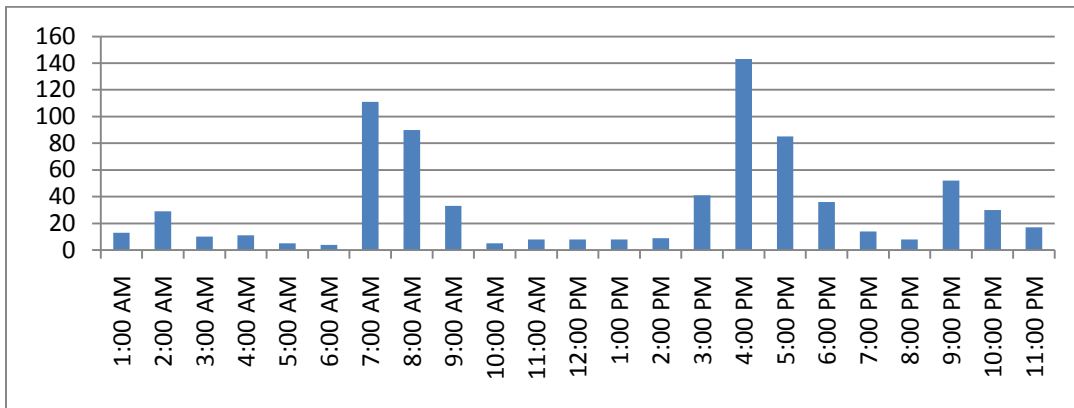


Figure 25 – Hillsborough County Events by Hour of Day

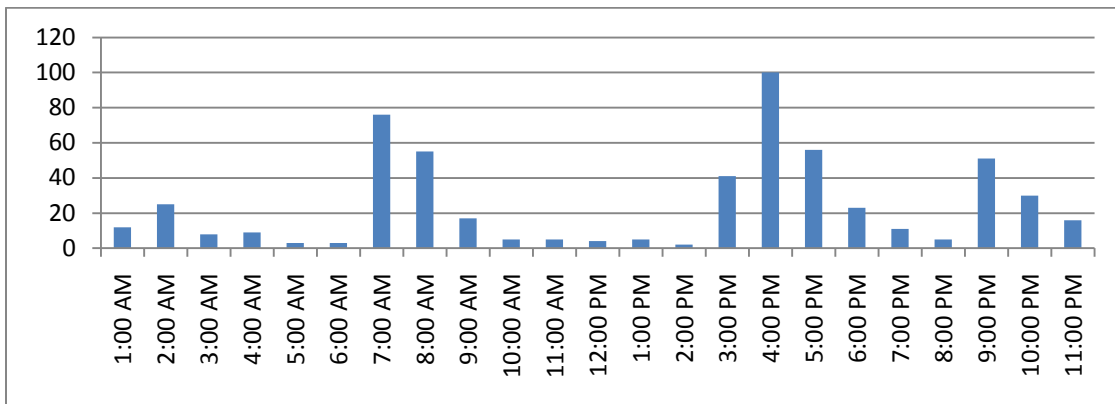


Figure 26 – Hillsborough County Interstate Events by Hour of Day

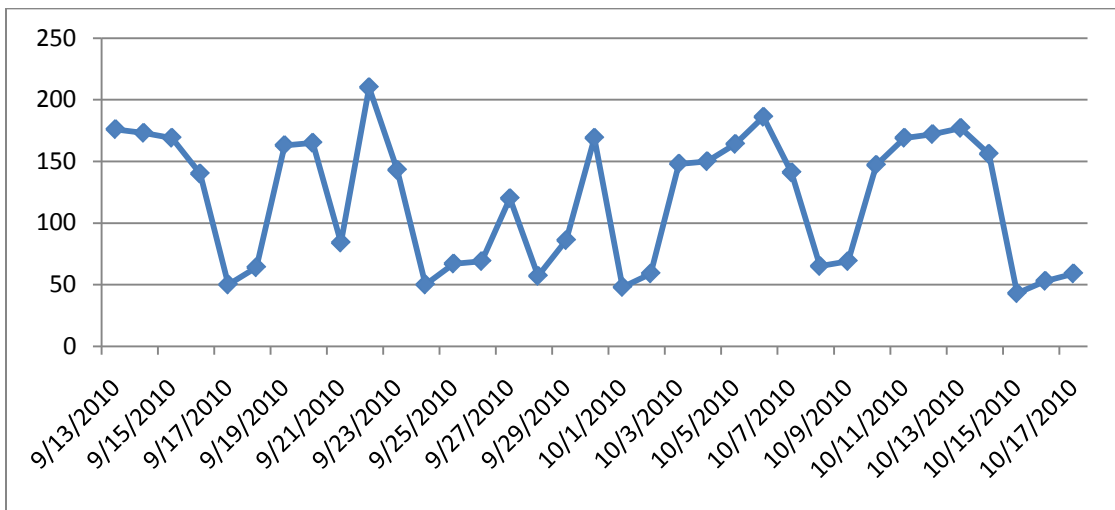


Figure 27 – Number of Daily Events by Date

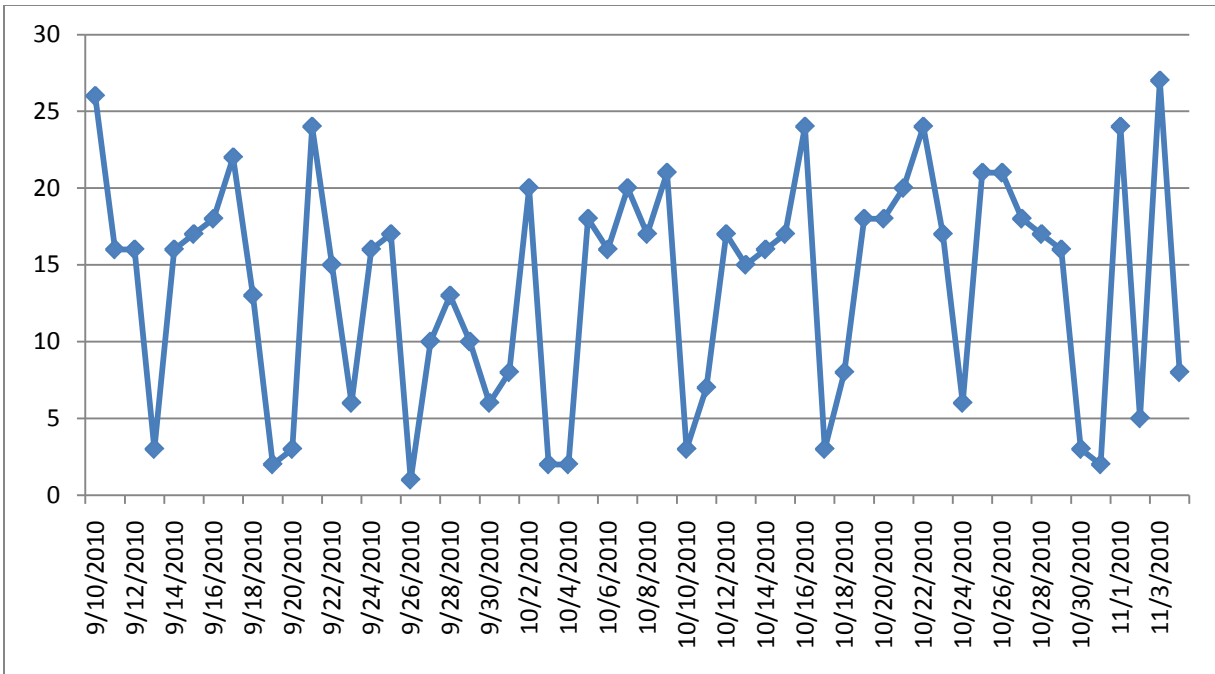


Figure 28 – Hillsborough County Events by Date

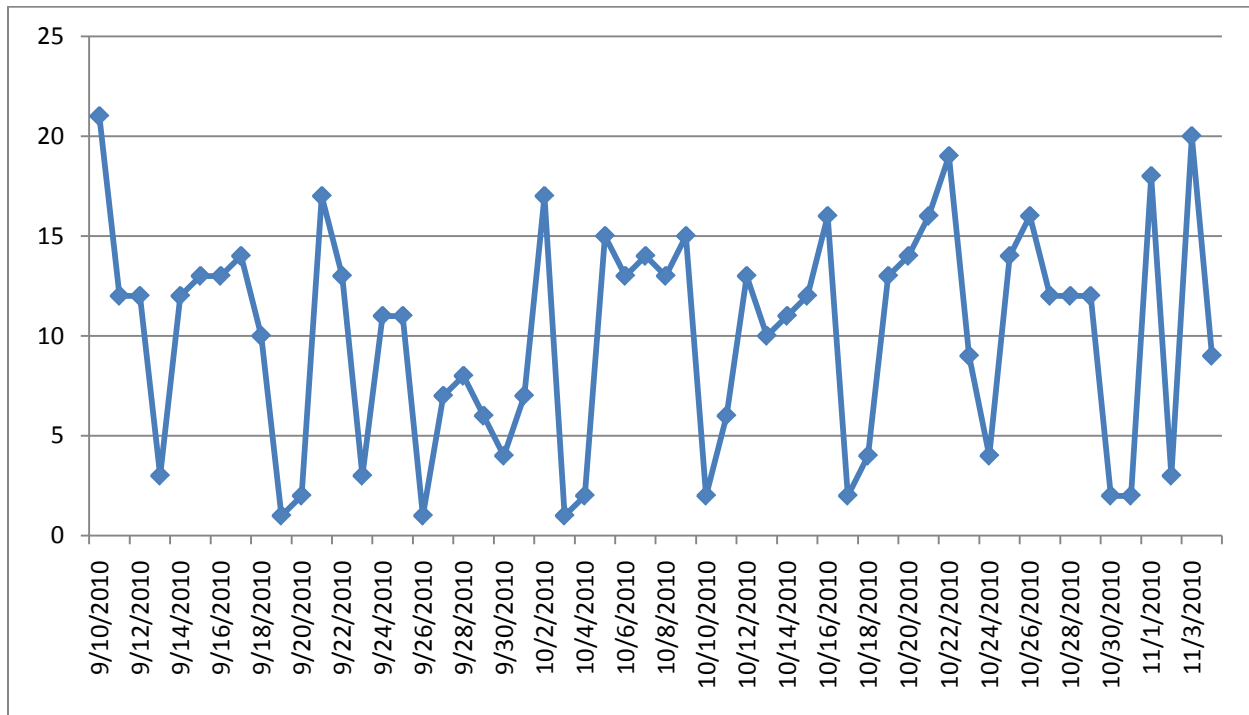


Figure 29 – Hillsborough County Interstate by Date

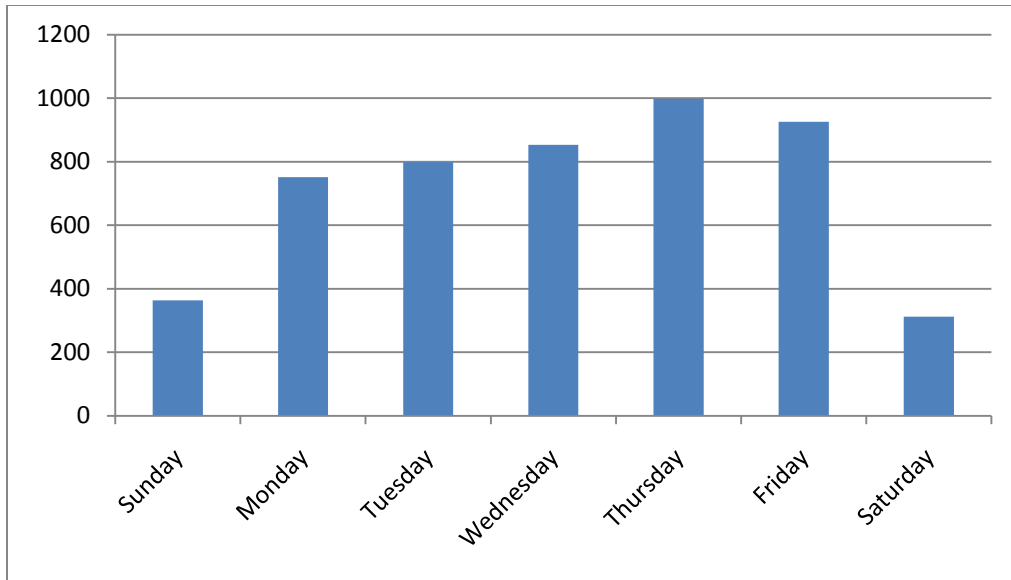


Figure 30 – Average Number of Events for Florida per Weekday

RETRIEVING REAL-TIME ESTIMATED TIME OF ARRIVAL DATA FOR PUBLIC TRANSPORTATION

Using transit is an option for coping with traffic delays, however, FL511 system does not include real-time transit information. It includes options to request transit information, then the user is directed to the local transit agency for further information.

The lack of transit data in the FL511 system meant that the research team needed to retrieve real-time transit data from another data source to provide a multimodal information service. Hillsborough Area Regional Transit (HART) real-time estimated time of arrival (ETA) information API was chosen to provide real-time estimates for when a bus will arrive at a nearby transit stop. The interaction between USF, HART's ETA API, and the FL511 API is shown in Figure 31. Further information about the HART ETA API can be found in a previous research report [9].

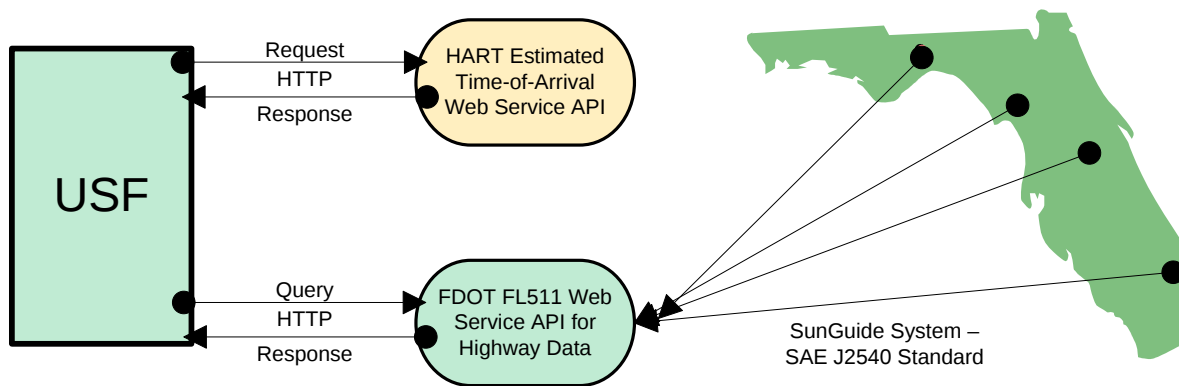


Figure 31 – System Architecture Showing USF, HART, and FL511 API Interaction

CHAPTER FOUR

FAST CLUSTERING OF GLOBAL NAVIGATION SATELLITE SYSTEM DATA AND TRIP SEGMENTATION

RELATIONSHIP BETWEEN CLUSTERING, PATH PREDICTION, AND TRAC-IT

As previously stated, past research projects sponsored by FDOT and NCTR developed the initial concept of real-time Path Prediction, which uses the real-time location of the user, along with their personal past travel behavior formatted as spatial paths, to predict the path they will travel in the immediate future [10]. TRAC-IT system features a TRAC-IT mobile phone application that can capture real-time user positions as frequently as once per second on any mode of transportation (e.g., car, public transportation, biking, walking), which provides Path Prediction with spatial travel behavior data for each user [6].

Past versions of Path Prediction simply generated a predicted geographic line based on the real-time position of the user, and the spatial intersection of that real-time point with lines in a spatial database representing historical travel behavior. For example, if a user is leaving his work location and typically travels to home, Path Prediction will use the real-time knowledge of his presence at work and his past trips traveling from work to home to predict this path as a possible immediate path that he may take. Past versions of Path Prediction relied solely on spatial queries based on the user's real-time GPS position and the intersection of this position with past paths to predict possible immediate paths.

Although this previous method of Path Prediction gave predictions while the user was moving (e.g., driving, walking, cycling, or via bus), it did not provide predictions when a user was not actively traveling (e.g., while he is at work). To determine future paths while the user is standing still for long periods of time, additional spatial-data mining of his travel behavior must be performed in order to identify precise locations that they frequently visit as well as the paths that connect these locations in his travel behavior. This research complements the original simple spatial capabilities of the original Path Prediction algorithm and opens the door for probabilistic methods of personalized path prediction in addition to simple spatial query-based approaches.

This section of the report discusses the enhancement of Path Prediction through the development of a Fast Clustering algorithm for GPS data that performs spatial-data mining in order to identify and process Points-of-Interest (POIs) that the user frequently visits, as well as trips, which are defined as the spatial and temporal properties that define movement from one POI to another. Later sections of this report discuss a basic probabilistic engine using a Naïve Bayes classifier (i.e., an independent feature model) to illustrate the capability to predict both the destination for the user's next trip in real-time while he is standing still for a long period of time at a location, as well as an estimated departure time from his current location. Destination and Departure Time predictions both utilize POIs produced by the Fast GPS Clustering Algorithm described in this section of the report.

BACKGROUND

The use of global navigation satellite systems (GNSS) in both civilian and military applications has exploded over the last decade on a worldwide basis, with the European Union and Russia launching their own GNSS to complement and compete with the widely popular U.S. NAVSTAR GPS. As a result, GNSS position data (e.g., latitude, longitude, time, speed) are calculated at rates up to once per second by millions of GNSS receivers and stored in databases to track people (e.g., GPS-enabled cell phones for consumers, ankle bracelets for convicted felons), vehicles (e.g., in-car navigation systems, commercial driver monitoring), goods (e.g., asset management systems, sensors for cargo monitoring), and animals (e.g., migration and homerange area recording). TRAC-IT is an excellent example of such an application for GPS-enabled mobile phones. TRAC-IT typically records about one GPS fix every four seconds while the user is moving, which leads to a large amount of GPS data for a single user, and much more for thousands of users.

This explosion of tracking data has led to a significant problem: how can these data be analyzed and transformed into information and then action on a near real-time basis in order to react to the events being monitored? One method to identify Points-of-Interest (POIs), or a location that a user visited as part of their daily travel (i.e., origin or destination of a trip), is to require the user to manually mark these locations. Locations can be marked through the use of the active data collection version of the TRAC-IT mobile application, which allows users to press a button on their phone and type in a written description (e.g., Walmart) each time they stop at a location. However, in previous field tests of TRAC-IT participants reported that remembering to indicate when they were starting or ending a trip at a location was the biggest challenge they faced [3]. Requesting that users label locations after data is collected is also possible, but this method requires even more effort from the user and also relies on the user's memory to identify locations, which has been shown to be unreliable for travel surveys in past studies [11] [12]. Requiring users to manually label their locations also places a significant burden on the participant, especially when performing longitudinal studies or operating location-based services which require a large amount of historical travel behavior data, and therefore this inconvenience must be offset with an incentive. This incentive is an additional cost to the survey or location-based service. If an automated method of identifying user POIs in extremely large datasets existed, it would allow passive tracking (i.e., no user interaction required) over long periods of time in a very cost-effective manner.

The following sections present the components and methodology of an unsupervised, Fast GPS Clustering algorithm that quickly translates a large collection of GNSS position data into a series of POIs defining spatial dimensions where a user has stopped for a significant amount of time and Trips defining the spatial and temporal properties for movement from one POI to another (Figure 32).

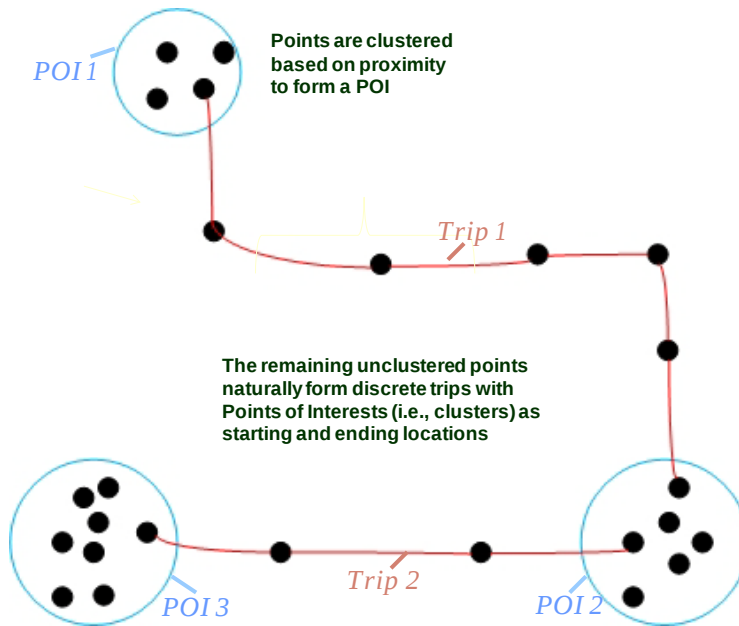


Figure 32 – POIs Identified Using the Fast Clustering Algorithm to Segment GPS Data into Trips

After a user's GPS data has been transformed into POIs and Trips, this information can be used to predict his future travel behavior based on personal travel history in terms of precise places and the number of times those places have been visited. The clustering algorithm presented in this report avoids the scalability pitfalls of hierarchical clustering algorithms and is specifically designed to handle large tracking datasets, where a single day's worth of data for one user can total 86,400 records. The algorithm has been implemented in a Java desktop application using PostGIS as the spatial database.

It is impractical to use traditional hierarchical clustering algorithms to obtain clusters due to the length-processing time, as the size of a set of GPS points can be significant. Several related algorithms have been developed that aim to reduce the time required to cluster large datasets. Two of them, in particular, which are discussed in the following section, have been designed for use in spatially-aware applications but differ from the approach created for this project.

RELATED WORKS

Qing Cao et al. present an agglomerative clustering algorithm that finds common trips rather than locations from a set of points [13]. In their algorithm, they use a specified number of events to indicate the nature of an origin or destination. The operator of the GPS device is responsible for manually indicating the start and end of a trip. They then use a grid based agglomerative clustering approach which compares trips for similarities using cells of the grid.

BIRCH is a clustering algorithm that performs well on large datasets and under memory limitations [14]. BIRCH scans the data and is able to quickly cluster it based on the tightness between points of a cluster and the centroid of a cluster. It is able to discard noise in a dataset by maintaining clusters only in which the points are densely distributed. Like our algorithm, BIRCH uses a balanced tree. In BIRCH, the balanced tree is used to compactly represent the characteristics of a cluster. However, on the first pass of the algorithm, the resulting clusters may still have some outliers or noise. Hence, depending on the accuracy required by the user, BIRCH might require multiple passes to disqualify outliers and transfer points between clusters to

achieve the desired resolution. YuFang and Zhongshi use a modified k-means clustering algorithm to track the density of urban populations where individuals are tracked by mobile phones [15]. To find the initial cluster centers, they found the maximum distance between all of the points in the dataset and divide the magnitude by k . The algorithm then tries to minimize the error within a cluster by continuously looping through the dataset until they get the best possible set of clusters. However, this algorithm performs only moderately better in terms of running time than standard k-means clustering.

THE FAST CLUSTERING APPROACH

The criterion for clustering is based on a buddy system where individuals (i.e., nodes) become friends (i.e., clusters of nodes) based on distance and transitivity. Fast GPS Clustering performs significantly faster than traditional hierarchical clustering since the number of comparisons of nodes for possible clustering merging that are performed through the algorithm are significantly reduced. These concepts are illustrated in Figure 33.

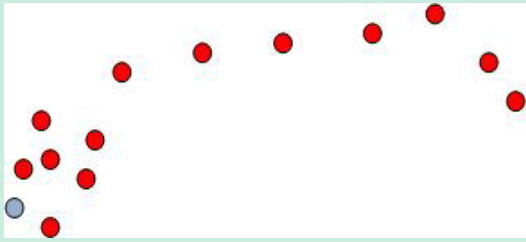
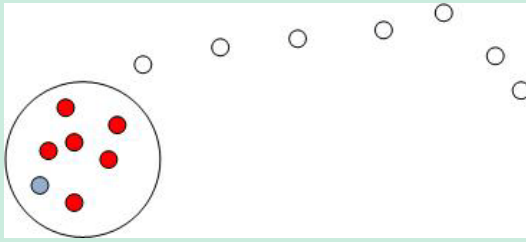
Traditional Hierarchical Clustering	Fast GPS Clustering
	
In the traditional hierarchical clustering algorithm it is necessary to compare a single point (blue) to every other point (red) in the dataset for each iteration of the algorithm.	In the <i>Fast GPS Clustering</i> algorithm a given point is only compared to points that are within its neighborhood of a given radius $r > 0$.

Figure 33 - Fast GPS Clustering Improves Performance over Traditional Hierarchical Clustering by Only Comparing Each Point to Others within its "Neighborhood"

NOTATION

The research team first introduces some notation to help explain the design of the Fast GPS Clustering approach. The tree rooted at a node v is denoted by T_v . When a node or data are inserted into T_v , the root of this tree might change, so T'_v denotes the modified tree, where v' is the node that is now in the former position of v . As most of the algorithms are recursive, T_v describes the tree rooted at some node v , but v may have a parent and thus reside in some larger tree. This larger tree is noted by $\hat{T}_v$. For the height and number of nodes in T_v the notation $h(T_v)$ and $n(T_v)$ are used, respectively.

The algorithm is given a set of points for which it then assigns an enumerated index ($1 \dots n$) to each element. As in the case of hierarchical clustering, the distances are computed between each pair of elements, which takes $O(n^2)$ time, where n is the number of elements.

DISTANCES BETWEEN ELEMENTS

For each pair of elements and the distances between them, a unique triple (i.e., a combination of three variables) (i, j, r) was assigned, where i is the index of the first element, j is the index of the second element, and r is the distance (a positive real number) between them. These triples are stored in an AVL tree using the computed distance as the key. An AVL tree (named after its inventors initials) is a self-balancing binary search tree where the difference in heights of the two children subtrees of a node is at most one. Due to the properties of an AVL tree, the triple can quickly be removed with the next smallest distance while maintaining an ordered and balanced set (Table 3).

Table 3 – Clustering Algorithm Example

Distance (m)	Element i index	Element j index
0.42465015713986803	1	11
.	.	.
.	.	.
.	.	.
8.505083873505768	314	651
.	.	.
.	.	.
.	.	.
32.59953978573408	864	n-1
.	.	.
.	.	.
.	.	.
39.990662609170734	867	879

STRUCTURE OF A CLUSTER

As in agglomerative hierarchical clustering (i.e., tree-based clustering, where a tree is gradually formed by grouping related nodes in a hierarchy), the algorithm starts with n disjoint elements. Initially, each element is the root of its own AVL tree. To this AVL tree, a map was assigned and maintained, denoted by M_k , of the indices of all elements that are within the tree.

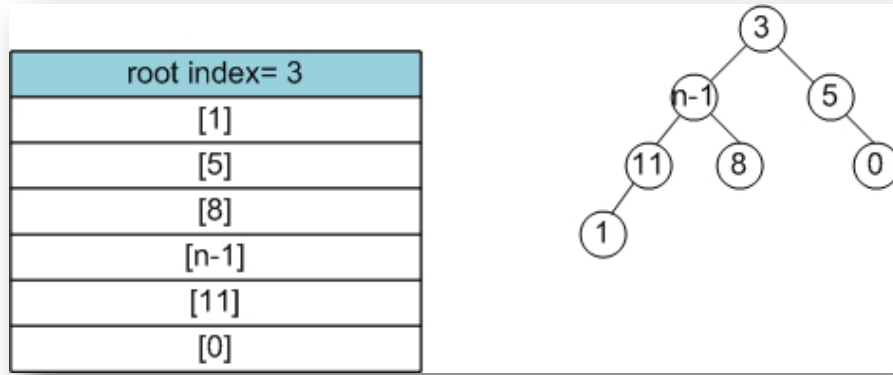


Figure 34 - A Cluster is Assigned a Map of its Elements
for Quick Access via an Enumerated List

Hence, any node of a particular cluster can be quickly accessed by its index using the same enumerated list of elements built earlier. This method of quickly accessing a cluster by its index is called map, with the AVL tree rooted at the k th element, a cluster denoted by C_k , $k = 1 \dots n$. Finally, T_k denotes the AVL tree of cluster C_k . To further clarify, the indices in $k = 1 \dots n$ are used as the label of the node rather than v when referring to the list of triples (i, j, k) .

To ensure that disjoint pairs of clusters are being merged, the cluster map is assigned a root index, which is the index of the element, the root of the cluster (i.e., AVL tree). Thus, for the triple (i, j, k) the i th and j th elements are shown in the enumerated list but refer to the root of T_i i and T_j j respectively. This ensures that the algorithm does not create loops by performing unions within a single cluster or attempting to merge only a portion of the tree (i.e., a subtree rooted at the element) with another cluster. This method does not, of course, exclude the situation for singleton clusters where the root index would refer to the single node itself. The root index also ensures that unions are performed only between disjoint clusters. This is easy to verify by checking the root index of each cluster's cluster map (i.e., if these values refer to the same index then they are necessarily in the same cluster). Note that no time is wasted traversing a tree from the child to the root as the root index allows the algorithm to retrieve the root of the tree indexed in the enumerated list of elements in $O(1)$.

Now introduced is the notation C_{i+j} to denote union of C_i and C_j and a similar notation T_{i+j} for the union of their trees. Unlike hierarchical clustering, there is no need to update a matrix every time C_i and C_j are merged. In this version of clustering, updating a matrix is an extraneous operation because a list of shortest distances already exists. The distance tree maintains a list of shortest distances between elements, but these elements are nodes of clusters. Thus, the cluster map of C_{i+j} is updated by adding the indices of C_j to cluster C_i rather than updating a table or a matrix. To limit the running time, the "shorter" cluster, in terms of AVL tree height, is merged into the "taller" cluster. In other words, the responsibility of knowing what clusters the elements belong to is a property of the clusters themselves, rather than a cohesion table or matrix. After merging T_i and T_j , the cluster map is then updated and the root index for each element in

C_{i+j} is updated by simply changing their cluster map reference. Thus, if any of the elements in C_{i+j} are either indexed by i or j in a later distance triple (i, j, k) , the algorithm will know to retrieve the root of T in which the element resides, rather than the element itself.

$\max(T_v)$ and $\min(T_v)$ denote the maximum and minimum elements respectively of T_v . Although all of the clusters are pairwise disjoint, this does not guarantee a convenient relationship between them. In particular, some of the elements in C_i might interleave keys in cluster C_j . This might occur after $h_i > 2$ or $h_j > 2$. The nodes of which C_k consists are time-stamped GPS points, while time is used as the ordering criterion of the AVL tree. Ordering the clusters by time gives us a chronological order of visitation of points, which provide information on when the user visited a particular location.

ALGORITHM COMPONENTS

To denote the time of a node (i.e., GPS point) v , t_v is used. Three algorithms that can be used to unite two disjoint AVL trees T_i and T_j have been developed. The entire merging scheme is designed so that the most ideal merge between T_i and T_j occurs when $t_{\max(T_i)} < t_{\min(T_j)}$ and $h_j \leq h_i$. The symmetric case where $h_i < h_j$ can also exist. If either T_i or T_j fail these criteria, then some nodes of T_j will interleave some nodes of T_i . In other words, $t_{\min(T_i)} < t_{\min(T_j)} < t_{\max(T_j)} < t_{\max(T_i)}$. Thus, if the interleaved portion of T_j is removed, T_i and T_j can be merged in $O(\log(n(T_j) + n(T_i)))$.

FRAGMENTATION ALGORITHM

The fragmentation algorithm separates a tree by a key κ into two trees $T_{v\alpha}$ where $t_v > \kappa$ for $v \in T_{v\alpha}$ and T'_v where $t_v < \kappa$ for $v \in T'_v$. It simply partitions the tree T_j into trees. The fragmentation always assumes that $h_j \leq h_i$. κ is assigned the value of the time $t_{\max(T_i)}$ or $t_{\min(T_i)}$. The actual value of κ will depend upon the relationship between T_j and T_i . To fragment T_j , all nodes $v \in T_j$ are removed such that the $t_v > \kappa$. To accomplish this task, the algorithm recursively searches the tree as in a normal binary tree search, assuming there will be a node such that its time is κ . However, since $T_i \cap T_j = \emptyset$ (and because two separate points cannot be visited at the same time) it will never find such a node. Thus, it reaches a null node (i.e., the search returned no satisfactory element) with the previously visited node $v_0 \in T_j$ having time $t_{v_0} < \kappa$ or $t_{v_0} > \kappa$. Since the algorithm recursively searched for this nonexistent node, it simply starts popping (i.e., removing) the stack elements once it reaches the null node.

All of the nodes $v \in T_j$ can easily be collected such that the $t_v < \kappa$ or $t_v > \kappa$, by storing them in a two-element array U where $t_{U[0]} < \kappa$ and $t_{U[1]} > \kappa$. Once v_0 is reached for each pop of the stack, the algorithm inspects each node v to see whether $t_v < \kappa$ or $t_v > \kappa$. If $t_v > \kappa$, simply make v a child of the node (or the first such node) in $U[1]$.

Function fragment(Avl tree T , AvlNode κ)

Input: an avl tree T , key κ
Output: two roots of two avl trees

begin

- if** $T = \text{null}$ **then**
 - return** AvlNode[] {null, null};
- direction := $\kappa.\text{data} < T \rightarrow \text{root}.\text{data} ? \text{left} : \text{right}$;
- AvlNode[] nodes = fragment (κ , subtree in direction);
- set child of root to nodes[direction];
- if** $T \rightarrow \text{root}.\text{data} < \kappa.\text{data}$ **then**
 - $T \rightarrow \text{root} \leftarrow \text{restoreBalance}(T \rightarrow \text{root}, \text{nodes}[0]);$
- else**
 - $T \rightarrow \text{root} \leftarrow \text{restoreBalance}(T \rightarrow \text{root}, \text{nodes}[1]);$
- if** $T \rightarrow \text{root}.\text{data} < \kappa.\text{data}$ **then**
 - return** new AvlNode[] { $T \rightarrow \text{root}, \text{nodes}[0]$ };
- else**
 - return** new AvlNode[] { $T \rightarrow \text{root}, \text{nodes}[1]$ };

As there are n nodes at the subtree rooted at v , the sum of nodes in its left and right subtrees must be $n-1$. If the size of the larger subtree is k and the size of the smaller subtree is c , then the running time for the restore balance operation would be given by $\log(n-1-k+c)$ where $c \leq k$. At a single node, zero operations are performed so that $\theta(1)=1$. Hence, the time spent at v is given by $T(n) = T(k) + 2\log(n-1-k+c)$. The coefficient 2 in the last term exists because the root has to be inserted back into the merged halves.

In the worst case, the entire tree is being re-assembled (i.e., no nodes have data greater than κ) by performing a series of unions on the left and right subtrees which takes $O(\log n)$. In the best case, only inserting the root v back into one of its subtrees; the balance restoration in the worst case is $\Omega(\log n)$, i.e $1/2 \log(n) = \log(\sqrt{n}) \leq \log(n/2)$, when $n > 4$. Hence, by the Master Theorem, the running time in the best case using recursion is given by:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ T(n/2) + \Theta(\log(n)) & \text{if } n > 1 \end{cases}$$

Function restoreBalance(AvlNode v_1 , AvlNode v_2)

Input: (node v_i , node $v_{i'child}$, direction)
Output: a node
begin
 $v_{opChild} \leftarrow v_i \rightarrow \text{child in opposite direction};$
 $T' := \text{null avl tree};$
if $v_{opChild} = \text{null}$ **and** $v_{i'child} = \text{null}$ **then**
 $v_i \rightarrow \text{BalanceFactor} \leftarrow \text{EH};$
return v_i ;
else if $v_{opChild} \neq \text{null}$ **and** $v_{i'child} \neq \text{null}$ **then**
if $h_{opChild} > h_{i'child}$ **then**
 $\quad \text{merge } v_{i'child} \text{ into } v_{opChild}$
else
 $\quad \text{merge } v_{opChild} \text{ into } v_{i'child}$
return $T' \rightarrow \text{root};$
else
 $v_{opChild} = \text{null} \oplus v_{i'child} = \text{null};$
make tree T' out of whichever one is not null;
add $v_i \rightarrow \text{data}$ to T' ;
return $T' \rightarrow \text{root};$

On termination, there are $m_1 \in \lfloor \log n(T_j) \rfloor$ nodes that satisfied $t_v < \kappa$ and $m_2 \in \lfloor \log n(T_j) \rfloor$ that satisfied $t_v > \kappa$, $m_1 + m_2 = \lfloor \log n(T_j) \rfloor$.

The value of κ depends on the relationship among the nodes of T_j to T_i . Assume $h_j \leq h_i$. Then,

$$\kappa = \begin{cases} t_{i_{max}} & \text{if } t_{i_{min}} < t_{j_{min}} < t_{i_{max}} \\ t_{i_{min}} & \text{if } t_{i_{min}} < t_{j_{max}} < t_{i_{max}} \\ \text{null} & \text{otherwise} \end{cases}$$

Assume $\kappa = t_{i_{max}}$, then $T_{j_{\alpha}} \cap T_i = \emptyset$. Thus, $T_{j_{\alpha}}$ and T_i can be merged to get a new tree $T_{j_{\alpha}+i}$. If $\kappa = t_{i_{min}}$, then $T_{j'} \cap T_i = \emptyset$ and $T_{j'}$ and T_i must be merged to get a new tree $T_{j'+i}$ by deleting the largest element in $T_{j'}$ and traversing the left children and comparing their balance factor to the right children. For the remaining fragments, either $T_{j'}$ or $T_{j_{\alpha}}$, a splicing algorithm is used to merge them with either $T_{j_{\alpha}+i}$ or $T_{j'+i}$, respectively. In both cases, the splicing algorithm works the same.

SPLICING TWO AMBIGUOUS TREES

The splicing algorithm is simply a modification of the fragmentation algorithm, where $U[0]$ or $U[1]$ is initialized to T_i , depending on κ . The tree produced by the splicing algorithm is called T_i , T_{i^*} . Let $n_0(T_i) = n(T_i)$ before splicing. If $\kappa = \max(T_i)$, then the running time is

$O\left(\sum_{i=0}^{m_1-1} \log(n_0(T_i) + i) + \log(m_2(m_2 + 1)/2)\right)$. Similarly, if $\kappa = \min(T_i)$, then the running time is $O\left(\sum_{i=0}^{m_2-1} \log(n_0(T_i) + i) + \log(m_1(m_1 + 1)/2)\right)$.

MERGING TWO TREES

If $\kappa = \max(T_i)$, then merging T_i and T_j can be done by (1) splicing T_j by κ and (2) performing a union, $T_j \cup T_{i^*} = T_{i^*+j}$ to get the final result T_{i^*} . For simplicity, call $m = (m_2(m_2 + 1)/2)$. Thus, the running time to merge T_i and T_j is $O\left(\log\left(m(m + n_0(T_i) + m_1 - 1) \sum_{i=0}^{m_1-1} (n_0(T_i) + i)\right)\right)$.

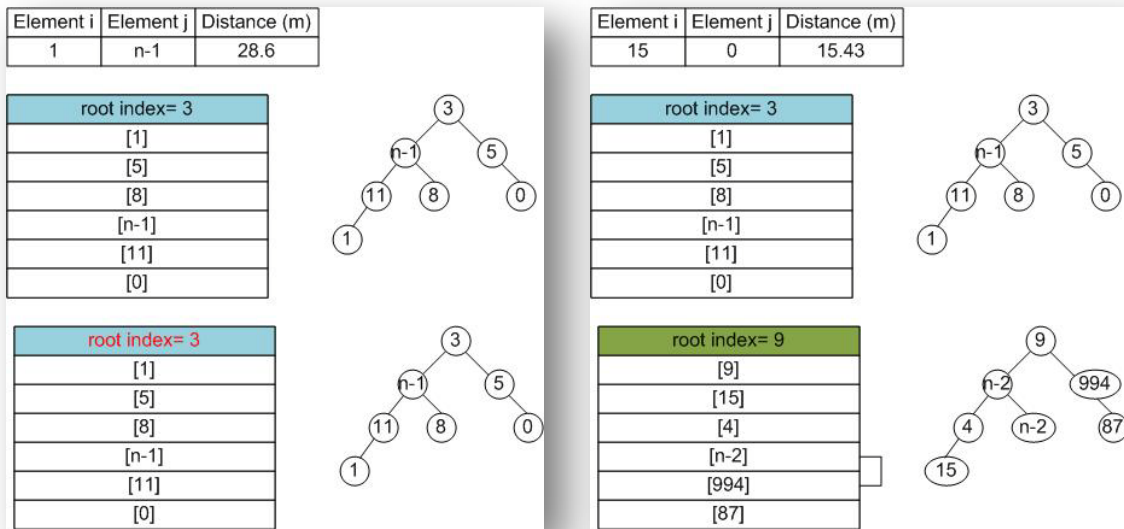


Figure 35 - Fast GPS Clustering References the Element at the Root Index to Properly Merge Two Disjoint Clusters

FINDING THE FINAL SET OF CLUSTERS

To find the set of clusters, the list of triples (i, j, k) of the distances tree T_d is simply exhausted by deleting the smallest element until T_d is empty. For each triple (i, j, k) , each disjoint cluster C_i and C_j is merged. There will be a total of $n(T_d)$ iterations. However, because only disjoint C_i and C_j are merging, the number of merges is dependent upon the set of pairs of clusters $S = \{(C_i, C_j) | C_i \cap C_j = \emptyset \text{ and } C_{i+j} \cap C_{j+1} = \emptyset\}$. Noting that any two clusters in this set must be disjoint for any given iteration (i, j, k) , including merged clusters, the final running time is thus dependent on the set S . Denote the size of the cluster map, a linked list, M_k by n_{m_k} . M_j is added to M_i . Let

$$\begin{aligned}\zeta &= \sum_{\forall C_i, C_j \in S} (\log(m(m + n_0(T_i) + m_1 - 1) \\ &+ \sum_{i=0}^{m_1-1} (n_0(T_i) + i) \\ &+ n_{M_j})\end{aligned}$$

Where m and m_1 are dependent upon the triple (i, j, k) . Then, ζ denotes the time to merge all disjoint clusters including the time to transfer the indices M_j to M_i . Thus, the time to find the final set of clusters is

$$O\left(\zeta + \sum_{i=0}^{\zeta-1} \log(n(T_d) - i)\right).$$

This final set of clustering identified by the clustering algorithm represents the spatial area, defined by GPS fixes, at which a user spent a significant amount of time. Each of these clusters, therefore, represents a POI for that particular user.

TRIP SEGMENTATION

Once POIs are defined for a set of GPS data, these data can then be further analyzed to produce trips, which are defined as the spatial and temporal characteristics of movement from one location to another. Starting with the first POI (i.e., POI_1) that occurs in a chronologically-ordered set of GPS data, the GPS fix that occurred most recently in time (i.e., the “youngest” fix) is identified. This most recent GPS fix in POI_1 , shown in Figure 36 is the point within POI_1 that is connected to a line, represents the first GPS fix in a series of points that defines a trip, $Trip_1$, leaving POI_1 . The subsequent GPS fixes are connected in chronological order until a point within another POI (i.e., POI_2), is found, which the terminating GPS fix is for $Trip_1$. $Trip_1$ is then defined as starting at POI_1 and ending at POI_2 , with a start time equal to the most recent GPS first in POI_1 , and with an end time equal to the earliest (i.e. “oldest”) GPS fix in POI_2 . This process is repeated beginning with the most recent GPS fix in POI_2 until the complete set of GPS data is processed. Figure 36 shows a total of three POIs, and two trips connecting those POIs in chronological order.

FAST CLUSTERING AND TRIP SEGMENTATION RESULTS

The Fast Clustering algorithm is able to produce a set of POIs (i.e., clusters), and the Trip Segmentation algorithm is then able to split GPS data into discrete trips traveling from one POI to another.

Figure 36 shows the “before” image of raw GPS data that was connected with lines, as well as the “after” image showing the results of Fast GPS Clustering and Trip Segmentation algorithms. Two POIs (i.e., POI A and POI B), were identified as well as three trips (i.e., Trip 1, Trip 2, and Trip 3). The end result in the database for this dataset would be two new records in a Location table, one for each POI, as well as three new trips in a Trips table. The spatial attributes of the data are also saved in a PostGIS spatial database, with POIs being represented as polygons and trips as polylines.

Figure 36 shows the performance increase of Fast GPS Clustering over traditional hierarchical clustering for various dataset sizes. These results were obtained with both algorithms implemented within a Java application and utilizing PostGIS as a spatial database.

One byproduct of Fast Clustering is the detection of smaller “pseudo-POIs,” as can be seen in the vicinity of POI B in Figure 36. These smaller clusters represent a dense cluster of GPS data that does not actually represent a meaningful destination to the user. Instead, these pseudo-POIs occur for events such as GPS noise when the user is standing still, or when the user pauses during a trip (e.g., stopped at a traffic light).

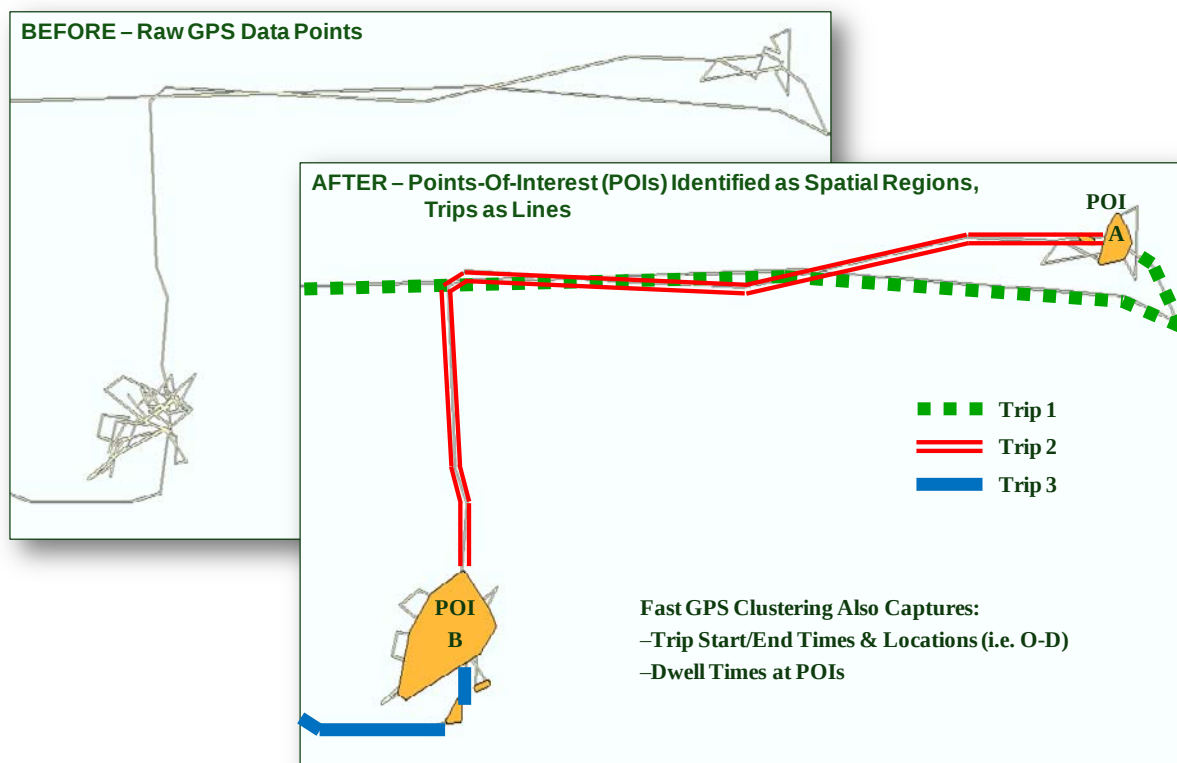


Figure 36 - Raw GPS Data Before and After Processing for Two POIs

Table 4 - Fast GPS Clustering Performs Significantly Faster than Traditional Hierarchical Clustering

NUMBER OF POINTS IN SET	COMPLETION TIME (HOURS: MINUTES) INCLUDING DATABASE INSERTION		TIME SAVINGS WITH FAST GPS CLUSTERING (MINUTES)
	Traditional hierarchical clustering	Fast GPS clustering	
275	:15	:12	3
790	1:42	:44	58
939	2:27	:59	88
1109	3:46	:37	143

Figure 37 shows a pedestrian trip where the user paused at several points in the path, which generated pseudo-POIs. These POIs can be filtered out of the Trip Segmentation algorithm to avoid falsely marking a

traveler's destination as a pseudo-POI and therefore falsely incorrectly increasing the number of trips for a user. Pseudo-POI filtering can be based on the dwell time at the POI (e.g., a true POI requires the user to have stayed in the same position for over 15 minutes) as well as the overall diameter of the POI (e.g., GPS noise tends to create large POIs when a user is standing still over a long period of time). While pseudo-POIs are a by-product of the intended detection of true POI destinations that have meaning to users, pseudo-POIs have an alternate use that can be very valuable: detecting areas of travel delay and congestion.



Figure 37 – Sample of Pseudo-POIs

Future work should examine using pseudo-POIs from many users to identify areas of frequent congestion or other causes of traveler delay. For example, if many pseudo-POIs appear at a particular traffic light, the signal timing of that light may need to be adjusted. Since TRAC-IT has knowledge of the user's path, the system could actually be used to detect areas where travelers face a long delay before making a right turn, or left turn, which could signal the need for additional lanes on a road. Pseudo-POIs could also be used for generating predictive traffic delays based on historical speeds and delays on roads, so that travelers could be warned that a traffic jam on a road appears imminent and be advised to take another route or delay their trip if possible. Drivers that could avoid problem routes would avoid contributing to the traffic delay.

Pseudo-POIs could also be used to identify high-value advertising areas where travelers tend to spend a lot of time without much movement, thus expanding the amount of time that a traveler would have to read advertisements. Since Fast GPS Clustering could be used on any mode of transportation, pseudo-POIs could locate high-value bus stops, road crossings, outdoor malls, or other pedestrian areas where travelers tend to dwell for longer periods of time. For vehicular traffic, high-value areas may be identified as lengthy stop light cycles, areas of heavy congestion, toll-road entry lines, etc. For digital advertising solutions where dynamic content is possible, the type and length of advertisements could be adjusted in order to present the best content to the targeted demographics.

FAST CLUSTERING CONCLUSION

In this section, an unsupervised, Fast Clustering algorithm for tracking data collected using GNSS receivers was described. It demonstrated the feasibility of the fast GNSS clustering algorithm using data collected using a mobile application for GPS-enabled cell phones. The fast GNSS algorithm takes raw GNSS data as input and outputs a series of POIs at which a user spent a significant amount of time, as well as Trips, which define the spatial and temporal properties for a user's travel from one POI to another. Since the fast GNSS algorithm allows quick batch processing of tracking data, large tracking datasets can be analyzed on a frequent and near real-time basis in order to take action on recent tracking events. This algorithm, therefore, supports new location-aware applications that are based on knowledge quickly extracted from large collections of tracking data without requiring a human to manually label any data. Pseudo-POIs, which signal areas of traffic congestion or delay, also have great potential to provide numerous other services such as informing traffic signal re-timing, road enhancements, predictive traffic alerts based on historical travel patterns, and high-value advertising placements.

CHAPTER FIVE

PREDICTING USER DESTINATIONS AND TIMES OF DEPARTURES

USING TRIPS, POINTS-OF-INTEREST, AND NAÏVE BAYES

A problem with many existing subscription-based services for traffic alerts, as well as car and mobile-phone-based navigation devices that include traffic alerts, is that the user must manually enter the location he is visiting and the location where he is headed in order to get the desired traffic alerts or location-based information. Subscriptions require the user to choose which roads he plans to traveling on, and then traffic alerts are always sent about those roads. The user must manually edit his subscription if he wants to expand or reduce the scope of alerts. Many navigation systems in cars require users to manually enter their immediate planned destination every time they travel in order to provide alerts for their planned route. Subscriptions are difficult to maintain for a large number of destinations, and for most subscription services the user will receive a large number of alerts related to destinations that he is not currently near. Manually entering locations into navigation systems can take a significant amount of time and is not expected to be done on regular trips for which the user knows the path to their destination. Intelligent algorithms can help predict where the user is headed, based on his real-time location and historical travel behavior to avoid the limitations of subscription-based traffic alerts and existing navigation systems. Once a user's POIs and trips have been created using the Fast GPS Clustering algorithm described in the previous section, this information can be used to predict his next destination, as well as the estimated time of departure from the users.

González et al. studied a group of 100,000 people whose position was tracked for six months using cell phone communication with cellular towers [16]. This study showed that human trajectories contain a high degree of temporal and spatial regularity, with each individual being characterized by a time-independent characteristic travel distance and a significant probability of return to a few frequented locations. This similarity in travel patterns shows that people's mobility patterns can be reproduced and/or predicted using travel histories.

This section of the report focuses on the design of an intelligent application that can use past travel behavior from a specific user to predict future destinations due to a specific start location and a specific departure time. This type of algorithm can enable a variety of services to users, such as traffic incident reporting and location-based advertising regardless of the user's transportation mode.

The following sections present an algorithm that predicts a user's destination in real-time using his travel history considering factors such as the time and the day of the week. The algorithm can also predict the hour of departure of a user from a specific start location. This algorithm uses data recorded by TRAC-IT application on GPS enabled mobile phones.

PROBABILISTIC PATH PREDICTION METHODOLOGY

The Probabilistic Path Prediction algorithm uses individually-recorded travel behavior from a specific user to predict that user's trip destination and his hour of departure from his current locations. This method allows the prediction to be more oriented to a specific user and to a specific departure location.

NAÏVE BAYES DESTINATION PREDICTION

The Destination Prediction algorithm retrieves the user identification number, the start location identification number, and the current day of the week from the TRAC-IT system on a periodic basis (e.g., every 15 minutes) for all active users. For each user, the algorithm then queries the database for a list of historical trips that start at the user's current location. Then, the destinations for all historical trips that start at the user's current location are retrieved. Various statistics are then calculated for the frequency of trips to each destination. Next, the algorithm uses a Naïve Bayes classifier to determine the probability of the user traveling to each different end location based on the number of times that this location was previously visited by that user, using his current location as the starting location. Finally, the algorithm stores the probabilities in a list ordered by decreasing probability and returns this list to the TRAC-IT system for possible action.

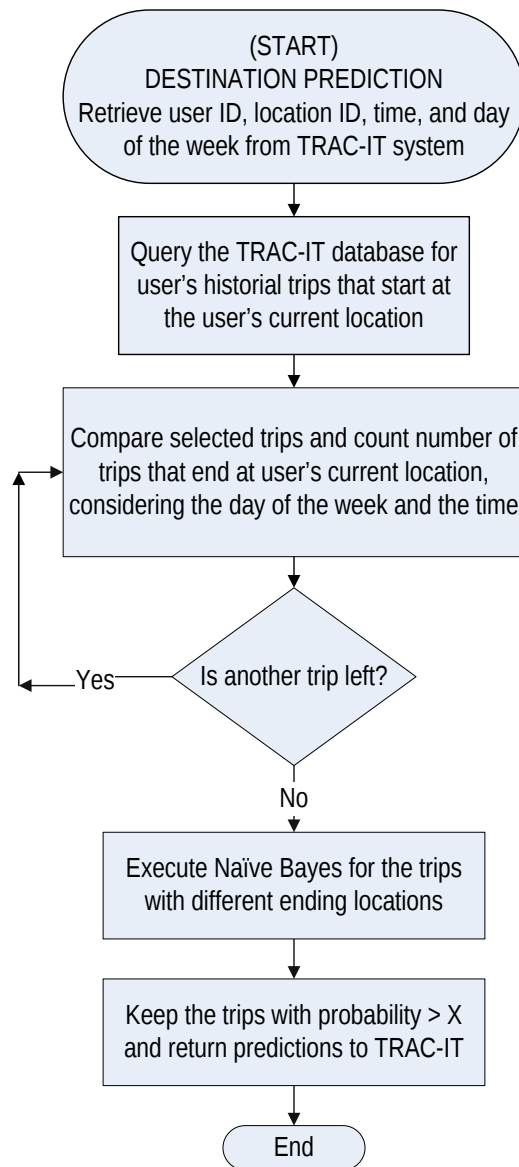


Figure 38 - Destination Prediction Flow Chart

NAÏVE BAYES DEPARTURE TIME PREDICTION

The Departure Time Prediction algorithm is triggered on a periodic basis (e.g., every 15 minutes) to retrieve the user identification number, the start location identification number, and the current day of week to predict the hour of departure of the user. The algorithm then queries the database for a list of trips that start at the same location as the user's current location. Next, the algorithm stores all the trips that start at different hours in a list and counts the number of times that a trip starts at the same hour. Next, the algorithm uses a Naïve Bayes classifier to calculate the probabilities of the user starting his or her trip at the different hours in the list. Finally, the algorithm returns this list of probable hours in decreasing order of probability to the TRAC-IT system for possible action. It is important to note that the algorithm makes the comparisons and categorizes times using only the hour without considering the minutes or seconds in order to allow an error margin for classification. Future work could also break down time categorizations further to use 15-minute blocks instead of an hour.

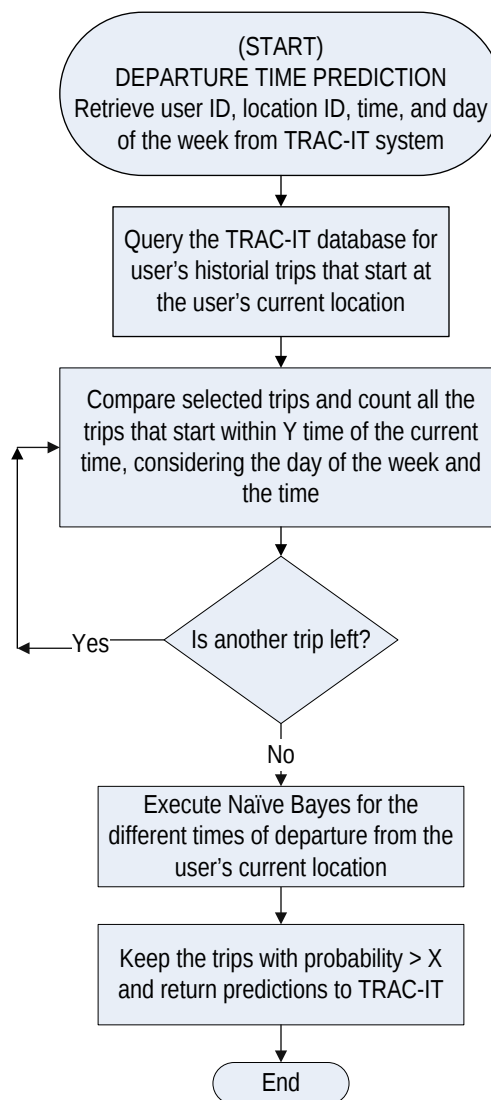


Figure 39 - Hour of Departure Prediction Flow Chart

NAÏVE BAYES CLASSIFIER

Naïve Bayes is a simple and very efficient technique that allows all the attributes and features that are equally important and independent of one another to make contributions to the calculations of the algorithm. In real-life, datasets' attributes are not equally important or independent, but this is a simple technique that has been very effective in practice. An example calculation of a user's probability for traveling to a single specific location can be represented as:

IMPLEMENTATION

The Naïve Bayes algorithm was developed using the Java SE software platform and was integrated into the TRAC-IT Glassfish Java Web application server for execution using real-time travel information. Microsoft SQL Server and PostGIS database management systems were used to store locations as well as user's information.

DESTINATION AND DEPARTURE TIME PREDICTION CONCLUSION

Predictive intelligent applications are very important for the future of location-based services on GPS devices and cell phones in order to provide pre-trip information as well as highly targeted personalized messages that avoid overwhelming the user with a large number of irrelevant alerts. This technique uses the Naïve Bayes method to predict user destination and departure times based on the number and time of visitations to each of the user's POIs as well as the user's real-time location, which represents the starting location of the trip. This type of algorithm enables a variety of location-based services, including providing relevant traffic information to users, without requiring the user to manually input information into their phone, so the user can elect to change his travel behavior before he reaches areas of congestion. In the future, other more advanced probabilistic techniques such as Markov models or Bayesian networks could be investigated to determine if these techniques provide more accurate predictions than Naïve Bayes.

CHAPTER SIX

TRAC-IT ARCHITECTURE INTEGRATION WITH PATH PREDICTION, FL511 API, AND TRANSIT INFORMATION

ARCHITECTURE OVERVIEW

To provide travel information services to cell phone users, the technology presented in each of the previous chapters must be integrated, including TRAC-IT mobile phone and server software, TRAC-IT database servers, FL511 API for traffic events, and HART ETA API. Figure 40 shows the arrangement of the different architecture components and how they are connected via communication. The figure also shows two additional components that have not been previously discussed: the bus stop database and the cellular carrier messaging gateway. The following sections explain these two architecture components, which provide valuable data and connections to the primary portions of the architecture.

BUS STOP DATABASE

To query the HART transit estimated time-of-arrival webservice API, the TRAC-IT server software must determine the closest bus stop to the user as well as the routes that visit that stop. When the TRAC-IT mobile phone user switches to the screen displaying transit information, the TRAC-IT mobile phone application requests that the TRAC-IT server software provide the bus stop ID and related information for the closest bus stop to the user. The TRAC-IT server software then queries a SQL Server database which contains the bus stop information for the local transit agency, and passes the bus stop ID and other information back to the mobile phone application. The TRAC-IT mobile phone application then is able to query the HART webservice API directly.

The bus stop database is kept up to date via updates from a separate software application used in the Travel Assistance Device (TAD) system, which provides real-time navigation services for public transportation users [9]. The TAD system reads the transit agency's data formatted according to the General Transit Feed Specification (GTFS) and imports these data into the bus stop database on a regular basis.

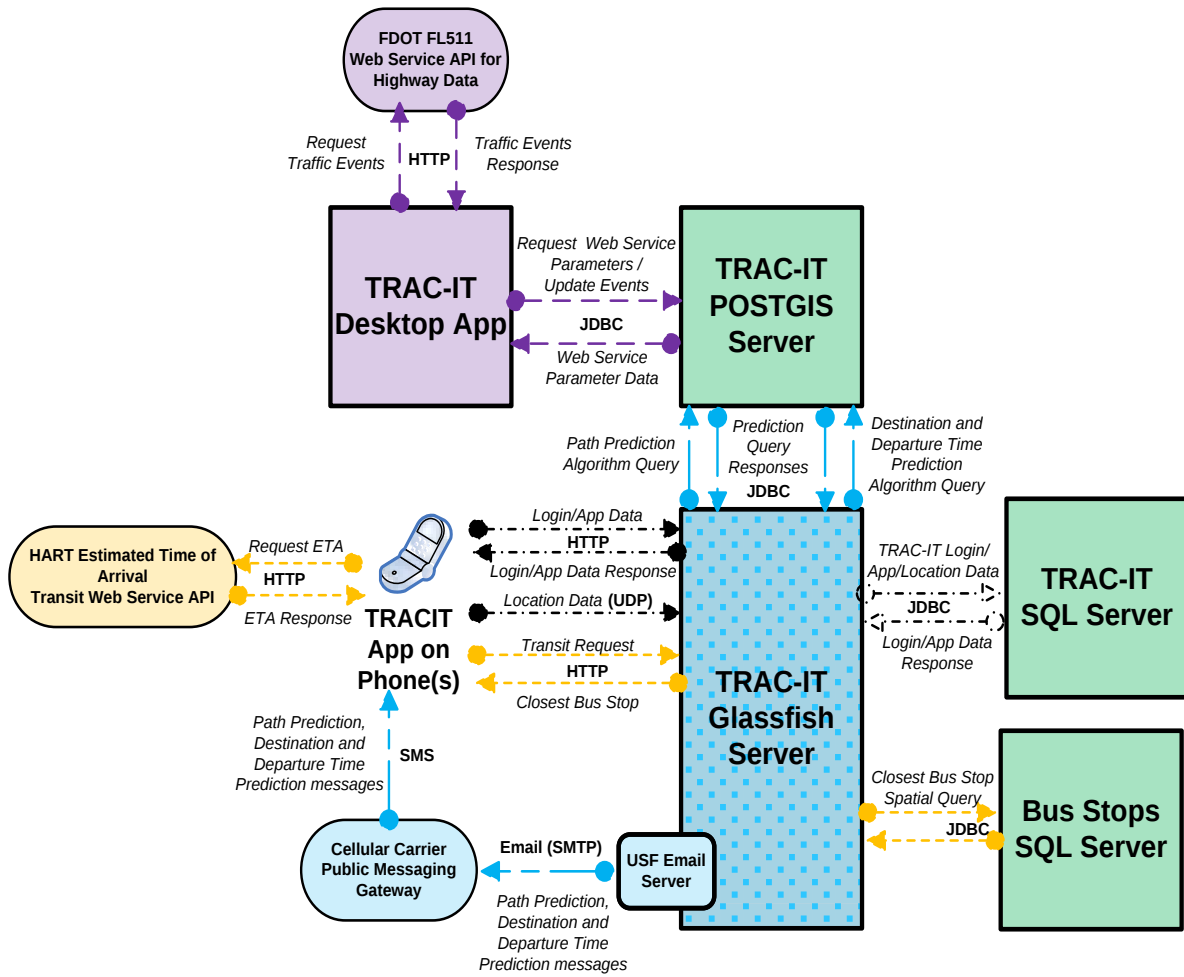


Figure 40- TRAC-IT Architecture Supporting Path Prediction Service

SENDING TRAFFIC ALERTS TO PHONES

Once the path prediction software determines that an alert should be sent to a phone, the TRAC-IT server needs to issue a message to that phone. In this proof-of-concept, messaging to phones was tested using the carrier public gateway, which allows an application to address an email to a phone number at a specific domain related to the wireless carrier (Figure 41).

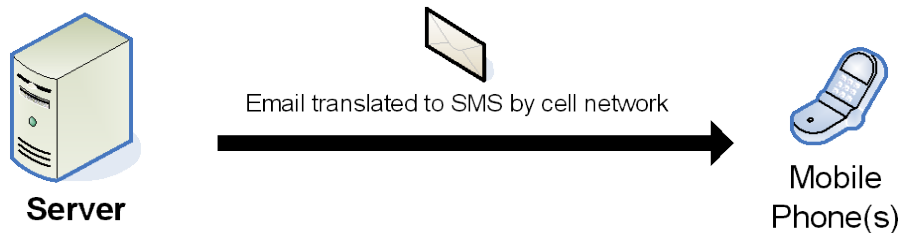


Figure 41 - TRAC-IT Server Issues Message to Phone Using the Public Messaging Gateway

For example, an email addressed to *8135551234@messaging.sprintpcs.com* will be translated to a text message and delivered to a phone with the phone number 813-555-1234. The sender must have knowledge of the carrier to which the phone number belongs, since each carrier has its own public gateways and will deliver messages only to devices on its own network. It should be noted that these public gateways are traditionally used to send messages to a small number of users. Large numbers of text messages sent in bulk within a small time period may be blocked by the carrier, depending on its policies regarding spam. In this case, private SMS gateways maintained by the carrier or carrier partners, such as the Sprint Services Framework (http://developer.sprint.com/site/global/services/overview/core_services/sms/p_sms.jsp) may be necessary if a large number of text messages must be sent simultaneously. Typically, these private gateways require a fee or service agreement with the maintaining party, while public gateways are free to use but may be subject to use restrictions.

Table 5 contains the public gateway email formats for various U.S. carriers tested as part of this project. Emails were created and sent using the public gateway email format, and deemed successful if a text message containing the message arrived at the mobile phone. Android phones, as well as several flip-model feature phones, were used to test this service.

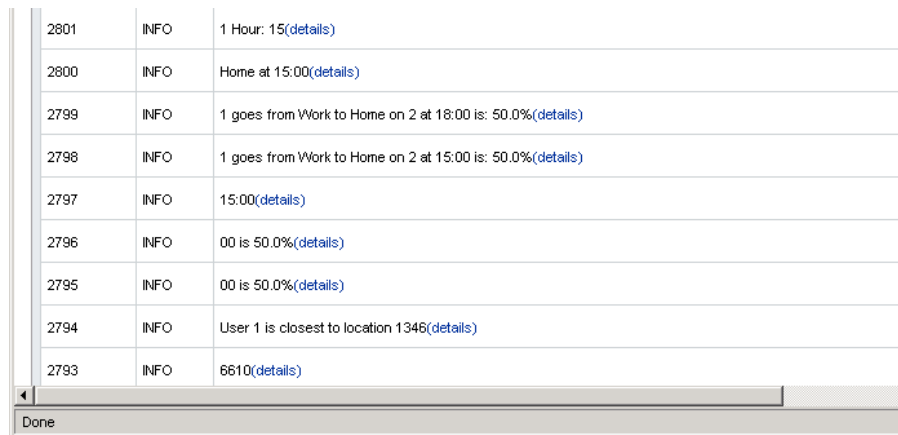
Table 5 - Public Messaging Gateways for Carriers

WIRELESS CARRIER	PUBLIC GATEWAY EMAIL FORMAT
Sprint	phoneNumber@messaging.sprintpcs.com
Nextel	phoneNumber@messaging.nextel.com
AT&T	phoneNumber@txt.att.net
Verizon Wireless	phoneNumber@vtext.com
T-Mobile	phoneNumber@tmomail.net

PROOF-OF-CONCEPT WALKTHROUGH

DESTINATION AND DEPARTURE TIME PREDICTION

Figure 42 shows the example output from an execution of the Destination and Departure time prediction algorithm that uses a Naïve Bayes classifier. The TRAC-IT server was set to execute the Naïve Bayes classifier every 30 minutes for all active sessions (i.e., all cell phone users that are logged into the system). The user ID corresponding to each session is retrieved from the database, and the nearest location is found for this user (i.e., Location ID = 1346, Description = “Work”) with respect to that user’s real-time cell phone location. Naïve Bayes is then used to determine which destination the user is most likely to visit next (i.e., “Home”), as well as what time he is likely to leave his current location (i.e., “3 p.m.” with 50% probability, “6 p.m.” with 50% probability). If there is a tie between two ending locations, as in the above example, the most recently visited location is chosen as the favored location. Once the destination location is predicted, any trip information related to the path between the user’s current location and his destination can be delivered to him. The time of departure information can be used to deliver alerts to the user only in a certain time window (e.g., 30 minutes) before he typically leaves his current location. The length of this time window could be set by the user. Departure time prediction keeps alerts from being delivered to the user during a long period of time (e.g., 8 hours) that he might spend at a work or other location on a regular basis. For example, if a user arrives at his “Work” location at 8:00 a.m. and typically leaves work at 5 p.m., he should not be notified of traffic incidents until closest to his departure times since incidents that occur early in the day are likely to be resolved by the time he leaves work.



2801	INFO	1 Hour: 15(details)
2800	INFO	Home at 15:00(details)
2799	INFO	1 goes from VWork to Home on 2 at 18:00 is: 50.0%(details)
2798	INFO	1 goes from VWork to Home on 2 at 15:00 is: 50.0%(details)
2797	INFO	15:00(details)
2796	INFO	00 is 50.0%(details)
2795	INFO	00 is 50.0%(details)
2794	INFO	User 1 is closest to location 1346(details)
2793	INFO	6610(details)

Figure 42 – Example Output from Destination and Departure Time Prediction Engine

PATH PREDICTION

To test the Path Prediction algorithm with the TRAC-IT system, the Path Prediction software was integrated into the TRAC-IT Java Web application server software. The Path Prediction algorithm was executed at every five “critical points.” Critical points are identified when the user changes direction in order to create a simplified place (e.g., when the user is standing still no critical points will be generated [17][11]. The TRAC-IT server software then compares those critical points to the paths stored in the TRAC-IT spatial database and determines which paths overlap with the user’s current position in real-time. Figure 43 shows a graphical illustration of paths, shown as grey and yellow polygons, that intersected with the user’s real-time position, shown as the push pins in the lower-right corner of the map.

Once the paths are predicted, the TRAC-IT server software compares the predicted paths against the currently active set of traffic incidents which were retrieved from the FL511 API webservice. If the predicted paths intersect with an active incident, then the event information associated with that incident is sent to the user using the cellular carrier public messaging gateway.

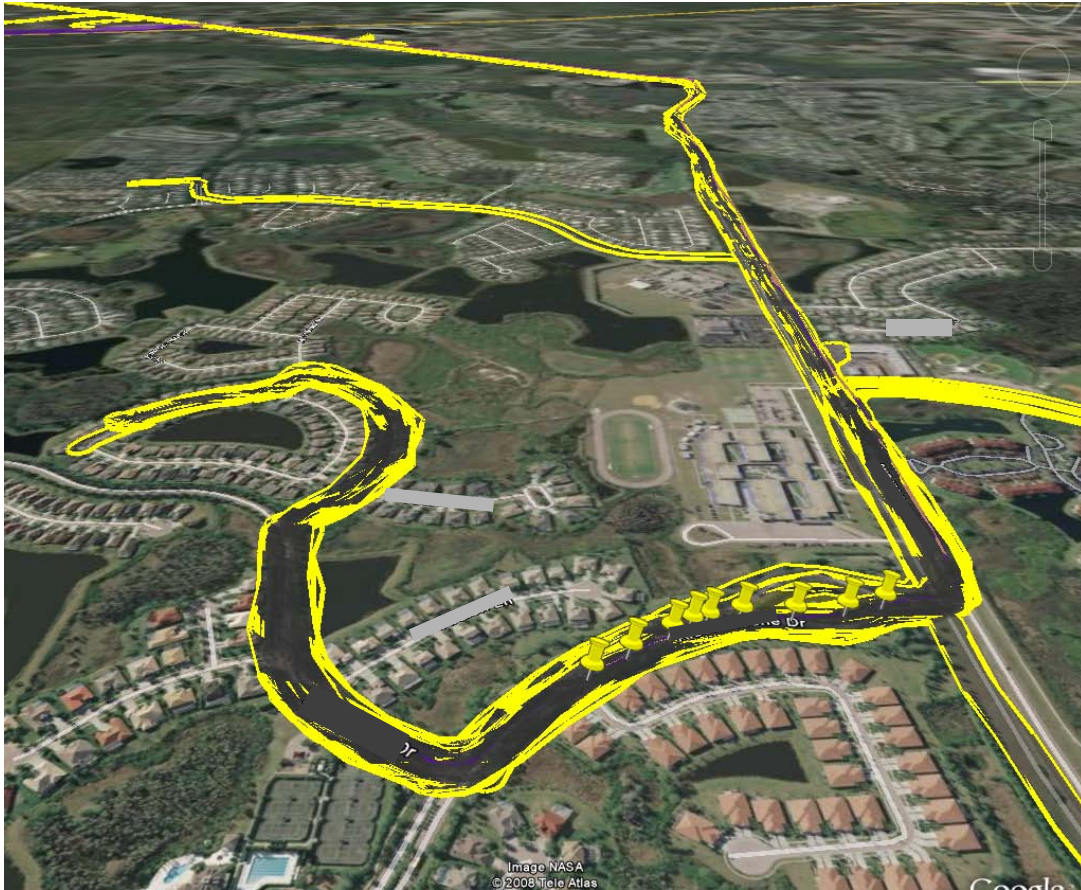


Figure 43 - Path Predictions for a User's Path in Real-Time

Figure 44 shows the TRAC-IT Web application server logs which shows the check for any intersecting events that overlap the user's predicted path, the transmission of the details of the incidents from TRAC-IT to the cellular carrier's public messaging gateway, and a screenshot of the text message containing accident information that was received by the cell phone.

Log Entry Detail	
Timestamp	Jan 11, 2011 12:18:47.069
Log Level	INFO
Logger	javax.enterprise.system.stream.out
Name-Value Pairs	_ThreadID=32;_ThreadName=Thread-292;
Record Number	396
Message ID	selecting
Complete Message	SELECT a."TRIPID" FROM "tblTrips"AS a WHERE ST_Area(ST_Intersection(ST_Buffe 28.05883026123047, -82.41614532470703 28.058780670166016, -82.4160995483; -82.4161148071289 28.058732986450195),4269),32617),1.0),a."geom"))/ST_Area 28.05883026123047, -82.41614532470703 28.058780670166016, -82.4160995483; -82.4161148071289 28.058732986450195),4269),32617),1.0))>*.10

Log Entry Detail	
Timestamp	Jan 11, 2011 12:18:47.022
Log Level	INFO
Logger	javax.enterprise.system.stream.out
Name-Value Pairs	_ThreadID=32;_ThreadName=Thread-292;
Record Number	395
Message ID	Sending accidents
Complete Message	Traffic congestion in Alachua on I-75 north from Exit 399 US-441 to at Exit 404 CR-2



Figure 44 - TRAC-IT Web Application Software Log and Screenshot of Text Message

TRANSIT INFORMATION TESTS

The research team tested the accuracy of the transit information provided by the TRAC-IT system to the mobile phone by visiting a variety of bus stops in the Tampa Bay area and comparing the estimated time-of-arrival to the actual arrival time of the bus. Table 6 shows that 90 percent of arrival estimates, 27 of 30, were within 1 minute of bus time arrival.

CHALLENGES

While most of the technology in this proof-of-concept system performed as expected, the use of PostGIS as the primary spatial database server and the programmatic access of the PostGIS database from the TRAC-IT Java Web application hosted within a Glassfish server provided some challenges. Glassfish maintains its own database connection pool based on Java DataBase Connectivity (JDBC) drivers. Therefore, Glassfish relies on the JDBC drivers for PostGIS in order to create and establish connections to the PostGIS database. Glassfish creates a number of PostGIS connections and places these connections in a pool in memory so they can be allocated to Web applications (e.g., TRAC-IT server application) as they are requested. While the combination of Glassfish, PostGIS, and JDBC drivers seem to work fine for a period of time, occasionally the Glassfish server will run out of connections in the pool and start sending error messages. One common problem that creates this connection pool exception is the absence of software in the TRAC-IT server application that properly closes all database connections when the software is finished with the connection. However, the TRAC-IT server application was carefully checked, and it appears that all connections are being closed properly. Therefore, given the current configuration, the problem likely lies either in the management of the database connection pool by Glassfish or the handling of the creation and destruction of connection objects by the PostGIS JDBC drivers. Glassfish v2.1, PostgreSQL 8.3, and PostGIS JDBC drivers version 1.5.0 were tested as part of this project. Future work should examine the use of newer versions of Glassfish to see

if the problem is resolved, or consider utilizing SQL Server 2008 for spatial operations in place of PostGIS. While PostGIS is a free open-source spatial database server, it may not have the widespread use of SQL Server and therefore SQL Server may have a more stable platform for these specific spatial operations.

Table 6 – Estimated Time of Arrival Information Provided to
TRAC-IT Mobile Phone Users

#	ROUTE	STOP	ETA ACCURATE (WITHIN ~1 MIN)?
1	39-Busch Blvd	Busch @ 37th	Yes
2	5-40th St	Busch @ 40th	Yes
3	5-40th St	McKinley @ Linebuagh	Yes
4	5-40th St	McKinley @Bouganvillea	Yes
5	5-40th St	McKinley @ FDOT	Yes
6	6-56th St	Fowler @ 51st St	Yes
7	6-56th St	Fowler @ 52nd St	Yes
8	57-UATC Netpark	Fowler @ 62nd St	Yes
9	57-UATC Netpark	Fowler @ Raintree	Yes
10	57-UATC Netpark	Fowler @ Gillette	No
11	57-UATC Netpark	Fowler @ Gillette	Yes
12	57-UATC Netpark	Fowler @ Riverhills	Yes
13	57-UATC Netpark	Fowler @ Morris Bridge	Yes
14	57-UATC Netpark	Morris Bridge @ Boardwalk	Yes
15	57-UATC Netpark	Davis @ Christi	No
16	57-UATC Netpark	Davis @ Christi	Yes
17	57-UATC Netpark	Davis @ Copeland	Yes
18	57-UATC Netpark	Davis @ Copeland	No
19	57-UATC Netpark	Temple Terrace @Hamilton Hill	Yes
20	57-UATC Netpark	Temple Terrace @78th St	Yes
21	6-56th St	56th st @ Busch	Yes
22	39-Busch Blvd	Busch @ 50th	Yes
23	39-Busch Blvd	Busch @ Hyaleah	Yes
24	39-Busch Blvd	Busch @ Temple Ct	Yes
25	39-Busch Blvd	Busch @ 46th	Yes
26	39-Busch Blvd	Busch @ 42nd	Yes
27	39-Busch Blvd	Busch @ 30th	Yes
28	39-Busch Blvd	Busch @ 26th	Yes
29	39-Busch Blvd	Busch @ 2308	Yes
30	39-Busch Blvd	Busch @ 19th	Yes

CHAPTER SEVEN

TRAFFIC TEXT-TO-SPEECH: A PROTOTYPE FOR REDUCING DISTRACTED DRIVING

BACKGROUND ON TEXT-TO-SPEECH

Recently, the dangers of texting while driving have gained much attention, with recent studies stating that the risk of someone who texts while driving being involved in a car accident is 8 to 23 times greater than the crash risk of someone who does not text while driving [18]. In many states, services such as FL511 exist that will keep travelers informed of any traffic delays or accidents that may affect their travel. However these services typically require a driver to open and read a traffic alert text message which contributes to the distracted driving problem. This action can be dangerous and conflicts with the No-Texting-While-Driving rule in many states.

This portion of the project focused on implementing a prototype intelligent software application for cell phones that “speaks” a traffic alert message to an individual. These messages are only delivered to the user when they are moving at a speed less than an established threshold speed, or when they are completely stopped. This reduces the amount of risk related to receiving traffic alerts and at the same time allows the driver to receive the information without having to interact with the mobile phone. An Android mobile phone application was implemented that receives a User Datagram Protocol (UDP) packet from a server, calculates the driver’s actual speed, and speaks the message if the driver is traveling at or below the threshold speed, or is stopped. If the user is traveling above the threshold speed, the application will wait until they stop moving before announcing the traffic alert. This application can also log into the TRAC-IT system to track travel behavior and record the travel path of the phone, and provide personalized alerts based on predicted travel behavior.

The original TRAC-IT mobile phone application was implemented using Java ME for standard “flip” phones. While Java ME was sufficient to support the first version of the TRAC-IT mobile application, Java ME devices have proven to have varying performance characteristics that require modifications to mobile applications for each device on which the application runs. This fragmentation of behavior causes a great deal of effort on the software developer’s part to keep up with the large number of devices that are released. Additionally, Java ME was designed to run on standard “flip” phones and is typically not available on higher-end smartphones that have touch screens and more powerful processors. Therefore, a TRAC-IT software application for a smartphone platform should be created to both increase the number of devices on which TRAC-IT can run as well as take advantage of new smartphone capabilities, including Text-To-Speech capabilities.

Google Android, the most pervasive Java-based smartphone platform, has emerged on many different smartphones from a variety of manufacturers, including Motorola, HTC, Samsung, Kyocera, Sanyo, and LG, and is available on every major U.S. cell network, including Verizon Wireless, AT&T, Sprint, Nextel, and T-Mobile. In the second quarter of 2010, Android took the lead as the top operating system for smartphones sold in the U.S., with 33 percent of the share of phones purchased in Q2 2010, while RIM Blackberry came in

second at 28 percent and Apple's iOS for iPhone held on at 22 percent [19]. In September 2010, Gartner declared that Android will capture the No. 2 worldwide operating systems title in 2010, ahead of RIM's Blackberry and Apple's iPhone, and challenge Nokia's Symbian operating system for the No. 1 position by 2014 [20]. Android as a mobile device programming platform is also beginning to push into the tablet PC market as well as the "feature phone" or the less capable and cheaper "flip phones." More information about the transition for application developers from Java ME to Android is discussed in the Addendum. While Apple and Verizon announced in January 2011 that the iPhone would be available on February 10, 2011 on the Verizon network, this only makes the iPhone available on two of the five major cellular carriers (<http://www.verizonwireless.com/b2c/splash/iphone.jsp>). Therefore, to reach all carriers with a single mobile application, Android is preferred to the iPhone for an initial smartphone version of TRAC-IT. However, future research may investigate expanding TRAC-IT to the iPhone platform in order to reach iPhone users on AT&T and Verizon. For all these mobile applications, the same server-side TRAC-IT system code can be reused.

Android offers a Text-To-Speech (TTS) API that can be personalized through the language, pitch, and speech rate. This technology offers the opportunity to further increase safety measures on the TRAC-IT program and improve upon the current industry standard method of sending text messages for traffic alerts. In an application created using the Android Software Development Kit (SDK) and TTS API, messages received as part of the traffic alerts and travel suggestion are "spoken" to you. An additional feature also checks the driver's speed via the GPS feature of the phone that would only allow an alert to be read to a driver, if the driver were traveling at or below an established threshold speed.

METHODOLOGY FOR DEVELOPING TRAFFIC TTS

This portion of the project used the Android SDK and the Java programming language, which has its own design and APIs for Android that is different from Java Micro Edition and Java Standard Edition. Developing

The application requirements were to:

- Implement UI to enter username and password on Android.
- Implement login/logout functionality for TRAC-IT webservices on Android, and location listener that starts after user logs in.
- Implement TTS and GPS listener to reduce distracted driving by speaking messages only when the user is traveling at a slow speed.

Traffic TTS was originally divided into two software components for Android for testing purposes: Android application for TRAC-IT, which logs into the TRAC-IT system and sends location data for the phone to the server and UDP Receiver, which receives sample traffic alerts from the server via the UDP protocol and speaks them to the driver, based on the speed determined by the GPS.

These two components are described separately below as they were originally implemented for testing, and then a following section describes the final integration of these components into one Android application, Traffic TTS.

TRAC-IT ANDROID APPLICATION

The first software developed as part of the project was an Android application that could login to the TRAC-IT Web services. This software reads the username and password that the user types in when the login button is pressed, as shown in Figure 45, and contacts the TRAC-IT webservice using HTTP. If the user inputs invalid

login information, they are notified via text on the screen. If they enter the correct information, a session ID is created at the server and passed back to the device. All subsequent location data that is sent to the server, until the user logs out, contains this session ID. All location data for the user gathered from a trip under that session ID are stored in the TRAC-IT database.



Figure 45 – Initial Login Screen for Traffic TTS

After the login has been successful and a session ID has been created, another Android activity is launched. When the two application components were originally designed separately, an activity would let the user choose between TRAC-IT Android and UDP Receiver, as shown in Figure 46.

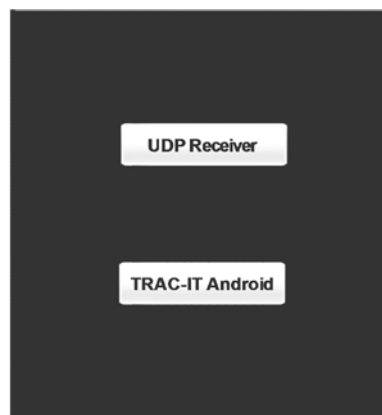


Figure 46 – Original Choice between UDP Receiver and TRAC-IT Android Functionality

If TRAC-IT Android is chosen, the TRAC-IT client is launched. This screen has two important parts: the location data (i.e., latitude, longitude, and speed) being collected from GPS or the network (e.g., wi-fi, cellular) that is sent to the TRAC-IT server, and the second feature that allows you to input any text in the box, which will be input to the Android TTS API so it is spoken out loud (Figure 47). Language can be set using the Android internal Text-To-Speech setting to enforce the setting on all applications. This can be done in the home screen by pressing *Menu > Settings > Text-To-Speech > Always use my settings*.

This Android activity software uses embedded Global Positioning System (GPS) technology to determine the real-time position and speed of the user. “Wake locks,” which request that the phone’s CPU continue to run the application in the background, are used to prevent the GPS from being interrupted if the phone goes to sleep due to inactivity. Wake locks must be properly handled to avoid accidentally locking the phone in the “awake” state when the application is not running. When the application is closed, it stops monitoring GPS to save battery energy. The connection with the TRAC-IT server is also terminated when this activity exits.

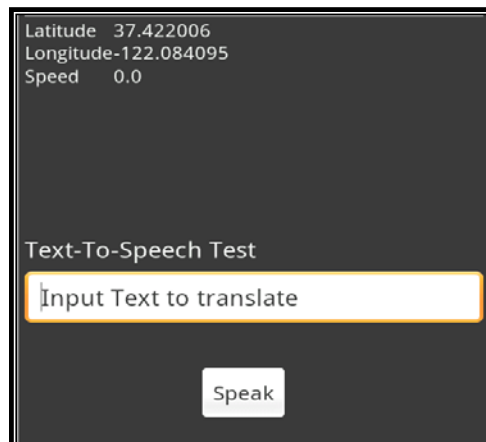


Figure 47 – Original Interface for TRAC-IT Android Application

UDP RECEIVER APPLICATION

A second application was created to implement the component of the application that receives messages from a server, as seen in Figure 48. After the login screen, if UDPRT is chosen, a UDP packet receiver can be started by clicking on the *Start UDP* button. A desktop Java application was used to send UDP packets to the phone's current IP address and port number. These packets simulate traffic messages which are sent to the phone. On the phone, a thread is started to process the received UDP data. On Sprint's cellular network, as well as most wi-fi networks, the phone's IP address tends to be dynamic and changes every time the phone reconnects. A phone's IP address can be determined from the home screen by pressing *Menu > Setting > Wireless & networks > Wi-fi settings*. By clicking on the active wi-fi connection the IP address will be shown. The thread that was started on the device will continue to wait until a UDP packet containing a message is received. This activity also uses the Android *LocationListener*, which updates the application with newly-calculated GPS positions. This application is set so that every 10 seconds the user's speed will be monitored. If the user is traveling under the speed threshold, the message received in the payload of the UDP packet will be translated into a string and spoken to the user. If the user is traveling faster than the speed threshold, the application will wait for the user's speed to drop below the threshold before announcing the traffic alert. After the message is processed, it is erased and the receiver awaits a new UDP packet from the receiver.



Figure 48 – Original Interface for UDP Receiver Application

Efficient code was implemented so that the *Start UDP* button will give correct message if pressed twice when the thread is still running. This code prevents the application from crashing. Every time the application is closed, the GPS component is turned off to help avoid an impact on battery life. The UDP receiver thread will then be interrupted and the user is redirected back to the home screen.

PROTOTYPE TESTING AND FINDINGS

Testing was done with the TRAC-IT Android application to gather some information during typical travel using a HTC Hero on Sprint's EV-DO Rev A network with Android version 2.1 update 1. The typical walking speed that was observed from the GPS was between 1.0 and 1.5 meter per second (m/s), and typical jogging speed was 2.0-3.0 m/s. The application was tested using a 1.0m/s as a threshold speed for receiving messages. If the phone receives a message at a speed above that threshold, the message would be spoken to alert the user of an incoming message and to reduce his speed. The TTS API was set to speak after a GPS location update. If a GPS fix is not available, the application will continue to try and get a GPS fix until successful.

Upon testing the TTS engine, no limits were found for spoken message length. Paragraphs were typed and tested using the engines in English and Spanish and no problems were encountered. In a final version of this type of application, the user would be able to select the language. As to the clarity of the TTS engine, the pitch was set to 0.8 and the speech rate to 0.8, which resulted in an understandable message.

TRAC-IT Android uses location-based services that will update the application every time the phone's interval GPS calculates a new position. As a result, the limitations of GPS apply to this application. Whenever TRAC-IT Android or UDP receiver was started indoors, it would sometimes take up to five minutes to get a GPS fix.

INTEGRATION OF APPLICATION COMPONENTS INTO TRAFFIC TTS

The TRAC-IT Android and UDP Receiver components were integrated into a single application, Traffic TTS. The combination of the two applications provides an ease of usability by eliminating the need to switch between views.

The initial view of the application is still the main logon screen where the user enters a username and password to log into the TRAC-IT system. The view selection screen (Figure 46) has been removed in the current version since the applications no longer operate separately from one another. Because the combined application must maintain functionality of both of the previous clients, a new user interface was designed in

order to facilitate the needs of both programs. The combined interface Figure 49 shows the new view layout. Messages that are printed to the screen by the UDP Receiver client are now printed in the center of the screen below above the “Start UDP” button. The speech test from the TRAC-IT Android client was also kept in order to ensure TTS functionality.

The current version of Traffic TTS has been tested with the Sprint HTC Hero with Android 2.1 update1 and T-Mobile HTC MyTouch 3G Slide android devices. The testing has shown that the combined application retains the same previous behavior as was witnessed in the individual views.

TRAFFIC TTS FUTURE WORK

Future research projects should extend Traffic TTS to include listening for SMS messages from specific numbers, such as the FL511 text messaging number (i.e., instead of only listening for UDP messages sent from the TRAC-IT server). This would enable the Traffic TTS application to allow normal delivery of non-traffic alert SMS messages, but it would intercept and speak FL511 traffic notifications to the traveler in a very similar way to how Traffic TTS currently speaks UDP messages from TRAC-IT. These modifications would allow Florida mobile phone users to opt-out of data collection of their travel behavior via Traffic TTS if desired and simply receive SMS notifications from FL511 that are read to them when they are traveling at or below the threshold speed. However, if the user opts out of TRAC-IT, they would not receive filtered messages based on their real-time and historical travel behavior. In other words, users would benefit from the Traffic TTS delivery system of the traffic message to the user via speech, but the user would receive all messages sent to them from the Florida 511 system since path prediction would not be filtering any of these messages. Therefore, if path prediction provides a valuable service of highly relevant alerts, path prediction may be sufficient incentive for some users to participate in TRAC-IT data collection travel surveys without any additional monetary incentive from the travel behavior surveyor.

TRAFFIC TTS CONCLUSION

The implementation of a prototype Traffic TTS provides a novel interface that delivers traffic alerts to users only when they are traveling at speeds at or below the established threshold without affecting the normal operation of the user’s phone. Additionally, Traffic TTS speaks the message to the traveler and does not require the user to look at his cell phone to receive the message. These features will allow DOTs to send travel information text messages to users but strongly promote that users only receive messages when they have stopped moving.

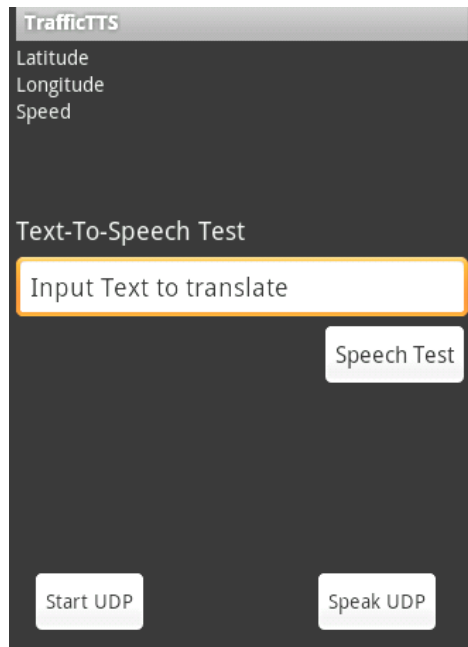


Figure 49 – Final Traffic TTS Interface, with TRAC-IT Functionality

CHAPTER EIGHT

CONCLUSIONS AND RECOMMENDATIONS

This project successfully demonstrated Path Prediction technology that can provide highly-targeted, personalized alerts based both on the real-time location and past travel history of each individual user. Fast GPS Clustering is able to rapidly process large GPS datasets in order to create POIs and trips, which provide a basis for probabilistic travel predictions. Fast GPS Clustering also produces pseudo-POIs, which signal areas of traffic congestion or delay, that have great potential to provide numerous other services such as informing traffic signal re-timing, road enhancements, and predictive traffic alerts based on historical travel patterns. A simple probabilistic engine using the Naïve Bayes method was also created to demonstrate the feasibility of probabilistic predictions using the results of the Fast GPS Clustering algorithm.

This project also demonstrated a prototype for technology that delivers traffic information only when the user is traveling below the established speed threshold or has stopped moving, and can speak traffic information to the traveler using TTS APIs so the user does not have to look at the phone, both without affecting the primary functionality of the mobile phone or the ability to receive other text messages. Therefore, DOTs can provide traffic incident alerts in a manner that does not interfere with the traveler's normal use of the cell phone. Additionally, the mobile application developed as part of this project automatically displays real-time transit information for transit stop the closest to the traveler's real-time position, without the traveler having to manually enter any information.

DOTs should be encouraged to provide travel information to the general public as openly as possible and allow personalized customization for the number and types of alerts that a user will receive. The existing My Florida 511 service is an excellent first step towards more personalized traffic information in Florida. By further adopting technology such as the hyper-personalization techniques based on the user's current location and personal travel behavior history discussed in this report, DOTs can further filter the deluge of real-time traffic information to travelers and ensure that users are only getting information that is directly relevant to them. Hyper-personalization can reduce the number of irrelevant alerts sent to a user, helping to retain users and preventing travel alerts from becoming another form of digital noise. Advanced personalization of messages will become even more critical as DOTs begin monitoring arterial roads in real-time in addition to highways. If the number of alerts based on two interstates in the Tampa Bay area reached an average of 13 email messages (and many more text messages) per day of the week, with an average of 17 emails on Fridays, the number of alerts generated from interstates and arterials will reach extreme numbers very quickly. In addition, manually subscribing to arterial roads will become more difficult and time-consuming for users, since travelers will typically use many different arterial roads. The Path Prediction technology does not require users to manually create subscription lists, as it builds a list of roads frequently traveled by each user, which is then used to generate personal traffic alerts based on the traveler's real-time location and likely immediate travel behavior.

The creation of APIs, such as FDOT's 511 Feed API, is an excellent way to provide travel information to third parties that can create new types of services based on this information without any cost to the DOT. It would not have been possible to test the technology created as part of this project with real-time traffic information without access to some type of API that provides programmatic access to real-time traffic information.

FUTURE RESEARCH

Future research projects should work with the Florida Intelligent Transportation Systems (ITS) Program and the FDOT 511 Working Group to deploy the Traffic TTS mobile application to users. This deployment should include extending Traffic TTS to include listening for SMS messages from specific numbers, such as the FL511 text messaging number (i.e., instead of only listening for UDP messages sent from the TRAC-IT server). This would enable the Traffic TTS application to allow normal delivery of non-traffic alert SMS messages to a user's phone, but the application would intercept and speak FL511 traffic notifications to the traveler in a very similar way to how Traffic TTS currently speaks UDP messages from the TRAC-IT system. These modifications would allow Florida mobile phone users to opt-out of data collection of their travel behavior via Traffic TTS if desired and simply receive SMS notifications from FL511 that are read to them when they are traveling at or below the threshold speed. However, if the user opts out of data collection, then they would not receive filtered messages based on their real-time and historical travel behavior. In other words, users would benefit from the Traffic TTS delivery system of the traffic message to the user via speech, but the user would receive all messages sent to them from the Florida 511 system since path prediction would not be filtering any of these messages. Therefore, if Path Prediction provides a valuable service of highly relevant alerts, it may be a sufficient incentive for some users to participate in TRAC-IT data collection travel surveys without any additional monetary incentive from the travel behavior surveyor.

For the prototype development, this research project focused on accessing the FL511 traffic information resources. Testing the prototype with different real-time traveler information resources that have more comprehensive coverage on the traffic network will be a next step that enables the application with more useful information for travelers who may not use highways.

When testing Path Prediction as part of this project, challenges were encountered when executing the Path Prediction software within Glassfish Java Application server v2.1, PostGIS database server based on PostgreSQL v8.3, and PostGIS Java Database Connectivity (JDBC) drivers v1.5.0. Glassfish occasionally exhausted the database connection pool when executing spatial queries, which indicates that either the application or underlying software is not correctly releasing database connections. Future work should examine the use of newer versions of Glassfish to see if the problem is resolved, or consider utilizing SQL Server 2008 for spatial operations in place of PostGIS. While PostGIS is a free open-source spatial database server, it may not have the widespread use of SQL Server and therefore SQL Server may have a more stable platform for these specific spatial operations. The feasibility of the deployment of Path Prediction as a reliable service will depend upon the reliability of the underlying tools, and therefore these issues should be investigated in the early steps of the deployment process.

The Google Android platform appears to be the best smartphone platform to adequately support location-based services and applications running in the background that is available from many different device manufactures and all major U.S. cellular carriers. The Traffic TTS application, partially based on the existing TRAC-IT Java Micro Edition (Java ME) mobile application created by USF, was developed as part of this project to deliver alerts to users only when they are traveling at speeds below a threshold, and reads the message aloud to the user using a TTS API. It was created on the Android platform and successfully tested on several devices from different cellular carriers and device manufacturers. Google Android is truly an open development platform, allowing developers to use location-based services based on technology such as integrated GPS without needing the permission of the wireless carrier, as was common with the Java ME platform. Battery life is also likely to be even more of an issue on smartphones than on lower-end feature phones, so future research should examine the impact of Traffic TTS and TRAC-IT on smartphones. In future research, TRAC-IT could also be extended to other smartphone platforms such as iPhone's iOS and RIM's

Blackberry OS as well to reach additional smartphone users. Since Blackberry OS utilizes Java ME, much of the existing TRAC-IT mobile software could be reused. For all these mobile applications, the same server-side TRAC-IT system code could be reused.

Future research should also examine the scalability of the Path Prediction system. Since numerous spatial queries are executed in a short amount of time, the capabilities of commercial databases to support such queries should be investigated. The accuracy of the Naïve Bayes method implemented in this paper for Destination and Departure Time predictions should also be examined and compared against other more sophisticated probabilistic methods such as Markov models. Departure Time predictions can also be subdivided beyond hours so that every 15 minutes is defined as a “block” of time. The resulting accuracy of Departure Time predictions based on these modifications should also be analyzed.

Further work should be performed to better understand the scalability benefits of Fast GPS Clustering over traditional Hierarchical Clustering on a larger number of datasets. Pseudo-POIs, an unintended byproduct of Fast GPS Clustering, identify spatial areas and times of traveler delay or traffic congestion and have great potential to provide numerous services such as informing traffic signal re-timing, road enhancements, automated vehicle idling detection, and predictive traffic alerts or location-based advertising based on historical travel patterns. Future work should examine the use of pseudo-POIs for these, and other, transportation applications.

Future work could also enable the TRAC-IT mobile app to effectively act as “probes,” using GPS speeds from users (with their permission), in order to feed a road traffic speed estimation system, such as that maintained by INRIX. This would enable road travel speed estimation for arterials and other roads not monitored by installed sensors.

DOTs and researchers should examine how many travel information emails and text messages are sent out per each incident, as too many messages can quickly result in message fatigue and may result in fewer users as a result of cancelled subscriptions. DOTs and researchers may be able to draw upon studies from the field of marketing and advertising to determine what a “reasonable” frequency of traffic alerts would be before users begin ignoring messages. The current design of sending messages out when an incident is created, updated, and cleared should be revisited in light of this research to determine if there is an alternate design methodology which could result in fewer messages per incident. For example, a traffic alert message could be given an estimated time-to-live, or expected duration, in the initial alert sent to the user, such as:

This alert is due to a new traffic incident: [1/6/2011 9:24:12 AM]
Region Tampa Bay
County Hillsborough
Highway I-275
Type Incidents
Severity intermediate
Reported At 1/6/2011 9:22:43 AM
Description Crash in Hillsborough on I-275 south ramp to Exit 39 Memorial Hwy, off-ramp right lane blocked. Last updated at 09:22:59AM.
Expected duration: 20 minutes

Then, additional emails could be sent only if the expected duration changes or if the actual duration extends beyond the expected duration. This technique would reduce the number of messages sent to the traveler by removing some of the “cleared” alerts for the incidents that have an estimated duration that was predicted correctly. Another method is to allow users to choose to receive message summary digests at given intervals (e.g., 15 minutes or 1 hour) that would contain all changes to incident statuses that occurred during this time

period, instead of the current design that pushes a message out for every alert, even if they are seconds apart.

DOTs and researchers should also consider the integration of real-time transit information when they are constructing state-wide traveler information systems. Currently, no real-time transit information appears in data available from the FL511 API system. To achieve real-time location-based multimodal traveler information within a single mobile application, the research team integrated data from a separate API maintained by HART that provides estimated arrival times for each bus stop. HART and other transit information could be integrated into the FL511 system to provide multimodal traveler information via a single statewide API.

REFERENCES

- [1] Mala Raman and Carol Schweiger. 2003. "Guidance for Developing and Deploying Real-time Traveler Information Systems for Transit." Federal Transit Administration, http://ntl.bts.gov/lib/23000/23600/23663/RTTIS_Final.pdf
- [2] 23 CFR Part 511 [FHWA Docket No. FHWA-2006-24219] RIN 2125-AF19 Real-Time System Management Information Program, FHWA, DOT. January 14, 2009, accessed February 10, 2009 at <http://edocket.access.gpo.gov/2009/pdf/E9-392.pdf>
- [3] National Center for Transit Research. 2008. "Smart Phone Application to Influence Travel Behavior (TRAC-IT Phase 3)." Center for Urban Transportation Research, University of South Florida. <http://www.nctr.usf.edu/abstracts/abs77709.htm>
- [4] INRIX TRAFFIC! Website, <http://www.inrixtraffic.com/>.
- [5] Florida's Statewide 511 website, <http://www.fl511.com/511FAQ.aspx?lang=en>.
- [6] Sean J. Barbeau, Miguel A. Labrador, Nevine L. Georggi, Philip L. Winters, and Rafael A. Perez. 2009. "TRAC-IT: A Software Architecture Supporting Simultaneous Travel Behavior Data Collection and Real-Time Location-Based Services for GPS-Enabled Mobile Phones." Proceedings of the National Academy of Sciences' Transportation Research Board 88th Annual Meeting, Paper #09-3175, January.
- [7] Sean J. Barbeau, Rafael A. Perez, Miguel A. Labrador, Alfredo Perez, Philip L. Winters, and Nevine L. Georggi. "LAISYC – A Location-Aware Framework to Support Intelligent Real-time Applications for GPS-enabled Mobile Phones." Submitted to be published in IEEE Pervasive Computing in 2011.
- [8] Philip L. Winters, Sean J. Barbeau, and Nevine L. Georggi. 2008. "TRAC-IT: Traveling Smart: Increasing Transit Ridership by Automatic Collection (TRAC) of Individual Travel Behavior Data and Personalized Feedback: Phase 2." Final Report, National Center for Transit Research and Florida Department of Transportation, March.
- [9] Sean J. Barbeau, Nevine L. Georggi, and Philip L. Winters. 2010. "Integration of GPS-Enabled Mobile Phones and AVL: Personalized Real-Time Transit Navigation Information on Your Phone." Proceedings of the National Academy of Sciences' Transportation Research Board 89th Annual Meeting, Paper # 10-2571, January.
- [10] Narin Persad-Maharaj, Sean J. Barbeau, Miguel A. Labrador, Philip L. Winters, Rafael Perez, and Nevine Labib Georggi. 2008. "Real-time Travel Path Prediction using GPS-enabled Mobile Phones." 15th World Congress on Intelligent Transportation Systems, New York, Paper #30413, November 16-20.
- [11] Murakami, E., Wagner, D. P., Neumeister, D. M. (1997) "Using Global Positioning System and Personal Digital Assistants for Personal Travel Surveys in the United States," International Conference on Transport Survey Quality and Innovation, Grainau, Germany. http://gulliver.trb.org/publications/circulars/ec008/session_b.pdf.
- [12] Murakami, E. and D. P. Wagner. (1999) Can using Global Positioning System (GPS) improve trip reporting? *Transportation Research Part C*, 7(2/3):149-165.
- [13] Qing Cao, B. Bouqata, P. D. Mackenzie, D. Messier, and J. J. Salvo. 2009. "A Grid-based Clustering Method for Mining Frequent Trips from Large-scale, Event-based Telematics Datasets." IEEE International Conference on Systems, Man and Cybernetics, accessed January 11 <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5345924&isnumber=5345886>.

- [14] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. 1996. "BIRCH: An Efficient Data Clustering Method for Very Large Databases." SIGMOD Rec. 25(2): 103-114.
- [15] YuFang Dan and Zhongshi He. 2010. "A Dynamic Model for Urban Population Density Estimation Using Mobile Phone Location Data." Industrial Electronics and Applications (ICIEA), 5th IEEE Conference, accessed January 11, <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5514844&isnumber=5514667>.
- [16] Marta C. González, César A. Hidalgo, and Albert-László Barabási. 2008. "Understanding Human Mobility Patterns." Nature, June, 779-782, accessed January 11, <http://www.nature.com/nature/journal/v453/n7196/full/nature06958.html>.
- [17] Sean J. Barbeau, Miguel A. Labrador, Alfredo Perez, Philip Winters, Nevine Georggi, David Aguilar, and Rafael Perez. 2008. "Dynamic Management of Real-Time Location Data on GPS-enabled Mobile Phones." Presented at UBICOMM 2008 – The Second International Conference on Mobile Ubiquitous Computing, Systems, Services, and Technologies, Valencia, Spain, September 29 – October 4.
- [18] Matt Richtel. 2009. "In Study, Texting Lifts Crash Risk by Large Margin," NY Times, July 27, <http://www.nytimes.com/2009/07/28/technology/28texting.html>.
- [19] The NPD Group. "Motorola, HTC drive Android to Smartphone OS lead in the U.S.," August 4, 2010. http://www.npd.com/press/releases/press_100804.html
- [20] Gartner. "Gartner Says Android to Become No. 2 Worldwide Mobile Operating System in 2010 and Challenge Symbian for No. 1 Position by 2014," September 10, 2010. <http://www.gartner.com/it/page.jsp?id=1434613>