

A Location-Aware Architecture Supporting Intelligent Real-Time Mobile Applications

by

Sean J. Barbeau

A dissertation submitted in partial fulfillment  
of the requirements for the degree of  
Doctor of Philosophy  
Department of Computer Science and Engineering  
College of Engineering  
University of South Florida

Co-Major Professor: Rafael Perez, Ph.D.  
Co-Major Professor: Miguel Labrador, Ph.D.  
Hyun Kim, Ph.D.  
Thomas Weller, Ph.D.  
Dewey Rundus, Ph.D.

Date of Approval:  
June 15, 2012

Keywords: global positioning systems, location-based services,  
mobile phone, Java Micro Edition, Android

Copyright © 2012, Sean J. Barbeau

UMI Number: 3518695

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 3518695

Published by ProQuest LLC 2012. Copyright in the Dissertation held by the Author.

Microform Edition © ProQuest LLC.

All rights reserved. This work is protected against unauthorized copying under Title 17, United States Code.



ProQuest LLC  
789 East Eisenhower Parkway  
P.O. Box 1346  
Ann Arbor, MI 48106-1346

## **DEDICATION**

This work is dedicated to my family and friends, especially my wonderful, beautiful, loving, and supportive wife Carlene. I love you more than you will ever know. This is also dedicated to Zach, my new son – I hope that this work will inspire you and show that with hard work, dedication, and the support of loved ones anything is possible

.

## **ACKNOWLEDGMENTS**

I would like to thank my major professors, Dr. Rafael Perez and Dr. Miguel Labrador, for their mentoring, patience, and guidance throughout my untraditional doctoral journey. I would also like to acknowledge the feedback from my committee, including Dr. Rafael Perez, Dr. Miguel Labrador, Dr. Hyun Kim, Dr. Thomas Weller, and Dr. Dewey Rundus, who helped shape and revise this research.

I would like to thank Phil Winters, my supervisor at the Center for Urban Transportation Research (CUTR), for his trust and supervision as I built my research career. Thank you to Nevine Georggi and Ed Hillsman for their partnership on many research projects and the rest of the CUTR Transportation Demand Management Team for their support and collaborations. Thank you as well to CUTR Management for their support of the many research projects that I have been a part of at CUTR. I would also like to thank the many Research Experience for Undergraduates (REU) and graduate students who have contributed in many different ways to the research projects I have been a part of surrounding the work in this dissertation: Alfredo Perez, Isaac Taylor, Marcy Gordon, Khoa Tran, Leon Augustine, David Aguilar, Josh Kuhn, Ismael Roman, Oscar Lara, Narin Persad, Dmitry Belov, Jeremy Weinstein, Paola Gonzalez, Tiffany Burrell, Francis Gelderloos, Joksan Flores, Jorge Castro, Richard Meana, Theo Larkins, Hector Tosado, and Marcel Munoz.

I would like to thank the organizations that contributed funding to the many research projects I have worked on, especially the Travel Assistance Device and TRAC-IT projects that built on my dissertation research, including the National Center for Transit Research, Florida Department of Transportation, U.S. Department of Transportation Research and Innovative Technology Administration, Federal Transit Administration, Transportation Research Board, and National Science Foundation.

I am grateful for the support of Sprint-Nextel's Application Developer program, and in particular the assistance of Ryan Wick, Sprint's Lead Developer Advocate, in providing access to the Location API on Sprint devices as well as facilitating the donation of cellular service and devices to USF that supported our research projects.

Last but certainly not least, I would like to thank my family and friends for their love and moral support throughout my years as a full and part-time student. Thank you to my wonderful wife Carlene, without whom this dissertation and the rest of my graduate work never would have been completed – she has put in more hours supporting me than I have working on my research. Thanks to Zach, my son, for being my inspiration. Thank you to my loving and supportive Mom and Dad, who provided crucial support and instilled a love of learning early in my life, and Momma Brown and Matt who have provided love and support in my college and post-college years. Thank you to my brother Ryan, for his many years of friendship and humor, and sister-in-law Daphna for her constant moral support despite juggling med school (Congrats Dr. Daphna!). Thanks to Sugar (I miss you!) for her companionship during the many hours studying, even if she spent most of it barking at squirrels.

## TABLE OF CONTENTS

LIST OF TABLES .....	iii
LIST OF FIGURES .....	v
ABSTRACT .....	xi
CHAPTER 1: INTRODUCTION .....	1
1.1 Mobile Applications.....	2
1.2 Positioning Technologies.....	2
1.3 Location-Aware Mobile Applications .....	4
1.3.1 Cross-Platform Application Environments.....	5
1.3.2 Multitasking Virtual Machines .....	6
1.4 Problem Statement .....	7
1.5 Contributions.....	11
1.6 Structure of Dissertation .....	14
CHAPTER 2: KNOWN LBS ARCHITECTURES .....	15
2.1 Commercial LBS Applications .....	15
2.2 Known Location-Aware Architectures .....	18
CHAPTER 3: PROPOSED ARCHITECTURE – LOCATION-AWARE INFORMATION SYSTEMS CLIENT (LAISYC) .....	29
3.1 Note to Reader .....	29
3.2 Architecture Overview.....	30
3.3 Mobile Device-Side Components .....	31
3.3.1 Positioning Systems Management Modules .....	33
3.3.1.1 GPS Auto-Sleep .....	33
3.3.1.2 Location Data Signing.....	49
3.3.2 Communications Management Modules .....	51
3.3.2.1 Session Management.....	51
3.3.2.1.1 Available Communication Protocols .....	52
3.3.2.1.2 LAISYC Application Data Transport .....	54
3.3.2.1.3 LAISYC Location Data Transport .....	59
3.3.2.1.4 Device-Side Implementation of Session Management.....	60
3.3.2.2 Adaptive Location Data Buffering.....	63
3.3.2.3 Critical Point Algorithm.....	70
3.3.2.4 Location Data Encryption .....	79

3.4	Server-Side Components .....	83
3.4.1	Communications Management .....	84
3.4.1.1	Session Management.....	84
3.4.1.2	Adaptive Location Data Buffering.....	86
3.4.2	Data Analysis .....	86
3.4.2.1	Critical Point Algorithm.....	86
3.4.2.2	Spatial Analysis.....	87
CHAPTER 4: EVALUATION .....		89
4.1	Note to Reader .....	89
4.2	Evaluation Overview .....	90
4.3	LAISYC Component Evaluation .....	90
4.3.1	GPS Auto-Sleep.....	91
4.3.2	Location Data Signing .....	117
4.3.3	Session Management and Adaptive Location Data Buffering....	121
4.3.4	Critical Point Algorithm .....	129
4.3.5	Location Data Encryption.....	150
4.4	Innovative Location-Aware Applications Developed Using LAISYC ..	151
4.4.1	TRAC-IT.....	151
4.4.2	Travel Assistance Device (TAD).....	161
CHAPTER 5: SUMMARY AND CONCLUSIONS .....		179
5.1	Note to Reader .....	179
5.2	Summary of Problem Statement and Needs .....	179
5.3	Summary of Contributions.....	181
5.4	Future Work.....	187
5.4.1	Location-Aware Mobile App Development .....	187
5.4.2	Potential LAISYC Improvements.....	188
5.4.2.1	GPS Auto-Sleep .....	188
5.4.2.2	Critical Point Algorithm.....	192
5.4.2.3	Location Data Buffering.....	192
5.4.2.4	Position Estimation .....	193
5.4.2.5	Privacy Filter .....	193
LIST OF REFERENCES.....		196
APPENDIX A. REPRINT PERMISSIONS .....		212
ABOUT THE AUTHOR .....		END PAGE

## LIST OF TABLES

Table 1 - The Location-Aware Information SYstems Client (LAISYC) modules are designed to meet the various critical needs of intelligent real-time mobile applications in Location-Based Services.....	12
Table 2 - SOAP-encoded messages add a significant amount of overhead to web service requests, approximately 3.7 times as many characters, as shown in this example.....	56
Table 3 - GPS Auto-Sleep state machine values chosen for experimentation.....	95
Table 4 - Horizontal error statistics for indoor GPS accuracy tests.....	109
Table 5 - While the positional error between the two devices is substantially different, the error in speed is much less dramatic.....	110
Table 6 - When using the 0.1 meters per second min_speed_threshold, the Critical Point Algorithm is able to produce significant data filtering savings with only a slight impact on accurate walking paths.....	138
Table 7 - Resulting statistics from a walk and a car trip that were both processed using the Critical Point Algorithm with different angle thresholds .....	147
Table 8 – The Critical Point Algorithm is able to reduce GPS datasets by more than 77% on average while maintaining an average distance error percentage under 10%. .....	148
Table 9 - TRAC-IT was used as part of a USDOT-funded research project to collect over 4 million GPS data points from 30 users over 2 months .....	157
Table 10 - 95% of sessions had less than 3.95% of lost UDP packets .....	157
Table 11 - When TRAC-IT used LAISYC, device battery life nearly doubled while reducing overall location data packet loss by 2.16% and adding encryption.....	158
Table 12 - Field tests of the TAD app in Tampa, Florida produced ideal prompts 87% of the time at random stops.....	173



Table 13 - Field tests of TAD with STAGES students were more challenging, primarily due to close proximity of stops near the USF campus .....	174
Table 14 - The improved bus stop detection algorithm delivered ideally-timed alerts to riders in 33 of 33 tests .....	176

## LIST OF FIGURES

Figure 1 - The LAISYC architecture consists of software on the mobile device and web application server, with a database server holding persistent server-side data .....	31
Figure 2 - LAISYC mobile phone-based modules .....	32
Figure 3 - High-sensitivity GPS receivers can acquire a GPS position more rapidly, and with less dependence on the time elapsed since the most recent GPS fix, than low-sensitivity receivers .....	36
Figure 4 - GPS Auto-Sleep uses a state machine with various logic evaluations that control the transition between states, which represent changes to the GPS sampling interval values .....	39
Figure 5 - Navigation mode for GPS Auto-Sleep controls GPS sampling interval directly based on a distance-to-goal (e.g., next turn for real-time driving directions) .....	47
Figure 6 - Relationships between HTTP, TCP, UDP, and SOAP as networking protocols .....	53
Figure 7 - The Session Management modules use HTTP for application data and UDP for location data for communication between the mobile device and server .....	61
Figure 8 - A timeline of Location Data Buffering which shows a TCP failure that results in a series of buffered location data fixes, which are transmitted to the server on the next successful TCP transmission .....	67
Figure 9 - Adaptive Location Data Buffering occasionally checks for an open connection with the server via TCP to increase the probability of successful UDP transmissions .....	69
Figure 10 - The Critical Point Algorithm filters out GPS fixes that are not necessary to recreate the user's path .....	74

Figure 11 - Azimuth calculations are used in the Critical Point Algorithm to determine change in direction.....	75
Figure 12 - The Critical Point Algorithm maintains a reference to three points that are used to determine whether the second of the three points is a critical point .....	75
Figure 13 - LAISYC uses a hybrid cryptosystem to protect the exchange of the AES key using HTTPS with SSL, and then uses the AES key to encrypt the location data sent over UDP.....	81
Figure 14 - 128bit AES is used to encrypt the location data in the UDP payload, with the exception of the session ID which is used by the server to identify the correct symmetric key per device session .....	82
Figure 15 - LAISYC server-side modules .....	83
Figure 16 - Even modest increases in the interval between GPS fixes produce extended battery life on the order of hours .....	93
Figure 17 - A growth function for the state[i] <sub>interval</sub> values was chosen to grow like an $x^2$ or $2^x$ function until it reaches the middle state, at which point it quickly accelerates in growth beyond an $x^3$ function .....	97
Figure 18 - Sample GPS Auto-Sleep values are chosen for an exponential growth in the interval between GPS fixes, while the timeout values have an upper-bound of 32 seconds; if a GPS fix cannot be acquired, the interval + timeout line illustrates an upper bound for the total time elapsed at each state.....	98
Figure 19 - The largest potential loss of beginning travel path is worst-case scenario when the user travel path is sampled just before they begin moving, since the next GPS sample occurs $\max\_gps\_activity_{state[n]}$ seconds later.....	99
Figure 20 – When high-sensitivity GPS is able to acquire a fix, it tends to deliver this information close to the expected interval value with an average delay of only 9 seconds.....	102
Figure 21 - Proactive GPS scheduling (left) starts the GPS hardware slightly before the scheduled interval value expires, while reactive GPS scheduling (right) waits until the interval period has completely expired before attempting a GPS fix.....	103

Figure 22 - GPS Auto-Sleep can miss a substantial part of the beginning trip path if it must transition through all states before starting to record high-resolution travel behavior .....	104
Figure 23 - Speed thresholds for the GPS Auto-Sleep state machine are selected using observations of speed when stationary and indoors .....	106
Figure 24 - GPS Auto-Sleep can quickly react to real movement using the <i>high_speed_threshold</i> and rapidly begin sampling GPS via direct transitions to state[0] to reflect a more accurate travel path .....	107
Figure 25 - Scatter plots of indoor horizontal positional accuracy tests.....	109
Figure 26 - Reliability of accuracy estimates for individual assisted GPS data points was shown to be poor on the evaluated devices, the Motorola i580 (left) and Sanyo 7050 (right) .....	111
Figure 27 - To evaluate the accuracy of GPS Auto-Sleep, the ground truth state of traveling was manually coded against the behavior of the state machine .....	113
Figure 28 - GPS Auto-Sleep is able to successfully track the moving or stationary state of the user with a high degree of accuracy. ....	115
Figure 29 - Execution time for key generation using DSA and RSA asymmetric cryptography .....	118
Figure 30 - Signature generation test results show that Location Data Signing using DSA and RSA is feasible for implementation on real mobile devices.....	119
Figure 31 - Estimated battery life with and without Location Data Signing .....	120
Figure 32 - The information exchanged between the mobile device and server for the HTTP POST vs. XML-based JAX-RPC battery life tests .....	124
Figure 33 - XML-based JAX-RPC mobile device to server communication clearly has a substantial negative impact on mobile device battery life when compared to HTTP-POST.....	124
Figure 34 - The location data format used for the payload contents of UDP and TCP packets in the power consumption tests .....	126

Figure 35 - (a) While at 4 second transmission intervals TCP and UDP have similar power consumption, (b) at 10 second transmission intervals it is evident that TCP consumes approximately 38% more power than UDP.....	127
Figure 36 - a) All GPS data points generated from a phone are shown on the left, while b) only the critical points generated by the Critical Point Algorithm are shown in the right .....	129
Figure 37 - The Critical Point Algorithm can more than triple battery life by filtering GPS data and transmitting at an interval of 60 seconds instead of 15 seconds.....	131
Figure 38 - The Critical Point Algorithm maintains a constant memory requirement during execution by using at most three location data pointers.....	133
Figure 39 - We observed the GPS speed recorded while a user was casually walking, which includes some speed values of 0 meters per second.....	135
Figure 40 - When comparing a) all points to b) critical points using a min_speed_threshold of 0.1 meters per second, the general walking path of the user is preserved, with some filtering at the beginning of the trip (bottom left of each image). .....	136
Figure 41 - Over 97% of the GPS drift shown here at an indoor stationary location can be filtered out by the Critical Point Algorithm when using a 0.1 meters per second min_speed_threshold .....	138
Figure 42 - Sampled GPS data points create an approximated path of the user with some uncertainty .....	139
Figure 43 - The distance of the path generated from Critical Point Algorithm will always be shorter or equal to the distance of the path using all GPS data points .....	141
Figure 44 - Running the Critical Point Algorithm with increasing angle thresholds gradually reduces the number of points that represent the line, which increases the distance error percentage.....	143
Figure 45 - As the angle threshold for the Critical Point Algorithm increases, there is a general trend towards fewer critical points being generated and an increase in the distance error percentage for both walking and car .....	144

Figure 46 - For car trips, the Critical Point Algorithm is able to dramatically reduce the full GPS dataset, a), to far fewer critical points , b), with lower angle_threshold values because of longer straight paths .....	145
Figure 47 - Location Data Encryption using 128-bit AES encryption for UDP payloads is feasible on mobile devices, although it does have a slight impact on battery life .....	150
Figure 48 - The TRAC-IT mobile application is based on the LAISYC framework to enable simultaneous travel behavior data collection and real-time location-based services .....	153
Figure 49 - The TRAC-IT mobile application provides a user interface to record input from the individual for mode of transportation, purpose, and vehicle occupancy as well as location data.....	155
Figure 50 - Path Prediction compares the traveler's real-time location, shown as yellow push-pin markers, against paths from the traveler's travel history, shown as yellow shaded buffers, to predict the immediate travel path.....	159
Figure 51 - Path Prediction successfully demonstrated that real-time location-based messages could be sent to the phone using LAISYC and a history of the traveler's behavior .....	160
Figure 52 - The Travel Assistance Device mobile application alerts the transit rider of an upcoming destination bus stop .....	162
Figure 53 - TAD was implemented using the LAISYC framework to support real-time location-aware services.....	164
Figure 54 - New transit trip itineraries can be created for the TAD mobile app user via the TAD website.....	165
Figure 55 - Travel Assistance Device mobile app interface that alerts the rider when to exit the bus .....	166
Figure 56 - The initial bus stop detection algorithm for the Pull the Cord Now alert was defined by a radius surrounding the destination stop .....	167
Figure 57 - The TAD website shows real-time location updates from the LAISYC framework supporting the TAD mobile and web app .....	170

Figure 58 - LAISYC Spatial Analysis module on the server compares the real-time location of the user against spatial buffers surrounding the rider's planned route, to determine if the user has become lost .....	171
Figure 59 - The planned travel path of the bus is used to detect if the rider is lost, versus an estimated path created by connecting bus stop locations, since an estimated path can produce false-positive lost alerts .....	172
Figure 60 - Some TAD alerts were given early or late due to incorrectly geocoded bus stops, where the actual bus stop position (marker "A") differed from the database location of the bus stop (blue bus icon).....	174
Figure 61 - An improved algorithm for notifying the user when to exit the bus is based on detecting the departure from the second-to-last bus stop [126].....	176
Figure 62 - Battery life issues related to GPS appear to be an even bigger challenge with smart phones, including Android devices .....	185
Figure 63 - Future work on LAISYC can include the addition of two new modules: Privacy Filter and Position Estimation.....	188

## **ABSTRACT**

This dissertation presents LAISYC, a modular location-aware architecture for intelligent real-time mobile applications that is fully-implementable by third party mobile app developers and supports high-precision and high-accuracy positioning systems such as GPS. LAISYC significantly improves device battery life, provides location data authenticity, ensures security of location data, and significantly reduces the amount of data transferred between the phone and server. The design, implementation, and evaluation of LAISYC using real mobile phones include the following modules: the GPS Auto-Sleep module saves battery energy when using GPS, maintaining acceptable movement tracking (approximately 89% accuracy) with an approximate average doubling of battery life. The Location Data Signing module adds energy-efficient data authenticity to this architecture that is missing in other architectures, with an average approximate battery life decrease of only 7%. The Session Management and Adaptive Location Data Buffering modules also contribute to battery life savings by providing energy-efficient real-time data communication between a mobile phone and server, increasing the average battery life for application data transfer by approximately 28% and reducing the average energy cost for location data transfer by approximately 38%. The Critical Point Algorithm module further reduces battery energy expenditures and the amount of data transferred between the mobile phone and server by eliminating non-essential GPS data (an average 77% reduction), with an average doubling of battery life as the interval of



time between location data transmissions is doubled. The Location Data Encryption module ensures the security of the location data being transferred, with only a slight impact on battery life (i.e., a decrease of 4.9%). The LAISYC architecture was validated in two innovative mobile apps that would not be possible without LAISYC due to energy and data transfer constraints. The first mobile app, TRAC-IT, is a multi-modal travel behavior data collection tool that can provide simultaneous real-time location-based services. In TRAC-IT, the GPS Auto-Sleep, Session Management, Adaptive Location Data Buffering, Critical Point algorithm, and the Session Management modules all contribute energy savings that enable the phone's battery to last an entire day during real-time high-resolution GPS tracking. High-resolution real-time GPS tracking is critical to TRAC-IT for reconstructing detailed travel path information, including distance traveled, as well as providing predictive, personalized traffic alerts based on historical and real-time data. The Location Data Signing module allows transportation analysts to trust information that is recorded by the application, while the Location Data Encryption module protects the privacy of users' location information. The Session Management, Adaptive Location Data Buffering, and Critical Point algorithm modules allow TRAC-IT to avoid data overage costs on phones with limited data plans while still supporting real-time location data communication. The Adaptive Location Data Buffering module prevents tracking data from being lost when the user is outside network coverage or is on a voice call for networks that do not support simultaneous voice and data communications. The second mobile app, the Travel Assistance Device (TAD), assists transit riders with intellectual disabilities by prompting them when to exit the bus as well as tracking the rider in real-time and alerting caregivers if they are lost. In the most

recent group of TAD field tests in Tampa, Florida, TAD provided the alert in the ideal location to transit riders in 100% (n = 33) of tests. In TAD, the GPS Auto-Sleep, Session Management, Adaptive Location Data Buffering, Critical Point algorithm, and the Session Management modules all contribute energy savings that enable the phone's battery to last an entire day during real-time high-resolution GPS tracking. High-resolution GPS tracking is critical to TAD for providing accurate instructions to the transit rider when to exit the bus as well as tracking an accurate location of the traveler so that caregivers can be alerted if the rider becomes lost. The Location Data Encryption module protects the privacy of the transit rider while they are being tracked. The Session Management, Adaptive Location Data Buffering, and Critical Point algorithm modules allow TAD to avoid data overage costs on phones with limited data plans while still supporting real-time location data communication for the TAD tracking alert features. Adaptive Location Data Buffering module prevents transit rider location data from being lost when the user is outside network coverage or is on a voice call for networks that do not support simultaneous voice and data communications.

## **CHAPTER 1: INTRODUCTION**

Mobile phones have become one of the most ubiquitous computing devices in modern history. As a result of mass production, cellular carrier subsidies, and decreasing technology costs, more people have access to mobile phones today than any other time in world history. As of late 2011, there were an estimated 5.9 billion mobile-cellular subscriptions worldwide yielding a global penetration rate of 87%, with a 79% penetration rate in developing countries [1].

In developed countries such as the United States, mobile phones are becoming so common that wireless penetration is reaching the point of saturation with only a small percentage of the population not owning mobile phones. For example, in the United States as of June 2011 there are 322.9 million mobile subscriptions with a penetration rate of 102.4%, indicating that a large number of individuals have multiple subscriptions [2]. A contributing factor to this growth is that many individuals are giving up their landline telephones in favor of mobile phones. In April 2011, 26.6% of U.S. households were wireless-only, meaning that they use only a cell phone instead of a landline telephone to make calls [3]. As a result of increasing penetration and reliance on cell phones for a variety of everyday tasks, mobile phones have become important devices to many individuals around the world. A 2009 survey indicates that 82% of Americans never leave their house without their phone, while 42% stated “they cannot live without their phone” [4].

## 1.1 Mobile Applications

Cell phones have become immensely popular not only for their ability to make phone calls, but also for their ability to perform general computing tasks that previously required expensive personal computers. Perhaps one of the most popular features of modern smart phones is the ability to execute mobile applications. Mobile applications, or “apps,” are software products that are typically developed by a third-party that does not have a direct relationship with the device manufacturer (e.g., HTC, Samsung, Motorola, Apple, Research in Motion), cellular carrier (e.g., Sprint-Nextel, AT&T, Verizon Wireless), or operating system vendor (e.g., Google, Microsoft). Instead, the mobile app is created by software engineers and then directly sold and distributed to the customer, often through online software vending services such as the Google Android Market [5], Apple AppStore for the iPhone [6], Blackberry AppWorld [7], Amazon AppStore for Android [8], and GetJar for Java Micro Edition and Android [9]. As a result of these vending services and an increasing availability of smart phones, the number of mobile apps downloaded has proliferated over the last few years. An estimated 29 billion apps were downloaded worldwide in 2011 [10], an astounding increase of 20 billion downloads since 2010 [10]. Revenues for app developers are expected to increase rapidly over the next few years, with an estimated global app revenue of \$7.3 billion in 2011 and \$36.7 billion by 2015 [11].

## 1.2 Positioning Technologies

One key difference between mobile phones and desktop computers is that mobile phones constantly change geographic location, unlike desktop computers, which are tethered to a single physical location for months or years. Even laptops do not have the level of

mobility that cell phones offer. Laptops can be moved from one place to another, but typically they are in operation for only several hours at a time and then shut down before being moved. In contrast, mobile phones typically remain on during the entire day and can be actively used when the user is in motion.

During the emergence of cell phones in the late 1990s, the U.S. Federal Communication Commission (FCC) became concerned that extreme mobility of cell phones could cause problems for emergency responders attempting to locate a mobile 911 caller, since, unlike a landline phone that is associated with a street address, little is known about the real-time location of a mobile phone. Even if the 911 operator knows what cellular tower a mobile phone is communicating with, this information is of little help to responders since the coverage area of a single cell tower can be several square miles. As a result of the lack of positional knowledge for mobile 911 callers, the FCC issued the E911 mandate, requiring cellular carriers to implement technologies that could accurately locate mobile 911 callers within 50 to 300 meters, depending on the underlying technology [12]. U.S. carriers tested a wide variety of positioning technologies for their networks. Global System for Mobile Communication (GSM)-based U.S. carriers such as AT&T and T-Mobile chose network-based Uplink Time Difference of Arrival (U-TDOA) to support E911 position requests [13]. Code Division Multiple Access (CDMA)-based U.S. carriers such as Sprint and Verizon chose handset-based Global Positioning System (GPS) solutions for devices on their networks because GPS technology was already integrated into the network as a time reference for CDMA-based wireless communications [13, 14].

Since U.S. cellular carriers were mandated to invest a significant amount of time, effort, and funds into positioning technology implementations, carriers immediately began to investigate commercial applications of these technologies for mobile phone users so they could recover a portion of their investments through user fees. Early deployments of these technologies for commercial purposes become known as location-based services (LBS), which are a general class of services that provide users with some type of information based on their real-time or historical location.

Of the positioning technologies implemented for E911 purposes, GPS-based solutions are by far the most accurate, with an estimated 3-5 meters of positional accuracy under ideal conditions [15-19]. Since this level of accuracy is also sufficient to provide commercial services such as real-time driving directions to mobile phone users, GPS became an attractive technology not only for E911 purposes but also for general consumer LBS. As a result, U.S. T-Mobile and AT&T have since implemented GPS-based positioning technologies in their handsets in order to provide commercial services based on the technology [14]. Global trends of GPS penetration in handsets to support commercial services have also surged upwards, with 79.9% of cell phones shipped in the fourth quarter of 2011 (318.3M) having integrated GPS [20].

### 1.3 Location-Aware Mobile Applications

With the availability of positioning technologies such as GPS in mobile phones, and the advent of apps, third-party application developers became interested in utilizing location information within their applications. There were two major developments in mobile phones that made widely deployable location-aware mobile applications possible: the

emergence of cross-platform application environments for mobile phones such as Java 2 Micro Edition, now referred to as Java Micro Edition (Java ME), and the ability to run applications in the background (i.e., a Multitasking Virtual Machine). Both developments are discussed below.

### 1.3.1 Cross-Platform Application Environments

The diversity and rapid evolution of mobile phone hardware creates a significant challenge for application developers. If the developer were to design and implement software that directly interfaced with the hardware and operating system for each phone, they would be forced to redesign the application for nearly every single mobile phone model that is released by each manufacturer, an extremely costly task. To ease the burden on developers and create a sustainable mobile application ecosystem, applications platforms that hide some of the lower-level detail of the hardware and operating system (OS) implementation have emerged. Instead of directly accessing these hardware and OS components, application instead interact with interfaces that abstract the underlying implementation details. This design allows the underlying hardware or OS to change and evolve without modifying the higher-level interfaces. Applications can therefore indirectly interact with the underlying hardware without the burden of rapidly redesigning their applications for every new mobile phone model.

Java ME, designed after the cross-platform Java virtual machines initially created for portability of desktop and server applications, was the first cross-platform application environment to emerge for mobile phones. Google's Android is a newer cross-platform environment for smart phones that has recently emerged, although in this dissertation the

majority of focus is on Java ME since at the time of this research Java ME was the primary cross-platform environment that was widely accepted in the telecommunications industry [21, 22].

One drawback to the standardization of high-level application programming interfaces across multiple hardware and operating system platforms is that there must be consensus in the industry for how this interface is designed, and this can take time to develop. For example, the introduction of positioning technologies in mobile phones for E911 purposes in the late 1990s and early 2000s did not mean that this technology was immediately available to third-party application developers. In fact, a location application programming interface (API) was not standardized for Java ME until September 2003 [23]. The Java Specification Request (JSR) 179 Location API for Java ME, and the subsequent JSR 293 Location API 2.0, defined a set of functions that a mobile application developer could use to access location information on a Java ME handset that implemented the JSR 179 or JSR 293 standards [22-24]. For the first time, an application developer could develop a location-aware application that accessed positioning technologies such as GPS and could work on devices from many different manufacturers and cellular carriers without significant modification, a critical development in the emergence of location-aware mobile apps.

### 1.3.2 Multitasking Virtual Machines

The second major development in the emergence of location-aware mobile applications was the ability to run applications in the background. Many of the first Java ME mobile phones released in the early 2000s did not have Multitasking Virtual Machines (MVMs),



which prevented applications from being executed in the background while the user performed a different task (e.g., phone call, web browsing, phone in standby mode) in the foreground. In other words, only a single application could be executed at a time, and that application could not be executed in the background. This limitation prevented an application from monitoring the location of the phone unless the user was actively using the application, which severely restricted the scope of location-aware mobile applications that could be implemented by third party software developers. MVMs for Java ME were introduced in Motorola iDEN phones circa 2004 [25], which opened up opportunities for a new breed of location-aware applications that could monitor and act upon a user's geographic location, even if the user was not actively using the phone.

#### 1.4 Problem Statement

The ubiquity of mobile phones, the availability of positioning systems to application developers, and the popularity of cross-platform mobile apps creates an environment rich for innovation in the area of location-aware applications. However, while location-aware applications have been implementable since the mid-2000s, there have been few popular real-time commercial mobile applications that are based primarily on high-precision and high-accuracy positioning systems (e.g., GPS). The lack of evolution of location-aware apps can be attributed to several key limitations in current commercial applications:

- 1) Commercial location-aware apps are a “black box”
- 2) Commercial location-aware apps require active user management of location features due to impact on device resources (e.g., battery life)
- 3) Commercial location-aware apps are often limited to “locate->send” functionality

- 4) Commercial location-aware apps are often lacking device-based intelligence

These limitations are discussed in the context of existing mobile applications in Chapter 2 of this dissertation.

Typically, architectures discussed in academic literature would gradually address the difficulties faced by location-aware apps and provide solutions that could help advance the industry. However, there has also been little evolution of the capabilities of location-aware architectures over the last 10 years. Due to the potential negative impact of some hybrid positioning technologies (e.g., assisted GPS) on the cellular network, cellular carriers have limited access to Location APIs on Java ME devices to industry partners [22]. Limited access to Location APIs, as well as the significant financial costs of mobile devices and data service plans, have largely reduced academic experimentation to the use of software emulators or laptops as proxies for cell phones. Emulators and laptops are simplistic models of logical program execution for mobile applications and do not appropriately model real-world conditions such as energy consumption of positioning technologies or wireless communication.

Lack of sufficient real-world experimentation with actual mobile devices has produced four primary shortfalls in known location-aware architectures:

- 1) Battery energy limitations are not addressed. Many architectures are designed without acknowledging that mobile devices have a finite energy supply, and that positioning systems such as GPS, wireless communications, and use of the CPU to execute the architecture components all have a significant impact on battery energy levels. Recent research [26-38] confirms that battery life is a significant

limiting factor for mobile applications running on modern mobile devices, and that GPS is a significant consumer of energy [28, 29, 32, 33, 35, 36, 38, 39].

Currently, only two existing location-aware architectures [32, 33] even directly address battery life. Comparison between these two architectures and our research is provided in Chapter 2.

- 2) Cellular data transfer limitations are not addressed. Many architectures are designed without consideration of constrained cellular network bandwidth and potential financial charges to the end-user for excessive data traffic.
- 3) Lack of integration with existing platforms on commercially-available devices (e.g., Java Micro Edition, Android). Many existing location-aware architectures presented in literature utilize custom operating systems or protocols which are not readily available on commercially-available mobile phones, and therefore cannot be widely deployed as mobile apps to existing phones.
- 4) Lack of evaluation of efficacy of location-aware architectures. Few location-aware architectures have actually been evaluated on real mobile devices, and as a result there is little quantifiable evidence of these architectures' efficacy with real devices. Only one existing location-aware architecture performs experiments with actual mobile devices [33], and we compare this location-aware architecture to our research in Chapter 2.

As a result, there is a demand for a new location-aware architecture that meets following needs:

- Need #1: Intelligently manage limited device and network resources. The architecture must acknowledge that location-aware apps can deplete significant device and network resources, and the architecture must demonstrate features that conserve these resources.
- Need #2: Support real-time applications. A significant portion of the architecture must be implemented on the mobile device to allow software to immediately act upon new data in real-time and immediately interact with the mobile user.
- Need #3: Support high-precision and high-accuracy positioning systems. Positioning technologies such as high-sensitivity assisted GPS must be usable within the architecture to support the most innovative types of location-aware apps that require highly accurate and precise location information.
- Need #4: Is fully implementable by third party mobile app developers. The architecture must take into account the availability of application programming interfaces (APIs) in existing cross-platform application environments such as Java Micro Edition or Android and ensure that the architecture can be implemented on such devices.

However, there are many challenges that must be addressed when creating a new architecture that meets these needs. Challenges can be categorized into the following key areas:

- 1) Collecting and acting on real-time data consume limited device resources. When an application is executed to record and process data, this requires use of CPU

and memory resources, which in turn use battery energy, and, if communicating with a server, increases network data traffic

- 2) Using high-precision and high-accuracy positioning systems consume limited device resources. GPS is the most accurate and precise positioning system widely available on mobile phones. However, it is also one of the highest consumers of battery energy, and for assisted or hybrid GPS solutions, network bandwidth.
- 3) Balancing tradeoffs between real-time app requirements and limited device resources is not trivial. Since monitoring and reacting to information also consumes the same limited device resources the software is trying to preserve, there are no simple solutions for highly accurate and precise location-aware applications that are always active.
- 4) Mobile hardware is proprietary and rapidly changing. Hardware and operating system functionality is abstracted by high-level software layers APIs (e.g., Android, Java ME), which limit control of underlying hardware

## 1.5 Contributions

This dissertation presents the Location-Aware Information SYstems Client (LAISYC), a modular mobile software architecture that meets the needs of intelligent real-time mobile applications and is fully implementable by third party mobile application developers.

Table 1 shows the relationship between each LAISYC module and the needs of intelligent real-time mobile applications that it addresses.

**Table 1 - The Location-Aware Information SYstems Client (LAISYC) modules are designed to meet the various critical needs of intelligent real-time mobile applications in Location-Based Services**

<b>LAISYC Modules</b>	<b>Need #1: Intelligently manages limited device/network resources</b>	<b>Need #2: Still supports real-time applications?</b>	<b>Need #3: Supports high-precision and high-accuracy positioning systems</b>	<b>Need #4: Fully implementable by 3<sup>rd</sup> party mobile app developer</b>
<b>Session Management</b>	X	X		X*
<b>GPS Auto-Sleep</b>	X	X	X	X*
<b>Critical Point Algorithm</b>	X	X	X	X
<b>Adaptive Location Buffering</b>	X	X		X*
<b>Location Data Encryption</b>	X	X		X
<b>Location Data Signing</b>	X	X		X

\*Interacts directly with the mobile device platform via Application Programming Interfaces (APIs)

We reference the needs listed in Table 1 throughout this dissertation as we discuss specific examples of how LAISYC meets each need.

Each module in LAISYC has been implemented and tested on mobile devices in Java Micro Edition as part of our research to demonstrate that each module is fully implementable by third party mobile application developers (Need #4). This prototype testing is especially important for the Session Management, GPS Auto-Sleep, and Adaptive Location Buffering modules because they interact with and depend upon features implemented in the mobile device platform. While we discuss the characteristics

of each module in detail in Chapter 3, the following paragraphs briefly state how each module meets the needs, as shown in Table 1.

The general communication framework between the mobile device and server is implemented in the Session Management module using a strategic combination of the HyperText Transfer Protocol (HTTP) [40], used for occasional transfer of application data, and the User Datagram Protocol (UDP) [41], a lightweight connectionless protocol used to transport real-time location data. Chapter 4 of this dissertation presents experiments showing that by using UDP as the main location data transfer protocol instead of the Transmission Control Protocol (TCP) [42], the impact on mobile device battery life is reduced (Need #1) while still supporting real-time location services (Need #2). The Location Data Buffering module supports efficient real-time communication (Needs #2 and #4) by increasing the probability of UDP location data being successfully received by the server via an occasional verification of an open data connection using TCP.

The GPS Auto-Sleep module intelligently adjusts the frequency of GPS recalculations (Need #3) based on the real-time and historical movement of the user (Need #2). This allows high-resolution tracking of the user using GPS when moving with a gradual transition to less frequent GPS fixes when the user stops moving, thereby conserving battery life and network traffic to transfer this data back to the server (Need #1). The Critical Point Algorithm filters a real-time stream of location data points (Need #4) and eliminates redundant points to produce a smaller data set that still accurately represents the path of the mobile device (Need #3). By reducing the amount of data required to send

a device's path from a mobile device to a server, the Critical Point Algorithm reduces the impact of path data transfer on the mobile device battery as well as the amount of information sent over the cellular network (Need #1).

To meet the security and data authentication needs of real-time mobile applications (Need #2), our research also presents the implementation of Location Data Encryption and Location Data Signing modules (Need #4) and evaluates the impact of these technologies on mobile device resources (Need #1).

## 1.6 Structure of Dissertation

The remainder of this dissertation is organized as follows: Chapter 2 provides a detailed review of known LBS architectures discussed in literature and compares existing literature to our work. Chapter 3 presents the proposed LAISYC architecture that is the main subject of this dissertation, and Chapter 4 presents an evaluation of the key LAISYC architecture components as well as two innovative real-time mobile apps, TRAC-IT and the Travel Assistance Device (TAD), that use LAISYC. Chapter 5 concludes the dissertation with an overview of the contributions and future research directions related to LAISYC.



## CHAPTER 2: KNOWN LBS ARCHITECTURES

This chapter reviews existing commercial LBS applications and known LBS system architectures, and explains the current limitations of these technologies.

### 2.1 Commercial LBS Applications

There are a number of LBS applications that are commercially available as of 2011, which can be organized into the following categories:

- **Location data recording:** These apps, such as My Tracks [43], records GPS trails and generates statistics/maps based on the path of the user as the user is biking or hiking. These applications typically store GPS data locally on devices, and can execute a bulk upload of data to online data stores such as Google Docs after an entire track has been recorded.
- **Navigation, mapping, and real-time traffic information:** Apps such as Google Maps [44], Google Navigation [45], Telenav [46], and INRIX [47] provide directions to the user for businesses and other locations and provide real-time turn-by-turn directions and/or real-time or predicted traffic information. These apps typically use GPS for navigation, cell network/Wi-Fi/GPS for location.
- **Social location apps:** Foursquare [48], Facebook [49], and Google Latitude [50] are all examples of applications that allow the user to manually “check in,” which indicates to their friends in their social network that they have arrived at a location.

Some apps, such as AT&T FamilyMap [51] and Sprint Family Locator [52], are designed to allow parents to see where a child is, based on the location of their child's phone.

- Location-based search catalogs: WHERE [53] and Poynt [54] are examples of location-based search engines that allow a social search of places based on the user's friends' ratings. They can also provide electronic coupons, local gas prices, and local weather information. WeatherBug [55] also provides local weather information.
- Phone finders: Apps such as Find My iPhone [56] and Where's My Droid [57] provide low-resolution or on-demand tracking capabilities that are designed to locate a lost phone from a website interface.

While providing a variety of services to the user, these apps and other apps that fall into the same general categories are all subject to the same limitations:

- 1) Commercial location-aware apps are a "black box." The design of the application and underlying functionality is not publically available and cannot be used to integrate with or improve other applications (an exception is MyTracks [43], which is open-source, but is a stand-alone mobile app without an active connection to a server). Therefore, each location app developer must start from scratch in implementing location-aware functionality in an application.
- 2) Commercial location-aware apps require active user management of location features due to the impact on device resources (e.g., battery life). Users are responsible for turning location-aware functionality on and off, which burdens the user whenever location-aware features are used. For example, if a user leaves the

MyTracks app on in order to record the phone's location using GPS, the mobile phone battery will die within a few hours. Instead, the user must repeatedly turn the MyTracks app on when traveling, and turn the MyTracks app off when they get to the destination. This effectively prevents a convenient 24/7 tracking application from being possible, given the energy demands of GPS.

- 3) Commercial location-aware apps are often limited to "locate->send" functionality.

Phones are often simply used to access the positioning technology in the device and send this information to a server, and the primary application features are available via desktop or web apps, not the mobile app. In other words, the software simply runs in the background and occasionally reports the rough location of the device to a server.

- 4) Commercial location-aware apps are often lacking device-based intelligence.

Location information is not often processed locally on the device, which limits the abilities of the app to intelligently manage constrained device resources while using positioning systems and wireless communication. This lack of on-board intelligence limits the frequency of use of GPS as well as the frequency of location reporting to a server to a large static interval (e.g., 10 minutes) to avoid having a severe impact on device battery life and cellular network data traffic.

The next section discusses known location-aware architectures and their limitations for supporting further innovation beyond today's location-aware features.

## 2.2 Known Location-Aware Architectures

Since the E911 mandate in the late 1990s, many location based services architectures have been presented in academic literature.

Some of the initial papers following the E911 mandate targeted the implementation of positioning technologies by cellular carriers. Zhao [13], Kupper [16], Barnes [58], and Rao et al. [59] provide a survey of the different technologies and standards under consideration for implementation by carriers, while Porcino [15] and Sunay [60] provide evaluations of various positioning technologies. In this dissertation we are concerned primarily with device-based (i.e., mobile terminal (MT)-based, mobile station (MS)-based) assisted GPS, since it is the most accurate and precise positioning technology widely available on mobile phones [13, 15-19] and is also the positioning technology typically exposed to application developers via APIs [18]. Soliman et al. [61], Ashjaee [62], Langley [63], Richton et al. [19], and Liu [64] all discuss the implementation details of first-generation assisted GPS systems for mobile phones which utilize both assistance information from the cell network as well as GPS hardware in the mobile phone. A weakness of first-generation GPS is that it could not acquire a positional fix indoors [65]. Subsequent evolutions of GPS technology, termed “high-sensitivity” or “indoor” GPS, are aided by a new hardware design that enables the GPS hardware to detect satellite signals and compute a position even in highly obstructed environments, such as indoors. Van Diggelen discusses the design, implementation, and testing of high-sensitivity GPS in his work [66-69]. Vittorini et al. [70], Lachapelle [71, 72], Zhang et al. [73, 74], Beauregard [75], and DeSalas et al. [76] all discuss further improvements to general high-sensitivity GPS design for additional accuracy and availability of position and

velocity measurements. Ballantyne et al. discuss integrated circuit (IC) designs within the mobile phone that can help reduce the amount of energy an individual GPS fix consumes [77]. Zandbergen et al. provide empirical accuracy evaluations of GPS data from mobile phones [17, 18], while Blunck et al. provide an analysis of the impact of body of the user on GPS signal reception in phones [78]. Other publications [79-85] examine issues related to increasing the precision and accuracy of indoor tracking via other technologies such as WiFi, ultrawideband, and Radio Frequency IDentifiers (RFID), although these techniques are not currently available positioning options for mobile application developers, and therefore are beyond the scope of this research.

While these papers on the intimate details of positioning systems served a critical role in the development of positioning systems for mobile phones, they are of greatest use to the engineers implementing these positioning systems in cellular networks and do not provide guidance to applications developers, other than to provide a rough order-of-magnitude analysis of the accuracy and precision of the underlying positioning technologies. These works discuss technologies which are largely hidden beneath application platform APIs, and therefore application developers do not directly interface with these technologies.

Once positioning technologies for cellular devices had matured and were implemented in cellular networks, the focus of many academic works turned to the realization of location-based services based on these positioning technologies. Mintz-Habib et al. [86] present a Voice over Internet Protocol (VoIP) emergency services architecture and prototype which is aimed at providing location information to public safety answering points (PSAPs)

when a mobile user calls 911. Jose et al. [87] present an architecture based on the Service Location Protocol (SLP) [88], but this work is designed for relative location between entities in the Internet and is not designed for high-accuracy or high-precision GPS-based devices.

Since the business model and cellular carriers' ultimate role in providing commercial LBS was initially uncertain, several papers presented architectures that could be implemented by cellular carriers or a commercial partner of the carrier. These architectures are either tightly coupled to the cellular infrastructure or maintain a centralized location data store and interface for all location-aware mobile applications running on the network. Zundt et al. present a peer-to-peer location architecture that is tightly-integrated with GSM networks [89], and Taheri et al. present a network location management scheme to enhance the efficiency of base station handoffs for GSM networks by using Hopfield Neural Networks [90]. Spanoudakis et al. [91], Kupper et al. [92, 93], and Treu et al. [94] all present architectures that enforce centralized control over all location-aware applications for mobile phone users, as the architecture serves as the location gateway for connecting applications. These architectures all assume that the carrier or commercial partner of the carrier has total control over the location-based services that are offered to cellular users on their network. In other words, application developers must enter into an agreement with the carrier or commercial partner to provide services to mobile phone users. This dissertation instead focuses on a location-aware architecture that can be fully-implemented at the application level by third party application developers and does not require a commercial relationship or programmatic interaction with a centralized system which controls all LBS for a cellular network.

Some location architectures provide conceptual models for location exchange between entities in a system, but do not define the exact protocols for the exchange of the location information and do not evaluate the impact of the architecture on important mobile device characteristics such as battery life or amount of data transfer over the cell network. Spanoudakis et al. [91] present their PoLoS Kernel server, which is designed to receive location information from cell phones using the Extensible Markup Language (XML)-based Simple Object Access Protocol (SOAP) [95] and share this information with Internet clients via a “Services Deployer.” Leonhardi et al. [96, 97] describe the conceptual exchange of XML-formatted documents between hierarchical entities in a location system that was implemented using a wearable computing system prior to GPS being available in mobile phones. Nord et al. [98] describe an architecture that has a primary purpose of abstracting positioning technologies used by a mobile device to network servers that wish to discover the location of the device using an XML-based “General Positioning Protocol.” Wu et al. [99] proposes a location architecture in which device positions are only sent on-demand to a server when a viewer requests to see the device’s position. The PoSIM system presented by Bellavista et al. [100] multiplexes between positioning technologies based on a rules defined by the software developer at compile time and asserted at runtime by a rule engine, and exchanges XML-based messages within the system. Chen et al. [101, 102] propose an XML-based “location operating reference model (LORE),” designed primarily for location-based messaging based on client subscriptions (e.g., user is subscribed to receive e-coupons to a store when they are in proximity of the store). For user privacy, Chen [101] also proposes that instead of sending location updates from the device to the server, the server sends all geo-

stamped XML-based subscription messages to all devices. Each device then compares the message's location area to its own location and determines if the message should be shown to the user. Ananthanarayanan et al. [103] propose StarTrack, a server-focused framework for abstracting spatial database operations on recorded user tracks to a set of conceptual primitives, alleviating the application developer from needing to understand low-level spatial database functionality.

In all these architectures, the impact of position updates (a function of both the frequency of GPS recalculations and the frequency of the data being sent to a server) on mobile device battery life is not directly considered. For architectures that use XML, experiments in Chapter 4 of this dissertation illustrate the drawbacks of using a verbose formatting scheme such as XML and SOAP for the transfer of location data between mobile phones and a server, as such a scheme has a significant impact on mobile device battery life due to the large amount of overhead data exchanged.

Several past LBS architectures have focused on the use of the Session Initiation Protocol (SIP), an application-layer protocol that is often used in the context of VoIP applications [86, 89, 104-109]. However, none of these SIP-based architectures were designed for GPS-enabled mobile phones in the Java ME environment. The optional SIP API for Java ME has not been widely implemented in mobile devices and therefore typically is not an available protocol for mobile developers to use in an application [110]. In fact, in the roadmap for the Java ME platform defined by the Mobile Services Architecture (MSA) specifications, the SIP API is only required to be supported in the high-end device segment, such as Personal Digital Assistants, in order for the device to be MSA-



compliant [111, 112]. Therefore, location-aware architectures targeting the majority of Java ME devices should not require support for SIP.

Some location-aware architectures have focused on the routing of location data between servers as part of a distributed system. In these systems, the mobile devices connect to a server on the periphery of the distributed system network, and then the server acts as a proxy for the mobile device to contact other entities in the distributed system, retrieve data, and relay that data back to the mobile device. Zhang et al. [113] present their GeoGrid architecture which maps the location of servers in the topology network to the actual geographic position of the servers, and provides routing algorithms for load-balancing and redundancy. Perez et al. [114] present Geotella, a peer-to-peer routing protocol modeled after Gnutella, as part of their scalable G-Sense global architecture to link location information from wireless sensor networks and mobile devices. These systems have the advantage of scaling to a larger number of simultaneous global users than traditional client-server architecture with a single centralized server. However, neither of these architectures directly considers the connection between the mobile device and server, which still must be a client-server architecture, and neither evaluates the impact of this exchange of location information on the mobile device's limited resources. In fact, Perez et al. [114] cites our research as the client-server architecture used in their system to exchange data between the mobile device and the server.

Out of the many location-aware architectures that have been presented in literature, only two have been designed with awareness of the negative impact that location-based

services can have on limited mobile device resources. The difference between our research and these existing location-aware architectures is presented below.

Kjaergaard et al. [33, 36, 37] presents Entracked, a software system for the Nokia N95 and N96 smart phones running the Symbian operating system that adjust the GPS recalculation frequency and position reporting frequency based on a software model of power consumption. The power consumption model is generated and updated via data from a power-sampling API on the device at the rate of 4Hz, and also samples GPS at a rate of 1Hz and an embedded accelerometer at a rate of 30Hz. However, the Entracked system is designed to deliver location information to network applications, not mobile applications. In other words, network applications query the Entracked server, which in turn queries the Entracked mobile software for the device position, and then relays this position information back to the network application. Therefore, Entracked does not support mobile real-time location-aware applications, which is the focus of our research. Also, since Entracked relies primarily on the accelerometer to decide when to turn GPS on and turn off, this software model cannot be used on devices that do not have embedded accelerometers. Entracked assumes that even when sampling GPS positions at large intervals (e.g., every 200 seconds) the GPS hardware would still need to remain constantly powered on (i.e., the hardware could not enter a low-power state in between samples). While this assumption is true for older GPS devices, for modern cell phones with high-sensitivity GPS even modest adjustments of sampling intervals (e.g., four seconds) in the app can yield significant energy savings, as we show in Chapter 4. This savings is produced by the internal GPS quickly acquiring a positional fix and then powering down between samples. Our research leverages these observations and

presents a power-saving technique, GPS Auto-Sleep, which does not require embedded accelerometers and therefore can function on even severely resource-constrained devices that have only embedded GPS. Another difference between Entracked and our work is that Entracked uses the Transmission Control Protocol (TCP) [42] to transfer location data between the device and the server. In Chapter 4, we demonstrate that the User Datagram Protocol (UDP) [41] is preferable for real-time location data transfer, and therefore UDP was chosen for our architecture. Langdal et al. [115] reimplement the features of Entracked in their modular graph-based PerPos middleware. However, the limitations discussed above also apply to the PerPos implementation of Entracked.

Farrell et al. [32] present an Early Distance-Based Reporting (EDBR) algorithm, a position reporting method which considers both the energy used by positioning sensors such as GPS as well as the energy used in the wireless transmission of this information. However, this method was designed primarily for reporting positions to a server for network-based applications, and not in the context of real-time mobile applications. The focus on network applications, and the tight coupling of the positioning sensor refresh interval and interval between location updates to a server, creates several limitations for real-time mobile applications. For example, Farrell et al. support only a distance-based reporting method, which will not produce any location updates to a server if the device does not move. Therefore, distance-based reporting does not support the use-case of a mobile application that is required to report a position to a server at a minimum time interval, regardless of movement. Also, since distance-based reporting sends a position to the server after a certain distance is exceeded, it can produce needless updates if the user is traveling in a straight line for an extended period of time (e.g., driving on a

highway). Our research presented in the next chapter de-couples the position reporting method (i.e., the Critical Point Algorithm) from the method used to refresh the positioning sensor (i.e., GPS Auto-Sleep) in order to support independent operations of each method, thus modularizing the system and extending the use cases for various positioning sensing refresh and position update reporting intervals. This allows our LAISYC framework to support various types of position update methods in the Critical Point Algorithm without changing the positioning sensor refresh rate. Additionally, the positioning sensor refresh rate can then be adjusted based on logic other than detecting movement for server updates. One example of alternate refresh logic is the manipulation of the refresh rate for a mobile navigation application that wants to only occasionally refresh a position when a large distance from the goal, but then needs to increase the refresh rate when getting closer to the goal. By reducing the GPS refresh rate and only updating the location occasionally when miles from a goal, the application can produce significant battery life savings, as we demonstrate in Chapter 4.

Farrell et al. also do not evaluate their algorithm on actual mobile devices; instead, they synthesize random positions from a simulator, with the assumption that objects move linearly and in a uniform manner, and use this data to evaluate their algorithm. Synthetic path data generated in this manner is problematic from several perspectives. Farrell et al. do not consider the uncertainty and error of a GPS position when evaluating their algorithm. As we show in Chapter 4, even with high-sensitivity GPS indoor position tracking produces a significant amount of errors in position that do not reflect the true geographic position of the device due to environmental noise [17, 18]. When a GPS device calculates a position repeatedly in the same geographic location, the error in

position creates a normal distribution [116]. Therefore, Farrell et al.'s assumption that the change in GPS positions while a user is stationary will be uniform is invalid. In our work, the LAISYC architecture is evaluated while it executes on actual mobile devices with real assisted GPS data, therefore removing the assumptions and limitations discussed above.

Our past research has investigated location-aware architectures in the context of bidirectional, multimedia, location-based messaging [117]. That architecture focuses primarily on a messaging infrastructure which piggy-backs location data in Multimedia Messaging Service (MMS) messages sent through a cellular carrier's publicly-accessible messaging gateway, thus avoiding the use of short-codes and messaging aggregators. However, the use cases for this architecture are the occasional exchange of messages, and therefore only occasional use of GPS. Since GPS is not used in an ongoing manner, this messaging architecture does not consider the impact of GPS on mobile device battery life, or the amount of data being sent over the cellular network.

This dissertation presents LAISYC, an architecture that supports real-time mobile applications that are "always-on" and in continuous communication with a server, as in traditional IP-based networks. LAISYC focuses primarily on the intelligent device-based modules but also discusses the structure of communication with the server and server-side components that support the overall framework. Unlike the other known architectures discussed in this chapter, LAISYC meets the needs of intelligent real-time mobile applications in Location-based Services as discussed in Chapter 1. Our research presents the results of field tests in Chapter 4 which evaluate key LAISYC modules in

order to quantitatively assess their impact on mobile device battery life in the context of the presented architecture. Our work on LAISYC is also summarized in publications in IEEE Pervasive Computing [118], Proceedings of IEEE UBICOMM Conference [119], the Transportation Research Record: Journal of the Transportation Research Board [120], Proceedings of the Intelligent Transportation Systems World Congress [121], the Journal of Navigation [18], and several issued [122-126] and pending patents [127-129].

LAISYC has been used to enable several real-time location-aware applications as part of research projects, including the Travel Assistance Device (TAD) mobile application that assists transit riders with intellectual disabilities in using public transportation through real-time navigation instructions [130], as well as TRAC-IT, a mobile app that supports simultaneous travel behavior data collection and real-time location-based services [131, 132]. TAD and TRAC-IT's relationship with LAISYC is discussed in detail in Chapter 4 as a demonstration of innovative location-aware applications implemented using LAISYC.

## **CHAPTER 3: PROPOSED ARCHITECTURE – LOCATION-AWARE INFORMATION SYSTEMS CLIENT (LAISYC)**

### 3.1 Note to Reader

Material presented in this chapter has been published in IEEE Pervasive Computing [118] (© 2011, IEEE), and we have received permission to reprint this work. The University of South Florida also has patents pending and issued on various technologies discussed in this chapter. The GPS Auto-Sleep technology is protected under U.S. Patent # 8,036,679 “Optimizing performance of location-aware applications using state machines” [122] by the University of South Florida. Material on GPS Auto-Sleep has also been published in the Proceedings of UBICOMM '08 [119] (© 2008, IEEE), and is reprinted here with permission of IEEE. The Session Management technology is protected under pending U.S. Patent Application # 13/082,094 and International Patent Application # PCT/US2009/059825 “Architecture and Two-Layered Protocol for Real-time Location-Aware Applications” [128] by the University of South Florida. Adaptive Location Data Buffering technology is protected under a pending U.S. Patent Application # 13/082,722 and International Patent Application #. PCT/2009/059985 “Adaptive Location Data Buffering for Location-Aware Applications” [129] by the University of South Florida. The Critical Point Algorithm is protected under pending U.S. Patent Application # 12/196,673 “Method For Determining Critical Points In Location Data Generated By Location-Based Applications” [127] by the University of South Florida and has also been published in the Proceedings of UBICOMM '08 [119] (© 2008, IEEE), and is reprinted

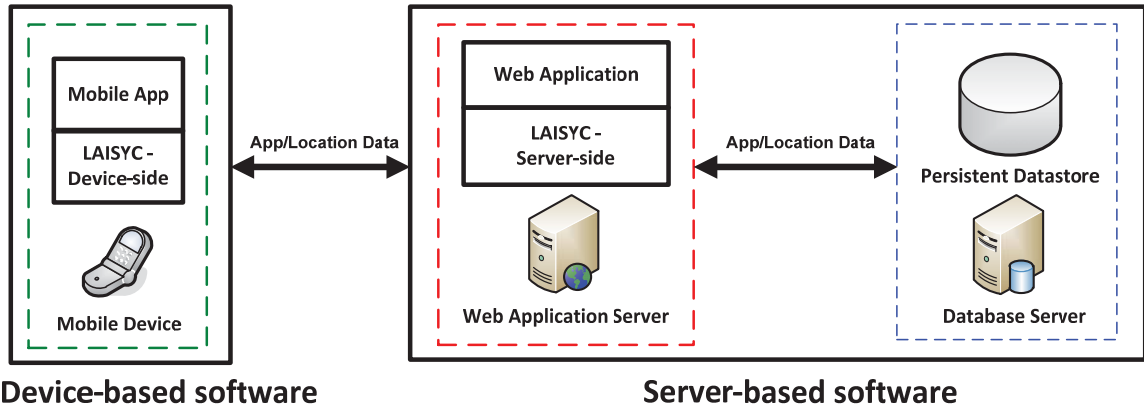
here with permission of IEEE. The Spatial Analysis technology is protected under pending U.S. patent Application # 11/855,694 “System and Method for Real-Time Path Prediction and Automatic Incident Alerts” and U.S. patent Application # 11/277,403 “System and Method for Transportation Demand Management” by the University of South Florida.

### 3.2 Architecture Overview

LAISYC was created to meet application needs for real-time, high-accuracy and high-precision location-aware applications. This architecture was designed to be fully-implementable by third party mobile app developers, and can intelligently manage limited device and network resources. LAISYC can support various types of location-aware applications, including real-time tracking, as well as delay-tolerant applications that record the user’s travel path. For maximum flexibility, an application can dynamically manipulate LAISYC module parameters according to real-time application needs, and therefore hybrid applications with both real-time and delay-tolerant features are also possible.

To support the needs of modern LBS discussed in Chapter 1, LAISYC is separated into device-based modules, which are implemented in software on the mobile device, and server-based modules, which reside on a web application server, such as Glassfish [133]. Figure 1 shows the high-level view of this device-server architecture. The mobile and web portions of the application supported by LAISYC sit on top of the respective LAISYC modules. The web application server supports a large number of client devices simultaneously and tracks individual sessions for each device.





**Figure 1 - The LAISYC architecture consists of software on the mobile device and web application server, with a database server holding persistent server-side data**

The web application server also acts as a proxy for the mobile device to access the database server, as mobile devices are not capable of directly interfacing with database servers due to a lack of mobile database drivers.

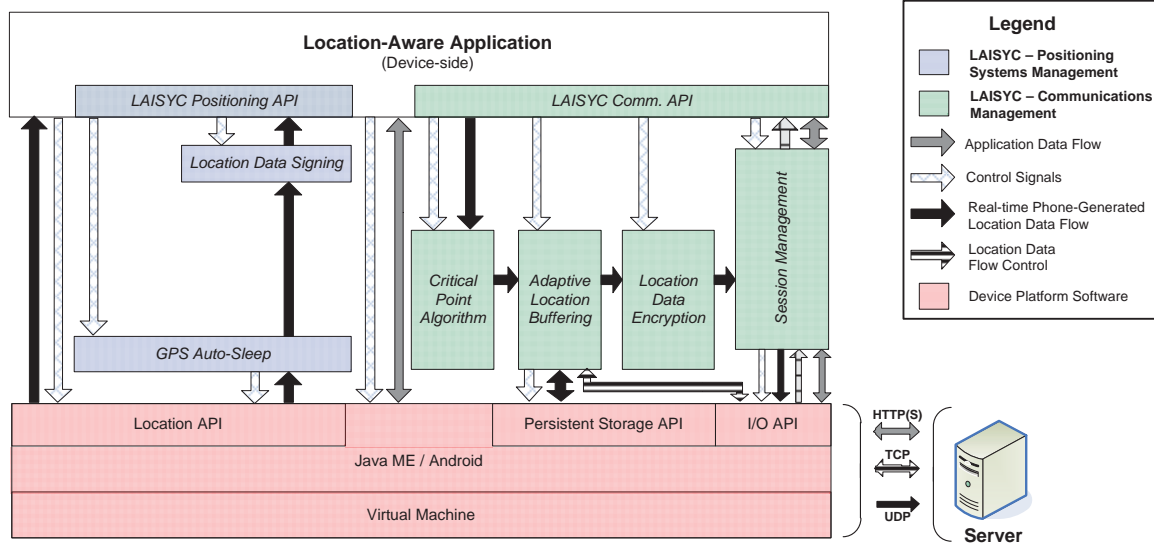
The following sections discuss each of the modules of LAISYC, and their respective position on either the device or the web server.

### 3.3 Mobile Device-Side Components

This dissertation focuses primarily on the design, implementation, and evaluation of the mobile device-side modules in LAISYC.

The LAISYC modules that reside on the device can be broken down into two categories, as shown in Figure 2:

- 1) Positioning Systems Management (Blue shaded modules in Figure 2)
- 2) Communications Management (Green shaded modules in Figure 2)



**Figure 2 - LAISYC mobile phone-based modules. [118] © 2011 IEEE**

LAISYC modules in each of these categories process two types of data:

- 1) Application data – all non-location information that is required for the successful operation of the application (e.g., usernames, passwords, application logic parameters). This data is typically exchanged with the server on an occasional basis, and its loss is not tolerable.
- 2) Location data – data generated by positioning systems (e.g., GPS) that represent the geographic position of a mobile device. This data can be frequently exchanged with the server with a rate of up to one transmission per second, and timeliness is of greater importance than 100% reliability. Therefore, occasional loss of individual device positions is tolerable for many applications.

Location data is generated from the positioning system (e.g., GPS) on the mobile device and is passed to the LAISYC framework through the Location API (i.e., JSR179 or JSR293 in Java ME, Location API on Android) that is part of the underlying platform.

The location data is first received by the bottom layer of Positioning Systems

Management (i.e., GPS Auto-Sleep), and is passed through each module until it reaches the application. The application can send control signals to each module, deactivating it if necessary. If the application deactivates a module, the location data pass through that module without any action by LAISYC.

After location data passes through Positioning Systems Management, the application can send location data to the server by passing it into the first module in the Communications Management group (i.e., Critical Point Algorithm). The data then propagates to the right until it reaches the Session Management module, which activates the wireless transmission of the location data. The mobile application also sends application data to the server by interfacing with the Session Management module.

Positioning Systems Management modules are discussed first in the following section, and Communications Management modules in a subsequent section.

### 3.3.1 Positioning Systems Management Modules

The Positioning Systems Management modules include GPS Auto-Sleep and Location Data Signing.

#### 3.3.1.1 GPS Auto-Sleep

The purpose of the GPS Auto-Sleep module is to save battery energy by dynamically adjusting the GPS sampling interval based on user movement.

Mobile phone platforms such as Java ME and Android typically provide two general modes of interaction between a mobile application and the underlying GPS hardware via a Location API [22-24, 134]:

- 1) Single-shot GPS request – In this mode, the application requests a single GPS position update from the platform using the Location API. The platform activates the GPS hardware, waits until a GPS position is calculated, and returns the calculated position to the application. The platform may support an application-defined or platform-defined timeout value, which is used to limit the length of time the GPS hardware remains active after a request. If the GPS cannot achieve a position fix within this timeout period, a null value may be returned to the application indicating a failure to locate the device (e.g., due to environmental obstructions).
- 2) Periodic GPS request – In this mode, the application specifies that it would like to receive recurring GPS updates from the platform at a fixed interval of time by registering a `LocationListener` with the Location API. The platform proceeds to calculate GPS positions using the underlying GPS hardware at the defined interval and executes an asynchronous callback to the application's `LocationListener` when each new position is calculated. Timeout values can also be used in this mode to establish an upper limit on the length of time the GPS hardware is active on each fix attempt. A parameter `maxAge` (i.e., maximum age) can also be passed into the `LocationListener` to define the maximum time allowed between when a location fix was calculated and when it can be returned to the application. `MaxAge` is typically used in a multitasking environment to allow the Location API to return the same location fix to more than one application that may be on a slightly different GPS update schedule. For example, if application<sub>A</sub> uses an interval value of 30 seconds started at time  $t_0$  and a `maxAge` value of 10

seconds, and application<sub>B</sub> triggers a GPS update from the Location API at time<sub>25</sub>, at time<sub>30</sub> the Location API could return the same GPS fix to application<sub>A</sub> because the GPS fix is less than 10 seconds old.

For long-term tracking of a device, developers use the periodic GPS request feature of the Location API.

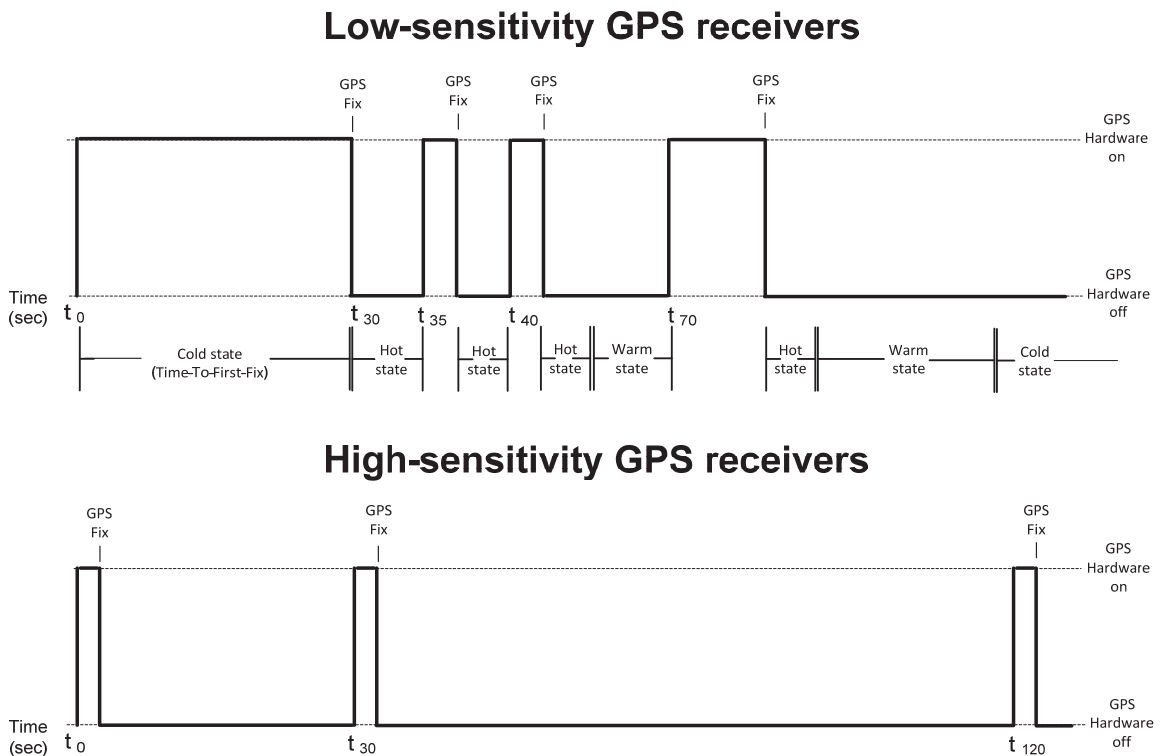
As discussed earlier, battery life is a key limitation for LBS on mobile devices. Since GPS technology requires GPS receiver hardware in the mobile device so the device can locate itself using satellite radio broadcasts, the use of GPS costs a significant amount of energy when activating this hardware.

In the early stages of our research with high-definition tracking on GPS-enabled mobile phones, it quickly became apparent that even on devices using modern high-sensitivity GPS, a simple solution of setting the LocationListener to periodically refresh its position every few seconds is energy-prohibitive, as this would exhaust the battery in a matter of hours. Typically, a mobile phone's battery must be operational during the day (approximately 16 hours) until the user can plug the device in and recharge the battery at night.

With further experimentation, we found that an application could request periodic updates at a larger time interval, such as five to ten minutes, and this would extend battery life to an acceptable length that would bridge the gap from one nightly battery recharge to another. However, GPS samples five to ten minutes apart do not meet our requirements for high-precision and high-accuracy tracking or real-time LBS.

This experimentation with larger time intervals between GPS fixes on high-sensitivity GPS-enabled mobile phones led to a valuable observation: high-sensitivity GPS hardware is still able to successfully achieve a GPS position fix even with long time delays between consecutive fixes. This ability to rapidly acquire a new GPS fix even if there has been significant delay since the most recent GPS fix is new to high-sensitivity GPS receivers [18, 68].

Previous generation GPS receivers exhibited a strong dependence on prior GPS observations when calculating a new GPS fix. These GPS receivers typically had three tracking modes: cold start, warm start, and hot start [135].



**Figure 3 - High-sensitivity GPS receivers can acquire a GPS position more rapidly, and with less dependence on the time elapsed since the most recent GPS fix, than low-sensitivity receivers**

These modes are illustrated in Figure 3, which shows the amount of time needed to acquire a GPS fix with different amounts of time elapsing since the most recent GPS fix.

A device starts in cold start mode, and even in complete open view of the sky it would typically take a minimum of 30 seconds to a minute to acquire a first GPS fix. This time period is known as the Time-To-First-Fix (TTFF). As an initial GPS fix is acquired, the GPS hardware then enters a hot state, in which it has current knowledge of satellite positions in the sky and the appropriate signal frequency offsets and code delays needed to successfully calculate the next GPS fix. However, as time begins to elapse from the initial hot fix, the GPS hardware's knowledge of the state of the GPS system begins to quickly decay as satellites change position in the sky and environmental factors change the GPS signal environment. As a result, while subsequent GPS fixes occurring within ten seconds following the initial hot fix will likely succeed if there is an open view of the sky, the likelihood of a successful GPS fix decays with the elapsed time after the most recent GPS fix.

The GPS receiver is said to enter a warm state, which can be from around 10 seconds to 1 hour after the most recent GPS fix, and then return to a cold state if more than 1 hour has elapsed since the most recent fix. Once in a cold state, the GPS receiver loses significant knowledge of the state of the GPS system and must enter an initial startup mode, which is again the TTFF. The exact decay time transitions from hot to warm to cold states can vary depending on the GPS manufacturer, as some receivers are only able to sustain a warm state for a few minutes after the most recent GPS fix.

High-sensitivity GPS largely eliminates the concept of hot, warm, and cold states, as high-sensitivity GPS hardware is quickly able to acquire a fix even from a cold start state, bringing the TTFF down to a few seconds or less, depending on the strength of the GPS signals. For example, in the view of an open sky, high-sensitivity GPS receivers can calculate a fix from a cold start in less than a second. Figure 3 illustrates that even if the length of time has been significant since the most recent GPS fix, high-sensitivity receivers still only require the GPS hardware to be on for a limited amount of time before successfully calculating a new fix.

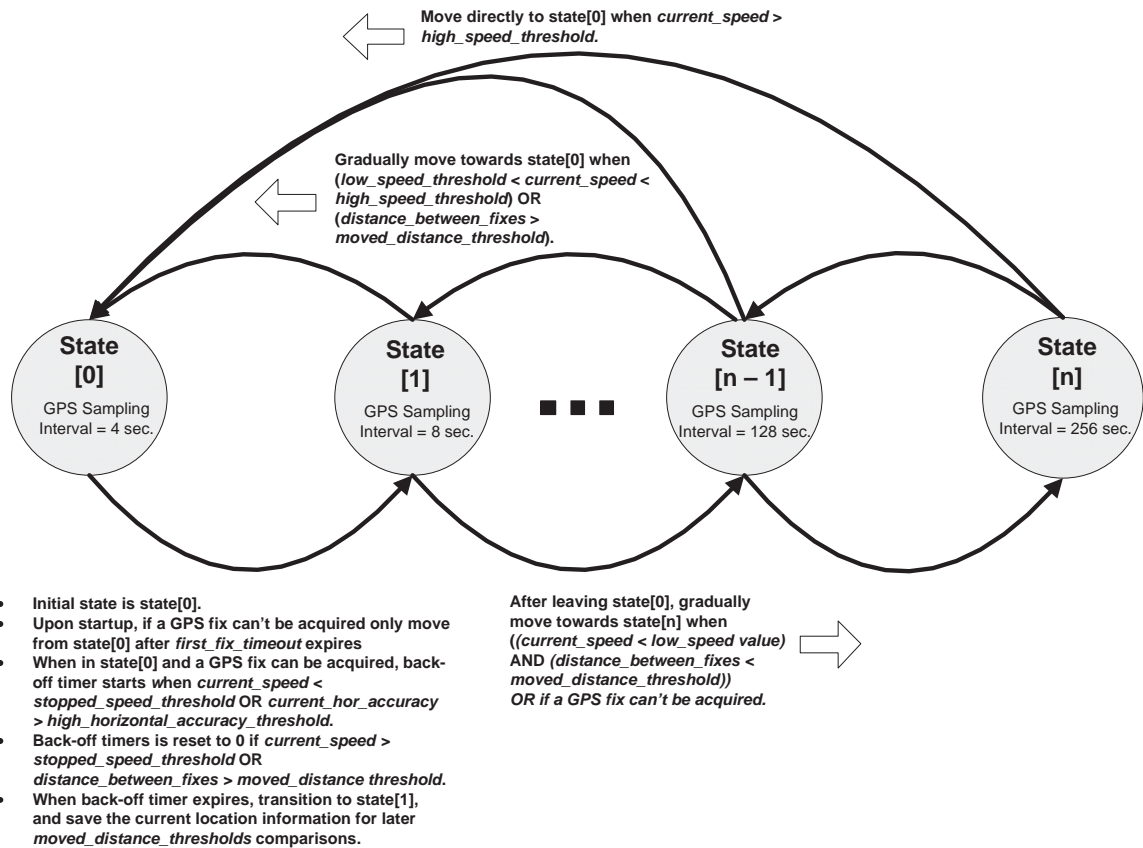
Through these experiments, we recognized that significant independence from previous GPS observations meant that, unlike older generation GPS, on high-sensitivity GPS-enabled mobile phones we could dynamically vary the interval of time between GPS fix attempts and produce significant battery energy savings. If a similar technique had been attempted on older generation GPS receivers, the length of time the GPS hardware would have been active during the TTFF to achieve a position fix from a cold start state would have largely offset any energy saved by using dynamic GPS sampling rates.

In LAISYC, the GPS Auto-Sleep module intelligently adjusts the GPS sampling rate of the mobile device based on real-time location information in order to save battery energy when the user is stationary, but still maintains a high-resolution sampling rate when the user is actively moving. For example, if a mobile device is indoors and cannot calculate a position due to GPS signal obstruction, a large amount of energy is wasted as the device continuously attempts to calculate a GPS fix every few seconds. The interval of time between position recalculations can gradually be increased (i.e., towards a sleep mode) in



order to prevent calculating the relatively same position information repeatedly. The application can be woken up and transition to a rapid position recalculation when it is determined the device is moving again with a high degree of confidence. The mobile application interfaces with the GPS Auto-Sleep module via the LAISYC Positioning API, which allows the application to turn GPS Auto-Sleep on and off and receive location updates from GPS Auto-Sleep.

We implement GPS Auto-Sleep using a finite state machine, as shown in Figure 4.



**Figure 4 - GPS Auto-Sleep uses a state machine with various logic evaluations that control the transition between states, which represent changes to the GPS sampling interval values. [119] © 2008 IEEE**

Changes between states in the state machine represent changes to the GPS sampling interval (i.e., time between sequential GPS samples), with a variety of rules based on real-time location data and previously observed location data controlling the state transitions. In addition to the GPS intervals, each state can also contain values for the timeout and maxAge parameters to be used with the respective GPS interval value.

The Java ME Location API provides the following values to GPS Auto-Sleep for each successful GPS position calculation [23, 24]:

- Latitude and longitude: the position of the user on the surface of the earth in decimal degrees, using World Geodetic System (WGS) 84 datum.
- Altitude: the altitude of the location in meters, defined as height above the WGS84 ellipsoid.
- Timestamp: the time at which the GPS position was calculated, based on the GPS receiver clock, which is synchronized to the GPS system.
- Speed: the device's current ground speed in meters per second (m/s) at the time of measurement.
- Heading: the heading of the device when the GPS fix was recorded, in degrees relative to true north, in range of 0-360 (e.g., 0, 360 = north, 90 = East, 180 = South, 270 = West).
- Estimated horizontal accuracy: the estimated accuracy of the location as the radius of a circular area indicating the 68% confidence level. In other words, the true location of the user should fall within a circle having the center of the

calculated position and a radius of the estimated horizontal accuracy value at a probability of approximately 68%.

During execution of GPS Auto-Sleep, several thresholds are used and compared against the location data provided by the Location API:

- *first\_fix\_timeout*: a time in seconds for which GPS Auto-Sleep will remain in state[0] when the mobile application is first started, if the device cannot achieve a GPS fix. After this amount of time has elapsed on startup, this threshold is not used for the duration of application execution. If the GPS receiver is refreshing its knowledge of the GPS system on startup, we do not want to immediately start transitioning to lesser sampling frequencies to give the receiver the best chance at achieving a first fix. This value should be set high enough to let the GPS receiver operate for enough time to refresh assistance data and observe GPS signals, but no longer than the amount of time expected for the receiver to calculate a GPS fix under typical conditions (e.g., outside, in light building coverage). In experiments with Sanyo 7050 and Sanyo Pro 200 phones, we have used 20 seconds for this threshold.
- *stopped\_speed\_threshold*: a speed value in meters per second that is used to determine if the user is currently stopped (i.e.,  $current\_speed < stopped\_speed\_threshold$ ). This threshold should be set so that the device has a high degree of confidence that the device is truly stopped if the  $current\_speed < stopped\_speed\_threshold$ . In experiments with Sanyo 7050 and Sanyo Pro 200 phones, we have used 1 m/s as this threshold.

- *high\_speed\_threshold*: a speed value in meters that is used to jump-start the GPS Auto-Sleep machine immediately and snap to a high GPS sampling frequency immediately if a very large speed value is observed. This value should be set so that the software has a high degree of confidence that the device is moving if the  $current\_speed > high\_speed\_threshold$ . In experiments with Sanyo 7050 and Sanyo Pro 200 phones, we have used 1.5 m/s as this threshold.
- *moved\_distance\_threshold*: a distance value in meters that is used to determine if the user has moved from a location when the GPS location was last sampled and the user was considered stationary (i.e.,  $distance\_between\_fixes > moved\_distance\_threshold$ ). This threshold should be set so that the device has a high degree of confidence that the device has truly moved if  $distance\_between\_fixes > moved\_distance\_threshold$ . In experiments with Sanyo 7050 and Sanyo Pro 200 phones, we have used 100 meters as this threshold. We used Vincenty's Inverse formula [136] to calculate the distance between two points on the WGS84 ellipsoid, which were shown by Vincenty to be accurate to within 0.5mm [136].
- *high\_horizontal\_accuracy\_threshold*: a distance in meters that is used to determine if the user has stopped moving and is inside a building, based on the high level of estimated horizontal accuracy uncertainty of the GPS fix. The general assumption is that if estimated horizontal accuracy is very high, then GPS signals are greatly obstructed and it is likely the user is inside a building. This value should be set so that the software has a high degree of confidence that the user is inside a building if the  $current\_horizontal\_accuracy >$

*high\_horizontal\_accuracy\_threshold*. In experiments with Sanyo 7050 and Sanyo Pro 200 phones, we have used 80 meters as this threshold.

- *backoff\_time\_threshold*: a time in seconds to wait after the user is believed to be stationary before allowing the state machine to transition from state[0] to less frequent GPS sampling in state[1] (i.e., “backing off” for actively sampling travel towards sleep mode). In other words, once GPS Auto-Sleep is actively sampling in state[0], a time greater than *backoff\_time\_threshold* must elapse before any state transitions take place. The backoff timer is started when  $current\_speed < stopped\_speed\_threshold$ , or  $current\_horizontal\_accuracy > high\_horizontal\_accuracy\_threshold$ , or if the GPS receiver cannot calculate a GPS position. The backoff timer is reset to 0 when movement is detected (i.e.,  $current\_speed > stopped\_speed\_threshold$ ) before the backoff timer has expired. We used this backoff timer because travel behavior tends to have a temporal locality, in that travelers are more likely to move if they have been moving recently. One example of this is traffic lights – we want to continue sampling for a typical duration of a traffic light to maintain high resolution sampling while the user is actively traveling, rather than briefly observing a pause in travel behavior (e.g., getting stopped at the light) and immediately reducing the GPS sampling rate. We assume that if the *backoff\_time\_threshold* has elapsed, then the user has likely stopped moving for the immediate future. In experiments with Sanyo 7050 and Sanyo Pro 200 phones, we have used 120 seconds for this threshold.

An example configuration of the state machine for GPS Auto-Sleep is:

- state[0] = 4 s between GPS samples, timeout = 2 s, maxAge = 2s
- state[1] = 8 s between GPS samples, timeout = 4 s, maxAge = 4s
- state[2] = 16 s between GPS samples, timeout = 8 s, maxAge = 8s
- state[3] = 64 s between GPS samples, timeout = 16 s, maxAge = 16s
- state[4] = 150 s between GPS samples, timeout = 32s, maxAge = 32s
- state[5] = 256 s between GPS samples, timeout = 32 s, maxAge = 32s

On application startup, the state machine will start in state[0] and will start periodic GPS sampling using the Location API's LocationListener with an interval of state[0] = 4 seconds.

If the device cannot acquire a GPS fix on startup, it will remain in state[0] for the duration of the *first\_fix\_timeout* until it transitions to state[1].

If the device can acquire a GPS fix on startup but is not moving (i.e., when *current\_speed* < *stopped\_speed\_threshold* OR *current\_hor\_accuracy* > *high\_horizontal\_accuracy\_threshold*), the back-off timer starts. The back-off timer is reset to 0 if *current\_speed* > *stopped\_speed\_threshold* OR *distance\_between\_fixes* > *moved\_distance\_threshold*. When the back-off timer expires, the state machine transitions to state[1], and saves the current location information for later *moved\_distance\_threshold* comparisons to determine if the device might be moving.

After leaving state[0] and arriving in state[1], the state machine will wait in state[1] 8 seconds for the next GPS fix attempt. After the next GPS fix is attempted, the location

information is evaluated to determine if the next state transition should be towards state[n] (i.e., the state machine assumes the user is stationary), or back towards state[0] (i.e., the state machine assumes the user is still moving). The state machine will move towards state[n] when  $((current\_speed < stopped\_speed\_threshold) \text{ AND } (distance\_between\_fixes < moved\_distance\_threshold))$  or if a GPS fix cannot be acquired. The state machine will move towards state[0] when  $(stopped\_speed\_threshold < current\_speed < high\_speed\_threshold) \text{ OR } (distance\_between\_fixes > moved\_distance\_threshold)$ . If the state machine transitions to state[2], it will wait 16 seconds until the next GPS fix attempt, and it will repeat the above evaluations until reaching state[n] (i.e., sampling GPS every 256 seconds) or arriving back at state[0]. If the state machine arrives back at state[0], it assumes the user is actively traveling again and resets and activates the back-off timer.

Once the state machine is in the sleep state (i.e., state[n]), it can conserve the most energy by calculating GPS fixes using a large interval of time. However, since we are concerned with measuring accurate distance of travel via high resolution GPS sampling, we want to be able to immediately resume high-frequency sampling (i.e., state[0]) if we observe a GPS fix that indicates that the user is moving with high probability. Therefore, we add the ability for the state machine to immediately transition from any state to state[0] if the  $current\_speed > high\_speed\_threshold$ . This “wake up” trigger is based on the speed of the device exceeding a certain threshold in the most recently calculated location data (i.e., the device has started moving).

The gradual transitions between a high-frequency sampling of state[0] and sleep mode of state[n] are a method to handle the uncertainty associated with GPS positions. As sequential GPS observations reinforce the certainty associated with the moving or stationary states, the GPS sampling frequency is adjusted accordingly. This ability allows GPS Auto-Sleep to capture location data for short walking trips that may look very similar to GPS noise. The first portion of the trip will only be occasionally sampled with the frequency of state[n], but as the distance from the last stopped location increases and the user's speed is observed to be slightly higher than the typical *stopped\_speed\_threshold* the sampling gradually increases until the GPS is being sampled at the high-resolution value of state[0].

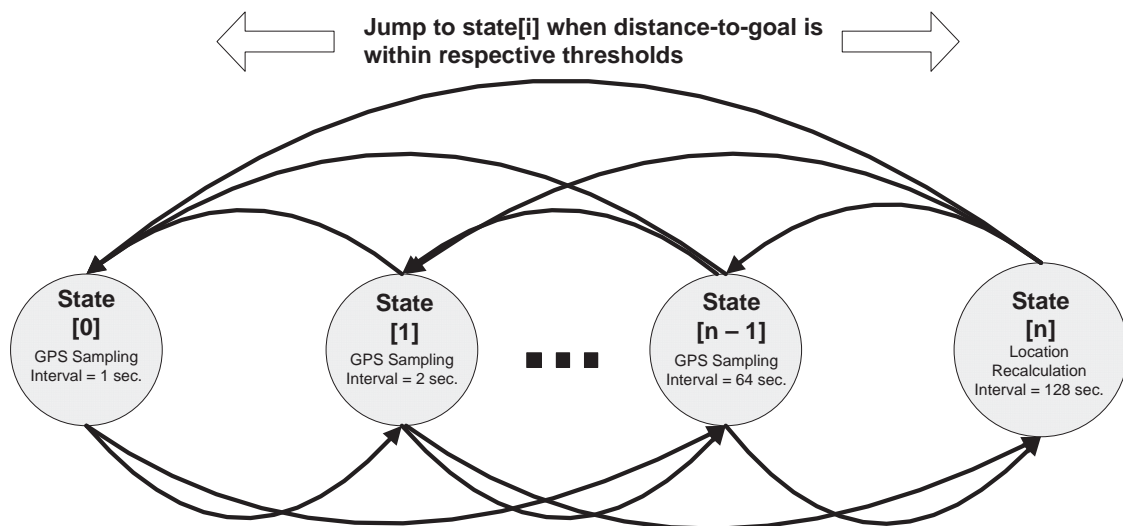
Gradual state transitions also ensure that outlier values do not have an extreme impact on the sampling frequency and cause sampling at a high frequency repeatedly when the GPS should actually be asleep. For example, the most frequent type of GPS outlier data is a position that may be 100 meters from the true location when the user is indoors. The state machine will only react by moving from state[n] to state[n-1], and if the next GPS sample is near the true location the state machine will return to state[n].

In addition to the general tracking functionality defined above, GPS Auto-Sleep also has a secondary navigation mode that can be utilized for location-aware applications that are based on the distance to a goal. This goal may be a fixed location (e.g., the next turn for real-time driving directions) or the location of a mobile device (e.g., real-time friend finder). When navigation mode is switched on and a goal is identified, the state machine can decrease the interval between position calculations as the mobile device gets closer to



the goal (i.e., increase the sampling frequency), and increase the interval between position calculations as the mobile device moves further away from the goal (i.e., put the device into sleep mode). This navigation mode is primarily designed for real-time navigation applications that do not require a high-resolution record of the travel history of the user, but do require high-resolution GPS sampling when nearing the goal to ensure the user is provided with timely instructions.

Navigation mode for GPS Auto-Sleep is also implemented using a finite state machine with interval, timeout, and maxAge values increasing from state[0] to state[n], but the state transition rules are different. Figure 5 shows the navigation mode state transition diagram. State changes for navigation mode occur based on the distance between the mobile device and a goal location, such as the next planned turn in a navigation application when a verbal prompt will be announced to the user.



**Figure 5 - Navigation mode for GPS Auto-Sleep controls GPS sampling interval directly based on a distance-to-goal (e.g., next turn for real-time driving directions)**

For example, if the distance between the current location and a goal location is greater than 5 miles, then the state machine will be at a preset state[n]. As the device approaches its goal location, the state would change to state[n-1] at a certain distance milestone in order to decrease the location recalculation interval. This would decrease the intervals between device location updates from 128 seconds when the device is more than 5 miles away, to 64 seconds when the device is between 5 and 3 miles away, and so on, until reaching 1 second GPS sampling when the device is .25 miles away. This assures that the device will not miss its goal since the location is updated very frequently when the device is physically near. State transitions also occur in the reverse order as the distance-to-goal decreases as the device moves away from the goal. Since real-time applications are time-sensitive, a state can transition directly to another state to avoid stepping through states when a lower interval (i.e. more frequent updates) is required immediately (e.g., in case of temporary GPS signal loss).

In conclusion, GPS Auto-Sleep is designed to address several of the needs for location-aware mobile apps outlined in Chapter 1. GPS Auto-Sleep is designed to increase mobile device battery life (Need #1) by dynamically adjusting the GPS sampling rate in real-time (Chapter 1 - Needs #2 and #3). GPS Auto-Sleep uses the existing periodic GPS request interface of the Location API, and therefore it is fully implementable by third party mobile app developers (Need #4).

In Chapter 4, we demonstrate a methodology for selecting values for each of the thresholds discussed in this section based on observed GPS data, so GPS Auto-Sleep can be configured appropriately for any GPS-enabled device. We also present an evaluation

of the state accuracy of GPS Auto-Sleep, as well as its effectiveness for increasing battery life.

### 3.3.1.2 Location Data Signing

The purpose of the Location Data Signing module is to add energy-efficient authenticity to location data generated by the mobile phone.

GPS data is increasingly being used by businesses and government entities in order to support key operations. These applications rely on GPS to report or verify mileage and time spent by workers on remote sites, support pay-as-you-drive car insurance through the identification of the length and location of car use, as well as to support variable transportation taxes. However, all of these uses of GPS data have a key weakness: GPS data can potentially be falsified through direct tampering with the data. Therefore, the integrity of raw GPS data cannot be independently verified.

The Location Data Signing module utilizes asymmetric cryptography (i.e., public and private keys) with certificates issued by a trusted third party in order to digitally sign data related to the GPS fix. These data can include the latitude, longitude, altitude, speed, heading, GPS timestamp, system timestamp, phone number of device, and identifying information for the phone and user including the International Mobile Equipment Identify (IMEI), Subscriber Identity Module (SIM) ID, mobile station ID (MSID), and Mobile Equipment Identifier (MEID), as well as the username and a hash of the password used to log into the application. By signing these data, Location Data Signing can prove that a particular GPS fix occurred on a particular phone with a specific user logged into the application at a specific time. Since this information is hashed and signed using a private

key by the application, the integrity of the GPS data can be verified by utilizing the public key and a hash of the message. Therefore, it can be shown that a GPS fix, including the location, speed, and time, is unaltered from the data that was originally calculated by a specific application on-board a GPS-enabled mobile phone. We designated Location Data Signing as an optional module in LAISYC, since it may only be required for applications that have a strict requirement for confirming the identity of the mobile device.

While symmetric cryptography is more efficient than asymmetric cryptography, only asymmetric cryptography can be used for digital signatures. To sign data, a private key is required, and to verify data a public key is required, and therefore symmetric cryptography cannot be used.

We chose the Digital Signature Algorithm (DSA) for implementation in the Location Data Signing module. While other options such as Rivest-Shamir-Adleman (RSA) and Elliptic Curve Digital Signature Algorithm (ECDSA) exist, DSA is the only algorithm that is not restricted by intellectual property or export constraints and can be used world-wide royalty-free [137].

In 2006 Jarusombat et al. [138] hypothesized that traditional digital signature algorithms such as RSA and DSA are too computationally intense for mobile devices and proposed their own location-based digital signature algorithm. In 2009, Xuan et al. [139] experimented with digital signature algorithm performance on emulators and demonstrated that traditional digital signature algorithms are indeed feasible for mobile virtual machines. However, Xuan et al.'s experiments were in context of general secure

web services and not in the context of location data, and were not executed on real devices. In Chapter 4, we present the results of experiments showing that actual GPS-enabled devices are indeed capable of frequent location data signing.

In conclusion, the Location Data Signing module is designed to address several of the needs for location-aware mobile apps outlined in Chapter 1. Location Data Signing is designed to add energy-efficient (Need #1) authenticity to location data generated by a mobile device in real-time (Need #2). The selection of DSA for Location Data Signing ensures that it is implementable by any third party mobile app developer (Need #4). In Chapter 4, we also evaluate the impact of Location Data Signing on mobile device battery life (Need #1).

### 3.3.2 Communications Management Modules

The Communications Management modules include Session Management, Adaptive Location Data Buffering, the Critical Point Algorithm, and Location Data Encryption.

#### 3.3.2.1 Session Management

The purpose of the Session Management module is to save battery energy and reduce data transfer costs while supporting real-time location data communication between a mobile phone and server.

Since location-aware applications are distributed between a mobile phone and server, the protocols used for communication between the mobile device and server must be carefully examined for efficiency and broad compatibility with many different client

devices. The selected protocols must also be appropriate for the type and frequency of exchanged data, to avoid an unnecessary impact on limited mobile device resources.

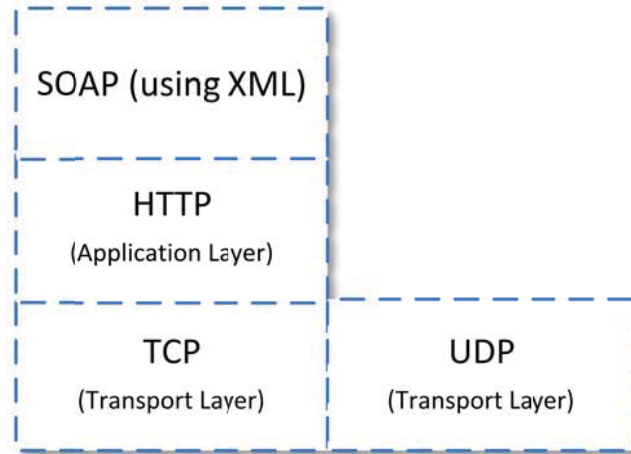
As mentioned earlier, there are two types of location data exchanged between the device and server: application data and location data. Application data is exchanged with the server occasionally and loss of this data is not acceptable. Location data is exchanged with the server frequently, and occasional loss of individual position data points is acceptable. Since LAISYC must be implementable by third party application developers (Chapter 1 - Need #4), the availability of networking protocols at the application level on mobile devices that are suitable for transporting these two types of data must be examined.

#### 3.3.2.1.1 Available Communication Protocols

Until the mid-2000s, HTTP was the only mandated networking protocol for Java ME devices, since many cellular networks were not capable of IP-based communication at that time [140]. However, as IP Multimedia Subsystems (IMS) were developed in the late 2000s to support packet-based communication on cellular networks, IP-based networking protocols, such as the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP), became accessible on a large number of Java ME mobile phones. In fact, the MSA roadmap that defines the evolution of the Java ME platform has required support for TCP for all MSA v1.0 compliant devices [111], and has mandated support for UDP as well for all MSA v2.0 compliant devices [112]. Therefore, HTTP, TCP, and UDP are the widely-available protocols on the mobile device that can be used by LAISYC for communication between a mobile device and server. Figure 6

shows the relationship between HTTP, TCP, and UDP as networking protocols.

Protocols from lower layers in the networking protocol stack, such as the link layer and physical layer, are not accessible to mobile applications.



**Figure 6 - Relationships between HTTP, TCP, UDP, and SOAP as networking protocols**

TCP and UDP are both transport-layer protocols, which focus on end-to-end communication between two nodes in a network. TCP includes many mechanisms to ensure reliable connectivity between entities, including in-order delivery of packets, detection of lost packets, retransmission of lost packets, data flow control, and congestion control. However, this increased reliability comes at a cost of increased communication between the nodes. UDP provides a much simpler and lightweight “send-and-forget” service which does not implement any detection of lost packets or flow control and therefore does not guarantee delivery of data. However, as a result, UDP imposes little overhead on the mobile device and server and is a timely and efficient protocol.

In today’s cellular networks capable of IP-based communications, application-layer protocols are implemented on top of one of the transport-layer protocols. For example,

HTTP is an application-layer protocol that uses a request-response, client-server model and is typically implemented on top of TCP to guarantee reliability for web applications. Therefore, each HTTP request to a server is wrapped with a TCP header, which implements the features of TCP for the packets carrying the HTTP information.

In LAISYC, we select HTTP as the primary application data transport protocol and UDP as the primary location data transport protocol. We discuss the rationale behind these decisions, as well as comparisons to other options, in the following two sections that focus on application data and location data, respectively.

#### 3.3.2.1.2 LAISYC Application Data Transport

Since reliability is required for LAISYC application data (e.g., session login and logout, server-side database accesses, application-specific distributed functions and logic), this data should be transported using TCP or a protocol relying on TCP.

The request-response model of HTTP fits well with the remote-procedure call-style, or web services, used by a client to send data to the server (e.g., username and password) and wait for a response back (e.g., session ID). Integrated development environments (IDEs), such as Netbeans and Eclipse, provide tools that enable rapid implementation of distributed functions using HTTP which would otherwise be tedious and time consuming to implement using TCP directly. Therefore, HTTP is a candidate for implementing web services from the mobile phone to the server.

A second option for a protocol to implement web services is SOAP. SOAP is a popular application-layer XML-based protocol often used to create enterprise web services that allow loosely-coupled servers to communicate with one another. Figure 6 also shows the



SOAP protocol's position in the networking stack, which is typically on top of HTTP. In other words, HTTP is used to transfer XML-encoded messages defined by the SOAP specification. SOAP defines complex functionality beyond HTTP such as token-based credentials, which allows an intermediary web application to receive, process, and forward data between the originating client and the destination server, without the originating client having exact knowledge regarding the destination server.

SOAP became popular in the mid-2000s as sophisticated enterprise networks evolved and a standardized method of exchanging XML-encoded messages between servers housed in different locations was needed. At first glance, it appears that since SOAP was developed to meet needs beyond HTTP, SOAP should be the logical choice to implement web services to carry application data in the LAISYC framework between a mobile device and server. As discussed in Chapter 2, many existing location-aware architectures use SOAP or XML to carry application and location data [91, 96-98, 100-102].

However, when considering the actual devices upon which LAISYC will be deployed, including the limited resources of mobile devices (e.g., battery energy, amount of data transfer), two problems quickly become apparent with SOAP and XML.

The first problem is the availability of SOAP-based communication on mobile devices. For Java ME, the JSR172 Web Services API [141], which implements the XML-based messaging protocol SOAP on top of HTTP, was defined for the Java ME platform in 2004 to allow mobile phones to directly access XML-based web services. However, the Web Services API is optional and only required to be supported in the high-end device segment in the Java ME MSA roadmap. Therefore, a limited number of device

manufacturers have implemented JSR172 and this API is typically not available on mid-to-low end devices [120]. SOAP support is limited in smart phones, as well. For example, the Google Android platform and iPhone iOS platform do not natively support SOAP.

**Table 2 - SOAP-encoded messages add a significant amount of overhead to web service requests, approximately 3.7 times as many characters, as shown in this example**

<b>SOAP-encoded web service request</b>	<b>HTTP-encoded web service request</b>
<pre> POST /busstoparrival/busstopws.asmx HTTP/1.1 Host: 99.999.999.999 Content-Type: text/xml; charset=utf-8 Content-Length: length SOAPAction: "http://tempuri.org/GetNextNVehicleArrivals" &lt;?xml version="1.0" encoding="utf-8"?&gt; &lt;soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"&gt; &lt;soap:Body&gt; &lt;GetNextNVehicleArrivals xmlns="http://tempuri.org/"&gt; &lt;n&gt;int&lt;/n&gt; &lt;RouteID&gt;int&lt;/RouteID&gt; &lt;DirectionCodeID&gt;int&lt;/DirectionCodeID&gt; &lt;BusStopID&gt;int&lt;/BusStopID&gt; &lt;TripID_External&gt;string&lt;/TripID_External&gt; &lt;/GetNextNVehicleArrivals&gt; &lt;/soap:Body&gt;&lt;/soap:Envelope&gt; </pre>	<pre> GET/busstoparrival/busstopws.asmx/ GetNextNVehicleArrivals? n=string&amp;RouteID=string&amp;Direction CodeID=string &amp;BusStopID=string&amp; TripID_External=string HTTP/1.1 Host: 99.999.999.999 </pre>

The second problem with SOAP from a mobile device perspective is that since all communication is wrapped within XML on top of HTTP, and each element has both an opening and closing XML tag, there is a significant amount of overhead to exchange messages between the device and server. Therefore, to represent the same amount of

information in a message, SOAP requires significantly more characters. Table 2 shows a comparison of the same web service request using SOAP on the left, and HTTP directly on the right. The SOAP message has approximately 3.7 more characters to represent the same amount of information. SOAP's additional overhead takes a toll on limited consumer data plans, as well as the overall cellular network bandwidth. Perhaps most importantly, as we show in Chapter 4 via experimentation, SOAP's additional overhead also has a heavy impact on mobile device battery life.

Since communication between the mobile device and server typically takes place over a wireless cellular network, the communication channel is prone to significant fluctuations in quality due to channel fading and movement of the mobile device, which can lead to a significant variation in transmission time and, subsequently, packet delays [142]. In third generation (3G) cellular networks, retransmissions at the link level are typically scheduled to give preference to mobile devices with higher quality connections, which can further delay end-to-end transmission time for packets originating from devices with poor signal quality [142]. Handoffs from one cell tower to another can also cause significant packet delays [142]. Since TCP was originally designed for wired networks, it was designed to interpret high packet delay as a sign of network congestion, and consequently TCP will reduce its transmission rate when it detects high packet delay in an attempt to better network conditions. However, as discussed above, in wireless networks high packet delay can originate from a variety of conditions that are not attributed to network congestion, and therefore slowing the device's transmission rate will not improve packet delay and will instead reduce the throughput of data communication from the device to server.

While there are a number of improvements proposed for TCP to help address these wireless network problems [143], these solutions are beyond the scope of our research since an application developer does not have influence over the implementation of TCP in a typical commercially-available mobile device. From a mobile application perspective, the one element that is under a developer's control when transmitting data via a web service is the amount of data being transferred. The amount of battery energy consumed by wireless communication is a function of how long the device radio is actively transmitting and receiving information. By reducing the amount of data being transferred from the mobile device to the server, not only is the impact on the user's limited data plan reduced, but battery life is extended. Reducing the amount of data being transferred also reduces the probability that interference in the wireless channel will result in lost packets transported via TCP, which reduces the need for retransmissions that would keep the radio on even longer. By eliminating the SOAP XML-based portions of messages between the device and the server, the amount of data wirelessly transmitted is significantly decreased. Therefore, even for the few mobile devices that natively support SOAP parsers, the use of HTTP-based web services is preferred.

Because of the above limitations of using SOAP-based web services on mobile devices, and because support for HTTP is required by the CLDC specification for all Java ME devices, we proposed that LAISYC use simple HTTP methods (e.g., GET, POST) for communicating application data from the mobile device to the server. Chapter 4 presents experiments illustrating the benefits of using HTTP-based web services instead of SOAP-based web services.

### 3.3.2.1.3 LAISYC Location Data Transport

Since location data are generated from a positioning technology on the mobile device such as GPS, these data must be transferred to a server to update the system on the mobile device's location. For real-time LBS (Chapter 1 - Need #2), this update rate can be up to once per second. Therefore, efficiency and timeliness is a top priority for location data transport in LAISYC.

We chose UDP, which is typically used for services where timeliness is favored over reliability (e.g., VoIP), as the protocol for efficient real-time location data transfer for LAISYC. The LAISYC framework treats streaming location data similarly to multimedia data in order to efficiently deliver timely location data from one entity in a location-aware information system to another. The choice of UDP for location data transport differs from previous location-aware architectures, largely because LAISYC is designed to meet the needs of real-time location-aware applications that are always on.

UDP is also preferable to TCP for location data because TCP's reliability mechanisms take a large toll on the mobile device. Retransmission of lost or significantly delayed packets over the wireless network via TCP costs precious battery energy, and since occasional loss of individual location data packets is acceptable in LAISYC, TCP's drawbacks outweigh its benefits. Therefore, UDP not only provides timeliness and scalability benefits for location-aware applications, but it also consumes less battery energy than TCP. In Chapter 4, we present results from experiments showing that under typical conditions the amount of GPS data lost via UDP is acceptable for most location-

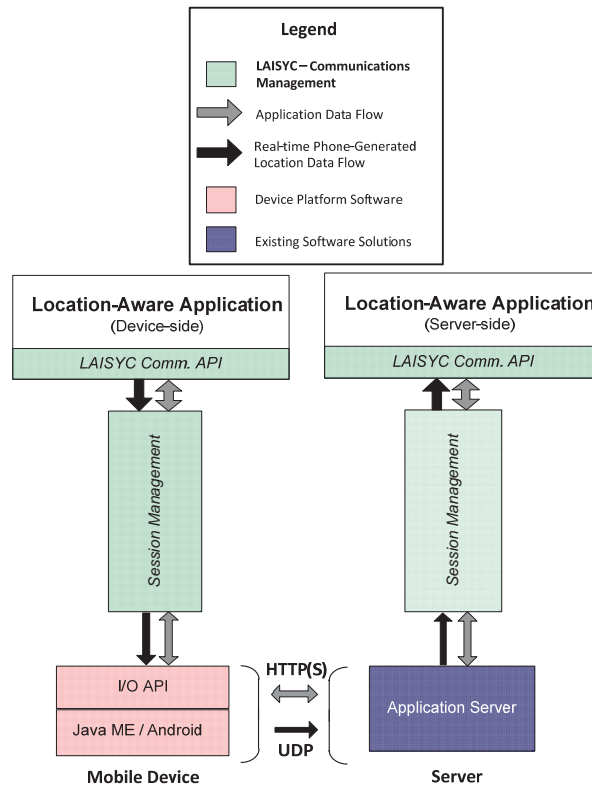
aware applications. The battery life benefits of using UDP, instead of TCP, as a location-data transport protocol are also demonstrated in Chapter 4.

Session Management modules are used on both the device and server to manage communication. The following section discusses the Device-Side Session Management module in detail, and the server-side module is discussed in the Server-Side Components section.

#### 3.3.2.1.4 Device-Side Implementation of Session Management

In summary, the LAISYC Session Management module within the mobile device splits data transferred to the server into two categories: application data, which is transported using HTTP, and location data, which is transported using UDP. Figure 7 shows this two-tier communication between the mobile device and server within the Session Management module in isolation from the rest of the LAISYC framework.

Two-tier protocols, using both application-layer and transport-layer protocols in the same application, have been utilized in the past to increase VoIP performance for mobile devices [144], but have not been used in previously presented location-aware architectures. Splitting application logic and location data has architectural advantages in addition to battery life advantages, such as allowing easier integration of location data with existing HTTP-based web applications, and supporting dynamic load balancing of incoming location data packets at the network level without examining the contents of payload (since we know any data transported over UDP is location data).



**Figure 7 - The Session Management modules use HTTP for application data and UDP for location data for communication between the mobile device and server**

Since location data will be arriving at the server with much greater frequency than application data, and can be treated as atomic packets that do not need immediate responses, the server may wish to handle UDP traffic differently than HTTP traffic. For secure HTTP communication, HTTPS can also be used in place of HTTP. We discuss UDP security in a later section.

The mobile application interfaces with the Session Management module via the LAISYC Communication API. For application and location data, the mobile application initiates communication with the server via a HTTP request, or transmission of location data via UDP. The module creates a session with the server for a device by calling a *createSession()* web service and passing in a variety of information including username,

password, phone number, and other authorizing information. The server responds with a unique session identifier that is used in subsequent communication with the server to aid the server in pairing location data received over UDP and application-specific web service instructions received over HTTP with a specific session. The server maintains a registry of connected devices that have open sessions at the server, which includes the current address (e.g., IP address) of each mobile device. The mobile device Session Management module prevents the application from having to directly manage sessions by implicitly controlling the creation and destruction of sessions surrounding the transfer of application and location data to the server. In other words, if a mobile application calls an application-specific web service or attempts to send location data through the Session Management module, the module will first check to see if there is an open session with the server, and if not, it will create one. Therefore, it is guaranteed that a session always exists at the server before any application or location data from the device is submitted to the server. To signal to the server that a session is finished, the module initiates a *destroySession()* web service.

The implicit management of sessions by the device-side Session Management module relieves the application from having to actively manage the concept of a session, which simplifies client-side application logic and also increases the efficiency of the server. For example, if the application can interact with the user using data cached from previous execution and can provide client-side functionality without needing to contact the server, then a session does not need to exist at the server. If no location data has been generated from the device since the application has started, there is no point in holding an open session at the server until data actually exists. In a system which will potentially have



thousands of simultaneous users, it is important to reduce server overhead for unneeded sessions whenever possible to allow the system's server-side memory requirements to scale at a rate potentially less than  $O(n)$ , where  $n$  is the number of devices.

In conclusion, the Session Management module is designed to address several of the needs for location-aware mobile apps outlined in Chapter 1. Session Management is designed to improve battery life (Need #1) while enabling real-time location data communication (Need #2) between the phone and server. In Session Management, HTTP-POST is selected over SOAP for real-time (Need #2) application data transfer to reduce the impact on mobile device resources (Need #1). UDP is selected over TCP as the location data transport protocol for real-time location data (Need #2) to reduce the impact on mobile device resources (Need #1). Session Management is based on protocols accessible to third party mobile apps (i.e., HTTP, TCP, UDP), and therefore it is fully implementable by any third part mobile app developer (Need #4).

#### 3.3.2.2 Adaptive Location Data Buffering

The purpose of the Adaptive Location Data Buffering module is to increase the reliability of real-time location data communication with the server in an energy-efficient manner.

Since UDP is utilized for location data transport, no end-to-end reliability exists for location data such as that provided by TCP. As discussed earlier, lack of reliability for each packet is a design trade-off in favor of the general efficiency of the system; while a large number of location fixes can be transferred to a server in a timely manner, there are no acknowledgments by the receiving entity that the location data has arrived, no retransmission of lost packets, and no guarantee of the order of delivery of packets. In

real-time tracking, the occasional loss of a few location data fixes is of no concern, since another location update will soon follow. However, location data is often referenced after-the-fact in order to provide certain metrics, such as distance traveled, as well as to identify the paths of users on particular days. Therefore, while a few occasional lost location data packets are acceptable, the loss of large numbers of contiguous fixes can introduce significant problems.

While we demonstrate in Chapter 4 that under ideal conditions UDP has an acceptable percentage of data loss on a cellular network, from our experiments we discovered that there are two primary causes of occasional large contiguous losses of location data:

- 1) Voice communication interference on devices that cannot handle simultaneous voice and data communication (e.g., CDMA devices)
- 2) Cellular network coverage gaps

In the United States, devices on CDMA networks (e.g., Verizon and Sprint) are not capable of simultaneous voice and data communication. As a result, if an application continues to transmit location data via UDP after a user picks up a voice call, these packets are lost. In our experiments we have confirmed that on several devices (e.g., Sanyo 7050, Sanyo Pro 200, and Sanyo Pro 700) the Java ME platform does not trigger any error messages when a UDP transmission is attempted during a voice call. Since a voice call could last a long period of time (e.g., 30 minutes) while the user is traveling, a large amount of data could be lost if no action is taken by the application.

The lack of simultaneous voice and data transmissions communication also has a secondary adverse effect on the cell phone user while an application is running in the

background: if the application is constantly transmitting data to the server, it will occupy the CDMA radio and any incoming voice calls will be sent to voicemail instead of ringing at the phone. Obviously, interfering with incoming voice calls is not a desirable trait for an application, and therefore we must address this problem in LAISYC.

Poor cellular coverage in certain locations (e.g., rural areas) can also result in lost UDP data packets. Since lack of a cell signal is typically geographically-correlated due to poor tower coverage, large contiguous chunks of the user's path can be lost if the device loses data communication with the server.

A simple solution to both the simultaneous voice and data problem and poor cellular coverage problem is to store all location data locally on a device and upload the location data at the end of the day. However, this solution does not support real-time location-aware applications, which is a requirement for LAISYC (Chapter 1 - Need #2).

Additionally, Java ME devices typically have limited persistent storage capacity and may not be able to store an entire day's worth of location data. An alternate solution is to use TCP instead of UDP, but as we discussed earlier, the entire suite of reliability mechanisms used by TCP are not necessary for most location-aware applications and these mechanisms also have a significant negative impact on mobile device battery life.

Our solution to these problems is the Adaptive Location Data Buffering module in LAISYC. Adaptive Location Data Buffering provides a basic quality of service mechanism when UDP is used as the location data transport protocol, but at a much cheaper cost than using TCP for every location data transmission. Adaptive Location Data Buffering is implemented through use of device-side APIs regarding cell signal

quality and cell network status, as well as occasional TCP transmissions to confirm end-to-end connectivity with the server.

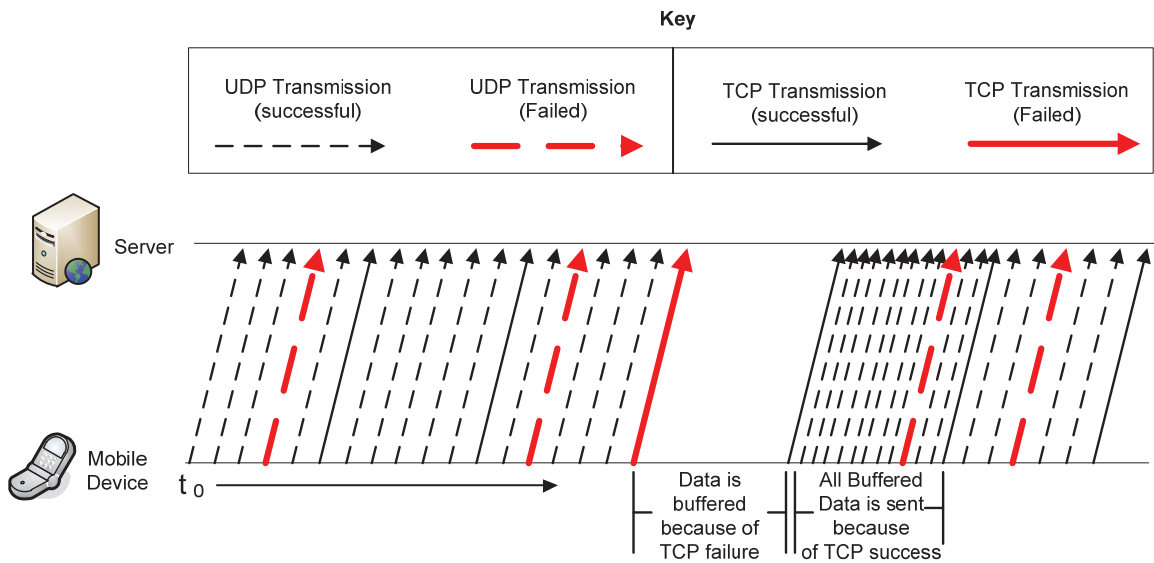
In Adaptive Location Data Buffering, under normal conditions UDP is continuously used to transfer location data to a server. Before each UDP transmission, the software checks device-side APIs (if available) in order to assess the current level of cellular signal in order to determine if a successful UDP transmission is probable given the current wireless environment. Additionally, if the Java ME environment supports error reporting for unsuccessful UDP transmission attempts, these exceptions can also be an indication of an unsuccessful location data transfer. If there is a low level of wireless signal, or if an exception is thrown, the location data is buffered to memory or to persistent storage such as the MIDP Recordstore. Once it is detected that UDP transmissions are likely to succeed, the buffered data is then sent via UDP and deleted on the device.

While the above method attempts to increase the probability that a UDP transmission will successfully be issued by the device, these methods do not verify an open connection with the server. Therefore, a more reliable method is required to occasionally determine if the server is properly receiving location data.

Adaptive Location Data Buffering also occasionally sends data via TCP to determine if there is a successful response from the server. If there is no response (e.g., the phone is off-network, the server is down, the user is on a voice call blocking data communication), then the software begins buffering location data until the next successful response via TCP. Upon the next successful response, the buffer is emptied and all location data is sent via UDP. TCP transmissions should only be sparsely attempted, since the benefits of

utilizing UDP over TCP as the primary location data transport protocol will disappear if TCP transmissions are too frequent. In Chapter 4, we demonstrate the energy tradeoffs between TCP and UDP when transmitting location data.

Figure 8 shows a simulated timeline of Location Data Buffering where location data is being transmitted via UDP. An occasional lost UDP transmission is unknown to the device, and acceptable for the system.



**Figure 8 - A timeline of Location Data Buffering which shows a TCP failure that results in a series of buffered location data fixes, which are transmitted to the server on the next successful TCP transmission**

When a TCP failure occurs, location data is buffered until the next successful TCP transmission. At this time, all buffered location data is sent to the server via UDP.

Figure 9 is a data flow diagram showing the execution of Adaptive Location Data Buffering as it is currently implemented in LAISYC.

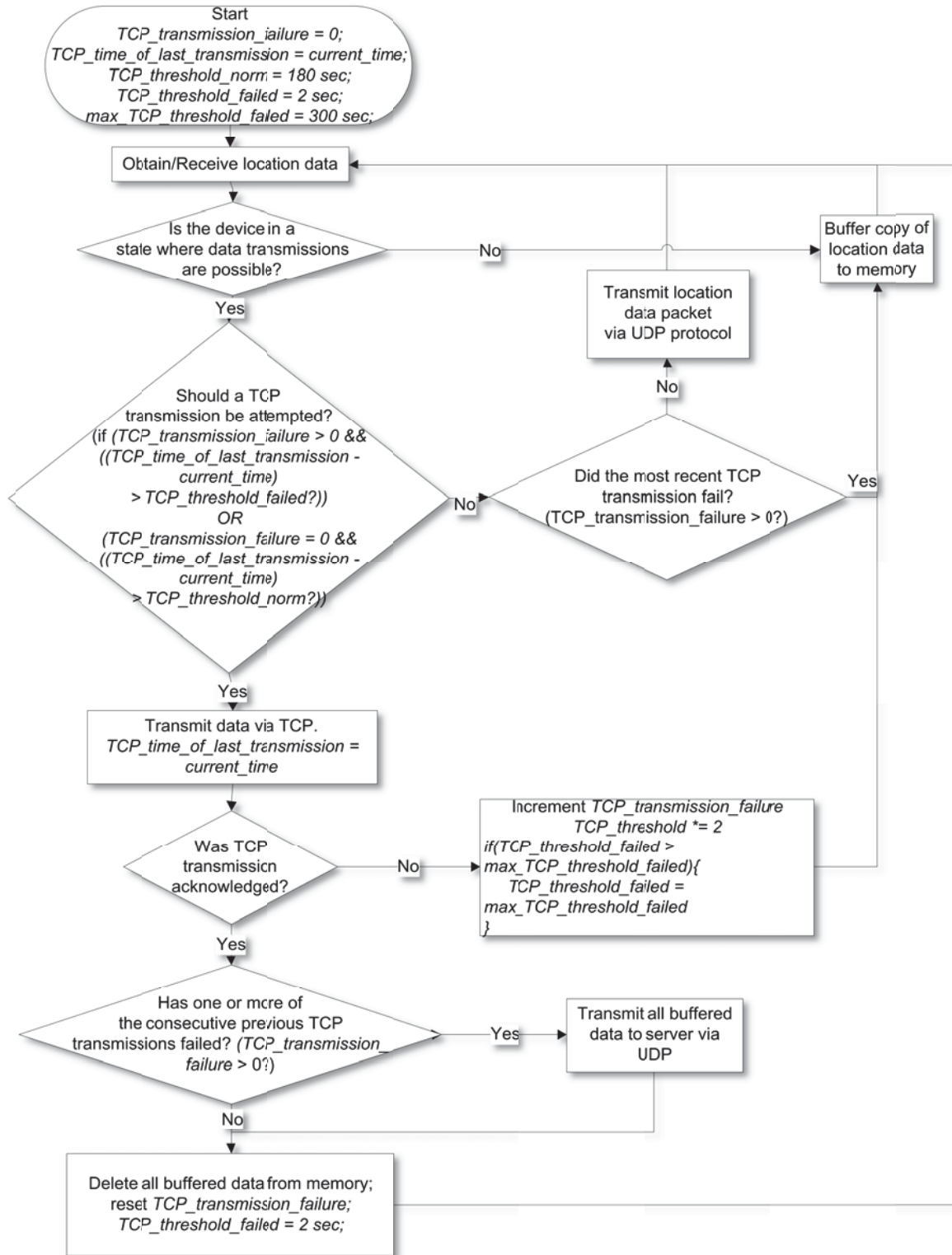
Sample execution of the Adaptive Location Data Buffering algorithm that traces this data flow diagram follows.

On the generation of each new location data point by the device's positioning system, we executed an algorithm (Figure 9) to determine whether or not a TCP check with the server to confirm an open data connection should occur.

We used two time-based thresholds to track whether or not a TCP check should occur:

- *TCP\_threshold\_norm* – minimum amount of time between TCP checks with the server when TCP checks are successful. The default value we used in LAISYC is 180 seconds.
- *TCP\_threshold\_failed* – minimum amount of time between TCP checks with the server when TCP checks are failing. This value starts at 2 seconds for the first failure, and doubles on each consecutive failure until it reaches a maximum threshold value of *max\_TCP\_threshold\_failed* (default value of 300 seconds in LAISYC). This enables a quick recovery and small overhead for intermittent failures, but also provides an exponential back-off with an upper limit to avoid contacting the server frequently during an extended failure of communication between the device and server.

If a TCP check with the server should not occur and if the last TCP check failed, then the location data is buffered in memory on the device. On Java ME phones, we buffered up to 2048 bytes to volatile memory, and subsequent data was buffered to persistent memory (i.e., the MIDP Record Store in Java ME).



**Figure 9 - Adaptive Location Data Buffering occasionally checks for an open connection with the server via TCP to increase the probability of successful UDP transmissions**

If a TCP check with the server should not occur and if the last TCP check was successful, then the location data is sent to the server via UDP like normal.

Adaptive Location Data Buffering also serves to reduce interference with incoming phone calls for devices that do not support simultaneous voice and data sessions. In early implementations of LAISYC we found that transmitting location data every four seconds resulted in the communication link being continuously occupied by data transmissions. Therefore, any incoming voice calls went directly to voicemail instead of ringing at the phone. By implementing a buffer size of ten to aggregate several GPS fixes and send all GPS data in a burst of UDP packets, interference with voice calls was eliminated.

In conclusion, the Adaptive Location Data Buffering module is designed to address several of the needs for location-aware mobile apps outlined in Chapter 1. Adaptive Location Data Buffering is designed to increase the reliability of real-time location data communication (Need #2) with the server in an energy-efficient manner (Need #1).

Adaptive Location Data Buffering is based on protocols (i.e., TCP, UDP) and persistent storage (i.e., MIDP Recordstore) accessible to third party mobile apps through platform APIs, and therefore it is fully implementable by any third party mobile app developer (Need #4).

### 3.3.2.3 Critical Point Algorithm

The purpose of the Critical Point Algorithm module is to reduce battery energy expenditures and the amount of data transferred between the mobile phone and server by eliminating non-essential GPS data.



Since GPS technology in a mobile device can generate a significant amount of location data, this data must be carefully managed to avoid wasting precious resources such as battery energy or cellular data transfer. If every GPS fix that is calculated on-board a mobile device is transferred to a server, a large amount of battery energy is consumed. Additionally, in the U.S. many cellular data plans have an upper limit on the amount of data that can be transferred from the mobile device over the cell network per month. If every GPS fix is sent over the cellular network, this data will have a large negative impact on the consumer's data plan.

In our research, we observed that a large number of GPS fixes generated on a mobile device may not contain useful information for many applications that are primarily interested in the travel path of the device. For example, GPS generates a large number of very close but different positions when the user is standing still; a single GPS fix could adequately represent this same information. Additionally, when the user is traveling in a straight line, a large number of points may lie upon the same vector, which can be represented using only the start and end point of the vector. Therefore, the path of the user could be accurately represented by using only a small portion of the GPS data generated by the mobile phone.

The Critical Point (CP) algorithm was created in order to filter out non-critical location data points out of a real-time stream of location data. Location data points are defined as a set of data containing latitude, longitude, and speed information at a minimum, and may include other information such as altitude, accuracy uncertainty, and heading. We defined non-critical data points as redundant data that does not contribute to the

knowledge of the path of a device. While the CP algorithm could be used with 3-dimensional data, since consumer-level GPS-enabled phones are not currently able to provide accurate altitude information [18], we focused on 2-dimensional data in the description and analysis of the CP algorithm.

The CP algorithm can be seen as reducing a stream of location data into a series of connected vectors. In other words, points along the vector are discarded since they do not contribute additional path information. A path will always have at least 2 critical points, which are the starting and ending points, since the simplest path is a straight line. Non-critical points are points that lie directly between two critical points so that if a line was drawn between the two critical points, it would intersect the non-critical points between them. Non-critical points are also gathered while a device is standing still (i.e., redundant location data).

Changes in direction along with speed information are used to identify a critical point. In other words, if a device is traveling in a straight line but changes direction, a new critical point must be recorded at this change in direction. The resulting path is a series of vectors with a critical point defining the vertex between vectors.

A flowchart describing the execution of the CP algorithm is shown in Figure 10. The CP algorithm is executed each time a new GPS position of the mobile device is calculated. Each time the algorithm is executed, it selectively retains memory of past input and uses this to determine whether or not a critical point exists. If a critical point exists, it will return the point that has been determined as critical. If a point has not been determined as critical based on current input, then it returns null. The CP algorithm uses a speed

threshold to determine the appropriate minimum speed that should be used to filter out data when the device is not moving. The CP algorithm uses azimuth calculations to determine the change in direction when the device is moving (Figure 11).

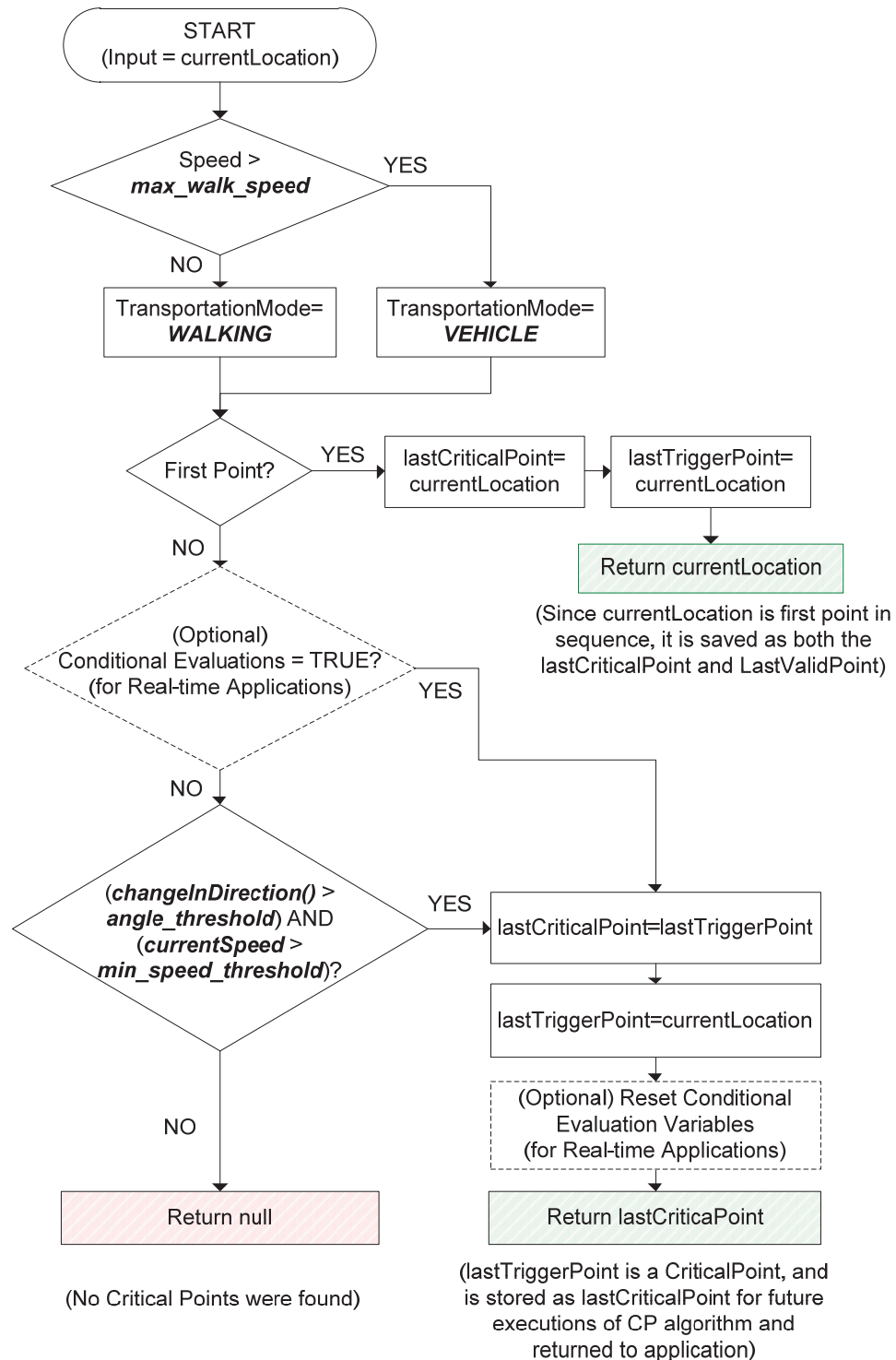
Azimuth is a measurement used to determine difference in angle given a reference plane and two points. We used Vincenty's Inverse algorithm to calculate the azimuth values [136], which has been shown by Vincenty to be accurate to within 0.000015 seconds [136] in angular Degrees-Minutes-Seconds (DMS) notation, where a degree of angle is equivalent to 60 minutes, and a minute is equivalent to 60 seconds.

For each execution of the CP algorithm, we evaluated the azimuth for two pairs of points:

- $Azimuth_1$  = Azimuth of the Last Critical Point and the Last Trigger Point in relation to true north (shown in Figure 11)
- $Azimuth_2$  = Azimuth of the Last Critical Point and the Current Point in relation to true north (shown in Figure 11).

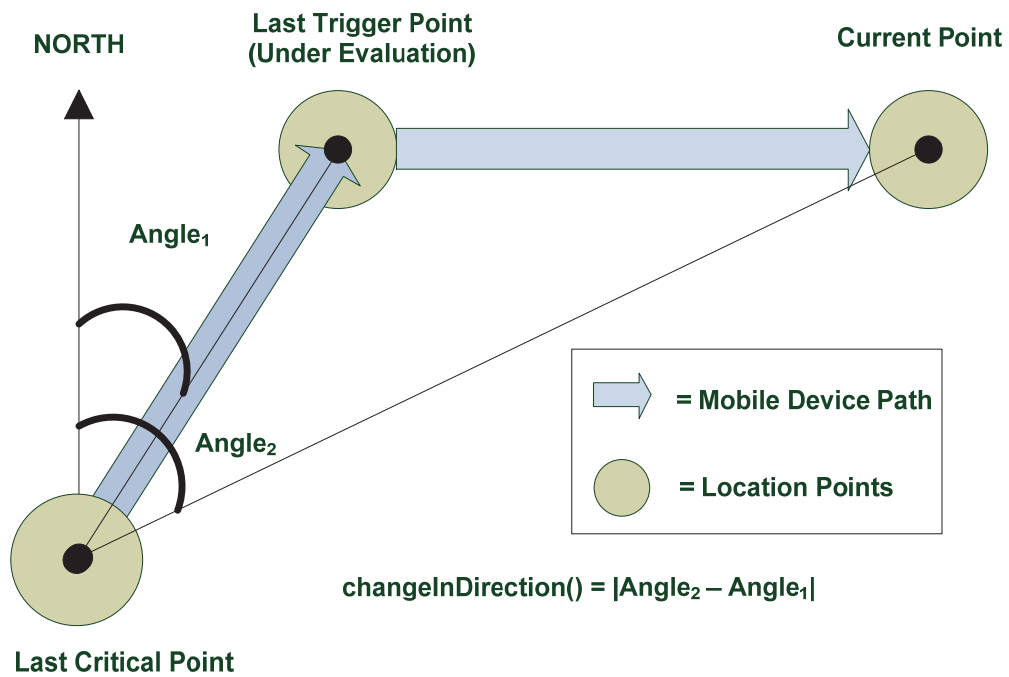
The absolute value of the difference of  $Azimuth_1$  and  $Azimuth_2$  yields the change of direction of the device for the Last Trigger Point.

The CP algorithm keeps references to the three points (Last Critical Point, Last Trigger Point, and Current Point) throughout its execution, which can be viewed as a three point sliding window over a stream of location data (Figure 12). The second of the three points is always the point under consideration by the CP algorithm to determine if it is a critical point.

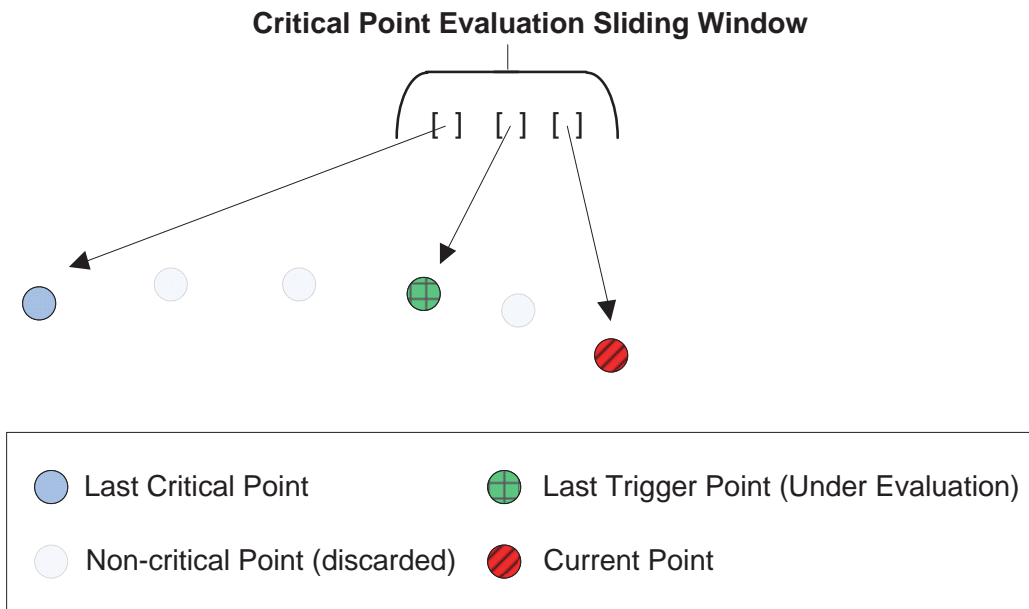


*changeInDirection()* = Angle Calculation (varies depending on the current speed)  
*angle\_threshold* = Minimum change in azimuth when a point is marked Critical  
*min\_speed\_threshold* = Minimum speed when device is considered moving

**Figure 10 - The Critical Point Algorithm filters out GPS fixes that are not necessary to recreate the user's path. [119] © 2008 IEEE**



**Figure 11 - Azimuth calculations are used in the Critical Point Algorithm to determine change in direction. [119] © 2008 IEEE**



**Figure 12 - The Critical Point Algorithm maintains a reference to three points that are used to determine whether the second of the three points is a critical point**

As a result, the most recent location point passed into the CP algorithm is not the same point that is returned by the algorithm as the critical point. This is because the CP Algorithm is stateful and remembers its past input for three location data points, and it uses this information to calculate changes in azimuth and speed that determine whether or not a point is critical of the second of the three points in the sliding window.

The change in direction threshold can also be dynamically adjusted based on the current speed. For example, one change in azimuth threshold can be used for speeds less than 10 meters/sec, and another azimuth threshold can be used for speeds greater than 10 meters/sec. In other words, the variation of changes in direction while walking may be high and a larger threshold value may be used to determine a critical point. Similarly, the variation of changes in direction while driving in a car at high speeds may be low and therefore a lesser threshold value can be used.

To support the above features, the following thresholds are used in the CP algorithm:

- *min\_speed\_threshold*: location data with speed values under this threshold are discarded as non-critical points, since the user is considered to be standing still
- *max\_walk\_speed*: if the speed is less than this threshold, the angle threshold for walking will be used, otherwise the angle threshold for a vehicle will be used
- *angle\_threshold*: if the absolute value of the difference of azimuth values exceeds this threshold, then the point is considered a critical point. One angle is used for walk trips, while another is used for vehicle trips.

The CP Algorithm also supports several optional evaluations (as seen in Figure 10) that can be used by real-time applications to transition between several different server update strategies:

- `HasTimerExpired()`?: A timer is started when a critical point is determined, and a new critical point is identified after a certain amount of time elapses. This would ensure that a position was reported at a minimum given interval, in case the device is stationary for long times or traveling in a straight line for an extended period of time. For example, after 5 minutes, if a critical point has not yet been determined, then the next point would be considered a critical point.
- `HasDistanceCounterExceededThreshold()`?: A distance counter is started after a critical point is found. While the device is traveling in a straight line, for each position update the distance would be increased. Once the device exceeds a threshold for distance traveled, then it declares the next point a critical point and sends this point to the server. This method assures that the server will receive position updates for a device before it travels more than a certain distance from the last reported point.
- `ReceivedLocationProbe()`?: If the device receives a location update request from a server, then the next point is automatically determined to be a critical point and sent to the server.

A sample execution of the CP algorithm follows.

A LBS application starts up and makes a request to the Location API to generate a new GPS position every 4 seconds. The critical points algorithm is executed every time a new

position is generated, beginning with the first fix. The first GPS fix will be determined as critical, and information about this fix is then saved within the algorithm. The device generates a new position 4 seconds later, and the application inputs this data into the CP algorithm. The algorithm outputs null, since it does not have enough information at this point to determine whether this second point is a critical point (i.e., it needs a third fix to calculate both azimuth values). However, it saves information about this fix for future CP calculations, as future calculations might determine this point as critical. The device then generates a third GPS fix, and the application inputs this into the Critical Point Algorithm. If the difference in azimuth between the first and second fix and the first and third fix exceeds an angle threshold value *and* the speed value for the third fix exceeds a speed threshold (i.e., the device is not stationary), then the second fix is determined to be a critical point, and the second fix is returned by the algorithm. Information about the second and third fix is then saved for future CP calculations. If the difference in azimuth values does not exceed the threshold for change in direction, then the CP algorithm saves information about the second fix and returns null. The device then generates a 4th fix, and the application inputs it into the CP algorithm. Assuming a critical point was just identified in the previous step, if the difference in azimuth between the second and third fix and second and 4th fix exceeds an angle threshold value and the speed value for the 4th fix exceeds a speed threshold (i.e., the device is not stationary), then the third fix is determined to be a critical point and the third fix is returned by the CP algorithm. Information about the 4th fix is saved for future CP calculations. This process continues until the final fix for series is calculated, at which point the final fix is determined to be a critical point.



In conclusion, the CP Algorithm module is designed to address several of the needs for location-aware mobile apps outlined in Chapter 1. The CP Algorithm is designed to reduce battery energy expenditures (Need #1) and the amount of data transferred between the mobile phone and server (Need #1) by eliminating non-essential GPS data (Need #3) in real-time (Need #2). The CP Algorithm uses attributes from location data provided by the Location API (e.g., latitude, longitude, speed), and therefore is fully implementable by any third party mobile app developer (Need #4).

We demonstrate the battery energy and data transfer savings of the CP Algorithm in Chapter 4, and also define a methodology for selecting values for the thresholds used in the CP algorithm.

#### 3.3.2.4 Location Data Encryption

The purpose of the Location Data Encryption module is to ensure the security of the location data being transferred between the mobile phone and the server in an energy-efficient manner.

The main threat to a breach of privacy by untrusted parties in a location-aware architecture is the interception of location data as it is being transferred from a mobile device to a server over the Internet. While secure TCP connections are supported by the Java ME platform through the use of SSL, there is no secure support for datagrams sent via UDP in Java ME [140, 145]. Therefore, the implementation of secure UDP transmissions is left to the application developer. Since we have chosen UDP as the primary location data transport protocol, as discussed in the Session Management section

earlier in this chapter, we must examine the potential for securing UDP communications to protect the privacy of the user.

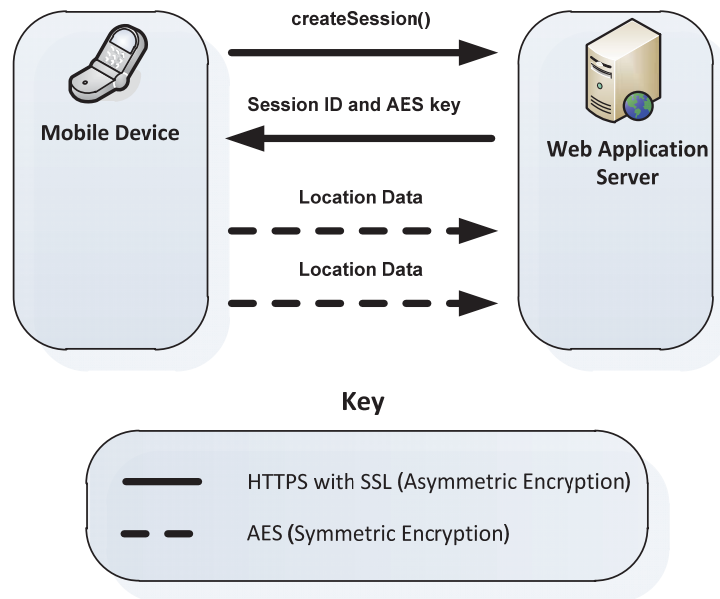
The Location Data Encryption module handles the encryption of location data in the payload of the UDP datagram to enable end-to-end security between the mobile device and the server. This module is optional, and is only needed for applications that require secure location data transfer. As discussed in the earlier Session Management section, the only information included in a UDP packet sent from a device to a server is a unique session identifier (i.e., an integer) and the latitude, longitude, speed, and other location data. The session identifier is not related to the user or device identifier and changes at least once daily (i.e., every time the application calls the *createSession()* server method), and therefore some applications may not need to encrypt the data transferred over UDP. However, we define this optional Location Data Encryption module for applications that require highly secure and private communication.

Symmetric encryption (e.g., Advanced Encryption Standard (AES)), which uses a shared key between two parties, is generally more efficient than asymmetric encryption [146]. However, symmetric encryption requires a secret shared key that is known only by both parties before any communication can take place. Since a device initiates communication with a server over a wireless network, we must use a different method to secure an initial information exchange between the device and server.

HTTPS uses asymmetric encryption, which does not require the exchange of a shared secret key. Instead, HTTPS uses public and private key cryptography. The device uses the server's public key to encrypt information, and then sends this information to the

server. Since the server is the only entity that has possession of the private key that can decrypt the information, the initial exchange between the device and server is secure even without exchanging a shared secret key. However, one drawback to asymmetric encryption is that it is less efficient and more computationally intense than symmetric encryption [146].

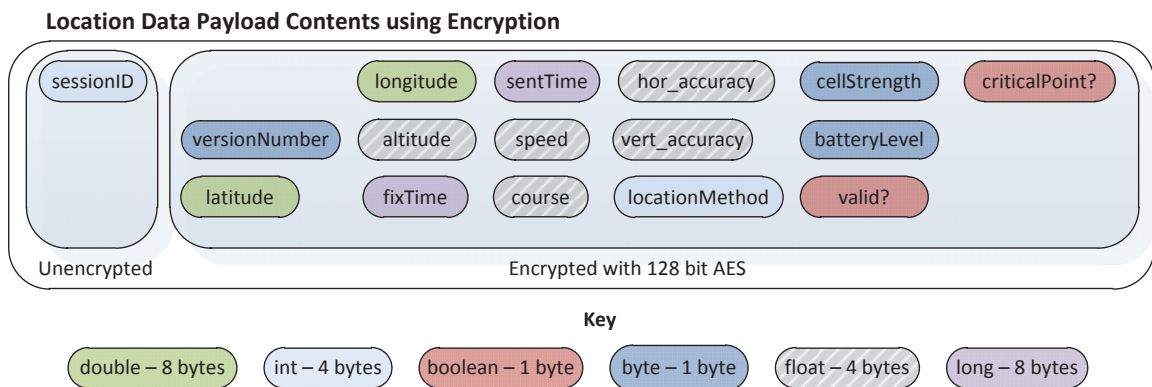
In LAISYC, we define a hybrid cryptosystem using both asymmetric and symmetric encryption to provide a secure and efficient exchange of information. Figure 13 shows the secure exchange between the device and server in this hybrid cryptosystem.



**Figure 13 - LAISYC uses a hybrid cryptosystem to protect the exchange of the AES key using HTTPS with SSL, and then uses the AES key to encrypt the location data sent over UDP**

We use HTTPS and SSL (i.e., asymmetric encryption) to protect the initial exchange of a symmetric encryption shared secret key, which occurs during the invocation of the initial `createSession()` web application method. The server uses a different symmetric key for each connected device, and a new symmetric key is generated at the start of each session

(i.e., approximately every 24 hours). After the device has the symmetric encryption key, it uses this key to encrypt all location data information, except for a session identifier, in the payload of UDP packets that is sent to the server. The session identifier is left unencrypted so that the server can identify the proper key to decrypt the data for each session. Figure 14 shows the contents of the UDP payload when encryption is used in LAISYC.



**Figure 14 - 128bit AES is used to encrypt the location data in the UDP payload, with the exception of the session ID which is used by the server to identify the correct symmetric key per device session**

RC4 and Advanced Encryption Standard (AES) are two popular candidates for symmetric encryption. According to Prasithsangaree and Krishnamurthy [146], tests executed on a mobile Pentium III processor in a laptop show that AES is more energy-efficient for packet sizes of less than approximately 100 bytes, while RC4 is more energy-efficient for packet sizes of more than 100 bytes. AES is preferred from a security perspective, since several weaknesses have been exposed in RC4 [146]. The typical payload size of a location data packet which contains location data and a limited amount of application-specific data to be transmitted over UDP is approximately 78 bytes. Therefore, we chose AES as the symmetric encryption method for LAISYC.

In conclusion, the Location Data Encryption module is designed to address several of the needs for location-aware mobile apps outlined in Chapter 1. Location Data Encryption is designed to ensure the security of the location data being transferred in real-time (Need #2) between the device and server in an energy-efficient manner (Need #1). AES was chosen since it is more secure and energy efficient than other methods and can be easily implemented by third party mobile app developers (Need #4) using existing libraries such as BouncyCastle [147]. In Chapter 4, we evaluate the battery life impact of Location Data Encryption.

### 3.4 Server-Side Components

The server-side modules in LAISYC exist to support the mobile device-side modules, and act as a proxy for database access. The LAISYC server-side modules are shown in Figure 15.

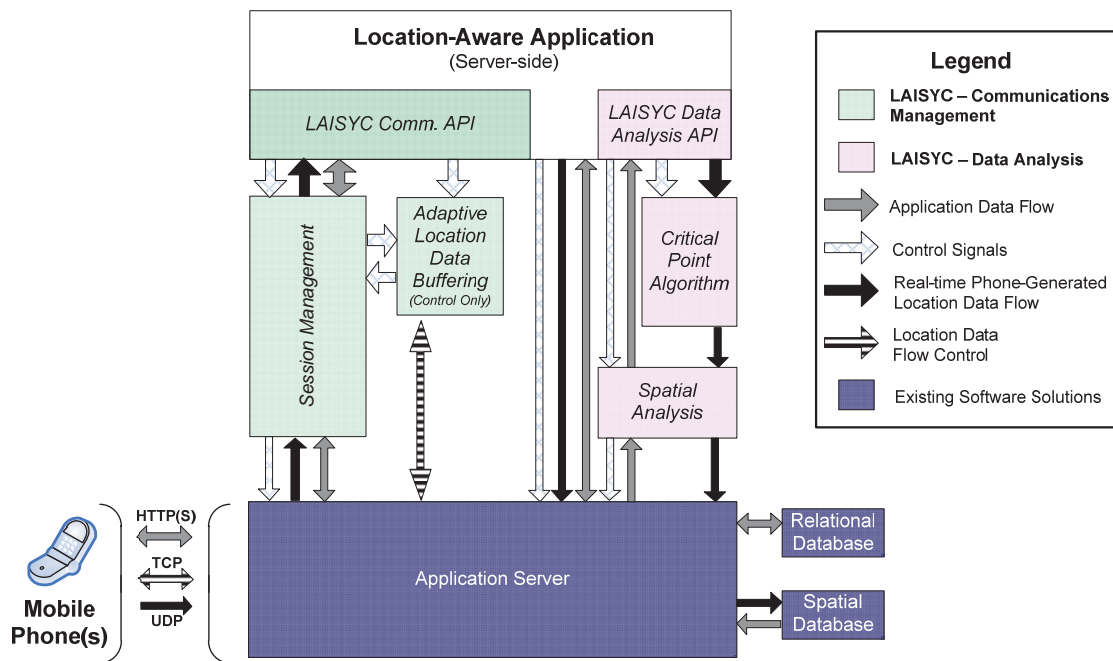


Figure 15 - LAISYC server-side modules. [118] © 2011 IEEE

As mentioned earlier, the focus in this dissertation is primarily the design, implementation, and evaluation of the device-side LAISYC modules. Therefore, the server-side modules are presented in this dissertation to describe how they support or benefit from the LAISYC device-side modules.

Server-side modules are broken down into two categories:

- 1) Communications Management (Green shaded modules in Figure 15)
- 2) Data Analysis (Pink shaded modules in Figure 15)

### 3.4.1 Communications Management

Communications Management on the server-side of LAISYC consists of the Session Management and Adaptive Location Data Buffering modules.

#### 3.4.1.1 Session Management

The server-side Session Management module coordinates communication with multiple connected client devices. In order to tie multiple web service calls over HTTP and location data sent via UDP together, upon the creation of a new session, a unique session identifier is passed back to the mobile device. This session identifier is then used in subsequent device-initiated communication between the device and server in order to identify the device. This identifier allows streamlined communication between the device and server, since login information does not have to be transferred and authorized for each communication between the device and server. HTTPS can be used to encrypt web service calls from the phone so that login information, session identifier, and

application-specific data are all protected. Support for HTTPS is mandated for MSA 1.0-compliant Java ME devices [111].

A limited amount of information for each session (e.g., session ID, device phone number and IP address, most recent location update) is kept in main memory inside the application server to enable a rapid response to the device based on incoming location data. Since the application is able to immediately execute threads to take action based on real-time location information received over UDP, timely location-based services can be executed from the server. While extremely time-sensitive actions such as real-time navigation must be handled by software executing on the mobile device, near real-time functionality with less stringent time constraints can be implemented server-side without experiencing the delay of first writing to a disk in the database management systems and then waiting for database triggers or a separate application to receive and process the information. The disk-based database contains a record of all user and location data and serves as the persistent backup of information contained in the application server memory.

The server-side Session Management module also handles the expiration of sessions for devices which have not communicated with the server in a certain amount of time in order to de-allocate memory assigned to that session. These “abandoned” sessions could be caused by an unexpected termination of the device’s client-side software or by a phone that is currently off-network and is unable to communicate with the server. Session information is always saved to a database management system to enable the transparent restart of the server in case of server hardware or software failures, as well as to allow the

application server to dynamically load and unload sessions to and from main memory. If the device tries to use a session that has been expired and removed from main memory, the Session Management module is able to reload the information from the database and reuse that session. Therefore, the expiration and re-initialization of the session is also transparent to the device.

#### 3.4.1.2 Adaptive Location Data Buffering

The server-side implementation of Adaptive Location Data Buffering responds to the TCP communication initiated by the Adaptive Location Data Buffering module on the mobile device to confirm that there is an open session between the mobile device and the server. Before taking any action, the Adaptive Location Data Buffering module confirms that a session exists for the given session ID through communication with the Session Management module.

#### 3.4.2 Data Analysis

Data Analysis consists of the Critical Point Algorithm and Spatial Analysis modules.

##### 3.4.2.1 Critical Point Algorithm

In LAISYC, we replicate the CP algorithm on the server-side, as well. The server-side CP algorithm is only used if the CP algorithm on the device is deactivated to allow all location updates to be transferred from the mobile phone to the server. Transferring all location data points may be desirable when tracking second-by-second. Therefore, before location data is input into any Location Data Analysis modules (e.g., Spatial Analysis), it is pre-filtered using the Critical Point Algorithm in order to reduce the information into a meaningful path that can be better analyzed.



### 3.4.2.2 Spatial Analysis

The purpose of the Spatial Analysis module is to provide near real-time location based services to mobile users that cannot be accomplished on the mobile phone due to processing or data storage constraints.

In order to provide intelligent location-based services to mobile users, it is desirable to provide information to the user which is highly relevant based on both their real-time position, as well as historical or future intended travel behavior. Location-based alerts should be given to travelers as soon as it is determined that the information is relevant, and before they reach the area to which the alerts pertain, in order to allow users to plan and react accordingly. However, to avoid inundating users with meaningless information, the information should be highly relevant and precisely targeted. For example, a traveler would ideally want to know of an incident along the typical path from home to the destination before even leaving their home. This would allow the user to take an alternate path to the destination or even delay the trip until a time when the congestion has cleared. However, a user would not want to be alerted of hometown incidents when traveling outside the hometown. A user would want to be notified as soon as possible, if wandering off the planned path.

One method of delivering relevant alerts to a traveler is to examine the real-time and/or spatial attributes of the traveler's past travel behavior in conjunction with a spatial database. LAISYC is designed to support real-time location information exchange from a phone to a server so that these types of services are possible. The Spatial Analysis module in LAISYC can utilize massive server-side spatial databases to provide services

that cannot be provided on the mobile device, due to memory and storage space constraints, as well as lack of spatial database support.

To demonstrate the ability for the LAISYC to provide real-time services based on spatial databases, we have focused on two specific implementations of the Spatial Analysis module:

- 1) Path Prediction and Traffic Incidents – Within our TRAC-IT application, we have implemented a spatial path-based prediction of the user’s travel to provide real-time traffic alerts based on the user’s real-time and historical location information. We discuss this application in detail in Chapter 4 along with the TRAC-IT system.
- 2) Lost user alerts – In our TAD application, we have implemented the ability to detect if a user has deviated from a planned transit route. We discuss this application in detail in Chapter 4 along with the TAD system.

## **CHAPTER 4: EVALUATION**

### 4.1 Note to Reader

Some experimental results presented in this chapter have previously been published in a variety of journals, and several patents are pending or issued on the related technology. Experiments related to GPS Auto-Sleep in this chapter have been published in IEEE Pervasive Computing [118] (© 2011, IEEE), and Proceedings of UBICOMM '08 [119] (© 2008, IEEE), and a 2011 issue of the Journal of Navigation [18] (Copyright © 2011 The Royal Institute of Navigation) are reprinted here with permission of IEEE and Cambridge University Press. Experiments for Session Management and Adaptive Location Data Buffering have been published in IEEE Pervasive Computing [118] (© 2011, IEEE) and the Transportation Research Board (TRB) Transportation Research Record [120] (© 2010, TRB) and are reprinted here with permission of IEEE and TRB. Portions of the experiments for the Critical Point Algorithm have been published in IEEE Pervasive Computing [118] (© 2011, IEEE) and Proceedings of UBICOMM '08 [119] (© 2008, IEEE), and are reprinted here with permission of IEEE. The Travel Assistance Device (TAD) technology is protected under U.S. Patents # 8,138,907 “Device to Assist Transit Riders with Special Needs” [123] and # 8,169,342 “Method of Providing a Destination Alert to a Transit System Rider” [126] by the University of South Florida. Descriptions and experiments related to TAD have been published in the Institution of Engineering and Technology (IET)’s Journal of Intelligent Transport Systems (© 2010

IET) [130] and Transportation Research Board (TRB)'s Transportation Research Record Journal [120] (© 2010) and is reprinted with permission of IET and TRB. Technology supporting TRAC-IT is protected under pending U.S. Patent Application # 11/855,694 "System and Method for Real-Time Path Prediction and Automatic Incident Alerts" and U.S. Patent Application # 11/277,403 "System and Method for Transportation Demand Management" by the University of South Florida. Portions of the material related to TRAC-IT have been presented at the Transportation Research Board (TRB) annual meeting and have been peer-reviewed by TRB [131], and have also appeared in a USF research project final report [148].

## 4.2 Evaluation Overview

Our evaluation of LAISYC is divided into two categories:

- 1) Evaluation of individual LAISYC framework components
- 2) Demonstration of innovative location-aware mobile apps developed using LAISYC

The first subsection of this chapter presents experiments performed with mobile devices in order to isolate and evaluate each component. The second subsection discusses two innovative location-aware mobile applications, TRAC-IT and the Travel Assistance Device (TAD), which have been developed and evaluated using the LAISYC platform.

## 4.3 LAISYC Component Evaluation

We set out to evaluate the various LAISYC components through a series of real-world tests on actual GPS-enabled mobile phones. This is particularly challenging, since at the time of these tests the Java ME Location API was a restricted API that could only be

accessed with the permission of the wireless carrier. Permission of this type is typically only given to commercial partners of the wireless carrier, since the use of assisted GPS and other network-assisted positioning technologies have a significant impact on cellular carrier network resources. However, we were able to obtain permission from Sprint to test our mobile applications that use the Location API on the Sprint and Nextel networks.

We developed several test mobile applications designed to isolate and test various aspects of the mobile software's impact on the mobile device. Each software test is discussed in the following respective sections for the LAISYC components.

#### 4.3.1 GPS Auto-Sleep

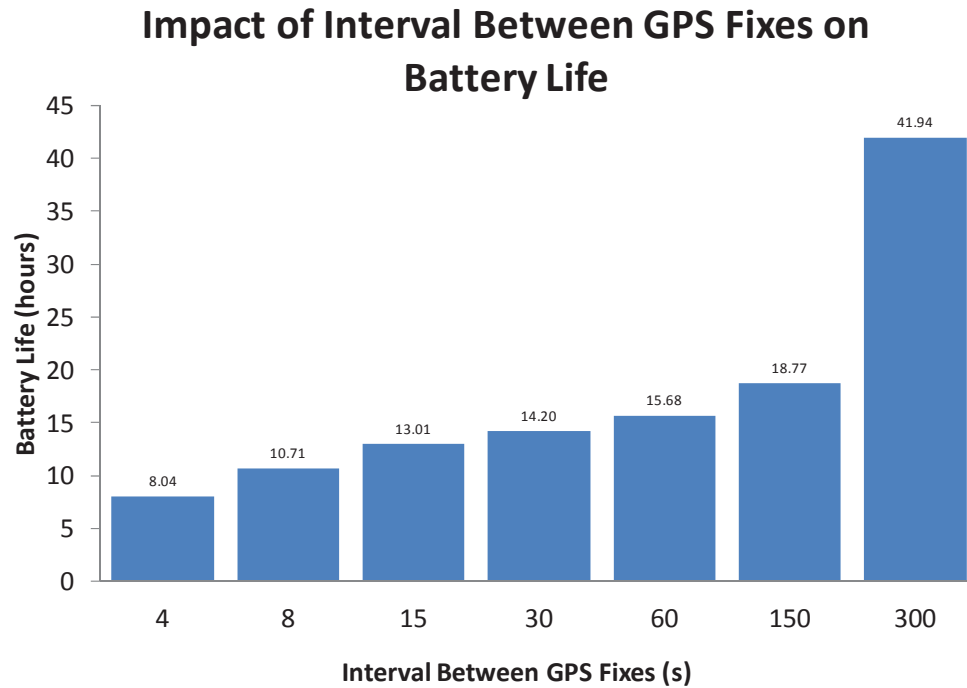
As mentioned in the GPS Auto-Sleep section in Chapter 3, during our research using high-sensitivity GPS-enabled mobile phones, we observed that the successful acquisition of individual GPS samples were significantly less dependent on previous GPS observations than previous mobile phone models with low-sensitivity GPS receivers. This observation led us to hypothesize that dynamic GPS sampling could capture high-resolution travel paths by using a high frequency sampling rate when the user is moving and saving a significant amount of battery energy by using a low frequency sampling rate when the user is stopped. GPS Auto-Sleep, implemented using a finite state machine, is the invention that controls the dynamic GPS sampling rate.

We first set out to demonstrate the feasibility of GPS Auto-Sleep through a series of controlled experiments using a Sanyo Pro 200 CDMA cell phone on Sprint's Evolution-Data Optimized (EV-DO) Revision (Rev.) A network with assisted GPS.

We implemented a test mobile application that simply registered given interval, timeout, and maxAge values with the JSR179 Location API LocationListener and then recorded timestamps to the persistent MIDP Recordstore every several GPS fixes. The device was charged until the battery life indicator on the outside of the device indicated a full charge, and then the test software was executed on the device until the battery was depleted to the point that the device powered itself off. After plugging in the device and powering it back on, we restarted the testing application and pressed a button to retrieve the timestamps from the most recently completed test. Through this method, we were able to record the length of time the phone was operational while attempting to acquire GPS at various sampling frequencies.

Figure 16 shows the result of these tests using the Sanyo Pro 200 and a series of sampling intervals varying from four seconds to 300 seconds (i.e., five minutes). The device was located on a table in the lower story of a two story building for these tests, and the phone was flipped closed during these tests, so the display was off.

From these experiments we can see that increases in the GPS sampling interval lead to a battery life savings in the order of hours. Even the increase between sampling GPS every four seconds to every eight seconds produces an increase of 2.67 hours in battery life, and the increase from eight seconds to fifteen seconds increases battery life another 2.3 hours. These results indicate that high-sensitivity GPS is able to turn on the GPS hardware to full power to acquire a GPS fix, and immediately reduce energy consumption by dropping to a lesser power level.



**Figure 16 - Even modest increases in the interval between GPS fixes produce extended battery life on the order of hours. [118] © 2011 IEEE**

An exponential increase in battery life of 23.17 hours can be seen between the GPS intervals of 150 seconds and 300 seconds. This large increase in energy savings indicates that various components in the phone (e.g., Central Processing Unit, memory, cellular modem) are able to reach a low power state due to the lack of GPS activity, unlike smaller interval values where these components remain active.

The trend of energy savings in relation to increasing GPS sampling intervals validates the general design of the GPS Auto-Sleep state machine. If we can achieve accurate state transitions, we could sample frequently when the user is moving, and less frequently when the user has stopped moving. Since U.S. travelers report traveling an average of approximately 91 minutes per day [149], occasional GPS sampling in the stopped state would cover the remaining 1,349 minutes of the day, which should adequately extend

battery life so that the user can carry the phone throughout the day without needing to charge the battery.

Since the GPS Auto-Sleep needs to run in real-time on the mobile phone, we next evaluate the complexity of the algorithm in terms of running time and memory. To keep up with real-time data, the algorithm must maintain a linear growth rate in relation to the number of GPS points processed, and must maintain a constant memory requirement throughout execution, or else the mobile device will eventually run out of memory, as the algorithm executes online for weeks or months at a time.

When a new GPS point is generated by the mobile phone, GPS Auto-Sleep makes several comparisons against constant thresholds that do not change. GPS Auto-Sleep only keeps one previous GPS data point in memory for the *moved\_distance\_threshold*. Therefore, the memory requirements of GPS Auto-Sleep,  $f(n)$ , is:

$$f(n) = O(1)$$

where  $n$  is the number of GPS data points processed.

For running time analysis, we can prove that GPS Auto-Sleep maintains a linear growth rate in terms of execution time with real-time data input by reviewing the processing steps in the algorithm. For each GPS data point, we measure the distance to the last saved location data point when the user is stopped to determine if the moved distance exceeds the *moved\_distance\_threshold*. We use the Vincenty inverse algorithm to compute the distance, which has been shown to execute in a constant amount of time [136], and therefore is  $O(1)$ . The amount of time to execute the comparisons of speed,



time, and uncertainty data against the respective thresholds is also  $O(1)$ . Since GPS Auto-Sleep is an online algorithm that executes these steps for each new location data point generated by the phone, the time complexity of GPS Auto-Sleep,  $f(n)$ , is:

$$f(n) = O(n)$$

where  $n$  is the number of GPS data points processed. Therefore, GPS Auto-Sleep scales linearly in execution time and maintains a constant memory requirement, as large numbers of location data points are processed. Thus, it can remain online for an indefinite amount of time.

The next steps for the design of GPS Auto-Sleep are the choice of thresholds used to control state transitions, the state values for interval, timeout, and maxAge, and the number of states used in the state machine.

For data collection in our experiments using GPS Auto-Sleep, we have configured the state machine attributes with the values shown in Table 3.

**Table 3 - GPS Auto-Sleep state machine values chosen for experimentation**

State	Interval (s)	Timeout (s)	MaxAge (s)
0	4	2	2
1	8	4	4
2	16	8	8
3	64	16	16
4	150	32	32
5	256	32	32

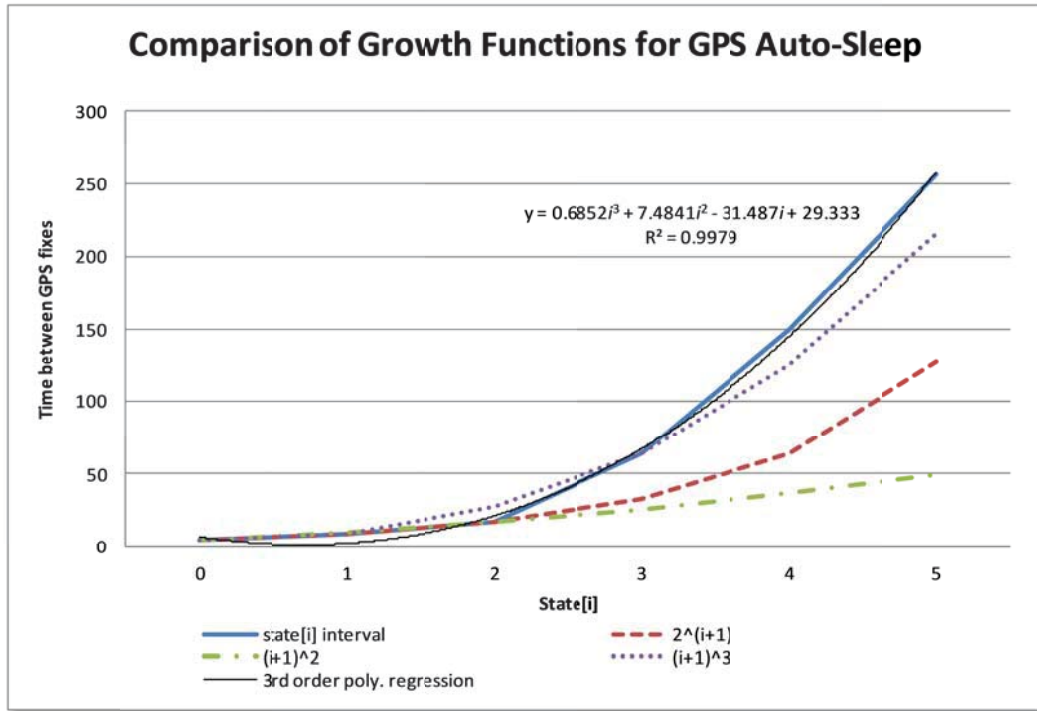
In our analysis of these attributes we simplify our calculations to focus on only the interval and timeout values. Using only the interval and time values, we can establish both an upper bound and lower bound on the amount of time required for our application to achieve a GPS fix without needing the maxAge parameter. Eliminating the maxAge parameter from consideration is also preferred, since we do not have any control over the GPS behavior of other applications that would affect this parameter.

Since the use of GPS by another application would only decrease the time needed for our application to acquire a GPS fix (i.e., the maxAge parameter never increases the time required to acquire a GPS fix), the lower bound of the time required to achieve a GPS fix is an ideal scenario when another application acquires a GPS fix just before our application's scheduled interval. In this scenario, the time elapsed between when a fix is scheduled and the fix is acquired by our application is zero, and therefore the lower bound on the total amount of time required to acquire a fix is equivalent to the interval value. The upper bound of the time required to achieve a GPS fix is equivalent to the sum of the interval and timeout values.

We chose interval values for states that exhibit exponential growth as we move towards sleep state, since as we build confidence that the device is not moving while transitioning through states, we want to rapidly enter the state that will save the most energy. The same exponential decay is desired when moving from the sleep state to the awake state, as we build confidence in the user's movement. Figure 17 shows the relationship of the chosen interval values (solid blue line) to several growth functions. As shown in Figure

17, the chosen interval values approximate a third order polynomial, which is determined using polynomial regression ( $R^2 = 0.9979$ ):

$$state[i]_{interval} = 0.6852(i + 1)^3 + 7.4841(i + 1)^2 - 31.487(i + 1) + 29.33$$



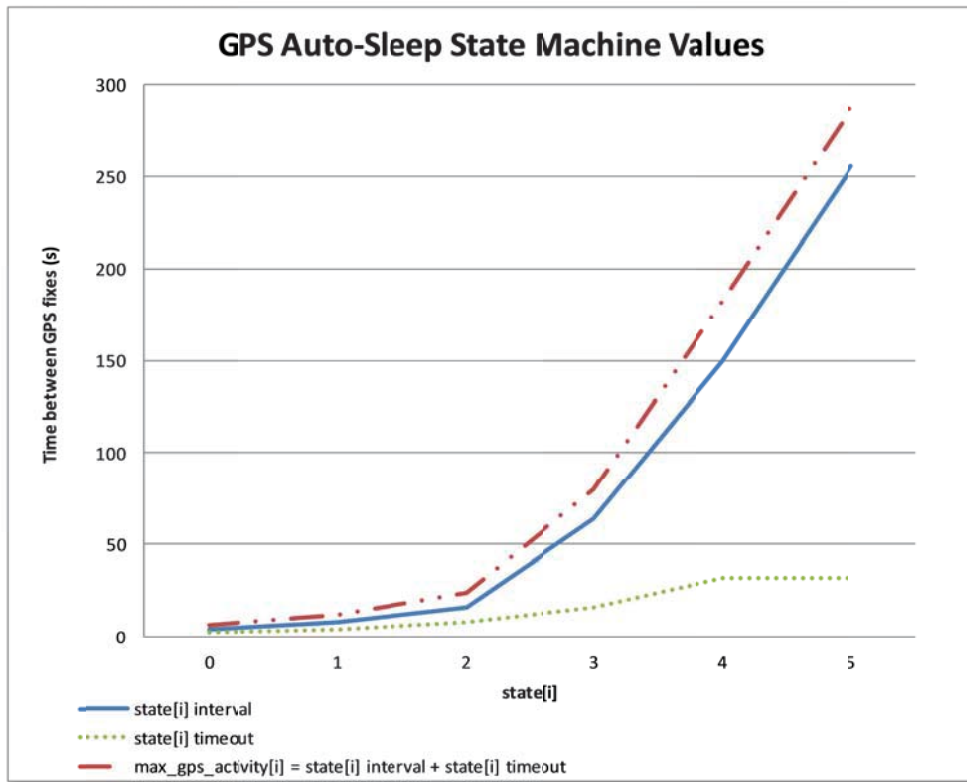
**Figure 17 - A growth function for the  $state[i]_{interval}$  values was chosen to grow like an  $x^2$  or  $2^x$  function until it reaches the middle state, at which point it quickly accelerates in growth beyond an  $x^3$  function**

The interval values' relationships with the timeout values from Table 3 are shown in Figure 18. The timeout values (beige dotted line) are defined as half of the interval value at each state, until an upper limit is reached at 32 seconds. In our research with the Sanyo Pro 200, allowing the GPS to continue to search for a signal after 32 seconds had elapsed did not yield a fix, and therefore to prevent wasting battery energy we do not want the GPS to be active longer than this threshold. In the case that a GPS fix cannot be acquired

at a particular state, the GPS may be active for a maximum period of time defined by the interval value and timeout value at a particular state:

$$max\_gps\_activity_{state[i]} = state[i]_{interval} + state[i]_{timeout}$$

These values are shown in Figure 18 as the red line composed of dots and dashes.

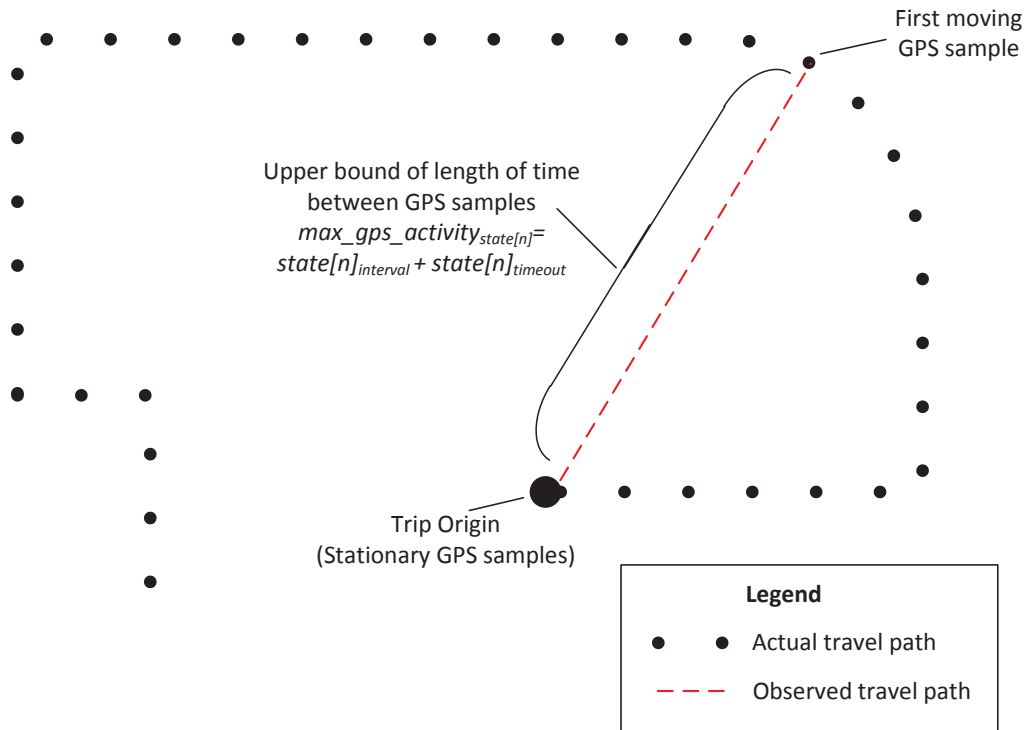


**Figure 18 - Sample GPS Auto-Sleep values are chosen for an exponential growth in the interval between GPS fixes, while the timeout values have an upper-bound of 32 seconds; if a GPS fix cannot be acquired, the interval + timeout line illustrates an upper bound for the total time elapsed at each state.**

One concern with the design of the state machine is the time it takes to transition from state[n] to state[0] due to the high-resolution travel path data that could be potentially lost for the time it takes to transition from asleep to awake. Through discussion with travel behavior data collection experts and a review of literature [149-154], we established five

minutes as the acceptable amount of the first portion of the travel behavior path to miss when transitioning from asleep to awake in the case of worst-case performance by GPS Auto-Sleep.

This loss of the first portion of the path is primarily of concern when distance of travel is being measured. Figure 19 shows the worst-case scenario when the state machine is in the sleep state of  $state[n]$  and the user begins moving immediately following a GPS fix acquisition. The longest amount of time that may elapse between successful GPS samples is  $max\_gps\_activity_{state[n]}$ . As a result, the user's travel behavior is not being monitored during this time, resulting in the observed travel path of the straight dashed red line in Figure 19, instead of the actual travel path shown in the black dots.



**Figure 19 - The largest potential loss of beginning travel path is worst-case scenario when the user travel path is sampled just before they begin moving, since the next GPS sample occurs  $max\_gps\_activity_{state[n]}$  seconds later**

The following formulas define the time cost for transitioning between states given the state transition rules defined in Chapter 3, with the goal of keeping the amount of time for lost travel path data under five minutes.

If a GPS fix cannot be acquired on startup, the maximum time elapsed from startup to fully asleep (i.e., state[n]) is:

$$\begin{aligned} &max\_elapsed\_time_{startup\_to\_asleep} \\ &= first\_fix\_timeout + \sum_{i=1}^{n-1} (state[i]_{interval} + state[i]_{timeout}) \end{aligned}$$

With the state values defined here, the  $max\_elapsed\_time_{startup\_to\_asleep}$  is 296 seconds, or almost five minutes, which is an acceptable amount of time for our tracking application.

The amount of time elapsed from the awake state to the asleep state during normal execution (i.e., not on startup) is a similar equation and value, with the only difference being the use of the back off timer instead of the  $first\_fix\_timeout$  value:

$$\begin{aligned} &max\_elapsed\_time_{awake\_to\_asleep} \\ &= backoff\_time\_threshold + \sum_{i=1}^{n-1} (state[i]_{interval} + state[i]_{timeout}) \end{aligned}$$

Since our goal is to capture high resolution travel behavior, a significant risk when fully asleep in state[n] is that the state machine will sample the GPS location when the device is stationary, and then the device immediately begins moving and the state machine waits state[n] amount of time (e.g., 256 seconds) before again sampling GPS. When considering the maximum possible elapsed time, we also must assume that nearly the entire timeout period has elapsed before acquiring a GPS fix. If we wait to fully

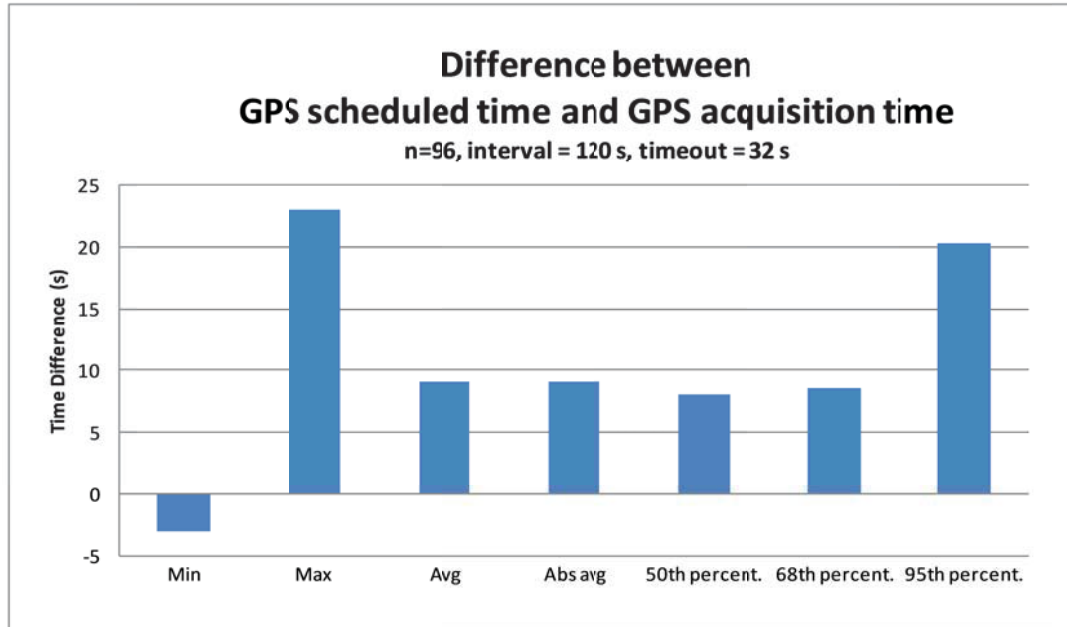
transition between state[n] to state[0] using single state transitions to resume high resolution tracking, and assuming the entire timeout period is used, the maximum elapsed time is:

$$\begin{aligned} &max\_elapsed\_time_{asleep\_to\_awake\_single\_transitions} \\ &= \sum_{i=1}^n (state[i]_{interval} + state[i]_{timeout}) \end{aligned}$$

With the state values defined here, the  $max\_elapsed\_time_{asleep\_to\_awake\_single\_transitions}$  is 564 seconds, or nearly ten minutes. This is far too long to wait for our needs of capturing high-resolution travel behavior.

We could consider using the  $avg\_elapsed\_time_{asleep\_to\_awake\_single\_transitions}$  instead of the  $max\_elapsed\_time_{asleep\_to\_awake\_single\_transitions}$  when calculating the acceptable amount of data loss if the theoretical  $max\_elapsed\_time_{asleep\_to\_awake\_single\_transitions}$  is found to be much larger than the  $avg\_elapsed\_time_{asleep\_to\_awake\_single\_transitions}$  values observed in our tests with actual devices. In our research, we found that once a high-sensitivity GPS-enabled cell phone is able to acquire a GPS fix in an environment, it typically returns the next GPS fix quickly, which significantly reduces the amount of time spent during the timeout stage of each GPS fix attempt. Figure 20 shows the difference between scheduled GPS times (i.e., when the Location API is scheduled to return a GPS fix based on the interval value) and the times when the Location API actually acquired and returned a GPS fix for a dataset, observed using a Sanyo Pro 200 on the bottom story of a two story building. The average time difference value here is approximately 9 seconds, which is less than a third of our maximum timeout value of 32 seconds. The negative value shown in the far-

left minimum column indicates that the Location API returned a GPS fix slightly before the GPS update to the application was scheduled.

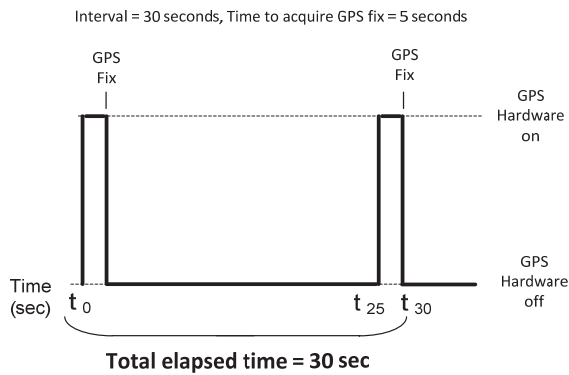


**Figure 20 – When high-sensitivity GPS is able to acquire a fix, it tends to deliver this information close to the expected interval value with an average delay of only 9 seconds.**

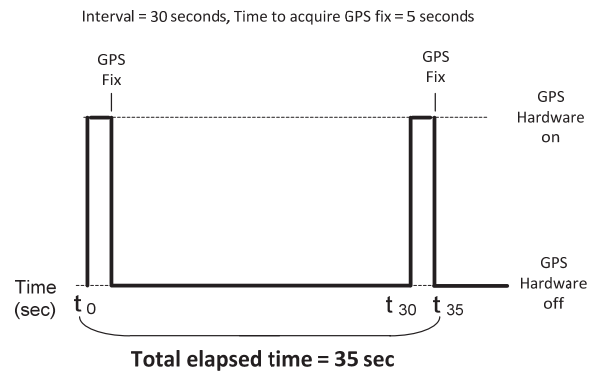
This observation indicates that the Location API is actually starting up the GPS hardware slightly prior to the time when the next GPS update is scheduled. We refer to this as proactive GPS scheduling, as opposed to reactive GPS scheduling, which waits until the scheduled interval expires before starting up the GPS hardware. The difference between proactive and reactive GPS scheduling is illustrated in Figure 21. Proactive GPS scheduling reduces the time the application has to wait for a fix to be acquired, since some of the wait time is moved prior to the scheduled update time. It should be noted that proactive GPS scheduling uses an estimate for the amount of time that is needed to acquire a GPS fix when scheduling the GPS hardware, since this value is not known a priori.



## Proactive GPS Scheduling



## Reactive GPS Scheduling



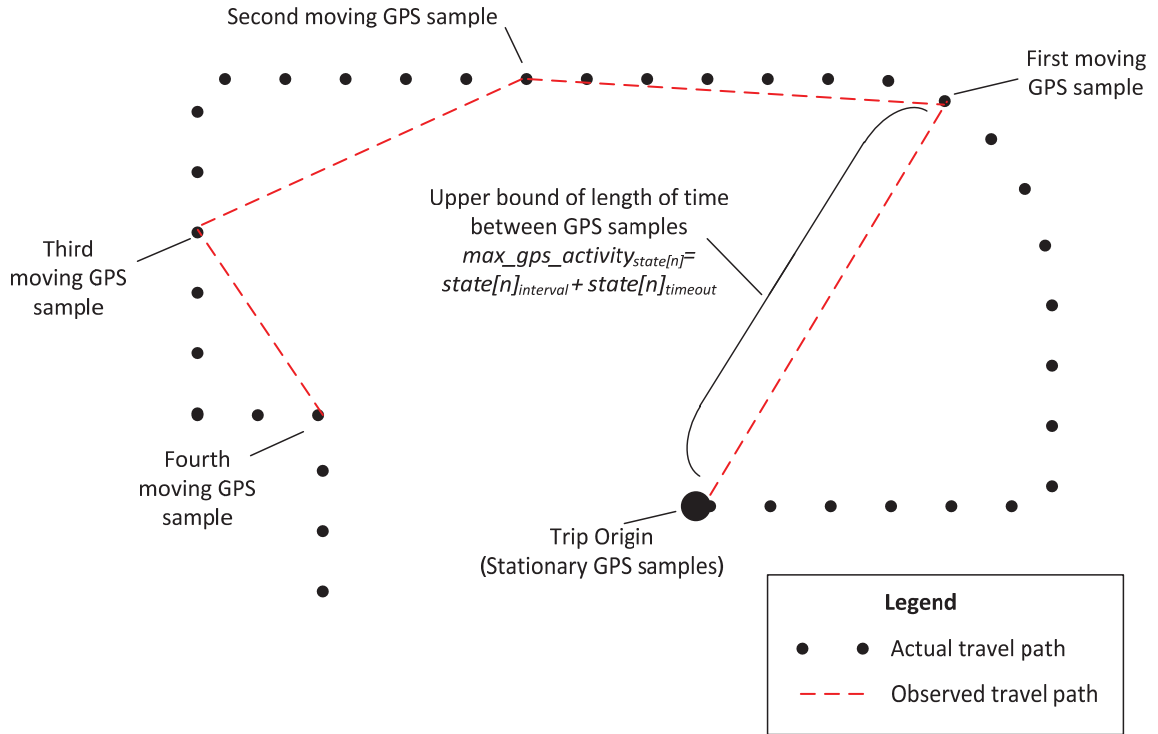
**Figure 21 - Proactive GPS scheduling (left) starts the GPS hardware slightly before the scheduled interval value expires, while reactive GPS scheduling (right) waits until the interval period has completely expired before attempting a GPS fix.**

However, even if the timeout values are completely eliminated via high-sensitivity GPS hardware, proactive GPS scheduling, or use of GPS by other applications, this would still yield a minimum amount of time required to transition from fully asleep (i.e., state[n]) to fully awake (i.e., state[0]) as:

$$\min\_elapsed\_time_{asleep\_to\_awake\_single\_transitions} = \sum_{i=1}^n (state[i]_{interval})$$

With the state values defined here, the  $\min\_avg\_elapsed\_time_{asleep\_to\_awake\_single\_transitions}$  is 472 seconds, or a little under 8 minutes. This value is too long to risk lost travel behavior. Figure 22 illustrates the potential loss of travel path information, if the state machine needs to transition through all states before beginning high-resolution tracking.

Therefore, we needed a new method to reduce the amount of time needed to transition from state[n] to state[0].



**Figure 22 - GPS Auto-Sleep can miss a substantial part of the beginning trip path if it must transition through all states before starting to record high-resolution travel behavior**

To increase our ability to capture high-resolution travel behavior data, we introduced the *high\_speed\_threshold* that allows direct state transitions from any state[i] to state[0] to “snap” back to high frequency GPS sampling. This reduces the *max\_elapsed\_time* value to:

$$max\_elapsed\_time_{asleep\_to\_awake} = state[n]_{interval} + state[n]_{timeout}$$

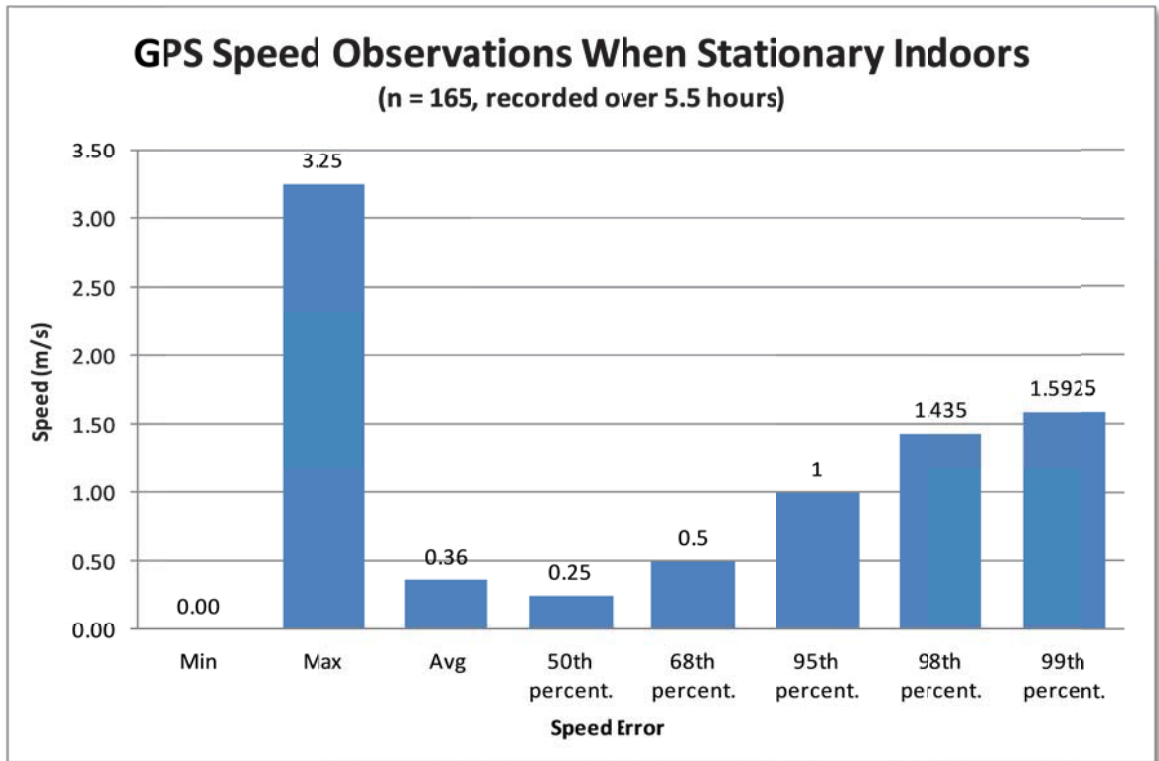
With the state values defined here, the *max\_elapsed\_time\_{asleep\\_to\\_awake}* is 288 seconds, which is under our five minute threshold and therefore an acceptable delay.

The *high\_speed\_threshold* must be chosen carefully to ensure proper operation of GPS Auto-Sleep. If *high\_speed\_threshold* is too small, the state machine will be constantly

waking up and wasting battery energy while it samples GPS at high frequency until the back off timer expires and the state machine gradually transitions to the sleep state again, which is upper-bounded by *max\_elapsed\_time<sub>awake\_to\_sleep</sub>*. If the *high\_speed\_threshold* is too high and we do not recognize true movement quickly, then we lose the ability to transition to state[0] within the five minute requirement and as a result, we risk losing a significant amount of travel behavior.

Fortunately, GPS speed measurements tend to be accurate, as speed is measured by the receiver using the Doppler shift of the GPS signal [155]. Additionally, research has shown that accurate GPS speed determination is preserved even when positional accuracy of GPS degrades due to reduced GPS signal quality [155]. Therefore, speed can be used as a threshold that is largely independent of position error, and therefore the *stopped\_speed\_threshold* and *high\_speed\_threshold* can be used to temper and correct movement in the state machine when the *moved\_distance\_threshold* may be affected by positional outliers.

To evaluate the *high\_speed\_threshold* and *stopped\_speed\_threshold* values for the execution of GPS Auto-Sleep on the Sanyo Pro 200, we recorded the GPS speed observation of the Sanyo Pro 200 while it was stationary indoors (i.e., true speed = 0 meters per second) over a 5.5 hour period, with an interval between GPS samples of two minutes. The device was located on a table in the lower story of a two story building. The observed speed error measurements are shown in Figure 23.

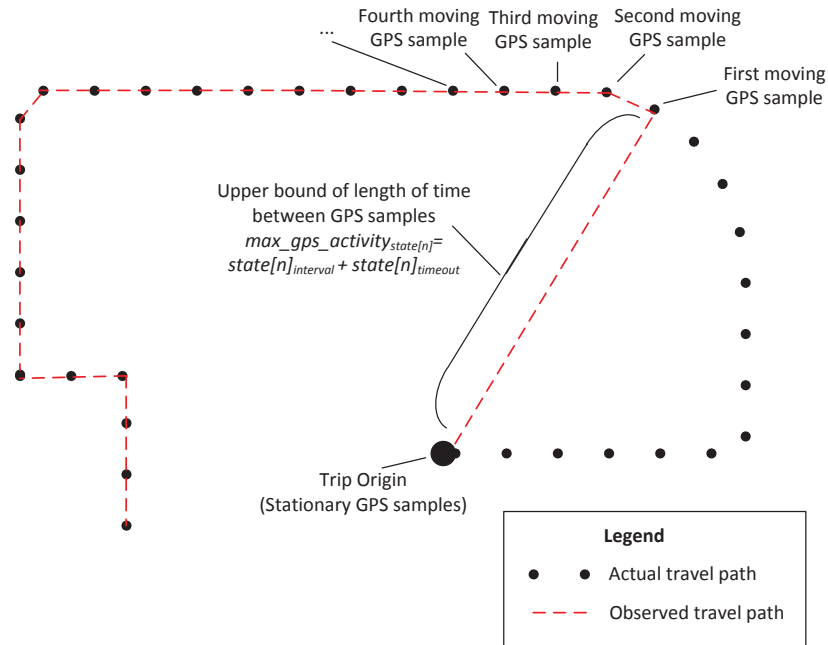


**Figure 23 - Speed thresholds for the GPS Auto-Sleep state machine are selected using observations of speed when stationary and indoors**

The 95<sup>th</sup> percentile of error is 1 meter per second, indicating that GPS speed tends to be close to actual speed when stationary indoors, even in a difficult GPS environment. In additional tests, the Sanyo Pro 200 tended to register a walking speed at a value slightly over 1 meter per second, approximately 1.1 meter per second. Since we want to respond to walking trips and slowly increase the sampling interval by stepping through the states from state[n] to state[0], we chose 1 meter per second as the *stopped\_speed\_threshold*. Since any direct transitions from any state[i] to state[0] need to be accurate to avoid unnecessary wake ups, we chose 1.5 meters per second as the *high\_speed\_threshold*, as this should have less than a 2% error rate based on our data and GPS Auto-Sleep will still be able to quickly respond to true movement by directly snapping to state[0]. We want this value to be as low as possible without triggering too many false wake-up periods

because we are interested in capturing non-vehicle travel behavior (e.g., walking, biking), which can be at fairly low speeds.

Figure 24 shows the behavior of GPS Auto-Sleep using the *high\_speed\_threshold*.



**Figure 24 - GPS Auto-Sleep can quickly react to real movement using the *high\_speed\_threshold* and rapidly begin sampling GPS via direct transitions to state[0] to reflect a more accurate travel path**

The state machine can now immediately snap to rapid GPS sampling to capture a better representation of the user’s travel path. We still cannot avoid the potential loss of data during the period of time between the most recent stationary GPS sample and the first moving GPS sample, since we must maintain this GPS sampling interval when stopped to save battery energy. However, it should be noted that this amount of time is an upper bound on elapsed time, and therefore the average amount of time elapsed between user movement and the first GPS sample is substantially less. Also worthy of note is that due to the initial sleep period before we detect movement, the calculated distance of observed

travel using GPS samples will typically be a lower bound on actual distance traveled (unless there is a significant amount of unfiltered GPS drift during the trip, which may occur at brief stops).

The final two thresholds that must be chosen for GPS Auto-Sleep are distance-based. The *moved\_distance\_threshold* is used to determine if the traveler has moved from the last-sampled GPS location when considered stationary, and is used to gradually step towards state[0] (i.e., awake) state-by-state. The *high\_horizontal\_accuracy\_threshold* is based on the estimated accuracy of the GPS fix, and is used to gradually step towards state[n] (i.e., asleep) state-by-state when the device reports that there is a large estimated error in the accuracy of the fix.

To determine a *moved\_distance\_threshold*, we performed an indoor accuracy test on two different mobile phones: the Motorola i580 mobile phone on the Sprint-Nextel iDEN network and Sanyo 7050 mobile phone on the Sprint-Nextel CDMA 1 x RTT data network.

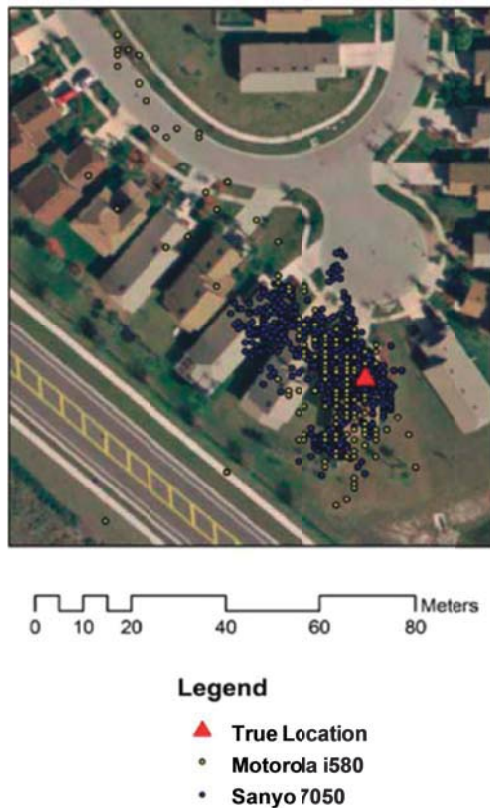
We chose to evaluate two different mobile phone models because in the early implementation of the LAISYC framework, we had anecdotal evidence that positional accuracy tended to differ between devices. These tests were performed inside a building made mostly of wood and concrete stucco, since when GPS Auto-Sleep is used to try to detect movement the device will typically be indoors. The reference ground truth location was determined by marking the location of the phones on a blueprint of the structure, and scanning the blueprint so it could be geo-referenced against a digital 6-inch resolution color aerial photo that was already geo-referenced in the Universal Transverse

Mercator (UTM) Zone 17N NAD 1983 coordinate system. Horizontal accuracy was determined by projecting the WGS 84 latitude and longitude coordinates from the GPS fixes into the UTM coordinate system and then measuring the Euclidean distance between the ground truth location and the observed GPS location. The horizontal error statistics from these tests are shown in Table 4.

**Table 4 - Horizontal error statistics for indoor GPS accuracy tests. Reprinted with the permission of Cambridge University Press. [18]**

Device	GPS Type	Sample Size	Horizontal Error Statistics (meters)						
			Min	Max	Avg	50th	68th	95th	RMSE
Motorola i580	Assisted	478	0.74	90.69	15.16	9.78	15.15	47.9	21.64
Sanyo 7050	Assisted	1513	0.16	32.04	8.78	6.23	9.33	24.44	11.33

A scatter plot of these tests is shown in Figure 25.



**Figure 25 - Scatter plots of indoor horizontal positional accuracy tests. Reprinted with the permission of Cambridge University Press. [18]**

The substantial difference between devices can be seen in both the statistics, with the Motorola i580 having almost twice the 95<sup>th</sup> percentile of error (47.9 meters) as the Sanyo 7050 (24.44 meters), as well as the scatter plot, where the GPS from the Motorola i580 drifts away from the ground truth location in the upper-left and lower-left corners of the image. These empirical measurements confirm our anecdotal evidence that there can be significantly different levels of positional accuracy between two different devices. There is much less difference in speed error measurements between the two devices (Table 5), confirming that GPS signal obstructions affect positional error substantially less than speed error.

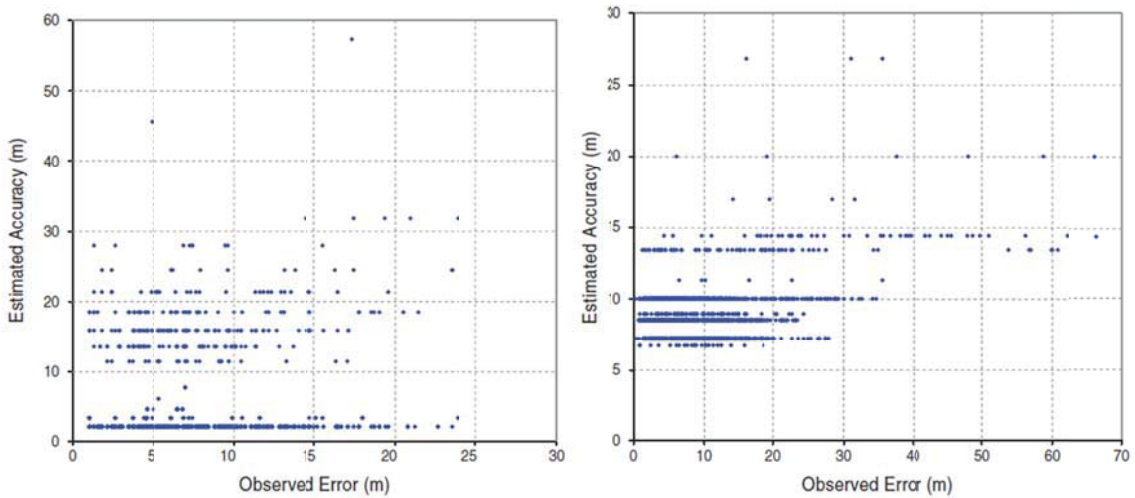
**Table 5 - While the positional error between the two devices is substantially different, the error in speed is much less dramatic**

Device	GPS Type	Sample Size	Speed Error Statistics (meters per second)					
			Min	Max	Avg	50th percent.	68th percent.	95th percent.
Motorola i580	Assisted	478	0.00	6.94	0.37	0.00	0.00	1.94
Sanyo 7050	Assisted	1513	0.00	1.25	0.13	0.25	0.25	0.25

Based on the observed positional error from these devices, we chose a value of 100 meters for the *moved\_distance\_threshold*. 100 meters is greater than any error we observed in our tests and will therefore be tolerant of moderate GPS drift from the ground truth location without producing a false-positive movement reading that would cause the state machine to move towards state[0] (i.e., awake) when the device is still stationary inside a building. 100 meters is also a short enough distance that, when combined with the *high\_speed\_threshold* to immediately detect fast movement, prevents the device from missing a substantial portion of the user's slow travel path before movement is detected.



We examine the estimated horizontal accuracy uncertainty values observed in this same test to determine the value for *high\_horizontal\_accuracy\_threshold*. Figure 26 shows the observed error and the estimated error for each GPS data point, demonstrating the connection between the two values.



**Figure 26 - Reliability of accuracy estimates for individual assisted GPS data points was shown to be poor on the evaluated devices, the Motorola i580 (left) and Sanyo 7050 (right). Reprinted with the permission of Cambridge University Press. [18]**

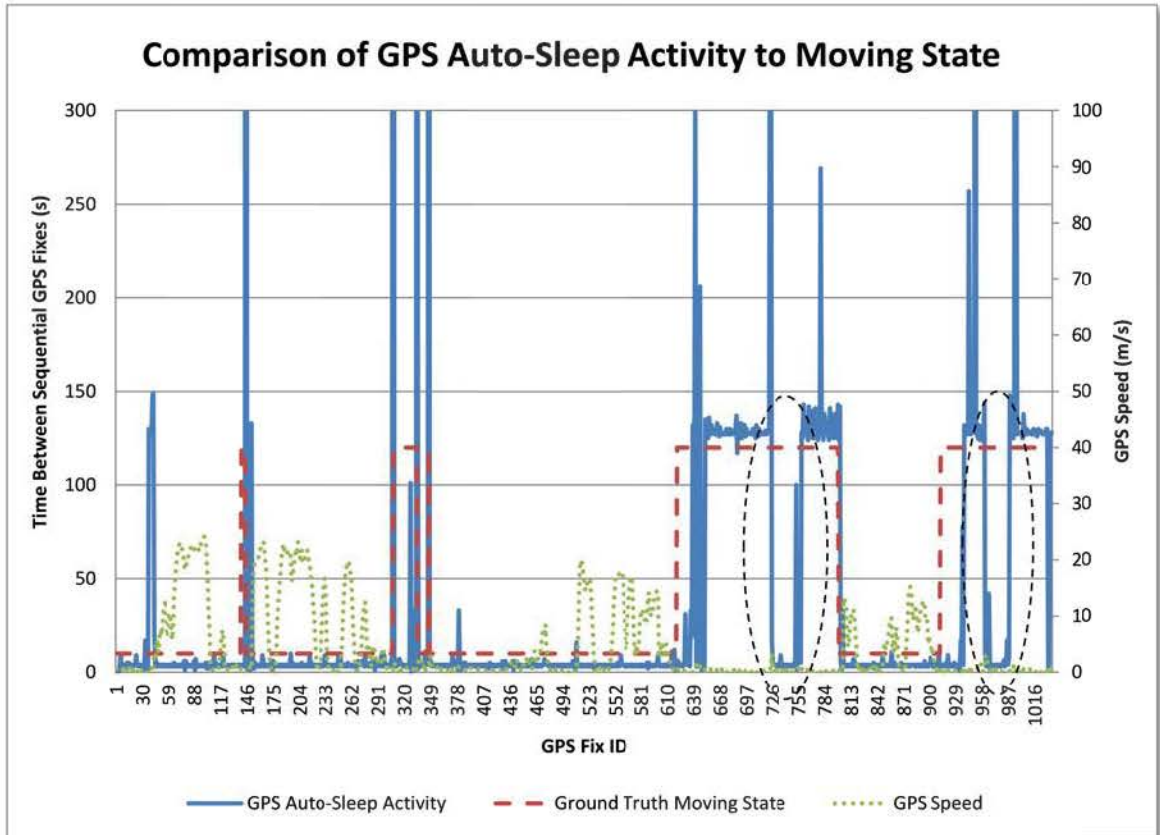
Unfortunately, when the observed error is compared to the estimated accuracy value produced by the Location API there is little correlation between actual and observed error. According to the JSR179 Location API, the “true position should be within a circle defined by the given estimated error uncertainty radius at the 68% confidence level” [23]. This means that in Figure 26, the data points in the scatter plot should fall above the 1:1 diagonal at least 68% of the time. For the Motorola i580 (left), observations only fall above the 1:1 diagonal 18.6% of the time, and for the Sanyo 7050 (right) observations only fall above the 1:1 diagonal 55.1% of the time. Surprisingly, this means that neither phone meets the requirements defined by the JSR179 Location API despite claiming to

be in compliance with the standard. This information is important for any app developer who makes real-time decisions in a mobile app based on the 68% confidence level specification for accuracy uncertainty required by the JSR179 Location API, because as evidenced by our experiments, at least two commercially-available mobile phones are not meeting this standard.

Based on this information, we set the *high\_horizontal\_accuracy\_threshold* threshold value to 80 meters, which is substantially larger than any value observed in these tests. Our reason for choosing this value is that the threshold will effectively be ignored for these devices, but will still remain in place for other devices implementing JSR179 that may meet the standard specifications as GPS Auto-Sleep is deployed to additional devices in the future. The relationship between estimated and actual error could also be reevaluated on future devices to determine if a correlation between observed and estimated error exists.

Once all the threshold values were chosen, we evaluated the performance of GPS Auto-Sleep for accurately tracking the movement of the user and transitioning between frequent GPS sampling and occasional GPS sampling based on real-time location data and the state transition rules with the chosen threshold values. We collected 30 days of normal travel behavior from members of the research team using a Sanyo Pro 200 CDMA cell phone on Sprint's Evolution-Data Optimized (EV-DO) Revision (Rev.) A network with assisted GPS. We manually post-processed this data after it was collected and marked each data point as stationary or moving based on written travel logs from the user.

A visualization of this manual coding of data is shown in Figure 27, with the solid blue line representing the behavior of GPS Auto-Sleep and the red dashed line representing the ground truth value of the traveling state as stationary (high) or moving (low).



**Figure 27 - To evaluate the accuracy of GPS Auto-Sleep, the ground truth state of traveling was manually coded against the behavior of the state machine**

The X axis is the GPS Fix ID for each GPS fix recorded during the test, and the primary Y axis on the left is the amount of time between sequential GPS fixes, with the sleep state of state[n] having the value of 120 seconds. The speed values for each GPS fix are also shown as a yellow dotted line and on the secondary Y axis on the right, so the relationship between speed and traveling state can be seen. The spikes off the top of the graph for GPS Auto-Sleep activity indicate long periods of time when a GPS fix was not

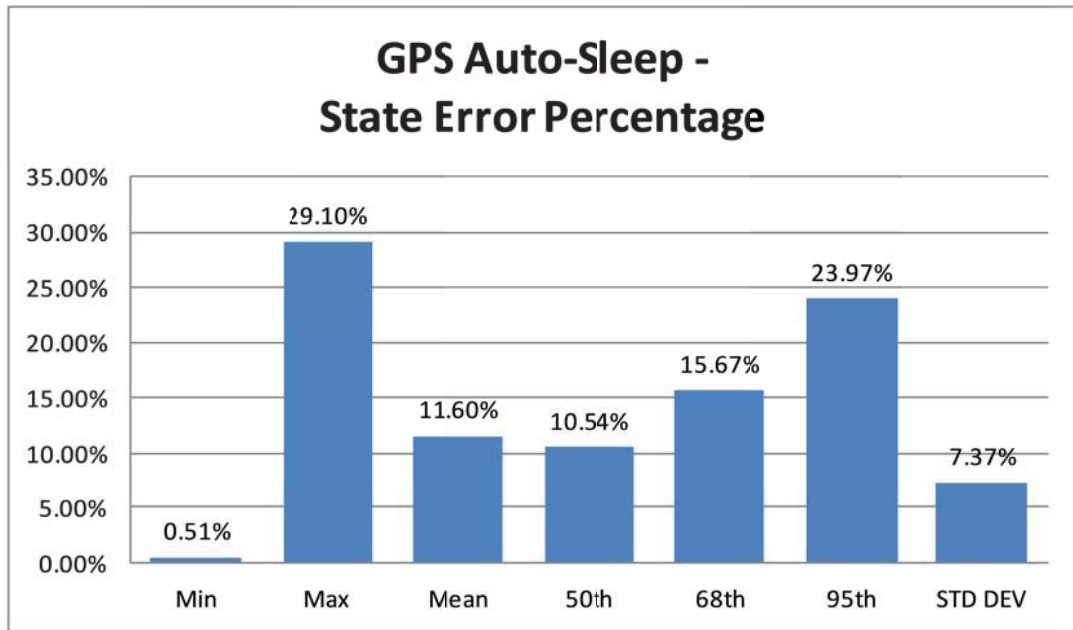
available (e.g., the user was deep inside a building), and therefore the time between GPS fixes was a large value. Ideally, the blue line for GPS Auto-Sleep activity should roughly trace the red dashed line (with the exception of the spikes off the graph for lost GPS fixes). Two areas on the graph are circled with black dashed ovals that indicate brief periods of error when the GPS Auto-Sleep transitioned from the sleep state[n] to state[0] for rapid GPS sampling every 4 seconds, when in fact the user was stationary. Close examination of the speed values at the leading edge of these periods show that the incorrect wake-ups of GPS Auto-Sleep were triggered by large outlier speed values.

To quantify the correct state percentage during tracking over the 30 sessions of collected data, we classified the GPS Auto-Sleep activity into two states, moving or stationary, based on the observed interval between GPS fixes, so that the GPS Auto-Sleep state could be directly compared to the ground truth values that were manually coded. The GPS Auto-Sleep states are defined as:

- 1) Moving – GPS Auto-Sleep is considered to be in a moving state if the interval between fixes is observed to be between 1 and 5 seconds.
- 2) Stationary – GPS Auto-Sleep is considered to be in a stationary state if the interval between fixes is observed to be greater than or equal to 8 seconds.

We consider the GPS Auto-Sleep activity to be incorrect (i.e., an erroneous state) if the GPS Auto-Sleep state does not match the ground truth manually coded state for each GPS fix.

The results of the analysis of the 30 collected sessions of GPS data are shown in Figure 28.



**Figure 28 - GPS Auto-Sleep is able to successfully track the moving or stationary state of the user with a high degree of accuracy.**

These results demonstrate that GPS Auto-Sleep is able to track the moving and stationary states of the user with a high degree of accuracy, with a mean error of 11.60% (i.e., a mean accuracy of 88.40%). The worst accuracy observed was 70.90% (an error of 29.1%), and on one session GPS Auto-Sleep was able to achieve 99.49% accuracy (an error of 0.51%). The 95<sup>th</sup> percentile of state error was 23.97%.

To confirm our hypothesis that the use of GPS Auto-Sleep during tracking saves battery energy, we examined a large scale deployment of GPS Auto-Sleep in support of the TRAC-IT mobile app used to collect high-resolution travel behavior as part of a U.S. Department of Transportation (USDOT)-funded research project. We discuss TRAC-IT in further detail later in this chapter.

As part of the USDOT study, we deployed TRAC-IT with GPS Auto-Sleep on Sanyo Pro 200's to 30 users and recorded their travel behavior for almost three months. A total of

1,857 data sessions containing a total of 4,023,917 GPS fixes were recorded during this period, for an average of 39.83 days of survey time per user. The average session length during the experiment was 15.44 hours, based on the difference between the oldest and newest GPS fix time in each session. Since the battery life observed when using a static GPS interval of 4 seconds without GPS Auto-Sleep was only 8.04 hours, a battery life of 15.44 hours when using GPS Auto-Sleep is substantially longer. As discussed in the Session management and Location Data Buffering evaluation section, wireless data transmissions consume a significant amount of battery energy. Since TRAC-IT transmits GPS data to our server via the mobile phone's cellular connection, in addition to collecting GPS data, the battery life of the phone using GPS Auto-Sleep without transmitting data to the server is substantially more than the observed 15.44 hours.

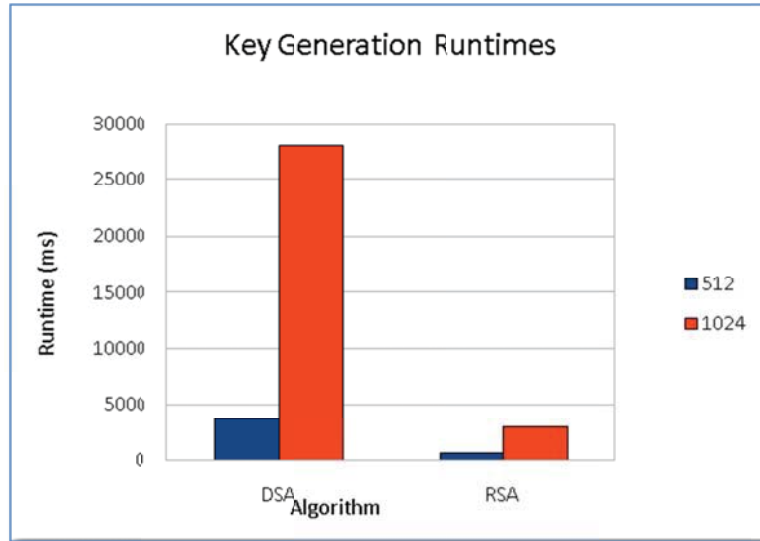
In conclusion, GPS Auto-Sleep addresses several of the needs for location-aware mobile apps outlined in Chapter 1. GPS Auto-Sleep is able to provide substantial battery energy savings (an approximate average doubling of battery life (Need #1), while maintaining acceptable movement tracking (approximately 89% accuracy) (Need #3). GPS Auto-Sleep operates in real-time on the mobile device (Need #2). We have also demonstrated a methodology for selecting the thresholds used in the algorithm (i.e., *first\_fix\_timeout*, *stopped\_speed\_threshold*, *high\_speed\_threshold*, *moved\_distance\_threshold*, *high\_horizontal\_accuracy\_threshold*, *backoff\_time\_threshold*) based on observed GPS data so that the algorithm can be implemented by any third party mobile app developer on any device with GPS and a Location API (Need #4).

### 4.3.2 Location Data Signing

Our primary motivation in analyzing Location Data Signing is to demonstrate that traditional asymmetric cryptography such as DSA and RSA are feasible for real-time execution on a mobile device, contrary to the claims of Jarusombat et al. [138]. We must examine two operations to evaluate asymmetric cryptography:

- **Key generation:** Key generation happens once daily at the start of a communication session with a server and creates both a public and private key. The private key is used to create the digital signature for individual GPS fixes, while the public key is distributed to others so that they can verify the digital signature for GPS fixes. Since key generation only happens occasionally, execution time of key generation is not of great concern.
- **Signature generation:** Signature generation happens frequently, potentially as often as once every GPS fix. Since GPS data can be generated at a rate of once per second, signature generation must be efficient in terms of execution time to be feasible for implementation on mobile devices.

To evaluate the impact of Location Data Signing, we developed a test mobile application that performed key generation and signature generation, and recorded the execution time by querying the system timestamp both before and after execution. We executed this application on an HTC G1 mobile device with Android 1.6. Figure 29 shows the results of execution time for key generation for both DSA and RSA using 512-bit and 1,024-bit keys.



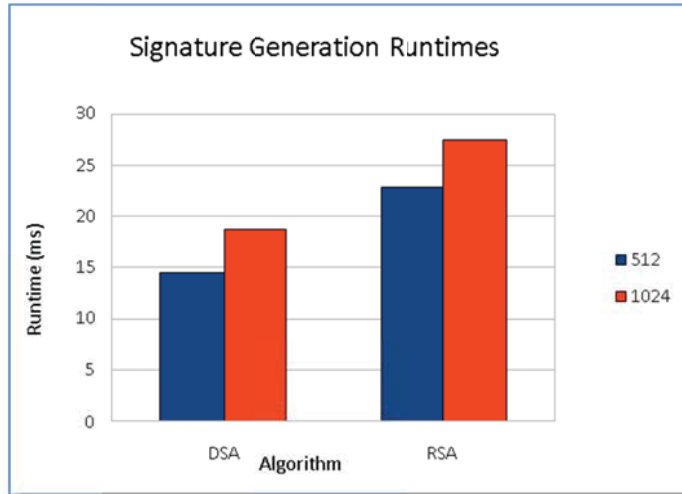
**Figure 29 - Execution time for key generation using DSA and RSA asymmetric cryptography**

Key generation for DSA is slower than for RSA, especially for the larger 1,024 key size. However, given that key generation would occur only approximately once daily, since it takes less than five seconds to generate a DSA 512-bit key, and RSA 512-bit and 1,024-bit keys, any of these key generation algorithms are feasible for execution on mobile devices. Even the DSA key generation for the 1,024-bit key that takes approximately 28 seconds could be considered feasible for real-world implementation, since key generation happens so infrequently.

Of greater concern for execution times is signature generation, since this operation will be taking place frequently, as often as every GPS fix. Since Location Data Signing must use asymmetric cryptography for every execution, and asymmetric cryptography is more computationally intense than symmetric cryptography used in Location Data Encryption [146], we must evaluate the time it takes to generate a digital signature for a GPS fix.

The results of the signature generation execution time tests are shown in Figure 30.





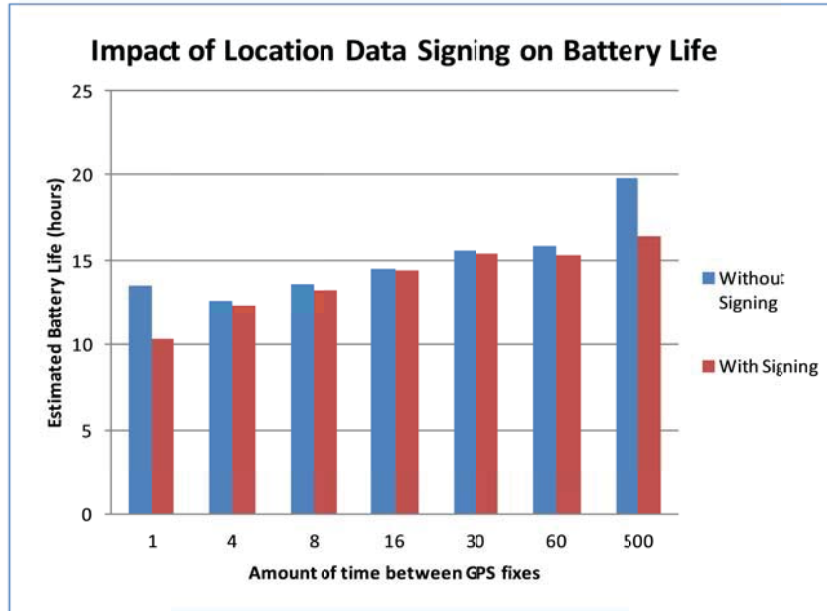
**Figure 30 - Signature generation test results show that Location Data Signing using DSA and RSA is feasible for implementation on real mobile devices**

DSA appears to take slightly less time than RSA for generating digital signatures using both 512-bit and 1024-bit keys. Both DSA and RSA can be executed in less than 30 milliseconds, which is much less than the minimum GPS sampling interval of one second. Therefore, we successfully demonstrated that Location Data Signing can be executed in real-time to generate digital signatures for location data. As mentioned earlier, we chose to implement Location Data Signing in LAISYC using DSA because it is the only algorithm that is not restricted by intellectual property or export constraints and can be used world-wide royalty-free [137].

To evaluate the energy usage of Location Data Signing signature generation, we executed the test application using DSA 512-bit with varying intervals on a Motorola Droid X while it was hooked up to the Agilent E3631 power supply. Based on the measured current and battery capacity, we use Peukert's Law to estimate the battery life for these tests [156, 157]:

$$C_p = I^k t$$

where  $C_p$  is battery capacity,  $I$  is the discharge current,  $k$  is the Peukert constant, and  $t$  is the discharge time in hours. Figure 31 shows the estimated battery life for a variety of location data signing intervals.



**Figure 31 - Estimated battery life with and without Location Data Signing**

As expected, there is a general trend towards longer battery life with less frequent GPS sampling and signature generation. Battery life is longer without location data signing for all intervals, with the largest differences (23.34% and 17.18%) happening at the one second and 500 second intervals, respectively. Average percent difference in battery life for the four to 60 second intervals is 1.95%, with an overall average percent difference of 7.18% for all intervals. The larger difference in battery life at the one second interval may be due to a heavy load on the CPU with frequent signature generation, which forces the CPU to throttle to a higher frequency to handle the additional processing load, which in turn costs additional energy. With occasional signature generation (i.e., intervals 4 through 60), the CPU can absorb the additional overhead of signature generation without

a substantial impact on CPU frequency, resulting in smaller differences in battery life. The larger difference in battery life at the 500 second interval is likely due to the fact that when the phone is generating a digital signature, the CPU stays on longer than normal, costing additional energy and preventing a quick return to a low-power state. Entering this low-power state quickly after a GPS fix is acquired is much easier when the phone is not generating a digital signature.

In conclusion, Location Data Signing addresses several of the needs for location-aware mobile apps outlined in Chapter 1. The results of the above experiments demonstrate that the Location Data Signing module is able to add authenticity to location data on mobile devices in an energy-efficient manner. The Location Data Signing module is fully implementable by third party application developers (Need #4), and can support real-time applications (Need #2) by frequently signing location data fixes as often as once per second. There is a slight impact on battery life due to Location Data Signing (Need #1) at intervals 4 through 60, and a substantial impact at frequent (one second) or infrequent (500 seconds) intervals of signature generation.

#### 4.3.3 Session Management and Adaptive Location Data Buffering

Since the Session Management and Adaptive Location Data Buffering modules are designed to work together in support of the general LAISYC communication framework, we describe the evaluation of both modules in this section.

The first evaluation of these components focuses on our choice of using HTTP directly for application data instead of SOAP, which encodes messages in XML and uses HTTP as a transport protocol. Our hypothesis was that the extra characters required to encode

messages in XML will increase the amount of time the cellular radio is active, and will result in decreased battery performance.

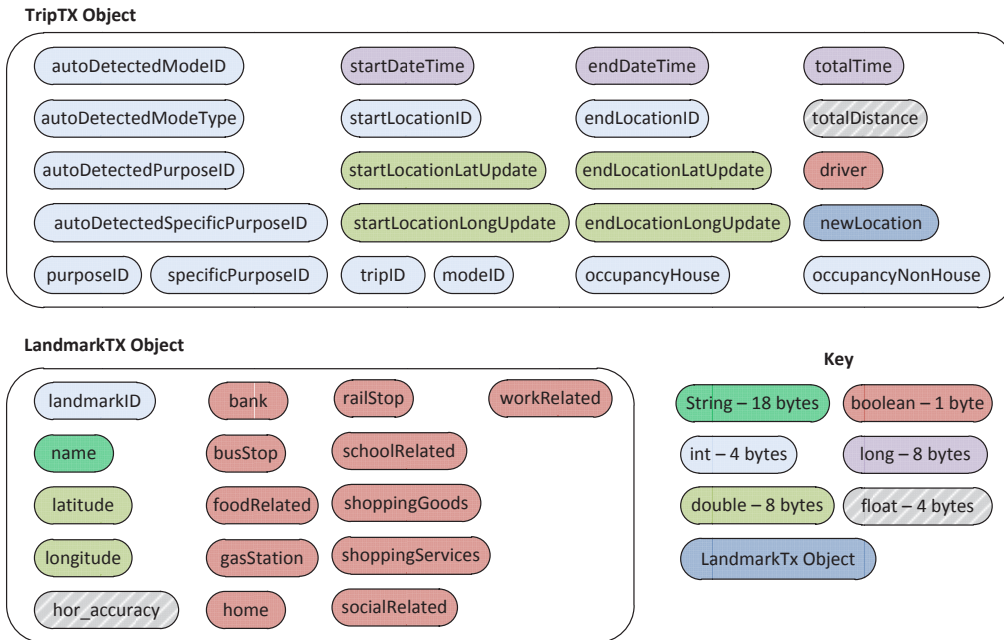
For this experiment, we implemented a custom Java ME application that was designed to query a web application at defined intervals of time and then record timestamps to the persistent MIDP Recordstore every several queries. Two methods of querying the server were implemented: one using HTTP POST methods to exchange information, and the other using the JSR172 J2ME Web Services Specification API to exchange SOAP-encoded messages via the Java API for XML-based Remote Procedure Calls (JAX-RPC) [141]. A device was charged until the battery life indicator on the outside of the device indicated a full charge, and then the test software was executed on the device using one of the methods of querying the server (e.g., HTTP POST or SOAP-based JAX-RPC) until the battery was depleted to the point that the device powered itself off. After plugging in the device and powering it back on, we restarted the testing application and pressed a button to retrieve the timestamps from the most recently completed test. Through this method, we were able to record the length of time that the phone was operational while querying the server before the device powered off for both HTTP POST and JAX-RPC.

Netbeans was utilized as the primary Java Integrated Development Environment (IDE) for implementing the mobile and web application. A Motorola i580 phone on the Sprint-Nextel iDEN network was utilized for this test since it supports both HTTP POST methods as well as JSR172 for SOAP-based web clients. Glassfish [133], the reference implemented for Java Enterprise Edition (EE) 5 and 6, was chosen as the primary Java application server to host the server-side web application. The Java API for Web

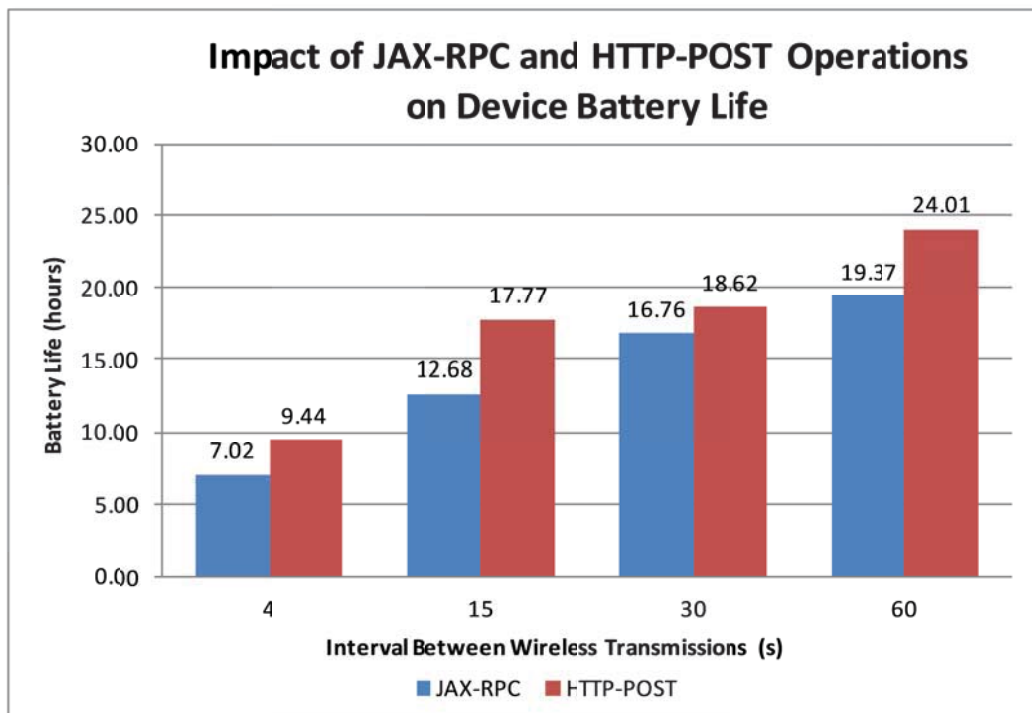
Services (JAX-WS) 2.0 [158] was used to create web applications within Netbeans that exchanged SOAP-based messages. To create server-side HTTP servlets with which the mobile phones could communicate directly via HTTP POST methods, Netbean's Mobile to Web Client tool was utilized to generate code stubs from the JAX-WS 2.0 web services for both the mobile phone and web server.

When defining the information exchange between the mobile and web application using both HTTP POST and JAX-RPC, we had to determine the exact set of information that would be exchanged between the mobile device and server. We chose to use the input and output of actual web services implemented for our TRAC-IT mobile application, which is discussed in detail later in this chapter. When generating the JAX-WS 2.0 web application, a TripTX object was defined for both the input and the output of the web application. The contents of the TripTX object in Java data types can be seen in Figure 32. Therefore, in these tests the exact same amount of information was exchanged via both HTTP POST and JAX-RPC, with the only difference being how the information was encoded and passed from the mobile device to the server.

Figure 33 shows the results of the HTTP POST vs. JAX-RPC tests on the Motorola i580. The potential energy savings when utilizing HTTP POST-based communication to transfer information instead of the heavyweight XML-encapsulated JAX-RPC can clearly be seen here. By utilizing HTTP directly instead of JAX-RPC and transmitting at 60 second intervals, battery life can be extended by approximately 4.6 hours.



**Figure 32 - The information exchanged between the mobile device and server for the HTTP POST vs. XML-based JAX-RPC battery life tests**



**Figure 33 - XML-based JAX-RPC mobile device to server communication clearly has a substantial negative impact on mobile device battery life when compared to HTTP-POST. [118, 120] © 2011 IEEE**

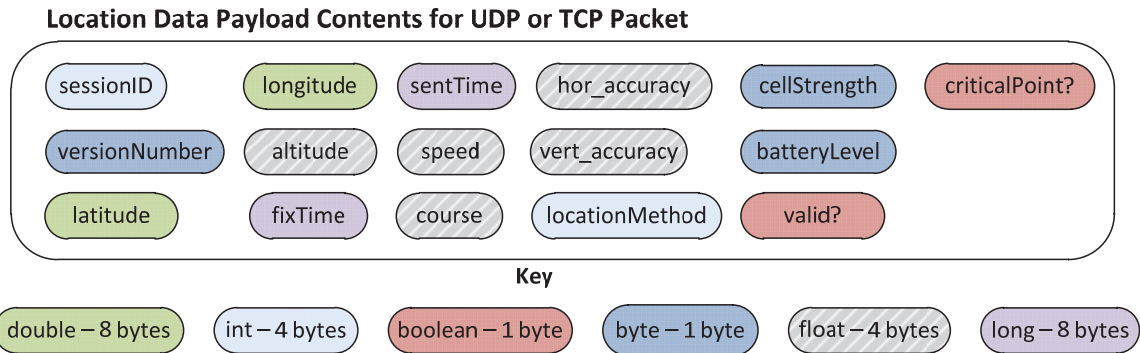
The trend continues even to frequent communication with the server every four seconds, which has an approximately 2.4 hour difference. On average, there is a 27.42% increase in battery life. These results justify our choice of plain HTTP as the main application data transport protocol, instead of an XML-based protocol on top of HTTP.

Our next evaluation focused on our choice of UDP as the transport protocol for location data. TCP is the primary alternative to UDP, but we chose UDP due to its lightweight design and general preference for systems where timeliness and scalability are of greater importance than reliability. Since the timeliness and scalability benefits of UDP over TCP are well understood, here we focus on demonstrating the battery-life benefits of UDP to better understand the tradeoffs between reliability and power consumption in relation to Adaptive Location Data Buffering.

Adaptive Location Data Buffering occasionally opens a TCP connection with the server to ensure that there are not large consecutive losses of location data when using UDP (e.g., when the device is in a gap of cellular coverage, when the user is on a voice call and the device does not support simultaneous voice and data operations). Therefore, we need to understand the power consumption differences between UDP and TCP to schedule the frequency of TCP checks with the server. Understanding these differences will help application developers choose TCP check frequencies that meet the reliability needs of their applications, but avoid negating the benefits of using UDP for location data by querying via TCP too often.

To evaluate the power consumption differences between UDP and TCP, we used an Agilent E3631 power supply to measure the current drawn by a Sanyo 7050 mobile

phone on the Sprint-Nextel CDMA 1xRTT network. We created another test mobile application that transmitted location data to a Glassfish server, with the choice of selecting either UDP or TCP as the transport protocol. The location data format used here is identical to that used by our TRAC-IT mobile application and is shown in Figure 34.



**Figure 34 - The location data format used for the payload contents of UDP and TCP packets in the power consumption tests**

We allowed both the UDP and TCP mobile applications to run on the mobile phone, in separate tests, and recorded the power consumption while the application was transmitting every 4 seconds for a total of 300 transmissions. We repeated these tests again with transmissions every 10 seconds for both UDP and TCP over an additional 300 transmissions. The phones were flipped closed during these tests, and therefore the display was off. The results of these tests are shown in Figure 35. On the left, Figure 35 (a) shows that when transmitting via both UDP (blue line) and TCP (red line) the device radio is constantly active, and therefore the difference in power consumption between UDP and TCP is negligible.



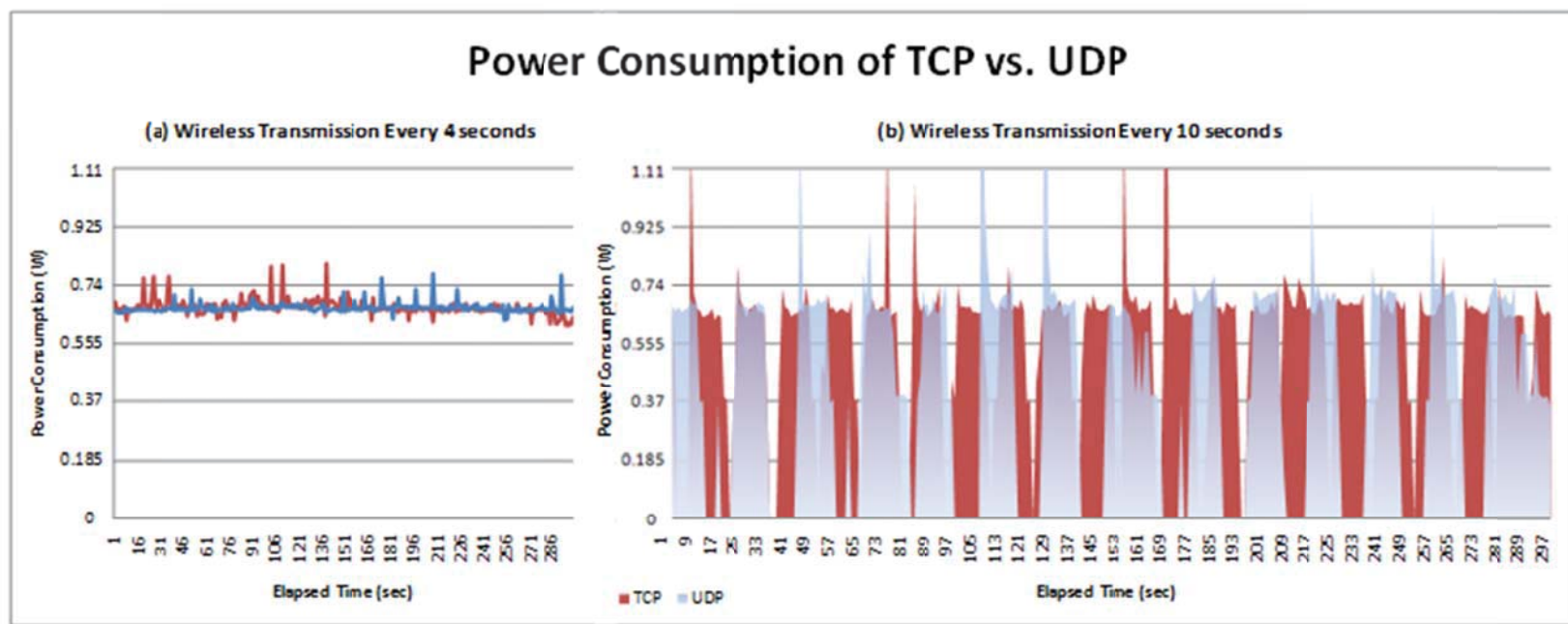


Figure 35 - (a) While at 4 second transmission intervals TCP and UDP have similar power consumption, (b) at 10 second transmission intervals it is evident that TCP consumes approximately 38% more power than UDP. [118] © 2011 IEEE

However, Figure 35 (b) shows that as soon as there is enough time in between transmissions for the radio to reach a power-off state, UDP (blue shaded area) is able to reach this state more quickly after each transmission than TCP (red shaded area).

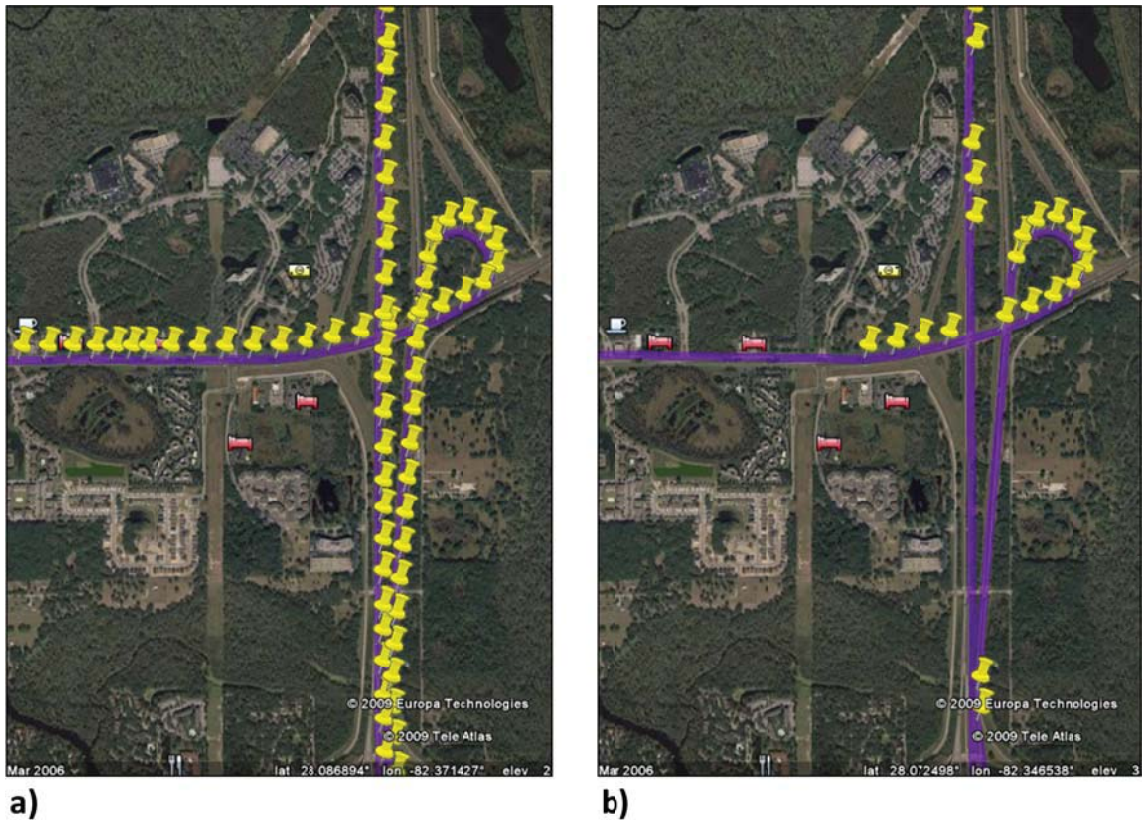
Examining the data more closely, the approximate energy used during UDP transmissions is 110 joules, while TCP uses approximately 152 joules during transmissions. This yields an average energy use of approximately 3.68 joules/transmission for UDP and approximately 5.08 joules/transmission for TCP. Therefore, TCP consumes approximately 38% more power than UDP for 10 second transmission intervals.

These results confirm our hypothesis that the reliability features in TCP (e.g., verification of packet arrival, retransmissions of lost packets) force the radio to stay in a power-on state longer than if UDP is used, which justify our choice of UDP as a location data transport protocol instead of TCP. Developers of mobile applications can use these results for guidance to balance their apps individual reliability requirements against the additional energy consumption of TCP.

In conclusion, the Session Management and Adaptive Location Data Buffering modules address several of the needs for location-aware mobile apps outlined in Chapter 1. Both modules contribute to battery life savings by providing energy-efficient (Need #1) real-time (Need #2) data communication between a mobile phone and server, increasing the average battery life for application data transfer by approximately 28% and reducing the average energy cost for location data transfer by approximately 38%.

#### 4.3.4 Critical Point Algorithm

Since the primary motivations for using the Critical Point Algorithm are the battery life and wireless data transfer savings, we first focus on quantifying these two characteristics. Sample output of the Critical Point Algorithm from a user entering and exiting a highway is shown in Figure 36.



**Figure 36 - a) All GPS data points generated from a phone are shown on the left, while b) only the critical points generated by the Critical Point Algorithm are shown in the right**

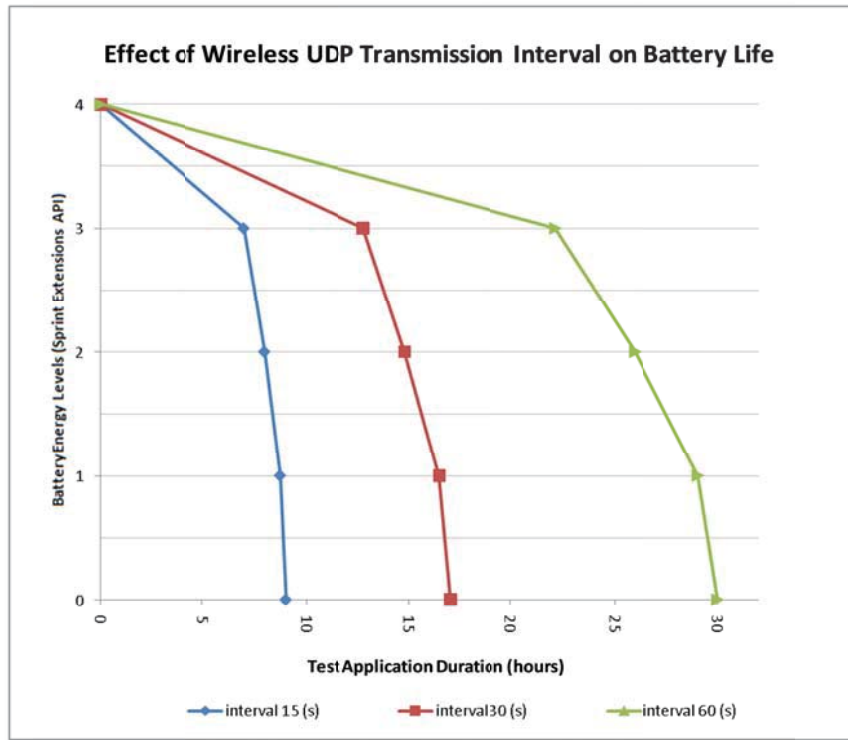
On the left, all GPS points generated from the mobile phone are shown, while on the right only the output from the Critical Point Algorithm is shown. The purple lines shown on the left and right represent the same approximate path of the user by connecting

sequential points, even though the output of the Critical Point Algorithm consists of far fewer points.

We hypothesized that the Critical Point Algorithm would save battery energy by filtering unneeded GPS data points before they are transmitted via UDP, effectively increasing the interval of time between UDP transmissions. Therefore, to evaluate the potential battery energy savings of the Critical Point Algorithm, we examined the effect of UDP transmission interval on battery life.

To evaluate this hypothesis, we created another custom Java ME test application that repeated UDP transmissions at a user-defined interval. GPS was not active during these tests, so GPS data was simulated by hard-coding a set of data that was observed during separate tests. Similarly to the test applications described earlier, this application recorded timestamps to the persistent MIDP Recordstore every several UDP transmissions. The device was charged until the battery life indicator on the outside of the device indicated a full charge, and then the test software was executed on the device until the battery was depleted to the point that the device powered itself off. After plugging in the device and powering it back on, we restarted the testing application and pressed a button to retrieve the timestamps from the most recently completed test. Through this method, we were able to record the length of time that the phone was operational while transmitting location data to the server using various transmission frequencies. A Sanyo SCP-7050 mobile phone using the standard Sanyo SCP-22LBPS 3.7V Lithium Ion 1000 milliampere-hour (mAh) battery on the Sprint-Nextel CDMA 1xRTT cellular network was used for these tests.

Figure 37 illustrates the difference in device battery life for an application utilizing UDP to transmit simulated GPS fixes to the server at fixed transmission intervals of 15, 30, and 60 seconds until the device's battery was completely depleted.



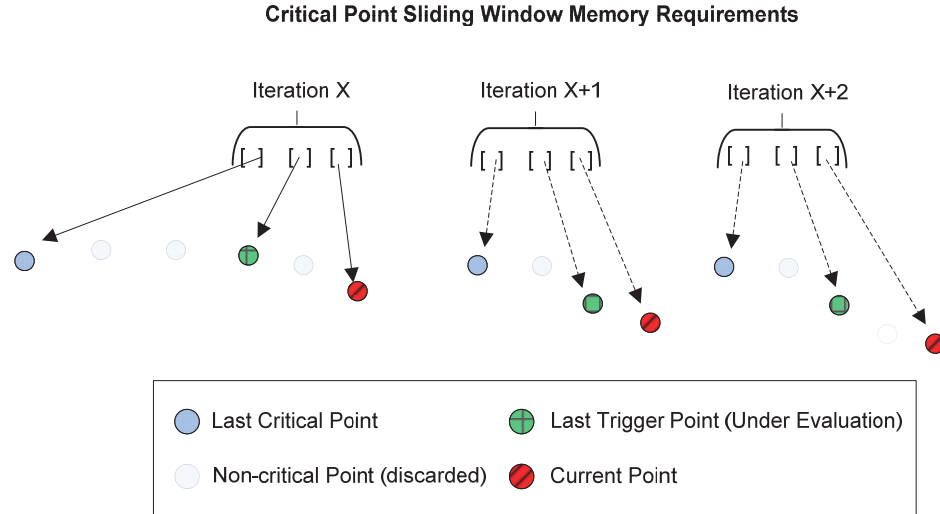
**Figure 37 - The Critical Point Algorithm can more than triple battery life by filtering GPS data and transmitting at an interval of 60 seconds instead of 15 seconds. [118, 119] © 2008, 2011 IEEE**

The Y axis of Figure 37 shows battery energy levels defined by an interval value provided by the Sprint Extensions API that were also recorded during the test, with values of 4 = FULL, 3 = HALF, 2 = LOW, 1 = WARNING, and 0 = POWER OFF. The X axis represents the duration of the test application in hours. For each interval, the decay of battery energy is evident as the battery energy levels dropped until the line met the X axis, at which point the mobile device powered off. It is clear that battery life is directly proportional to the length of transmission interval, meaning that less frequent

wireless transmissions result in a significant increase in battery life. By increasing the interval from 15 to 30 seconds, battery life is extended from approximately 9 hours to almost 17 hours. If the interval is increased further to 60 seconds, battery life reaches approximately 30 hours.

Now that we have shown that reducing the number of UDP transmissions can have a substantial effect on battery life, we must examine the feasibility of execution of the Critical Point Algorithm on a mobile device. Since the Critical Point Algorithm is going to run in real-time on the mobile phone and process a stream of generated location data, the algorithm must be efficient. For real-world implementation, the algorithm must maintain a constant memory requirement during execution, or the device will eventually run out of memory as it runs for a period of days or months. Additionally, the execution time of the algorithm should scale linearly regardless of the size of the dataset processed. If the execution time of the algorithm scales exponentially and attempts to loop through the entire dataset multiple times, then the software executing in real-time will inevitably fall behind the real-time data stream.

To prove that the Critical Point Algorithm maintains a constant memory requirement, we examined the amount of information required during execution. For each execution of the Critical Point Algorithm, we kept a maximum of 3 data points in memory: Last Critical Point, Last Trigger Point, and Current Point. We implemented the Critical Point Algorithm as a sliding window with pointers to the 3 data points, as shown in Figure 38.



**Figure 38 - The Critical Point Algorithm maintains a constant memory requirement during execution by using at most three location data pointers**

If a new critical point was found, then the first and second pointers were re-assigned to the next respective points. If no critical point was found, then the third pointer was again moved on to the next GPS data point, and the first and second pointers remain unchanged. Therefore, the memory requirements of the Critical Point Algorithm,  $f(n)$ , is constant:

$$f(n) = O(1)$$

where  $n$  is the number of GPS data points processed.

To prove that the Critical Point Algorithm could scale linearly in execution time with real-time data input, we examined the necessary number of steps to process one new GPS data point. For each GPS data point, we computed the azimuth between two sets of points: the Last Critical Point and Last Trigger Point, and the Last Critical Point and the Current Point. The Vincenty Inverse Algorithm was used to compute the azimuth, which has been shown to execute in a constant amount of time [136], and therefore is  $O(1)$ . We



also computed the difference between the two azimuth values, and compared the current speed of the user to a speed threshold for each data point, which could also be done in a constant amount of time. Since the Critical Point Algorithm would be executed once for each new GPS data point, and a constant number of steps completed for each new GPS data point, we determined the time complexity of the Critical Point Algorithm,  $f(n)$ , to be:

$$f(n) = O(n)$$

where  $n$  is the number of GPS data points processed. Therefore, the Critical Point Algorithm scales linearly with the number of GPS data points and can successfully run as a real-time stream processing algorithm.

Once we confirmed that the algorithm scales linearly in execution time and is constant in memory requirements with the amount of data processed, we next examined the threshold values that could be used by the Critical Point Algorithm for *min\_speed\_threshold*, *max\_walk\_speed*, and *angle\_threshold*.

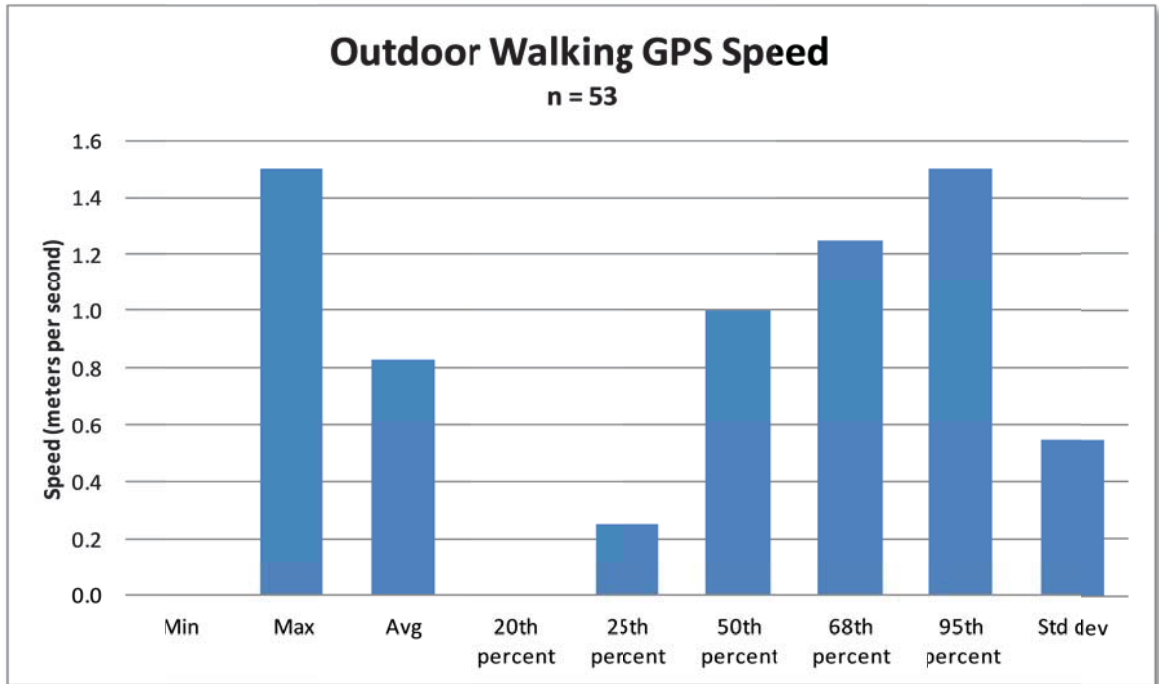
For *max\_walk\_speed*, previous research has indicated that mean maximum walking speed for fastest group of subjects studied was slightly over 2.5 meters per second [159].

Therefore, we chose 2.6 meters per second as our *max\_walk\_speed* threshold to distinguish whether the user is walking.

To choose the *min\_speed\_threshold*, we referred back to the speed tests performed while stationary. We wanted to eliminate points generated while the user was standing still, but we also wanted to capture points that represented the user's path and did not want to



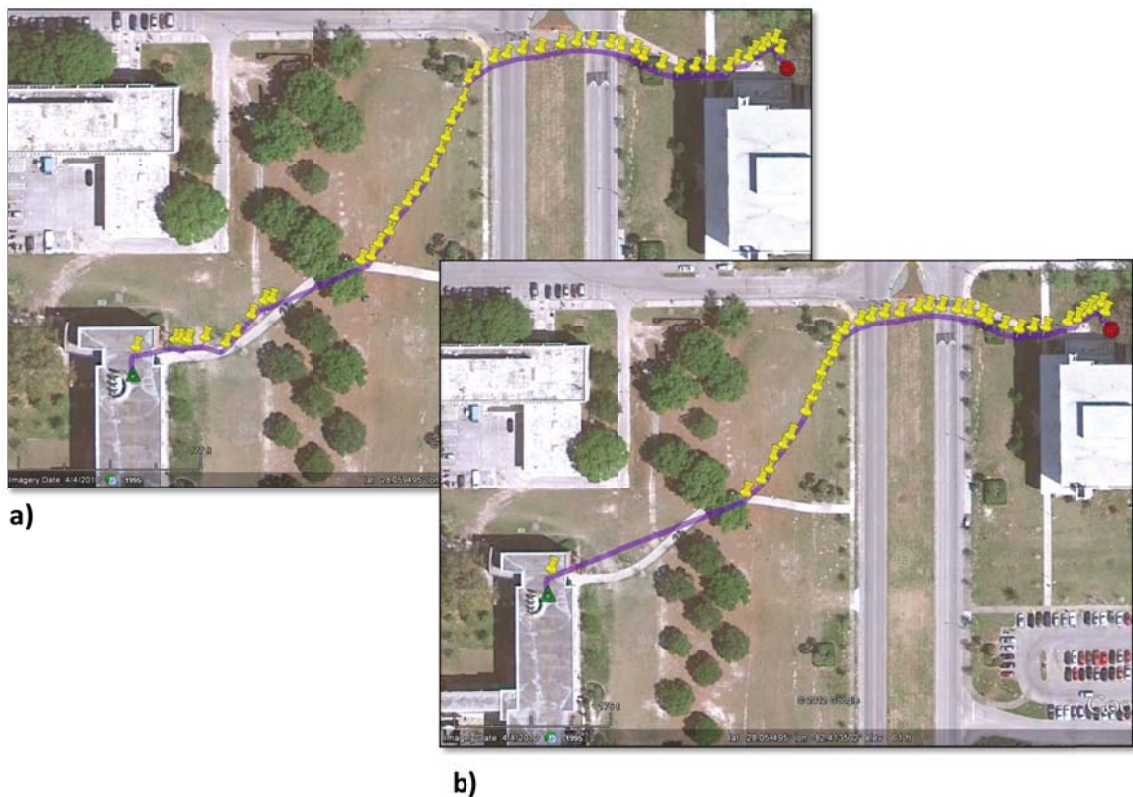
accidentally eliminate low speed points collected while the user was walking. We observed a user carrying a Sanyo Pro 200 at casual walking speed outdoors, and the resulting speed measurements are shown in Figure 39.



**Figure 39 - We observed the GPS speed recorded while a user was casually walking, which includes some speed values of 0 meters per second**

The minimum speed column in Figure 39 is of interest, since it shows that while the user is walking, the GPS generates points on a path while still recording the device's speed as 0 meters per second. If we chose 0 as our *min\_speed\_threshold*, then we would be allowing all points through the CP algorithm whenever there was a change in direction, and we would lose significant benefits of the CP algorithm when the user is standing still due to GPS drift. Therefore, to allow filtering of location data points while the user was stationary, we used a *min\_speed\_threshold* value of 0.1 meters per second. Based on the observed data, this would still allow over 75% of the true walking points through the CP

algorithm based on the speed threshold, and would yield a false negative rate based on speed when identifying critical points when walking of less than 25%. Since a user cannot travel far at approximately 1 meter per second, this false negative rate is considered acceptable. Figure 40 a) shows all GPS points recorded in this walking trip, while Figure 40 b) shows only critical points when using the 0.1 meters per second *min\_speed\_threshold*. For this test, we collected all GPS data points from the cell phone and then post-processed the data using the Critical Point Algorithm on the server, so we could compare all GPS data to just Critical Points using the exact same GPS data set.

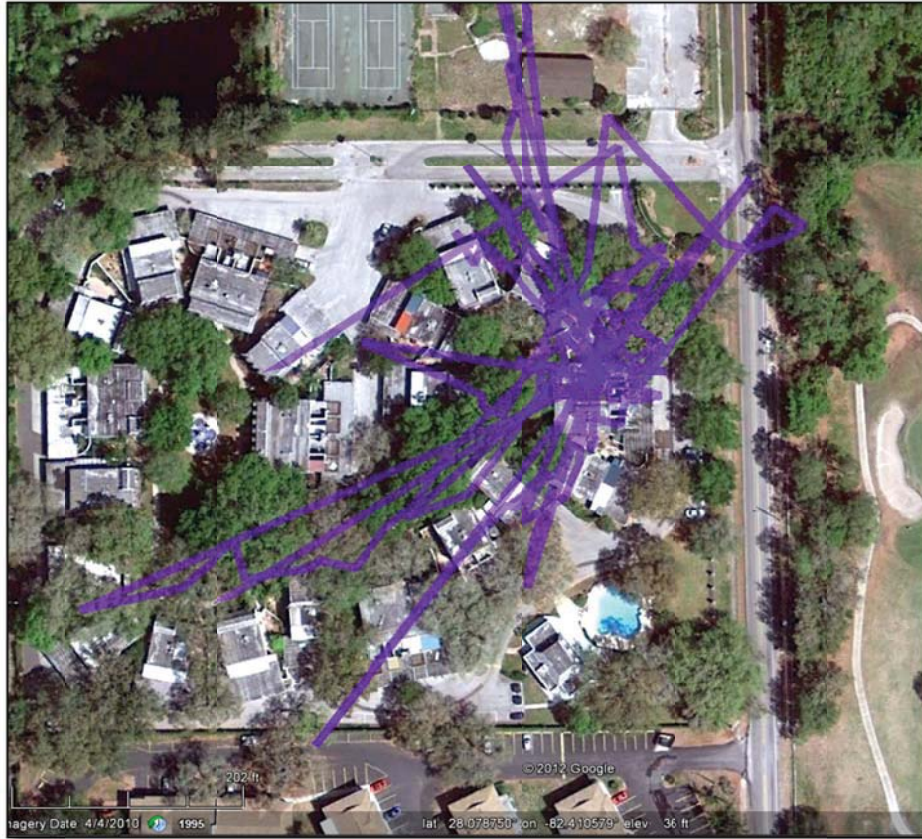


**Figure 40 - When comparing a) all points to b) critical points using a *min\_speed\_threshold* of 0.1 meters per second, the general walking path of the user is preserved, with some filtering at the beginning of the trip (bottom left of each image).**

The general walking pattern of the trip is preserved in both sets of data, with the primary filtering by the Critical Point Algorithm taking place at the very beginning of the trip, when the speed values are 0 meters per second. We believe this initial reading of 0 meters per second is due to Kalman filtering of the speed data happening in the Java ME platform or GPS firmware or hardware. In other words, the Kalman filter initially classifies the increase in speed as noise and filters this information out. However, once the user moves outside in the same general direction, the GPS speed values begin to reflect the user's true speed, as the Kalman filter reacts to the continuous outdoor movement.

To demonstrate the potential of savings of the *min\_speed\_threshold* value of 0.1 meters per second versus a value of 0 meters per second, we examined a day's worth of GPS data when the user stays in one location. During this time, the GPS drift can be substantial, such as that shown in Figure 41.

The Critical Point Algorithm filtered out over 97% of this data when using a 0.1 meters per second speed threshold, compared to only 74% of this data when using a 0.0 meters per second threshold (Table 6). In other words, we took advantage of the speed accuracy when the user was stationary to largely eliminate this erroneous movement, while still keeping an accurate record of the user's walking movements. Table 6 shows a detailed comparison for both the walking data and the stationary data when using the *min\_speed\_thresholds* of 0.0 meters per second and 0.1 meters per second. For these tests, an *angle\_threshold* of .5 degrees was used.



**Figure 41 - Over 97% of the GPS drift shown here at an indoor stationary location can be filtered out by the Critical Point Algorithm when using a 0.1 meters per second min\_speed\_threshold**

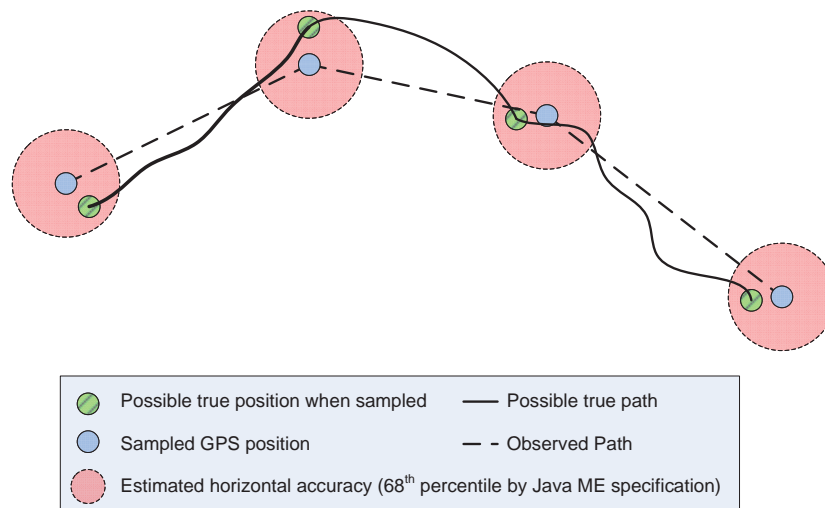
**Table 6 - When using the 0.1 meters per second min\_speed\_threshold, the Critical Point Algorithm is able to produce significant data filtering savings with only a slight impact on accurate walking paths**

	Min Speed	Number of Critical Points	Total Number of Points	% Savings	Bytes Saved*
Walking	0	50	53	5.66%	357
	0.1	39	53	26.42%	1,666
	Min Speed	Number of Critical Points	Total Number of Points	% Savings	Bytes Saved*
Stationary	0	904	3519	74.31%	311,185
	0.1	91	3519	97.41%	407,932

\* Based on 119 bytes per UDP payload

The final property of the Critical Point Algorithm to examine was the *angle\_threshold* for determining the magnitude of change in direction that should trigger a critical point to be generated. If the *angle\_threshold* is increased, fewer critical points will be generated, which will save battery energy and data transmission and storage costs. However, if fewer critical points are generated, the line defined by the remaining critical points becomes a less accurate representation of the user's path.

To illustrate the tradeoff between fewer data points and loss in path accuracy, ideally we wanted to compare the accuracy of a path generated by the Critical Point Algorithm against the true path of the user. However, determining the true path of the user is not trivial because the sampled GPS positions of the user are an approximation of the true position of the user, as shown in Figure 42.



**Figure 42 - Sampled GPS data points create an approximated path of the user with some uncertainty**

Therefore, the observed path reconstructed using the sampled GPS data, shown as the dashed line in Figure 42, is not equivalent to the true path traveled by the user, shown as

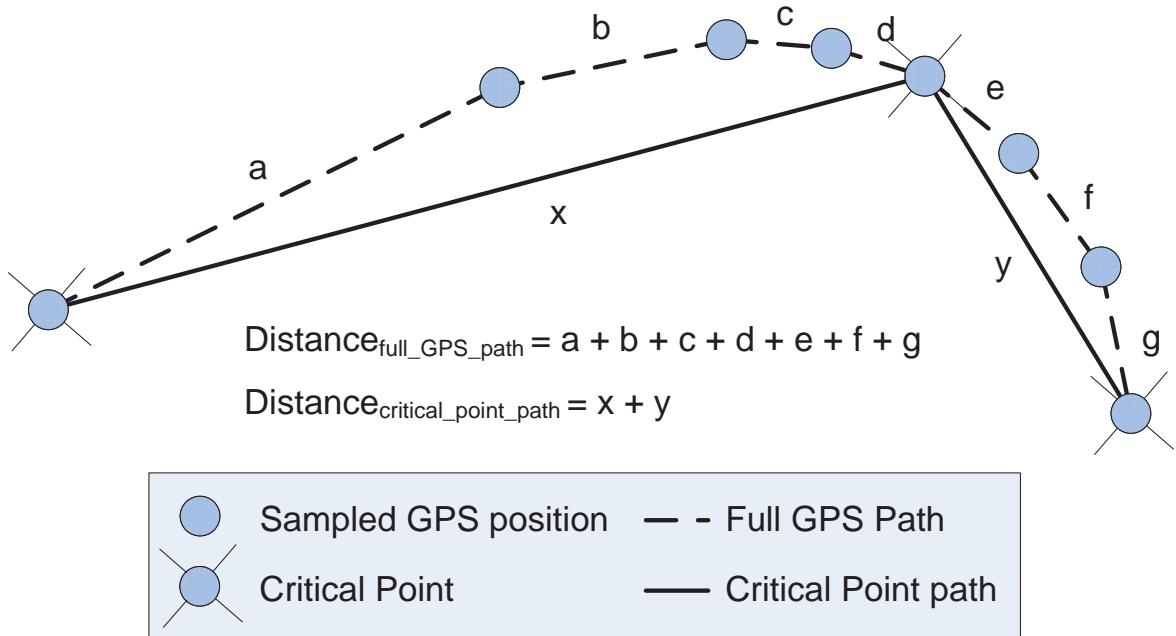


the solid line in Figure 42. The accuracy of the observed path is defined in part by the accuracy of the individual GPS data points, which is influenced by many environmental factors. The GPS sampling interval also has an impact on observed path accuracy, as the more frequent sampling will typically yield a better representation of the path.

We originally planned to use to our primary accuracy metric provided by the Java ME Location API, the estimated horizontal accuracy for each GPS fix, to estimate the true path of the user based on the collected GPS data. The red uncertainty circle around each GPS fix, defined by the estimated horizontal accuracy, is shown in Figure 42. However, as discussed earlier, in our experiments with GPS Auto-Sleep we found the estimated horizontal accuracy value to be unreliable and not within the specification defined by the Java ME Location API. Therefore, a methodology to evaluate line accuracy based on estimated horizontal accuracy would not provide a useful analysis.

When using the Critical Point Algorithm with our TRAC-IT mobile application to record travel behavior, we decided to approach the evaluation of the *angle\_threshold* used in the Critical Point Algorithm from a practical perspective. One of the key metrics that TRAC-IT was implemented to record is travel distance. In our research, we have found that the observed GPS path of the user recorded outdoors using a GPS interval of four seconds is a reasonable representation of the path for the purpose of most Location-based Services, including measuring travel distance. Therefore, we decided to analyze the impact of the *angle\_threshold* values on the distance of the path generated by the Critical Point Algorithm.

Figure 43 shows the difference between a path generated by the Critical Point Algorithm and the path defined by the complete GPS dataset.



**Figure 43 - The distance of the path generated from Critical Point Algorithm will always be shorter or equal to the distance of the path using all GPS data points**

The Critical Point Algorithm does not synthesize points; the set of critical points remaining after execution of the algorithm is always a subset of the points that appeared in the original GPS data:

$$\text{Critical Points} \subset \text{Full GPS Dataset}$$

Therefore, the distance of the full GPS path, Distance<sub>full\_GPS\_path</sub>, will always be greater or equal to than the distance of the path defined by the critical points, Distance<sub>critical\_point\_path</sub>:

$$\text{Distance}_{full\_GPS\_path} \geq \text{Distance}_{critical\_point\_path}$$

As the *angle\_threshold* increases,  $Distance_{critical\_point\_path}$  will decrease until eventually only the first and last GPS points remain. When only two critical points remain, the  $Distance_{critical\_point\_path}$  reaches its minimum value, which is the distance of a single straight line connecting the first and last GPS points.

We defined the error between the path created by all the GPS data points and the path created by critical points as the distance error percentage:

$$Distance\ error\ percentage = \frac{(Distance_{full\_GPS\_path} - Distance_{critical\_point\_path})}{Distance_{full\_GPS\_path}}$$

To assess the tradeoffs between angle threshold and number of critical points generated using the distance error percentage metric, we post-processed the same walking trip presented earlier using the Critical Point Algorithm and a range of angles from 0.5 degrees to twenty degrees, in 0.5 degree increments. The resulting lines, consisting only of critical points, are shown in Figure 44.

We repeated the same experiment on a trip via car and collected all GPS data points so we could post-process the results with many different parameters for the Critical Point Algorithm on the same dataset. The resulting critical point count in relation to the chosen angle and distance error percentage for the walk and car trips are shown in Figure 45, with the walking data set on the left and the car on the right.

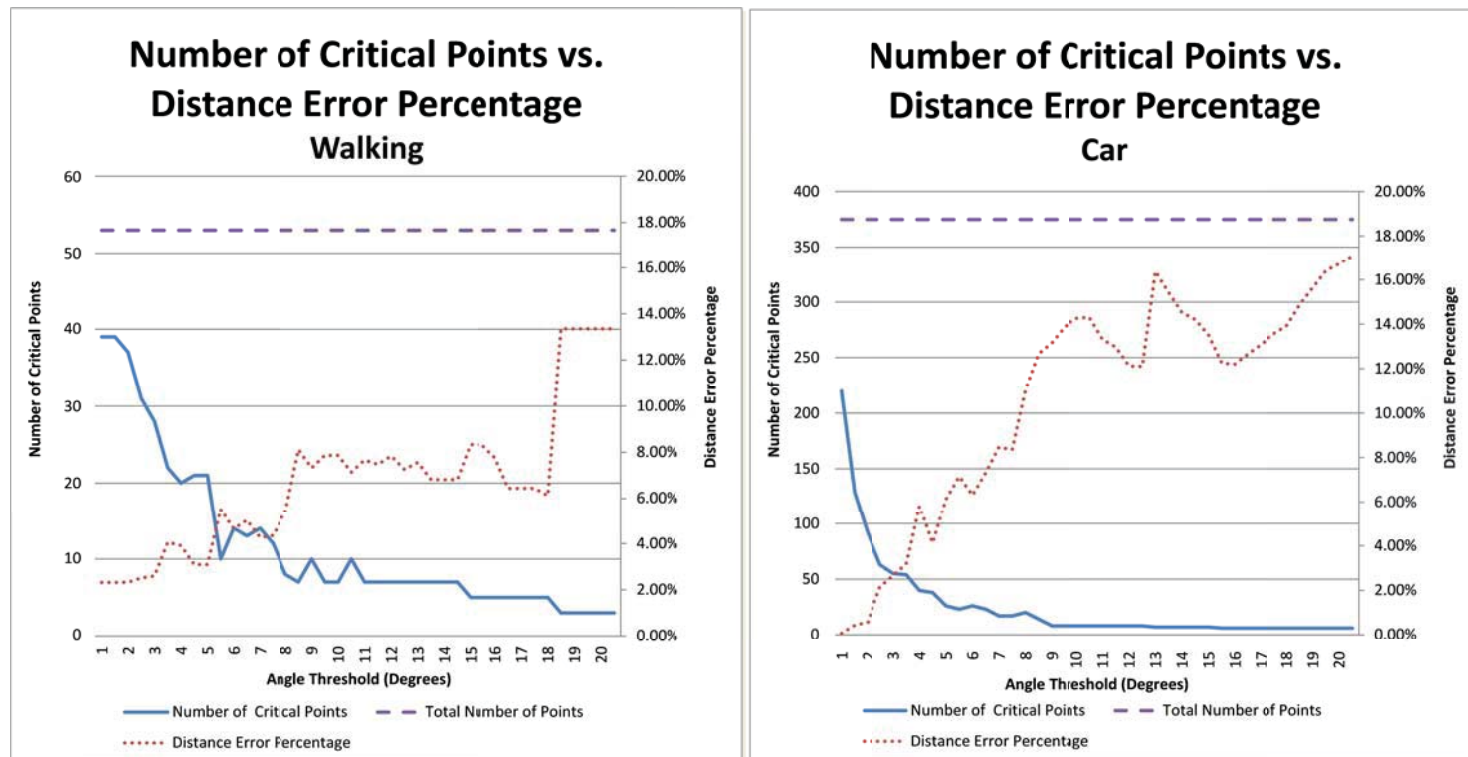
As we expected, there was a general trend towards fewer critical points and larger distance error percentages with larger angle thresholds for both walk and car trips.



For the walk trip, the line originally consisting of 53 points was reduced to 3 critical points when using an angle threshold of 18 to 20 degrees. The most dramatic change happened between thresholds of two to eight degrees, with the rate of change for the reduced number of points decreasing after approximately eight degrees. If we continued to increase the angle, the line is eventually reduced to just the beginning and ending points at the angle value of 65.5 degrees (not shown on graph).



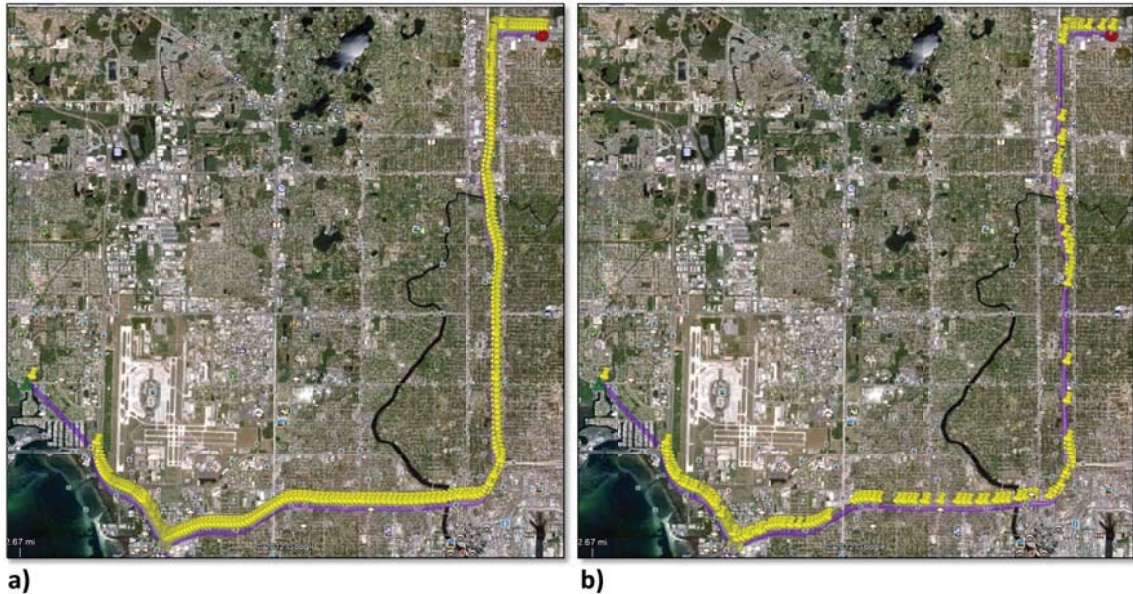
**Figure 44 - Running the Critical Point Algorithm with increasing angle thresholds gradually reduces the number of points that represent the line, which increases the distance error percentage**



**Figure 45 - As the angle threshold for the Critical Point Algorithm increases, there is a general trend towards fewer critical points being generated and an increase in the distance error percentage for both walking and car**

Inversely, the distance error percentage had a general trend of increasing as the angle threshold increased, since the original representation of the line became increasingly distorted. At eight degrees, as the rate of change of reduction of points began to level off, so did the distance error percentage. The brief local trends in increased number of critical points that occurred at angle thresholds 6 and 10 were due to shifts in the geometry of the line due to different vertices being used to connect vectors forming the critical point path. Several brief local trends in decreased distance error percentage at angles 4, 7, 10, and 16 were also due to shifting line geometries.

For the car trip, there was a substantial change in the number of points eliminated for angles 0.5 to approximately five degrees. After approximately five degrees, the rate of change for percentage of points eliminated began to level off.



**Figure 46 - For car trips, the Critical Point Algorithm is able to dramatically reduce the full GPS dataset, a), to far fewer critical points , b), with lower angle\_threshold values because of longer straight paths**

This immediate sharp decrease in points starting at 0.5 degrees occurred because of the larger number of points in the car trip (375 points) and larger distances in-between points, where the car traveled in a relatively straight line on the road, with turns happening mainly at intersections. Figure 46 shows all GPS points for a car trip on the left, and only critical points using an *angle\_threshold* of 0.5 degrees on the right. A large number of points along straight lines were eliminated quickly at lower angle thresholds, unlike walk trips that required larger angles to remove an equivalent percentage of points.

Table 7 shows these statistics for percentage savings for number of points when using the Critical Point Algorithm for the walk trip compared to the car trip, with the Critical Point Algorithm immediately producing a percentage savings of 41.33% at *angle\_threshold* 0.5 for the car trip and only 26.42% using the same angle for the walk trip. The car trip was reduced by over 90% of its points starting at angle 4.5, while for the walk trip, savings over 90% were not realized until angle 14.5.

Also unlike walk trips, as the reduction in points began to level off for car trips the distance error percentage continued to climb. There were a few shifts in geometry, like the walk trip, that produced local trends in decreasing distance error percentage.

Interestingly, unlike the walk trip, the shifts in geometry with local trends in decreased distance error percentage did not typically produce a larger number of critical points. In other words, there was only one small local trend in an increased number of critical points with an increasing *angle\_threshold*. This behavior was also a result of quickly eliminating the points along the straight lines in car trips using low *angle\_threshold* values.





therefore elimination of a nearby fix has a lesser impact on the measured travel distance than the removal of a fix that is further away.

Based on our observations from the above experiments, we chose the following values for the Critical Point Algorithm thresholds:

- *min\_speed\_threshold* = 0.1 meters per second
- *max\_walk\_speed* = 2.6 meters per second
- *angle\_threshold* = 4.5 degrees for walk trips, and 3 degrees for car trips.

For a final analysis on the expected data savings of the Critical Point Algorithm using these thresholds, we post-processed 1,314 trips of GPS data that were collected using the TRAC-IT mobile application. The results are shown in Table 8.

**Table 8 – The Critical Point Algorithm is able to reduce GPS datasets by more than 77% on average while maintaining an average distance error percentage under 10%.**

	Min	Max	Avg.	5th percentile	25th percentile	50th percentile	68th percentile	95th percentile
Total Critical Point Count	2	322	35	3	13	27	38	97
Total GPS Fix Count	20	3,710	193	31	74	130	188	511
% Savings	20.83%	99.40%	77.43%	47.97%	69.49%	80.00%	86.83%	95.84%
Bytes Saved*	595	403,172	18,883	2,380	6,426	12,138	17,493	54,788
Distance Critical Points (m)	0.00	1,043,805.50	7,437.09	328.14	1,162.37	2,675.00	4,049.37	22,815.61
Total Distance (m)	2.36	1,087,043.20	7,878.02	380.79	1,252.55	2,913.39	4,345.91	24,231.34
Distance Error Percentage	0.00%	100.00%	8.90%	1.94%	3.98%	6.20%	8.70%	24.11%

\* Based on 119 bytes per UDP payload

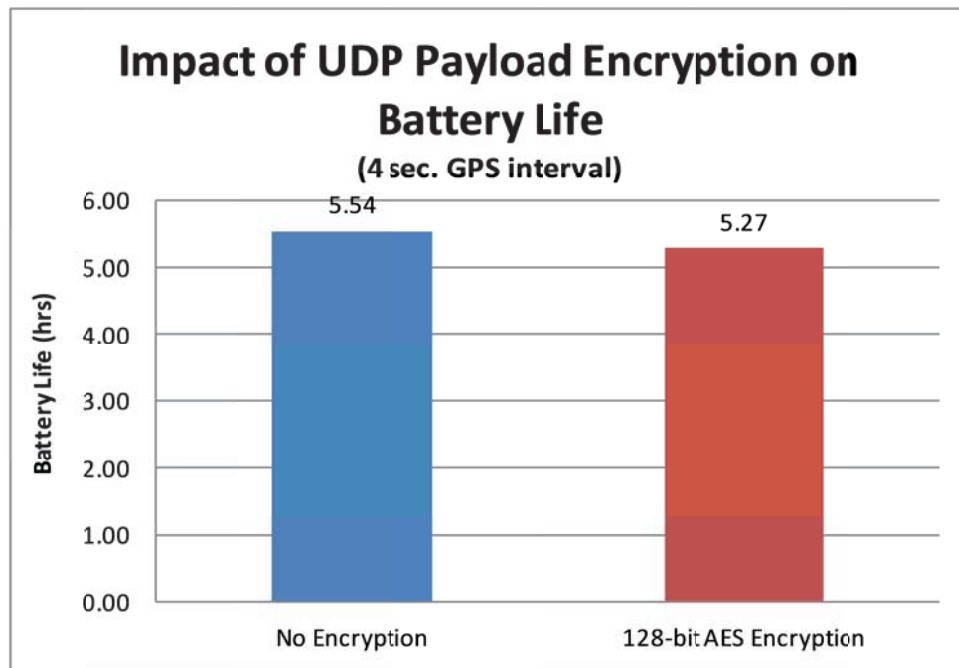
The average percent savings when using the Critical Point Algorithm was a reduction of approximately 77% of the GPS data points. The 5<sup>th</sup> percentile of percent savings was approximately 48%, which means that 95% of the time the percent savings is above 48%. Assuming that each GPS fix was 119 bytes, this translates to a substantial data transfer savings if the non-critical points were not transferred from the mobile device to the

server. On average, we saved almost 19 kilobytes in data transfer per trip. Since surveyed U.S. travelers report an average of 4.6 trips per day with a mean duration of approximately 20 minutes per trip [149], the amount of data saved adds up quickly. If location-aware services were to be provided to the 322.9 million U.S. cellular subscribers, the Critical Point Algorithm would save approximately 279.2 gigabytes of data transfer over the cellular network per day. Additionally, the average distance error percentage is kept under 10%, which is sufficient for our purposes of distance tracking. We could adjust the angle threshold value, if needed, to increase or decrease the distance error percentage and percent savings, depending on the needs of future applications.

In conclusion, the CP Algorithm module addresses several of the needs for location-aware mobile apps outlined in Chapter 1. The CP Algorithm reduces battery energy expenditures (Need #1) and the amount of data transferred between the mobile phone and server (Need #1) in real-time (Need #2) by eliminating non-essential GPS data (an average 77% reduction), with an average doubling of battery life, as the interval of time between location data transmissions is doubled. The CP Algorithm is able to maintain an average distance error percentage for GPS data under 10%, which ensures a high-precision and high-accuracy travel path (Need #3). We also presented a methodology to select values for the thresholds used in the CP Algorithm (*min\_speed\_threshold*, *max\_walk\_speed*, *angle\_threshold*) based on observed GPS data, allowing any third party mobile app developer to implement the algorithm on any GPS-enabled mobile device with a Location API (Need #4).

### 4.3.5 Location Data Encryption

To demonstrate that the Location Data Encryption of the UDP payload contents is feasible for use on Java ME phones, we implemented a test application that repeatedly encrypted GPS data every four seconds using 128-bit AES and occasionally saved timestamp information to the Java ME Recordstore until the battery of the device was depleted. AES was implemented using BouncyCastle Java libraries [147]. A second app was implemented that performed the same operations, but without encrypting the data. By comparing the output of these two applications, we determined the exact impact in terms of battery life on the mobile device. We executed these test applications on a Sanyo SCP-7050 mobile phone with a stock 3.7V Lithium Ion 1000 mAh battery on the Sprint CDMA 1xRTT network.



**Figure 47 - Location Data Encryption using 128-bit AES encryption for UDP payloads is feasible on mobile devices, although it does have a slight impact on battery life**



Figure 47 shows the results of the tests in battery life using encryption versus no encryption. There was a small impact on battery life, approximately 16 minutes in these tests, which is negligible for most mobile devices.

In conclusion, the Location Data Encryption module addresses several of the needs for location-aware mobile apps outlined in Chapter 1. Location Data Encryption ensures the security of the location data being transferred between the mobile device and server in real-time (Need #2), with only a slight impact on battery life (i.e., a decrease of 4.9%) (Need #1). Location Data Encryption can be implemented by any third party mobile app developer (Need #4) using existing software libraries such as BouncyCastle [147].

#### 4.4 Innovative Location-Aware Applications Developed Using LAISYC

Two separate research projects have implemented innovative location-aware applications using the LAISYC framework presented in this dissertation. LAISYC enables the use of efficient real-time, high-accuracy and high-precision location data in each of these systems. In this section we describe the two applications: TRAC-IT, a multi-modal travel behavior data collection tool that can provide simultaneous and real-time location-based services, and the Travel Assistance Device (TAD) mobile app to assist transit riders with intellectual disabilities.

##### 4.4.1 TRAC-IT

In order to solve transportation problems and effectively plan new roads or public transportation routes, transportation professionals require information about the current travel behavior of the general public. While road-based infrastructure such as loop-detectors can provide a count of cars traveling through a particular road, more descriptive

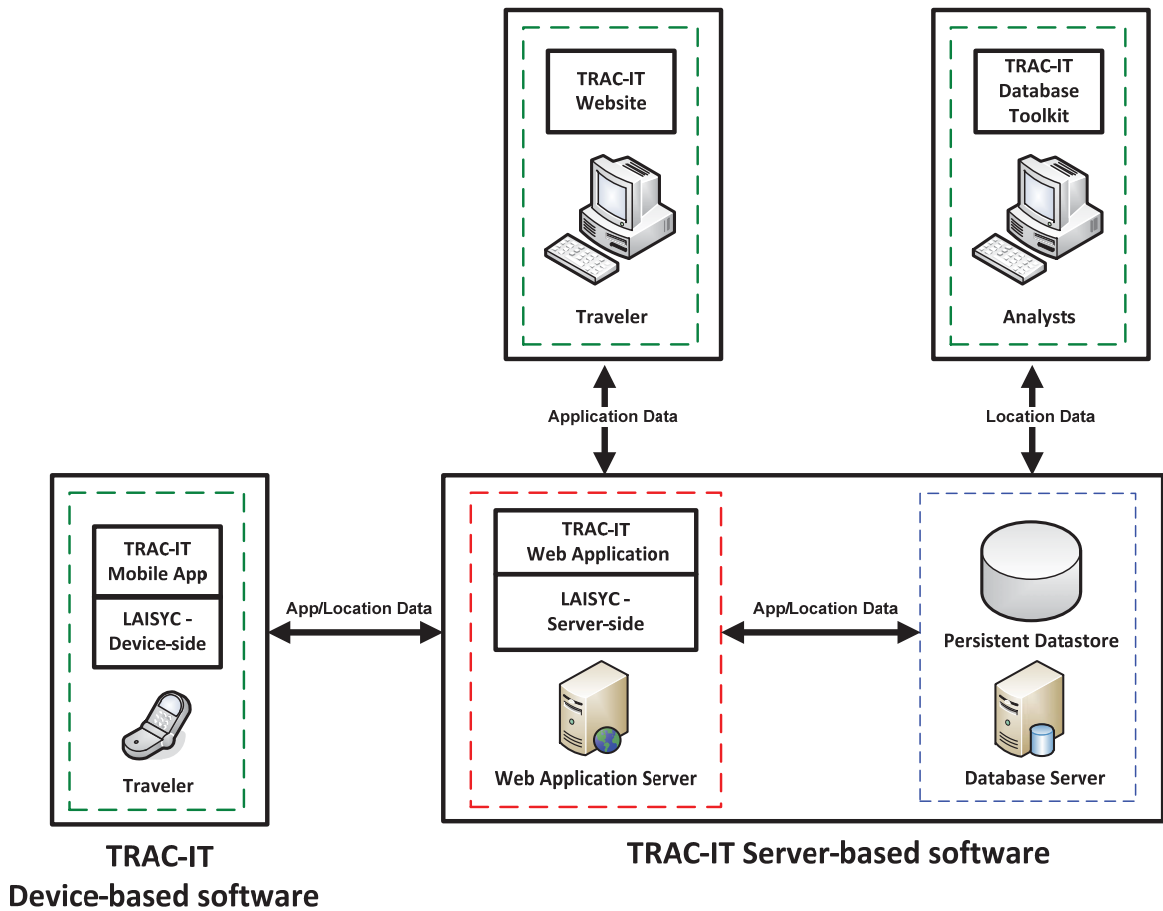
data for the purposes, origins and destinations of trips is desired. Additionally, road-based infrastructure does not provide any information about trips that are taken using public transportation, biking, walking, or carpooling.

In the past, paper diaries or phone interviews have been used to ask survey participants about their daily travel behavior. However, these manual survey methods typically only cover a day or two of travel behavior due to the burden on the participant. Due to this burden, past studies have shown problems with data accuracy and completeness in manual surveys when reported travel behavior was compared with vehicle-based GPS systems that also monitored the participant's travel behavior for the same period of time [160-162].

Vehicle-based GPS systems have the benefit of objective GPS data that is recorded at a particular time and location. However, like road-based infrastructure vehicle monitoring, vehicle-based GPS misses trips occurring via public transportation, biking, walking, and carpooling. Additionally, vehicle-based GPS could provide travel behavior data from more than one individual if the vehicle is shared within a household. Transportation professionals desire data per individual, over multiple modes of transportation, so behavior such as interactions within the household can be evaluated.

Our approach to enabling multimodal travel behavior data collection was to monitor the transportation behavior of an individual via TRAC-IT, a mobile application installed on a GPS-enabled mobile phone. However, for long-term travel data collection to be compelling for individuals participating in the study, an incentive for the individual to give up their privacy and donate their data would likely be required. One type of

incentive is a monetary reimbursement to the user. However, this quickly becomes expensive for the surveyor and could not be sustained over long periods of time with large populations. Another form of incentive could be services provided to the user. For example, Google provides free services and products such as Gmail, web search, Android, and others in exchange for access to a user's data. If we could provide valuable services to the survey participant, such as real-time personalized traffic incident alerts, this may be enough of an incentive for a user to contribute data to transportation professionals.



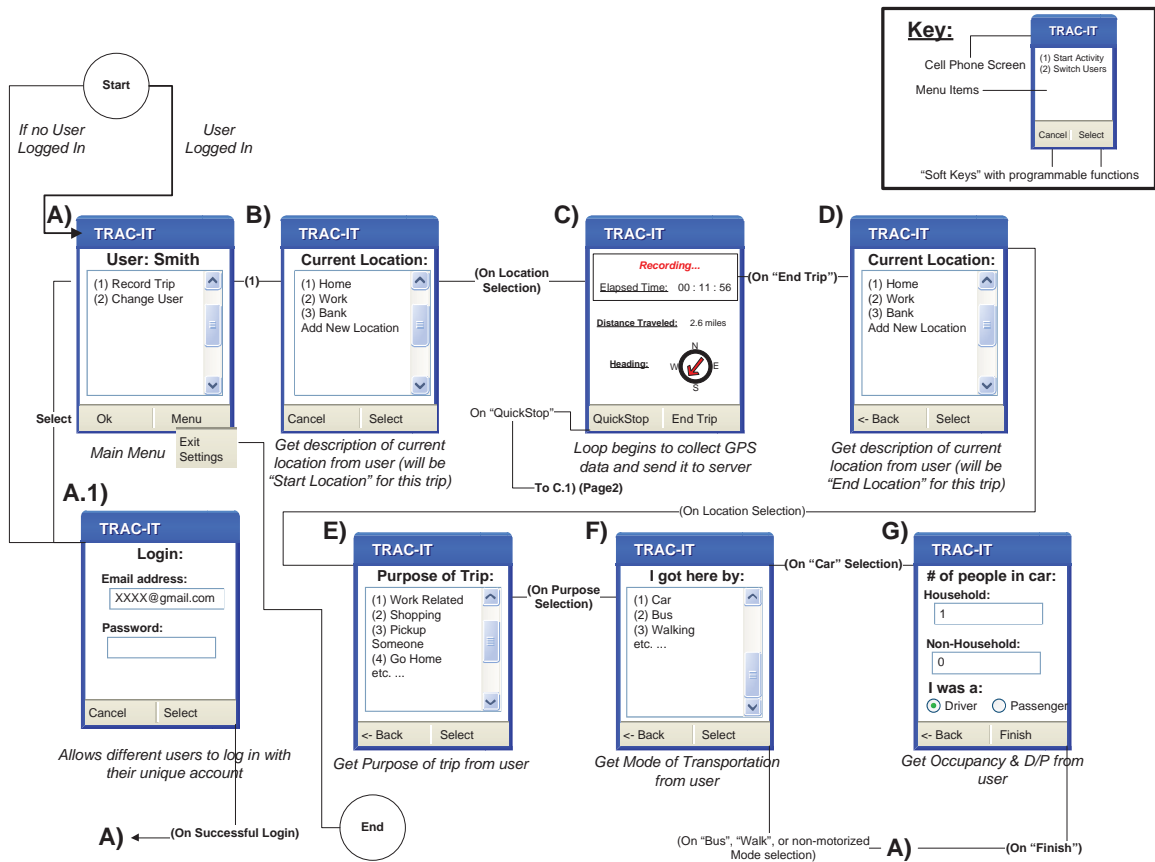
**Figure 48 - The TRAC-IT mobile application is based on the LAISYC framework to enable simultaneous travel behavior data collection and real-time location-based services**

To enable simultaneous data collection of multimodal travel behavior, as well as real-time location-based services, we implemented the mobile application TRAC-IT using LAISYC on Java ME. The TRAC-IT system architecture, based on LAISYC, is shown in Figure 48.

Glassfish was used as the Java Web Application Server to host the web application that communicated directly with the mobile phone, as well as provided a website for the traveler to manage account information (e.g., resetting passwords, etc.). We used SQL Server 2008 and PostGIS as the relational and spatial databases, respectively. Glassfish served as a proxy for database access for the mobile phone. We also created a Java desktop application, the TRAC-IT Database Toolkit, which was capable of a variety of both automated and manual post-processing analyses. For example, the TRAC-IT Database Toolkit running on the TRAC-IT server automatically generated Google Earth Keyhole Markup Language (KML) files for all trips taken by users and emails links to the user the day following the travel behavior so the user could examine the data and provide feedback to the analysts. Analysts could also use the TRAC-IT Database Toolkit to analyze and produce statistics for collected travel behavior, and evaluate the performance of the Critical Point algorithm, execute automated clustering algorithms to identify points-of-interest, use artificial neural networks to automatically classify the mode of transportation for a trip [163], as well as various other processing routines that have been implemented for the TRAC-IT system [131].

The TRAC-IT mobile application can execute in two modes:

- Passive data collection: The application runs on the background on a mobile phone without any interaction with the user, and collects only location data
- Active data collection: Every time when starting or stopping a trip, the user enters information such as trip purpose, mode of transportation, and vehicle occupancy via the TRAC-IT user interface (Figure 49). Location data is also simultaneously recorded.



**Figure 49 - The TRAC-IT mobile application provides a user interface to record input from the individual for mode of transportation, purpose, and vehicle occupancy as well as location data. [148]**

To evaluate real-time travel behavior data collection using TRAC-IT, we performed a variety of successful test deployments with the research team in both passive and active modes. These test deployments included GPS Auto-Sleep, Location Data Signing, the Critical Point Algorithm, Adaptive Location Buffering, Location Data Encryption, and Session Management, all using the parameters discussed earlier.

TRAC-IT was deployed in 2011 as part of a USDOT-sponsored research project in Tampa, Florida. We used the passive mode of TRAC-IT with 30 users on Sanyo Pro 200 mobile phones on the Sprint CDMA EV-DO Rev. A network. GPS Auto-Sleep was set in tracking mode with the parameters discussed earlier, and Location Data Buffering and Location Data Encryption were both used. We decided to turn off the Critical Point Algorithm so that we could collect a full GPS dataset from participants over a long period of time and use this data for a variety of post-processing and analysis routines, including the evaluation of the different Critical Point Algorithm parameters presented earlier.

From February 10, 2011 to April 29, 2011, TRAC-IT collected 1,857 sessions from 30 users (over 60 sessions on average per user) for a total of 4,023,917 GPS data points (Table 9).

Total survey time was calculated by the difference between the oldest and newest GPS times in each session, and a sum of differences over all sessions. TRAC-IT server uptime was over 99 percent during the data collection period.

We analyzed a subset of this data (899 sessions) to determine the reliability of UDP and Location Data Buffering during data collection.

**Table 9 - TRAC-IT was used as part of a USDOT-funded research project to collect over 4 million GPS data points from 30 users over 2 months**

<b>TRAC-IT Data Collection for USDOT-funded project</b>	
Date Range	2/10/2011 to 4/29/2011
Total Number of Users	30
Total Number of Sessions	1,857
Avg. Session Length (hrs)	15.44
Total Survey Time (days)	1,194.80
Avg. Survey Time per User (days)	39.83
Total Number of GPS fixes Received	4,023,917
Avg. Number of GPS fixes per Session	2,166.89
Avg. Number of GPS fixes per User	134,130.57

Each location data packet contained an integer that was incremented on each transmission, so the number of lost UDP packets could be determined by reviewing the missing counter numbers for each session. Table 10 shows that 95% of sessions had less than 3.95% of lost UDP packets, with an average UDP packet loss of 1.19%.

**Table 10 - 95% of sessions had less than 3.95% of lost UDP packets**

<b>UDP and Location Data Buffering - Packets Lost</b>		
	<b># Lost Per Session</b>	<b>% Lost Per Session</b>
Min	0	0.00%
Max	290	66.15%
Avg	15.67	1.19%
50th percentile	8	0.48%
68th percentile	13	0.88%
95th percentile	59.15	3.95%

We also compared the overall performance of TRAC-IT without LAISYC to TRAC-IT with LAISYC, as shown in Table 11. Without LAISYC, TRAC-IT battery life using only GPS sampling (i.e., not sending the location data to a server) was 8.04 hrs. When TRAC-IT both sampled GPS and sent the data to a server without using LAISYC, battery life dropped to 4.21 hrs.

**Table 11 - When TRAC-IT used LAISYC, device battery life nearly doubled while reducing overall location data packet loss by 2.16% and adding encryption**

	<b>GPS Sampling</b>	<b>Real-time server communication</b>	<b>Encryption</b>	<b>Battery Life</b>
<b>TRAC-IT</b>	4 s			<b>8.04 hrs</b>
<b>TRAC-IT</b>	4 s	UDP packet loss = 2.7% (n = 46,785)		<b>4.21 hrs</b>
<b>TRAC-IT w/ LAISYC</b>	Dynamic (4 s moving, 300 s stopped)	Adap. Loc. Data Buff. UDP packet loss = 0.54% (n = 2,642,309)	HTTPS - SSL UDP - 128-bit AES	<b>15.44 hrs</b> (avg, n = 1857)

When TRAC-IT used LAISYC, battery life was extended to at least 15.44 hrs on average even while sending data to the server and encrypting this data. Adaptive Location Data Buffering reduced the overall location data packet loss by 2.16%. These results clearly show the benefit of using LAISYC with a location-aware mobile app.

To demonstrate the ability of the TRAC-IT system to perform simultaneous data collection and real-time location-based services, we implemented a simple Path Prediction proof-of-concept using the LAISYC Spatial Analysis module. Path Prediction uses spatial representations of a user’s historical trips along with their real-time GPS position in order to predict the paths they may take in the immediate future [164]. Since human travel behavior has been shown to be highly repetitive in both space and time [154], historical trips can be effectively mined in order to anticipate the user’s future travel.

A spatial database was used to perform a series of intersection queries with the user’s real-time location (obtained using the LAISYC framework) and buffers surrounding



previously recorded trips. Each buffer that intersected the real-time location represented a potential path that the user may follow. Figure 50 shows a visualization of this process in Google Earth.



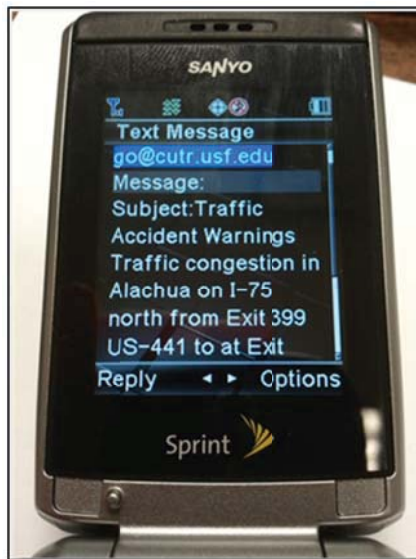
**Figure 50 - Path Prediction compares the traveler's real-time location, shown as yellow push-pin markers, against paths from the traveler's travel history, shown as yellow shaded buffers, to predict the immediate travel path**

The yellow push-pin represents the user's real-time location, and the yellow shaded buffers indicate the past historical paths that intersect with the user's real-time location.

Once these paths are identified, another series of intersection queries were performed to determine if any location-based alerts (e.g., traffic incidents) relevant to the user lay along the predicted paths. Intersecting location-based alerts could then be sent to the user via text message or within a mobile application. Based on the estimated impact to the

user for a false-negative or false-positive message hit, the messages could be pre-filtered accordingly.

We implemented and tested Path Prediction using the LAISYC framework, Glassfish as the Java Web Application Server, and PostGIS as the spatial database server. We created sample traffic incidents with descriptions in our database server with locations that overlapped the past travel behavior paths of a member of the research team to determine if incidents alerts would be triggered based on the user's real-time paths. Figure 51 shows a text message that was successfully sent by the server to the phone, based on the real-time location of the mobile phone, as well as the past travel paths.



**Figure 51 - Path Prediction successfully demonstrated that real-time location-based messages could be sent to the phone using LAISYC and a history of the traveler's behavior**

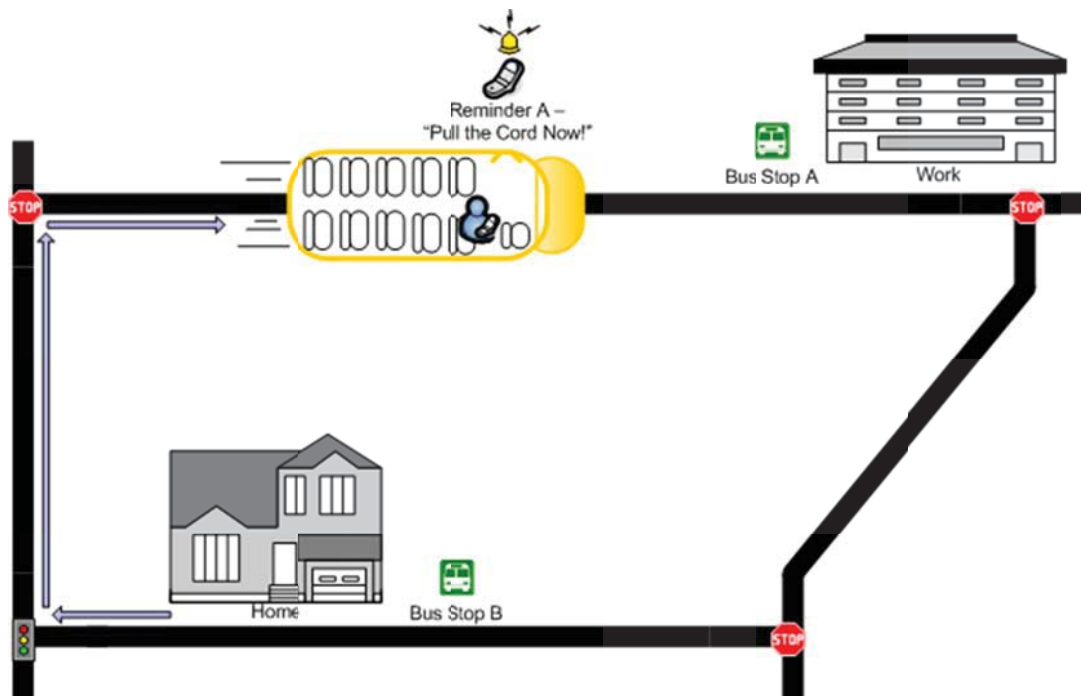
In conclusion, TRAC-IT is a multi-modal travel behavior data collection mobile app that can provide simultaneous and real-time location-based services (e.g., traffic alerts). In TRAC-IT, the GPS Auto-Sleep, Session Management, Adaptive Location Data

Buffering, Critical Point Algorithm, and the Session Management modules all contribute energy savings (Need #1) that enable the phone's battery to last an entire day during high-resolution, real-time GPS tracking (Needs #2 and #3). High-resolution, real-time GPS tracking is critical to TRAC-IT for reconstructing detailed travel path information, including distance traveled, as well as providing predictive, personalized traffic alerts based on historical and real-time data. The Location Data Signing module allows transportation analysts to trust information that is recorded by the application, while the Location Data Encryption module protects the privacy of users' location information. The Session Management, Adaptive Location Data Buffering, and Critical Point Algorithm modules allow TRAC-IT to avoid data overage costs on phones with limited data plans while still supporting real-time location data communication. The Adaptive Location Data Buffering module prevents tracking data from being lost when the user is outside network coverage or is on a voice call for networks that do not support simultaneous voice and data communications. TRAC-IT was successfully implemented and tested using LAISYC on actual mobile phones without any modification to device hardware or software (Need #4).

#### 4.4.2 Travel Assistance Device (TAD)

Traveling via public transportation such as a bus requires quick thinking and navigation skills. Identifying an upcoming bus stop as your correct destination and reacting to pull the cord or push a button to request that the vehicle stop in time is a challenging task, especially for the 16.4 million Americans, or 6.9 percent of the population, with intellectual disabilities [165]. For individuals who cannot perform this quick thinking on their own, transit agencies must provide equivalent door-to-door paratransit service.

Paratransit is a costly service to transit agencies at an average cost of \$17 per trip, versus \$1.70 per trip for regular fixed route transit [166], and can also be restrictive to riders by requiring 24 hour advance notice for trips, as well as long wait times. Transit agencies have instituted travel training programs in an attempt to train able riders to use fixed route transit. Travel training is an intense one-on-one instruction period in which travel trainers actually plan and travel with a transit rider on their personal trip and show the rider the various skills and steps required to successfully complete a trip via public transportation. However, in our work with transit agencies, travel trainers have indicated that one of the most challenging skills to master for these individuals is identifying and reacting to the riders upcoming destination stop. A traveler who cannot master this skill cannot ride fixed route transit independently.

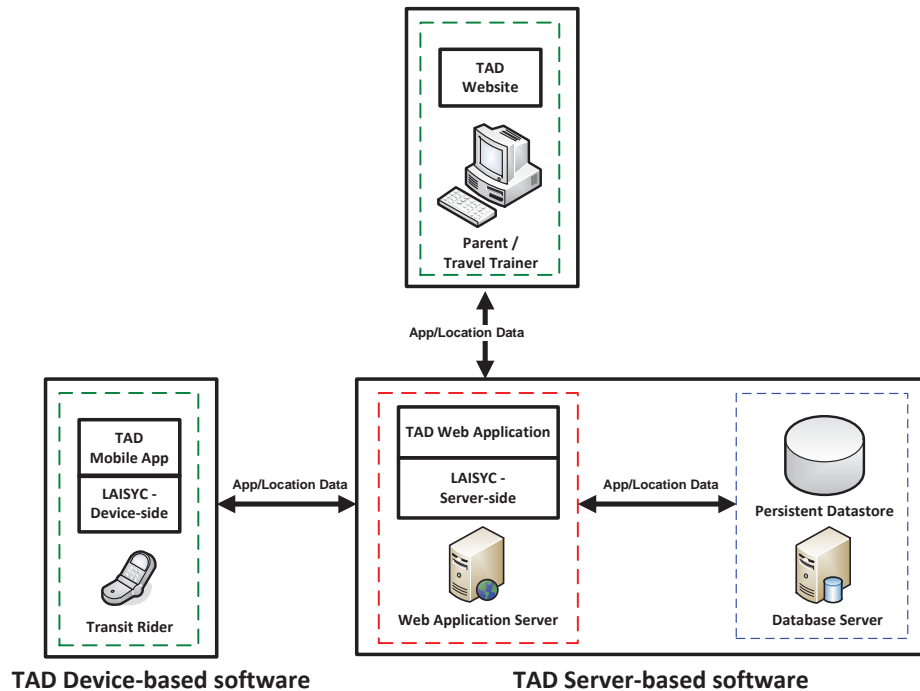


**Figure 52 - The Travel Assistance Device mobile application alerts the transit rider of an upcoming destination bus stop**

After discussing these challenges with the travel training and special education communities, we designed and implemented the Travel Assistance Device (TAD) mobile application for GPS-enabled mobile phones using LAISYC on Java ME.

We wanted to provide four services using TAD:

- 1) Website-based trip planning: Allow the travel trainer and caregivers of the transit rider to plan a transit itinerary, including the exact boarding and exit bus stops, via a website interface.
- 2) Real-time transit navigation prompts (the primary TAD feature): Alert the rider via audio, visual, and tactile prompts to identify an upcoming bus stop in real-time, much like a car-based navigation system, to help individuals with intellectual disabilities who had problems with this task. TAD alerts the transit rider twice: once with a “Get Ready...” notification several stops ahead of their destination, and repeatedly with a “Pull the Cord Now!” notification when the rider passes the bus stop previous to the destination stop until the rider confirms having received the alert by pressing a button (Figure 52).
- 3) Real-time location tracking: Allow the travel trainer and caregivers of the transit rider to always see the real-time location of the transit rider.
- 4) Automated lost alerts: Alert the travel trainer and caregivers if the transit rider wanders off the path of the planned trip.



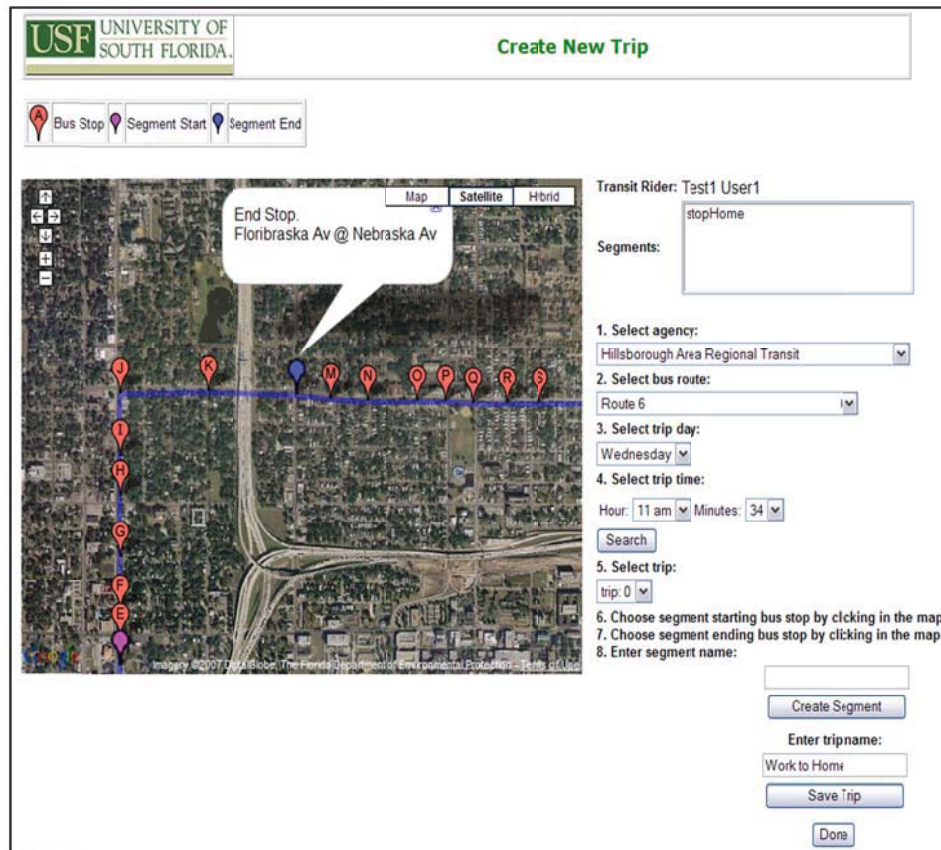
**Figure 53 - TAD was implemented using the LAISYC framework to support real-time location-aware services**

The TAD system architecture, based on the LAISYC framework, is shown in Figure 53.

We implemented the prototype TAD system and LAISYC framework on Java ME, and used Glassfish as the Java Application Server, to support the server-side portion of LAISYC and the TAD web application. SQL Server and PostGIS were used for relational and spatial database servers, respectively.

To support feature #1 of planning transit trips via a website, we implemented a web interface using the Google Web Toolkit (GWT). Figure 54 shows the map view of the travel trainer or caregiver choosing the boarding and exiting bus stops for a particular transit rider. A caregiver or travel trainer can simply choose the route they want the transit rider to use, clicking on the boarding bus stop, and clicking on the destination bus stop.



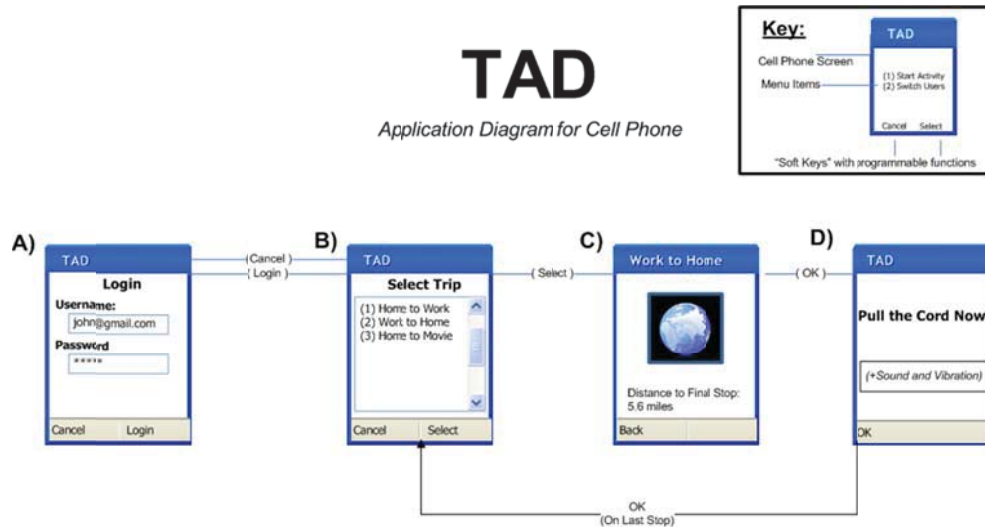


**Figure 54 - New transit trip itineraries can be created for the TAD mobile app user via the TAD website**

If the user must transfer from one route to another, multiple segments of the trip, each with separate boarding and destination stops, can be defined. Once the trip is completely planned, the caregiver or travel trainer enters a text description of the trip (e.g., Home to Work) and saves the trip to the TAD database.

To keep the TAD database updated with the most recent transit data, we created a desktop utility application that can import transit data in the General Transit Feed Specification (GTFS) format [167]. Since over 200 transit agencies in the United States share their bus routes, bus stops, and schedule data in GTFS format, TAD would be widely deployable to many different cities [168].

To provide feature #2 of real-time alerts to the transit rider on the GPS-enabled mobile phone, we implemented a user interface capable of visual, audio, and tactile interaction with the transit rider. The user interface shown to the rider is shown in Figure 55.

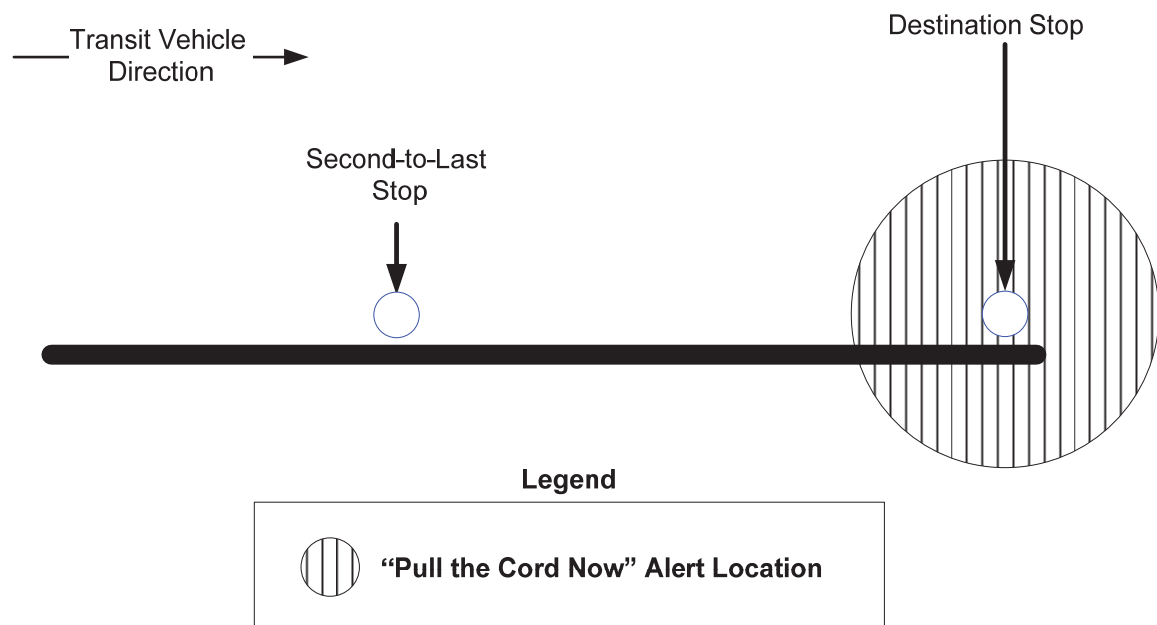


**Figure 55 - Travel Assistance Device mobile app interface that alerts the rider when to exit the bus**

Figure 55 A) shows the initial login screen given to the user on the first application startup. Once logged in, the user is shown a list of personal trips planned via the TAD website (Figure 55 B)) on subsequent mobile app startups. Planned trips for the currently logged-in user are pulled from the TAD database on each application startup. After choosing a trip to travel, the user is shown a simple screen that has a distance count-down when approaching the destination bus stop (Figure 55 C)). If the phone loses a GPS signal, a red circle with a line through it appears over the globe image to alert the user that TAD cannot provide them with real-time navigation instructions. If the phone has an active GPS signal lock, then the globe appears as shown in Figure 55 C) throughout the trip.



During the design of the real-time navigation feature of the TAD mobile app, we consulted Mark Sheppard, professional travel trainer for Hillsborough Area Rapid Transit (HART), and other members from the Association for Travel Instruction, as well as existing literature on real-time navigation instructions for individuals with disabilities. One past study found that auditory alerts are both the most preferred form of real-time navigation prompts for individuals with intellectual disabilities as well as the most effective form of prompts [169]. Other studies on users without cognitive disabilities have produced similar results [170-172]. Based on this information, we decided to create two alerts for the user when approaching the destination stop. When the rider is approximately 300 meters away from his or her destination stop, the TAD announces a recorded audio message “Get ready” twice and the phone vibrates several times.



**Figure 56 - The initial bus stop detection algorithm for the Pull the Cord Now alert was defined by a radius surrounding the destination stop**

When the phone is approximately 160 meters from the destination stop, it announces a recorded audio message to “Pull the cord now!” and the cell phone vibrates and shows the “Pull the Cord Now!” text on the screen (Figure 55 D)). A visualization of the radius that triggers this second alert is shown in Figure 56. The phone will continue to announce this message until the user presses a button to confirm that the message was received.

LAISYC was critical to implementing features #2 real-time navigation alerts, #3 real-time location tracking, and #4 automated lost alerts, since frequent GPS sampling on the mobile phone is required. TAD uses the GPS Auto-Sleep feature to dynamically control the GPS sampling frequency. When the user is not actively traveling on the bus (i.e., the mobile app is not on screens Figure 55 C) or D)), GPS Auto-Sleep is in the normal tracking mode that increases the frequency of sampling when the user is detected as moving, and reducing sampling when the user has stopped moving based on the intervals defined earlier. However, when the user selects a transit trip and the mobile app transitions to Figure 55 C), GPS Auto-Sleep switches to the navigation mode that controls the GPS sampling frequency based on the distance to the destination stop. For TAD, we chose distance thresholds of 800 meters, 1,500 meters, and 2,000 meters that would control GPS sampling with four respective interval values:

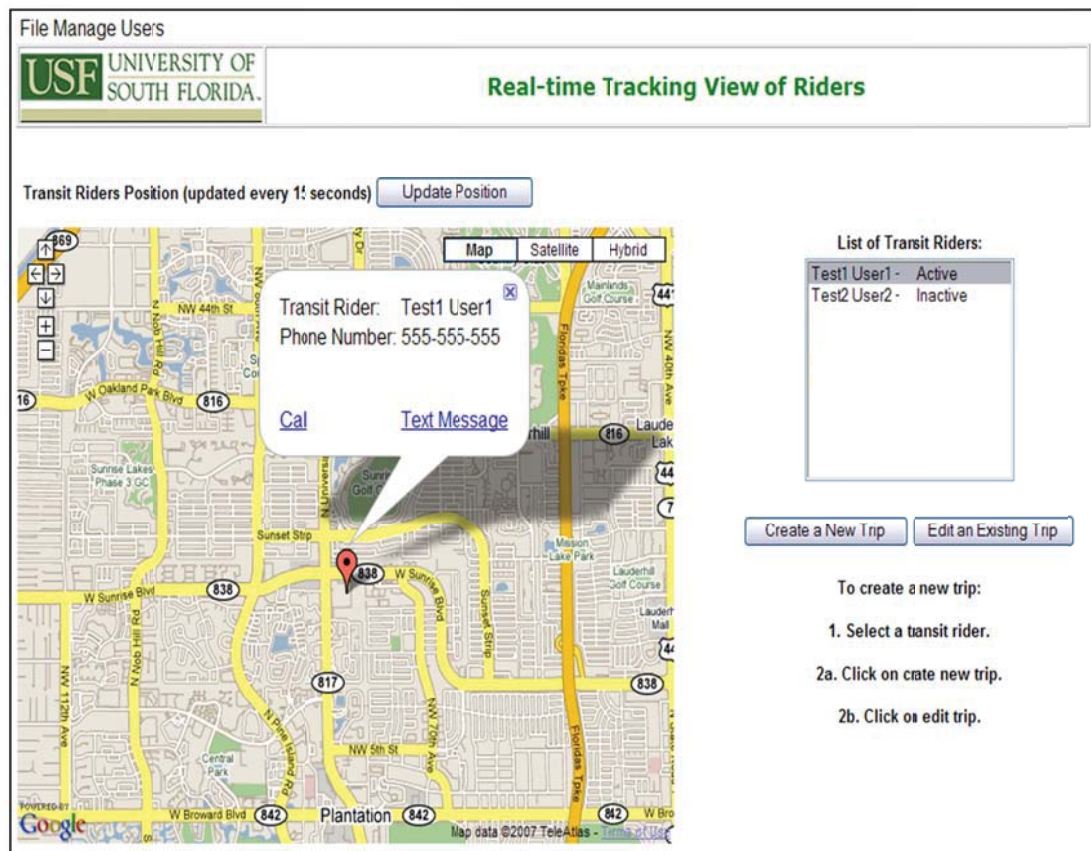
- State[0] = 1 second interval
- State[1] = 4 second interval
- State[2] = 16 second interval
- State[3] = 32 second interval

When the user was detected as being more than 2000 meters from their destination, state[3] interval of 32 seconds was used. When the user was between 1,500 meters and 2,000 meters from the destination, state[2] interval of 16 seconds was used. When the user was between 800 and 1,500 meters from the destination, state[1] interval of 4 seconds was used. When the user was closer than 800 meters to the destination, state[0] interval of 1 second was used. The TAD application then executed the comparison against the thresholds defined above to provide the “Get Ready...” and “Pull the Cord Now!” alerts. Once the user confirmed arrival at the destination, as in Figure 55 D), GPS Auto-Sleep switched out of navigation mode and back into tracking mode.

Location Data Signing was determined to be unnecessary for the TAD application, so this module was turned off.

Session Management handled all communication between the TAD mobile app and the server, using HTTPS for application data and UDP for real-time location data for the real-time tracking and automated lost alert features. The Critical Point Algorithm, Adaptive Location Data Buffering, and Location Data Encryption were all turned on for the TAD application. By using GPS Auto-Sleep to control GPS sampling frequency on the device, we were able to monitor the location of the user up to once-per-second, as the user neared the destination stop. However, real-time location updates to the server were controlled by the Critical Point Algorithm and Adaptive Location Data Buffering, and could occur less often than once per second to save battery energy and a reduction of data transfer over the cell network.

To show the real-time tracking location of the transit rider to the caregiver or travel trainer, we implemented a map-based website that showed the real-time location of the rider based on location updates from LAISYC on the phone (Figure 57).

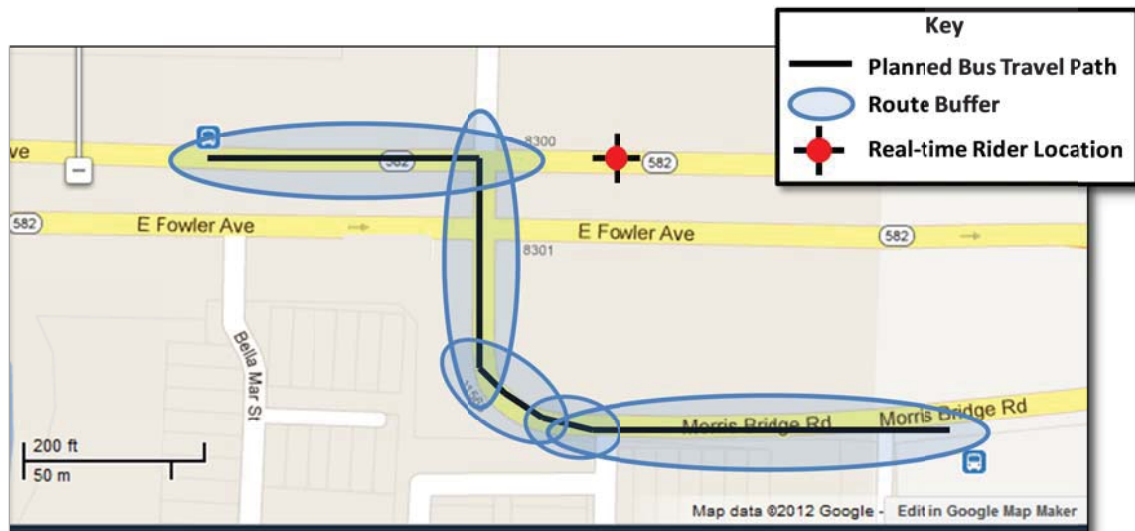


**Figure 57 - The TAD website shows real-time location updates from the LAISYC framework supporting the TAD mobile and web app**

The website was set to refresh the user's location from the real-time database every 15 seconds by default, but the website user could manually trigger an update by clicking on the Update Position button at any time.

To implement the feature #4 automated lost alerts, we used the real-time location data from LAISYC, as well as the server-side Spatial Analysis LAISYC module. Using the spatial data representation of each route from the shapes.txt file in the GTFS data, we

created a series of spatial area buffers surrounding the route in the PostGIS database. Based on the known route that the rider was traveling, the real-time location data from the rider's phone was compared to the spatial buffer surrounding the planned route (Figure 58).

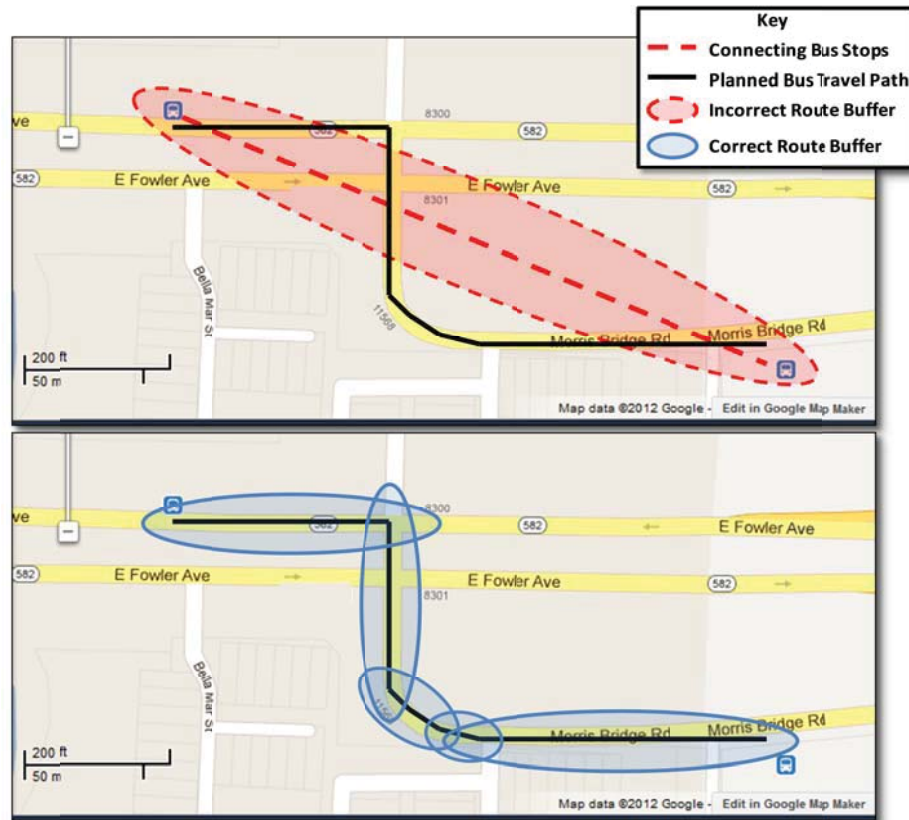


**Figure 58 - LAISYC Spatial Analysis module on the server compares the real-time location of the user against spatial buffers surrounding the rider's planned route, to determine if the user has become lost**

If the location was inside the buffer, the user was considered on-route. If the location was outside the buffer, the user was considered lost and an email and text messages was automatically sent to the caregiver and travel trainer. These comparisons between real-time location data and the route only occurred when the user had actively selected to travel via a route on the phone and the phone had entered navigation mode.

Having knowledge of the planned travel path of the bus was important, since creating a buffers based on “connecting-the-dots” between bus stops could produce false-positives

lost rider alerts. Figure 59 shows the buffers that are created when using only bus stops, versus the planned path of the bus.



**Figure 59 - The planned travel path of the bus is used to detect if the rider is lost, versus an estimated path created by connecting bus stop locations, since an estimated path can produce false-positive lost alerts**

In the top portion of Figure 59, the planned travel path of the bus is outside the buffer created by connecting the bus stop locations, which will produce a false-positive for a lost rider when the bus passes outside of the incorrect route buffer. Buffers based on the planned travel path will only generate an alert if the location of the user is far from the planned route.

Since the primary feature of TAD is the real-time notification to the user when to exit the bus, we focused on the evaluation of this feature. We conducted 50 planned transit trips

using a Sanyo 7050 mobile phone on the Sprint CDMA 1xRTT network on HART in Tampa, Florida. 38 trips were performed randomly on various stops in Tampa, Florida, while the remaining 12 trips were performed by six individuals with intellectual disabilities from the University of South Florida Successful Transition After Graduation for Exceptional Students (STAGES) program. Each trip was defined as a single boarding and exit of the transit vehicle, so TAD would provide one “Pull the cord now!” notification per trip. We defined *Ideal prompts* as prompts given between the stop prior to the destination stop and the destination stop. *Late prompts* were prompts given to the rider after the stop prior to the destination stop but would require fast reaction time by the rider to avoid missing the stop. *Early prompts* were prompts given to the user before they reached the stop prior to the destination stop. We categorized the results of the 50 test trips based on these groupings, and if the prompt was not ideal, we also analyzed why an ideal prompt did not occur. The results of these tests and subsequent analysis are shown in Table 12 for the random research team testing and Table 13 for the STAGES student testing.

**Table 12 - Field tests of the TAD app in Tampa, Florida produced ideal prompts 87% of the time at random stops**

<b>TAD Testing Conducted on Random Stops</b>	
<b>Number of Ideal Prompts</b>	<b>34</b>
<b>Number of Late Prompts</b>	
Incorrectly Geocoded Bus Stop	1
Close Proximity of Bus Stops	1
<b>Number of Times No Prompt Given</b>	
HART Service Change	1
Incorrectly Geocoded Bus Stop	1
<b>Total Number of Trips</b>	<b>38</b>



**Table 13 - Field tests of TAD with STAGES students were more challenging, primarily due to close proximity of stops near the USF campus**

Evaluation of TAD with STAGES Students	
Number of Ideal Prompts	5
Number of Early Prompts	
Received prompt while bus was stopped at 2nd-to-last bus stop	1
Number of Late Prompts	
Close Proximity of Bus Stops	2
GPS drift	1
User did not hear alert when it was first issued	1
Number of Times No Prompt Given	
Due to Lack of Connection to Wireless Carrier Location Server	1
Due to Incorrectly Geocoded Bus Stop	1
<b>Total Number of Trips</b>	<b>12</b>

Overall, in 38 of 50 trips TAD provided the “Pull the Cord Now!” prompt at the ideal place and time.



**Figure 60 - Some TAD alerts were given early or late due to incorrectly geocoded bus stops, where the actual bus stop position (marker "A") differed from the database location of the bus stop (blue bus icon)**

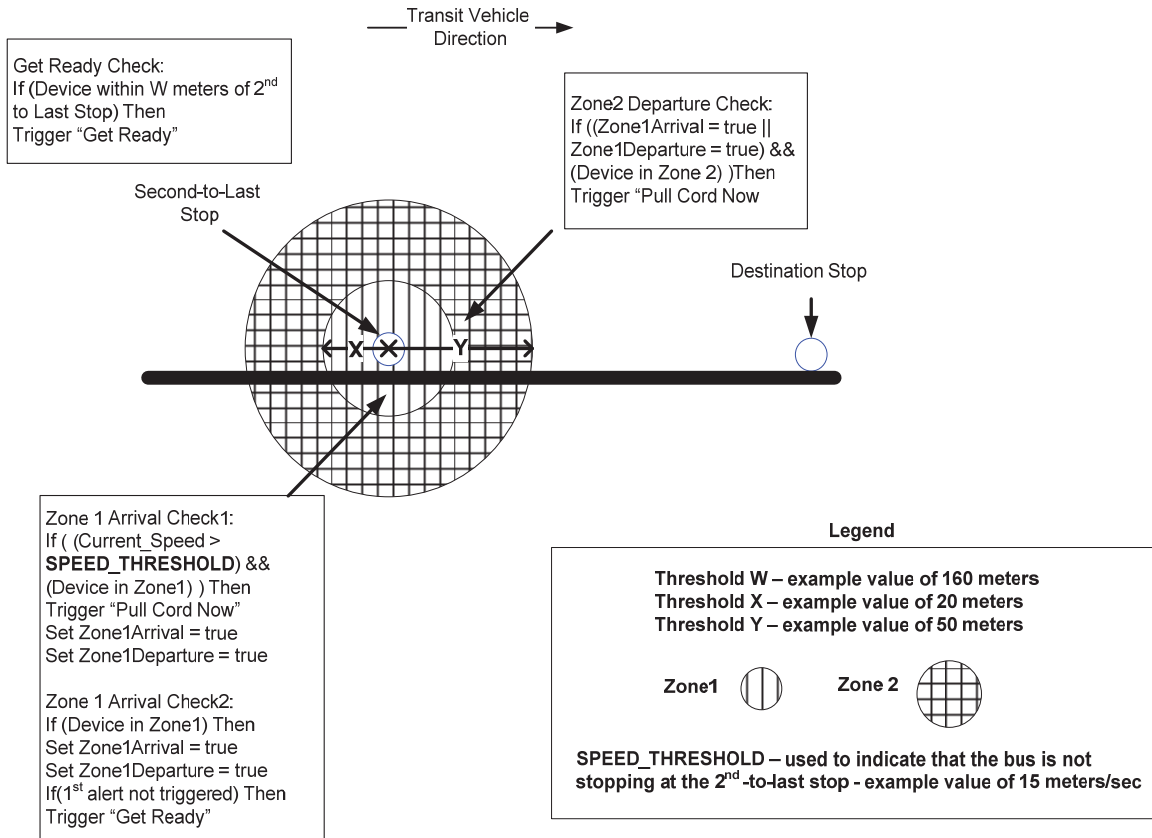


The majority of the remaining early, late, or missing prompts (8 of 12) were due to incorrectly geocoded bus stops (the database location of the bus stop did not match the actual bus stop location), or the challenge of alerting the rider at the correct time when bus stops were close together. We were able to monitor the location of each of the riders in real-time, and lost rider alerts were only issue to us when a research team member intentionally wandered outside of the route buffer.

Figure 60 shows one situation where the database location of the bus stop (the blue bus icon) did not match the true stop location (marker “A”).

Transit agencies are currently working to improve the quality of their bus stop inventories to support advanced systems such as TAD. Various emerging tools can assist agencies in this task [173-177], and the reliability of TAD and other advanced applications will be dependent on good data.

To address the remaining challenges of close bus stops and GPS drift, the research team modified the bus stop detection algorithm, so that instead of relying on a single radius surrounding the destination stop, the “Pull the Cord Now” alert was now based off of the entry into and exit of the phone from a circle surrounding the second-to-last stop (Figure 61). This design reduced both early and late alerts, since the “Pull the Cord Now” notification was given just after the user departs from the second-to-last stop. In subsequent tests with the new bus stop detection algorithm in Tampa, Florida, TAD provided the alert in the ideal location to users in 33 of 33 tests [178] (Table 14).



**Figure 61 - An improved algorithm for notifying the user when to exit the bus is based on detecting the departure from the second-to-last bus stop. [126]**

**Table 14 - The improved bus stop detection algorithm delivered ideally-timed alerts to riders in 33 of 33 tests**

<b>Evaluation of New Bus Stop Detection Algorithm</b>	
<b>Number of Ideal Prompts</b>	<b>33</b>
<b>Number of Early Prompts</b>	<b>0</b>
<b>Number of Late Prompts</b>	<b>0</b>
<b>Number of Times No Prompt Given</b>	<b>0</b>
<b>Total Number of Trips</b>	<b>33</b>

To further assess the effect of TAD on the bus riding behavior of individuals with intellectual disabilities, a research team from the Florida Mental Health Institute and the

Center for Urban Transportation Research conducted an additional study [178]. The team tested the ability of three individuals with intellectual disabilities to travel to a new location without TAD, and with TAD, for a total of 33 trials. Each of the individuals failed to both request a bus stop and exit the bus at the appropriate time when they were not carrying TAD. When they did carry TAD, each individual was able to both request the stop at the correct time, as well as exit the vehicle at the correct time. Therefore, this study concluded that the experiments provided supporting evidence that TAD was an effective tool for prompting individuals to pull the cord indicating their stop and exit the bus at the appropriate location and time [178]. The study also recommended larger scale tests to further evaluate TAD with different and more varied populations.

In 2010, USF partnered with DAJUTA, a Florida-based company, to provide TAD as a service to transit riders and transit agencies. More information about TAD as a commercial product can be found on DAJUTA's website at <http://dajuta.com/>.

In conclusion, TAD is a mobile transit navigation app that assists bus riders with intellectual disabilities by prompting them when to exit the bus, as well as tracking the rider in real-time and alerting caregivers if the rider is lost. In the most recent group of TAD field tests in Tampa, Florida, TAD provided the alert in the ideal location to transit riders in 100% (n = 33) of tests. In TAD, the GPS Auto-Sleep, Session Management, Adaptive Location Data Buffering, Critical Point Algorithm, and the Session Management modules all contribute energy savings (Need #1) that enable the phone's battery to last an entire day during high-resolution, real-time GPS tracking (Needs #2 and #3). High-resolution GPS tracking is critical to TAD for providing accurate instructions

to the transit rider when to exit the bus as well as tracking an accurate location of the traveler so that caregivers can be alerted if the rider becomes lost. The Location Data Encryption module protects the privacy of the transit rider while being tracked. The Session Management, Adaptive Location Data Buffering, and Critical Point Algorithm modules allow TAD to avoid data overage costs on phones with limited data plans, while still supporting real-time location data communication for the TAD tracking alert features. The Adaptive Location Data Buffering module prevents transit rider location data from being lost when the user is outside network coverage or is on a voice call for networks that do not support simultaneous voice and data communications. TAD was successfully implemented and tested using LAISYC on actual mobile phones without any modification to device hardware or software (Need #4).

## **CHAPTER 5: SUMMARY AND CONCLUSIONS**

### 5.1 Note to Reader

Portions of the technology presented in the future work section for GPS Auto-Sleep are protected by U.S. Provisional Patent “System and Method for Changing Positioning System Settings at Wirelessly-Obstructed Locations” by USF.

### 5.2 Summary of Problem Statement and Needs

While the exponential growth in the adoption of mobile phones provides many opportunities for new types of mobile apps, evolution in intelligent location-aware services has been limited due to several factors:

- 1) Battery energy limitations are not addressed. Many architectures have been designed without acknowledging that mobile devices have a finite energy supply, and that positioning systems such as GPS, wireless communications, and use of the CPU to execute the architecture components all have a significant impact on battery energy levels.
- 2) Cellular data transfer limitations are not addressed. Many architectures have been designed without consideration of constrained cellular network bandwidth and potential financial charges to the end-user for excessive data traffic.
- 3) Lack of integration with existing platforms on commercially-available devices (e.g., Java Micro Edition, Android). Many existing location-aware architectures

utilize custom operating systems or protocols which are not readily available on commercially-available mobile phones, and therefore cannot be widely deployed as mobile apps to existing phones.

- 4) Lack of evaluation of efficacy of location-aware architectures. Very few location-aware architectures have actually been evaluated on real mobile devices, and as a result there is little quantifiable evidence of these architectures' efficacy with real devices.

As a result of these limitations, there is a demand for a new location-aware architecture that meets following needs:

- Need #1: Intelligently manage limited device and network resources. The architecture must acknowledge that location-aware apps can deplete significant device and network resources, and the architecture must demonstrate features that conserve these resources.
- Need #2: Support real-time applications. A significant portion of the architecture must be implemented on the mobile device to allow software to immediately act upon new data in real-time and immediately interact with the mobile user.
- Need #3: Support high-precision and high-accuracy positioning systems. Positioning technologies, such as high-sensitivity assisted GPS, must be usable within the architecture to support the most innovative types of location-aware apps that require highly accurate and precise location information.
- Need #4: Is fully implementable by third party mobile app developers. The architecture must take into account the availability of application programming

interfaces (APIs) in existing cross-platform application environments such as Java ME or Android, and ensure that the architecture can be implemented on such devices.

### 5.3 Summary of Contributions

This dissertation presented LAISYC, a modular location-aware architecture for intelligent real-time mobile applications that is fully-implementable by third party mobile app developers and supports high-precision and high-accuracy positioning systems, such as GPS. LAISYC significantly improves device battery life, provides location data authenticity, ensures security of location data, and significantly reduces the amount of data transferred between the phone and server. We have designed, implemented, and successfully evaluated the following modules in real-world scenarios using actual mobile devices:

- GPS Auto-Sleep module: The GPS Auto-Sleep module saves battery energy (Need #1) when using GPS (Need #3) in real-time (Need #2), maintaining acceptable movement tracking (approximately 89% accuracy) with an approximate average doubling of battery life. We have also demonstrated a methodology for selecting the thresholds used in the algorithm based on observed GPS data, so that the algorithm can be implemented by any third party mobile app developer on any device with GPS and a Location API (Need #4).
- Location Data Signing module – Location Data Signing module adds real-time (Need #2), energy-efficient (Need #1) data authenticity to this architecture that is missing in other architectures, with an average approximate battery life decrease

of only 7%. We selected DSA as digital signature algorithm to ensure the module is fully implementable by third party application developers (Need #4).

- Session Management and Adaptive Location Data Buffering modules: The Session Management and Adaptive Location Data Buffering modules also contribute to battery life savings by providing energy-efficient (Need #1), real-time (Need #2) data communication between a mobile phone and server, increasing the average battery life for application data transfer by approximately 28% and reducing the average energy cost for location data transfer by approximately 38%. To implement these modules, we chose protocols available to third party mobile application developers (i.e., HTTP, TCP, and UDP) on Java ME and Android devices (Need #4).
- The Critical Point Algorithm module: The Critical Point Algorithm module further reduces battery energy expenditures and the amount of data transferred between the mobile phone and server (Need #1) by eliminating non-essential GPS data (an average 77% reduction) (Need #3) in real-time (Need #2), with an average doubling of battery life as the interval of time between location data transmissions is doubled. We have also demonstrated a methodology to select values for the thresholds used in the Critical Point Algorithm based on observed GPS data, therefore allowing any third party mobile application developer to implement the algorithm on any GPS-enabled mobile device with a Location API (Need #4).
- Location Data Encryption module: The Location Data Encryption module ensures the security of the location data being transferred between the mobile



device and server in real-time (Need #2), with only a slight impact on battery life (i.e., a decrease of 4.9%) (Need #1). Therefore, Location Data Encryption can be implemented by any third party mobile app developer (Need #4) using existing software libraries such as BouncyCastle [147].

The LAISYC architecture was validated in two innovative mobile apps that would not have been possible without LAISYC due to energy and data transfer constraints:

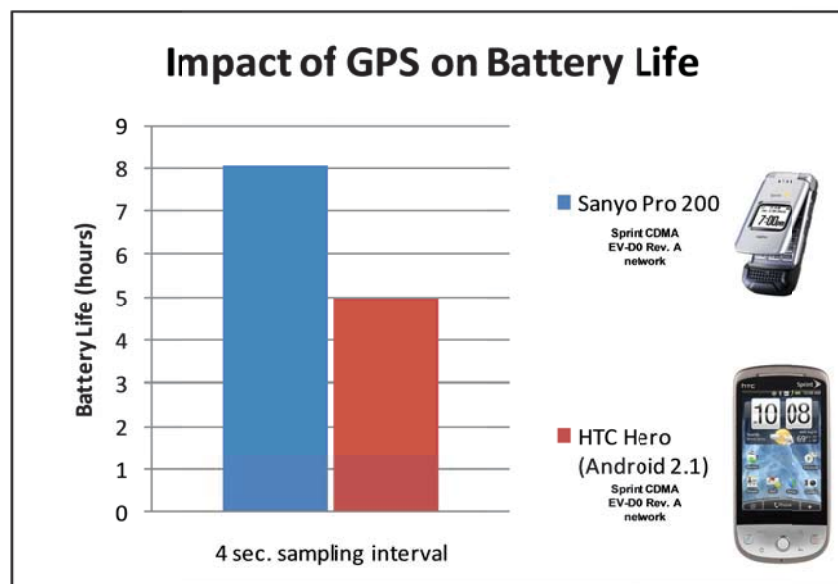
- TRAC-IT is a multi-modal travel behavior data collection mobile app that can provide simultaneous and real-time location-based services (e.g., traffic alerts). In TRAC-IT, the GPS Auto-Sleep, Session Management, Adaptive Location Data Buffering, Critical Point algorithm, and the Session Management modules all contribute energy savings (Need #1) that enable the phone's battery to last an entire day during real-time high-resolution GPS tracking (Needs #2 and #3). Real-time, high-resolution GPS tracking is critical to TRAC-IT for reconstructing detailed travel path information, including distance traveled, as well as providing predictive, personalized traffic alerts based on historical and real-time data. The Location Data Signing module allows transportation analysts to trust information that is recorded by the application, while the Location Data Encryption module protects the privacy of users' location information. The Session Management, Adaptive Location Data Buffering, and Critical Point Algorithm modules allow TRAC-IT to avoid data overage costs on phones with limited data plans, while still supporting real-time location data communication. The Adaptive Location Data Buffering module prevents tracking data from being lost when the user is

outside network coverage or is on a voice call for networks that do not support simultaneous voice and data communications. TRAC-IT was successfully implemented and tested using LAISYC on actual mobile phones without any modification to device hardware or software (Need #4).

- TAD is a mobile transit navigation app that assists bus riders with intellectual disabilities by prompting them when to exit the bus, as well as tracking the rider in real-time and alerting caregivers if the rider is lost. In the most recent group of TAD field tests in Tampa, Florida, TAD provided the alert in the ideal location to transit riders in 100% (n = 33) of tests. In TAD, the GPS Auto-Sleep, Session Management, Adaptive Location Data Buffering, Critical Point Algorithm, and the Session Management modules all contribute energy savings (Need #1) that enable the phone's battery to last an entire day during high-resolution, real-time GPS tracking (Needs #2 and #3). High-resolution GPS tracking is critical to TAD for providing accurate instructions to the transit rider when to exit the bus, as well as tracking an accurate location of the traveler so that caregivers can be alerted if the rider becomes lost. The Location Data Encryption module protects the privacy of the transit rider while they are being tracked. The Session Management, Adaptive Location Data Buffering, and Critical Point Algorithm modules allow TAD to avoid data overage costs on phones with limited data plans while still supporting real-time location data communication for the TAD tracking alert features. The Adaptive Location Data Buffering module prevents transit rider location data from being lost when the user is outside network coverage or is on a voice call for networks that do not support simultaneous voice and data

communications. TAD was successfully implemented and tested using LAISYC on actual mobile phones without any modification to device hardware or software (Need #4).

The contributions discussed above are likely to become even more important as mobile phone hardware continues to evolve. Early experiments indicate that the energy challenges related to location-aware applications discussed in this dissertation in context of Java ME are an even bigger problem on smart phones. Figure 62 shows the results of simultaneous battery life benchmarking tests using a GPS refresh interval of four seconds on a Sanyo Pro 200 with Java ME and an HTC Hero smart phone with Android 2.1.



**Figure 62 - Battery life issues related to GPS appear to be an even bigger challenge with smart phones, including Android devices**

Despite having a larger capacity battery, the HTC Hero managed a battery life of only approximately five hours, compared to the eight hours of battery life from the Sanyo Pro 200. These tests were performed without any other hardware such as the display being

activated, so the additional energy consumption is directly related to the GPS or CPU hardware required for an application to sample GPS every four seconds. Given that the HTC Hero has a processor capable of roughly 2.5 times the clock-rate of the Sanyo Pro 200 (528MHz versus 225MHz), it is no surprise that the CPU consumes additional battery energy.

Many users report having battery life problems with their smart phones [179]. This is because smart phones are used for many activities, including checking email, browsing the internet, watching videos, listening to music, etc. that all have a significant impact on battery life. Additionally, device hardware capabilities and power requirements are outpacing advancements in battery capacity at roughly twice the rate, creating a negative trend in battery performance [179]. Recent device features such as larger screens and 4G cellular network communication exacerbate the problem.

While users can attempt to budget their battery usage according to the features they want most, the applications that use the most battery energy are not always obvious. For example, recent research demonstrated that in the popular game Angry Birds, which does not provide any location-aware features, GPS was consuming around 19% of the energy spent while the application executed [38]. Further examination revealed that the advertising engine Flurry, used in Angry Birds, was responsible for turning on GPS during application execution. Furthermore, Flurry was responsible for about 45% of the total energy expended by Angry Birds. Therefore, in order to provide location-aware services for many different types of applications to smart phone users without a

noticeable impact on battery life, device-based intelligence such as LAISYC will be required.

#### 5.4 Future Work

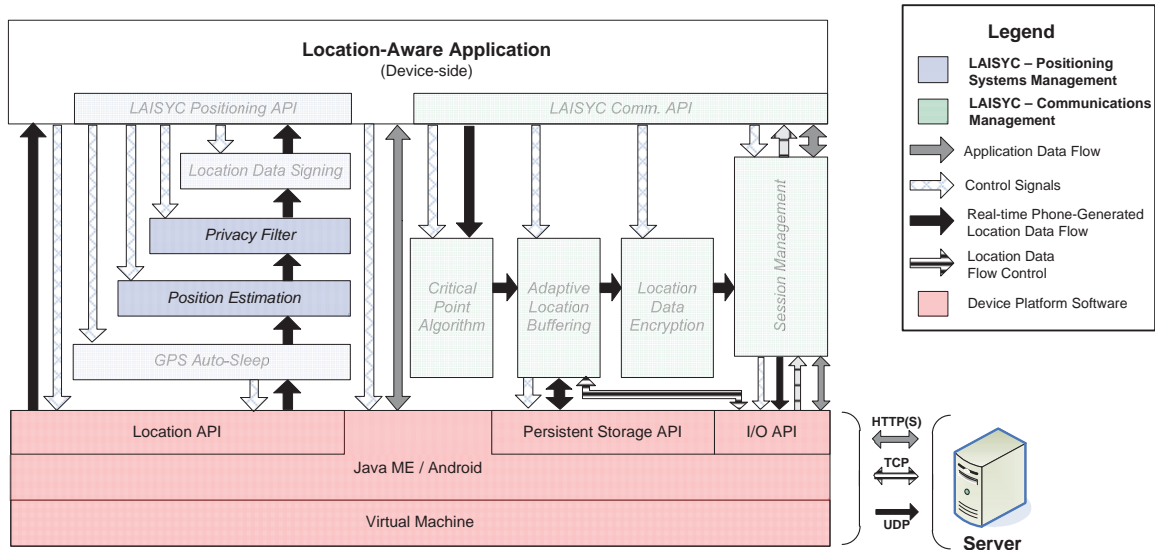
Our work with GPS-enabled mobile phones and LAISYC has provided insight into future research areas. For example, while LAISYC successfully supports real-time mobile applications given its current design, we have identified several potential areas of future work that would add new capabilities to LAISYC. We have also observed potential areas of improvement in the location-aware application development process. The following two sections outline these areas of future work.

##### 5.4.1 Location-Aware Mobile App Development

Many mobile apps that use location information have a large negative impact on mobile device battery life. One reason behind this phenomenon is that many mobile apps are tested on emulators before they are released instead of real devices, since devices are expensive. Current emulators do not model energy consumption of GPS or wireless communications, and therefore many developers do not realize the potential impact of their mobile apps until they receive feedback from their customers. There is a need for better software emulators that provide a model of energy consumption to mobile app developers so they understand the potential impacts of their application before releasing it.

## 5.4.2 Potential LAISYC Improvements

Since LAISYC is a modular framework, it allows integration of new components by simply defining input and output of location data from a module. Figure 63 shows the addition of two new modules: Privacy Filter and Position Estimation.



**Figure 63 - Future work on LAISYC can include the addition of two new modules: Privacy Filter and Position Estimation. [118]**

The following sections discuss these two new modules, as well as improvements that could be made to the existing GPS Auto-Sleep, Critical Point Algorithm, and Location Data Buffering modules.

### 5.4.2.1 GPS Auto-Sleep

While GPS Auto-Sleep currently tracks the true moving state of the user with approximately 88% average accuracy, we observed that the accuracy could potentially be improved by addressing the largest contributor to state errors: stationary GPS outliers. The most frequent errors in state transitions occur when the device is stationary and asleep in state[n] and the GPS generates an extreme outlier with a high speed value and a

distance far from the current location. When this occurs (in approximately 1-2% of all stationary GPS fixes on the Sanyo Pro 200 in our tests), the GPS Auto-Sleep snaps to rapid tracking of state[0] and must wait until the backoff timer times out before it can transition to the sleep state again.

To eliminate these false state transitions due to stationary outliers, we hypothesized that a Kalman Filter implemented at the application level could dampen the effect of these outliers on GPS Auto-Sleep. In subsequent research performed after the research presented in this dissertation, USF Masters student Isaac Taylor demonstrated that GPS Auto-Sleep accuracy could be improved from 88.40% to approximately 92% on average, without a substantial impact on tracking data through the use of Adaptive Kalman Filters [180].

The remaining source of state errors primarily occur when the user is traveling and reaches a destination location, and GPS Auto-Sleep must wait for the backoff timer to expire before gradually transitioning to the sleep state[n]. In other words, GPS Auto-Sleep still believes the user is actively traveling until this timeout expires. One way to potentially eliminate this timeout period is to have the device memorize locations previously visited by the user by tracking the user's location history. Then, when GPS Auto-Sleep recognizes that the real-time location of the user is approaching one of these historical locations, GPS Auto-Sleep could automatically transition to state[n] directly instead of waiting for the backoff timer timeout and a gradual state transition to state[n]. In order to contribute to this area, future work might evaluate possible methods for identifying, saving, and recognizing these custom user locations. Additional work with

GPS Auto-Sleep could also examine the potential for saving battery energy when the user is in a location with highly-obstructed wireless conditions. When one of these locations is recognized, the GPS interval could be increased until the user leaves the location, to reduce the impact of GPS hardware fruitlessly searching for a signal while the user is at the location.

GPS Auto-Sleep could also be used to increase the GPS sampling frequency at certain locations. This technique could be useful in context of location-aware advertising when an ad engine would like to obtain more detailed user information near advertising locations. However, the impact on device battery life would need to be carefully balanced against the value of the additional information.

One challenge of using GPS Auto-Sleep on smartphones is that many applications may be requesting GPS location information simultaneously but at different sampling frequencies. Therefore, an application that is requesting location updates every second could eliminate the energy benefit of another application using GPS Auto-Sleep. GPS Auto-Sleep could be moved into the underlying Location API in the platform to balance competing application requirements and the impact of GPS on device battery life.

However, this integration would require collaboration with device manufacturers or platform providers (e.g., Google for Android).

In our early work with GPS Auto-Sleep on Android devices, we have discovered an issue on many different devices that affects the ability of an application to request scheduled GPS updates at defined intervals. This is a problem for GPS Auto-Sleep, since it depends on the ability to request GPS updates at a specific interval for each state.



On many Android devices, when an application passes in a `minTime` parameter (i.e., interval of time between location updates) to the GPS location provider, the GPS provider typically ignores this value and proceeds to update the application via callbacks to the `LocationListener.onLocationChanged()` method every second (i.e., 1Hz update rate). A build of the Android Open-Source Project (AOSP) code 4.0.3 *Ice Cream Sandwich* on a Samsung Nexus S 4G has the same behavior of ignoring the `minTime` parameter, so the behavior is not due to an OEM modification of the platform source.

We believe we have narrowed down the problem to faulty capability reporting from native code to the `GpsLocationProvider` in the Android platform. In a custom AOSP build on the same Nexus S 4G, we hard-coded values in the `GpsLocationProvider` to indicate that the native code was not capable of handling GPS scheduling. The platform took over and properly followed the `minTime` parameter (60 seconds in this case) and delivered location updates to the app 60 seconds apart. Therefore, it seems that the native code is telling the platform that it can handle GPS scheduling, but then it does not, resulting in a 1Hz update rate no matter the `minTime` interval requested by the application.

We have worked with the Google Android team to arrive at a solution to this issue that should appear in the next Android version 4.1 *Jelly Bean*. Additional tests have been added to the Android Compatibility Test Suite to evaluate GPS scheduling compliance and the Location API documentation has been clarified to provide a strict expectation for GPS scheduling adherence [181], which should hopefully resolve this issue for Android devices version 4.1 *Jelly Bean* and above.

#### 5.4.2.2 Critical Point Algorithm

We currently use a single speed threshold to vary the angle threshold used at runtime between two values (i.e., a walk angle threshold, and car angle threshold) to filter points using the Critical Point Algorithm. Future work could examine if additional angle threshold values could be used to further decrease the number of critical points while not affecting the distance error percentage.

#### 5.4.2.3 Location Data Buffering

Currently, Location Data Buffering functions by occasionally checking in with the server after a timer expires via TCP, to ensure there is still an end-to-end connection. Instead of the current time expiration threshold, more complex evaluation functions to determine when a TCP transmission should occur are also possible. For example, the Critical Point Algorithm could be used to determine when a TCP transmission should occur, to increase the probability of Critical Points being successfully received by the server.

TCP-based checks with the server can also be utilized to increase system scalability by communicating location data flow control instructions back to the mobile device. For example, for many devices sending real-time second-by-second tracking updates to a web server, the server may eventually become overloaded with location data if enough devices are logged on simultaneously. In the subsequent TCP response for each device, the server could send a command back to the device to send fewer updates to the server until further notice. This would immediately reduce the load on the server, thereby allowing additional scalability, while providing a basic quality of service. When the number of devices logged on is reduced, the server could then send a command in the

next TCP response to each device allowing the phone to begin transmitting fixes more frequently again.

#### 5.4.2.4 Position Estimation

Position Estimation is one module that can be added to the LAISYC framework to estimate the position of the user when the raw output from a single positioning technology is unavailable or not sufficiently accurate. Existing work in position estimation by others could also be integrated into LAISYC as modules. For example, Shih-Hau et al. discuss localization techniques based on received signal strength of Wi-Fi access points and the use of an artificial neural network to infer position [83]. Beauregard presents a methodology to use artificial neural networks and GPS data to improve pedestrian navigation via a dead reckoning system [75], while Lachapelle seeks a similar goal via the combination of GPS and micro-electro-mechanical systems (MEMS) [72]. In their work on their Statistical Terminal Assisted Mobile Positioning (STAMP) system, Laoudias et al. present a statistical method based on historical position calculations to infer current position [182, 183], while Markoulidakis et al. present improvements on STAMP by using different Kalman filtering options on various input variables [184, 185].

#### 5.4.2.5 Privacy Filter

A Privacy Filter is another module that can be added to the LAISYC framework to further protect user privacy. Since the user must explicitly allow a mobile application to access their location according to the Java ME security model, the application on both the client and server is considered to be trusted by the end user. However, the privacy of the

user should be protected to ensure the trusted location-aware application only accesses user location when the user considers it appropriate and does not accidentally disclose sensitive location data. The current Java security model for the Location API has only blanket options for user approvals:

- Allow This Time
- Always Allow
- Allow Until Exit
- Never Allow

Therefore, the user must permit all location requests by the application, or the user is prompted each time the location-aware application wishes to access device location. Instead of these two extremes, there is a desire for the user to be able to define conditional approvals based on real-time information, including current location.

The Privacy Filter would allow the application to define conditional permissions for location requests, such as time limitations (e.g., requests are permitted from 9am to 5pm on Monday through Friday for business employees) or sensitive location restrictions (e.g., no requests allowed while in “private zones” near home). Using this method, the application would be protected from accidentally receiving sensitive location updates.

The Privacy Filter would also be a valuable addition on the Android platform. Currently, Android enforces only an install-time security model for application permissions. For example, if an application is going to access the user’s location, the user is only asked once when the application is installed if they would like to permit this. Once this initial permission is granted, the application can access the user’s location at any time during

execution without being required to ask the user during runtime. Therefore, since Android currently does not enforce a user-based runtime security model, the Privacy Filter module would be an important feature on Android devices.

## LIST OF REFERENCES

- [1] International Telecommunications Union (2011). "ICT Facts and Figures - The World in 2011."
- [2] CTIA - The Wireless Association. "Wireless Quick Facts - Mid-Year Figures." Accessed December 13, 2011 from <http://www.ctia.org/advocacy/research/index.cfm/aid/10323>
- [3] Stephen Blumberg, Julian Luke, Nadarajasundaram Ganesh, Michael Davern, Michel Boudreaux, and Karen Soderberg (2011). "Wireless Substitution: State-level Estimates From the National Health Interview Survey, January 2007–June 2010," U.S. Department of Health and Human Services, April 20, 2011.
- [4] Synovate. "Synovate mobile phones survey." Accessed January 31, 2010 from <http://www.synovate.com/insights/infact/issues/200909/>
- [5] Google, Inc. "Android Market." Accessed December 21, 2011 from <http://market.android.com/>
- [6] Apple, Inc. "iPhone - From the App Store." Accessed December 21, 2011 from <http://www.apple.com/iphone/from-the-app-store/>
- [7] Research In Motion Limited. "Blackberry App World." Accessed December 21, 2011 from <http://us.blackberry.com/apps-software/appworld/>
- [8] Amazon.com, Inc. "Amazon Appstore for Android." Accessed December 21, 2011 from <http://www.amazon.com/mobile-apps/b?ie=UTF8&node=2350149011>
- [9] GetJar, Inc. "GetJar." Accessed December 21, 2011 from <http://www.getjar.com/>
- [10] ABIresearch. (2011). "Android Overtakes Apple with 44% Worldwide Share of Mobile App Downloads." Accessed: December 13, 2011 from <http://www.abiresearch.com/press/3799-Android+Overtakes+Apple+with+44%25+Worldwide+Share+of+Mobile+App+Downloads>
- [11] Canalys. (2011). "App stores' direct revenue to exceed \$14 billion next year and reach close to \$37 billion by 2015." Accessed: December 13, 2011 from <http://www.canalys.com/newsroom/app-stores-direct-revenue-exceed-14-billion-next-year-and-reach-close-37-billion-2015>

- [12] Federal Communication Commission. "911 services web site." Accessed August 24 from <http://www.fcc.gov/911/enhanced>
- [13] Zhao Yilin (2002), "Standardization of mobile phone positioning for 3G systems," *Communications Magazine, IEEE*, Vol. 40 pp. 108-116.
- [14] GPS World. (2008). "GPS handset market poised for huge expansion." *GPS World*. Accessed: May 9, 2008 from <http://www.gpsworld.com/wireless/news/abi-gps-handset-market-poised-huge-expansion-3864>
- [15] D. Porcino (2001), "Location of third generation mobile devices: a comparison between terrestrial and satellite positioning systems," in *Vehicular Technology Conference, 2001. VTC 2001 Spring. IEEE VTS 53rd*, pp. 2970-2974 Vol. 4, 2001.
- [16] A. Kupper, *Location-Based Services: Fundamentals and Operation*. New York: Wiley, 2005.
- [17] Paul A. Zandbergen (2009), "Accuracy of iPhone Locations: A Comparison of Assisted GPS, WiFi and Cellular Positioning," *Transactions in GIS*, Vol. 13 pp. 5-25.
- [18] Paul A. Zandbergen and Sean J. Barbeau (2011), "Positional Accuracy of Assisted GPS Data from High-Sensitivity GPS-enabled Mobile Phones," *The Journal of Navigation*, Vol. 64 pp. 381-399.
- [19] Bob Richton, Giovanni Vannucci, and Stephen Wilkus, "Assisted GPS for Wireless Phone Location — Technology and Standards - Next Generation Wireless Networks." Vol. 598, S. Tekinay, Ed., ed: Springer Netherlands, 2002, pp. 129-155.
- [20] Jagdish Rebello. "Four Out of Five Cell Phones to Integrate GPS by End of 2011." Accessed December 13, 2011 from <http://www.isuppli.com/Mobile-and-Wireless-Communications/News/Pages/Four-out-of-Five-Cell-Phones-to-Integrate-GPS-by-End-of-2011.aspx>
- [21] Google, Inc. "Android." Accessed January 9, 2012 from <http://www.android.com/>
- [22] Sean J. Barbeau, Miguel A. Labrador, Philip L. Winters, Rafael Pérez, and Nevine Labib Georggi (2008), "Location API 2.0 for J2ME - A new standard in location for Java-enabled mobile phones," *Computer Communications*, Vol. 31 pp. 1091-1103.
- [23] Sun Microsystems, Inc., "Java Specification Request (JSR) 179: Location API for J2ME," ed, 2007.
- [24] Sun Microsystems, Inc., "Java Specification Request (JSR) 293: Location API 2.0," ed, 2007.

- [25] Motorola, Inc. "i860." Accessed June 1, 2012 from <http://developer.motorola.com/products/handsets-other/i860/>
- [26] Jeff Sharkley (2009), "Coding for Life--Battery Life, That Is," in *Google I/O 2009*, San Francisco, CA, May 27, 2009.
- [27] Robert Mayo and Parthasarathy Ranganathan, "Energy Consumption in Mobile Devices: Why Future Systems Need Requirements-Aware Energy Scale-Down - Power-Aware Computer Systems." Vol. 3164, B. Falsafi and T. VijayKumar, Eds., ed: Springer Berlin / Heidelberg, 2005, pp. 301-463.
- [28] Gerard Bosch Creus and Mika Kuulusa, *Optimizing Mobile Software with Built-in Power Profiling*: Springer, 2007.
- [29] Aaron Carroll and Gernot Heiser, "An analysis of power consumption in a smartphone," presented at the Proceedings of the 2010 USENIX conference on USENIX annual technical conference, Boston, MA, 2010.
- [30] Aqeel Mahesri and Vibhore Vardhan, "Power Consumption Breakdown on a Modern Laptop - Power-Aware Computer Systems." Vol. 3471, B. Falsafi and T. VijayKumar, Eds., ed: Springer Berlin / Heidelberg, 2005, pp. 165-180.
- [31] Rajesh Palit, Ajit Singh, and Kshirasagar Naik, "Modeling the energy cost of applications on portable wireless devices," presented at the Proceedings of the 11th international symposium on Modeling, analysis and simulation of wireless and mobile systems, Vancouver, British Columbia, Canada, 2008.
- [32] T. Farrell, R. Lange, and K. Rothermel (2007), "Energy-efficient Tracking of Mobile Objects with Early Distance-based Reporting," in *Mobile and Ubiquitous Systems: Networking & Services, 2007. MobiQuitous 2007. Fourth Annual International Conference on*, pp. 1-8, 6-10 Aug. 2007.
- [33] Mikkel Baun Kjaergaard, Jakob Langdal, Torben Godsk, and Thomas Toftkjaer, "EnTracked: energy-efficient robust position tracking for mobile devices," presented at the Proceedings of the 7th international conference on Mobile systems, applications, and services, Krakow, Poland, 2009.
- [34] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P. Dick, Zhuoqing Morley Mao, and Lei Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," presented at the Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, Scottsdale, Arizona, USA, 2010.
- [35] J. Eberle and G. P. Perrucci (2011), "Energy measurements campaign for positioning methods on State-of-the-Art smartphones," in *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, pp. 937-941, 9-12 Jan. 2011.



- [36] M. Kjaergaard (2012), "Minimizing the Power Consumption of Location-Based Services on Mobile Phones," *Pervasive Computing, IEEE*, Vol. 11 pp. 67-73.
- [37] Mikkel Baun Kjaergaard, Jakob Langdal, Torben Godsk, and Thomas Toftkjaer, "Demonstrating EnTracked a system for energy-efficient position tracking for mobile devices," presented at the Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing, Copenhagen, Denmark, 2010.
- [38] Abhinav Pathak, Y. Charlie Hu, and Ming Zhang (2012), "Fine Grained Energy Accounting on Smartphones with Eprof," in *EuroSys'12*, Bern, Switzerland, April 10-13, 2012.
- [39] Michael Matthews, Kenn Gold, and Peter Macdoran (March 1, 2006). "Testing the Limits of Power - A Methodology for Measuring the Power Consumption of Indoor-Outdoor Tracking GPS Receivers." *InsideGNSS*, Vol. 1, pp. 34-39.
- [40] Internet Engineering Task Force, "Request for Comments (RFC) 2616 - Hypertext Transfer Protocol -- HTTP/1.1," ed, 1999.
- [41] Internet Engineering Task Force, "Request for Comments (RFC) 768 - User Datagram Protocol," ed, 1980.
- [42] Internet Engineering Task Force, "Request for Comments (RFC) 793 – Transmission Control Protocol - DARPA Internet Program Protocol Specification," ed, 1981.
- [43] Google, Inc. "My Tracks." Accessed December 16, 2011 from <http://mytracks.appspot.com/>
- [44] Google, Inc. "Google Maps." Accessed December 16, 2011 from <http://maps.google.com/>
- [45] Google, Inc. "Google Maps Navigation (Beta)." Accessed December 16, 2011 from <http://www.google.com/mobile/navigation/>
- [46] Telenav. "Telenav GPS Navigator." Accessed December 16, 2011 from <http://www.telenav.com/products/tn/>
- [47] INRIX. "INRIX - Go Anywhere." Accessed December 16, 2011 from <http://www.inrix.com/>
- [48] Foursquare Labs, Inc. "foursquare." Accessed December 21, 2011 from <http://foursquare.com/>
- [49] Facebook, Inc. "Facebook." Accessed December 21, 2011 from <http://www.facebook.com/>
- [50] Google, Inc. "Latitude." Accessed December 21, 2011 from [www.google.com/latitude](http://www.google.com/latitude)

- [51] Location Labs. "AT&T FamilyMap." Accessed January 2012 from <http://familymap.wireless.att.com/finder-att-family/welcome.htm>
- [52] Sprint. "Sprint Family Locator." Accessed January 9, 2012 from <http://sfl.sprintpcs.com/finder-sprint-family/signIn.htm?EndTrialNotification=true&adcode=Main&ECID=vanity:familylocator>
- [53] Inc. WHERE. "WHERE." Accessed December 16, 2011 from <http://where.com/locations/dhvrpcb8hy18/places>
- [54] Poynt Corporation. "Poynt." Accessed January 9, 2012 from <http://www.poynt.com/>
- [55] Earth Networks. "WeatherBug." Accessed January 9, 2012 from <http://weather.weatherbug.com/>
- [56] Apple, Inc. "Find My iPhone." Accessed January 9, 2012 from <http://itunes.apple.com/us/app/find-my-iphone/id376101648?mt=8>
- [57] AlienmanFC6. "Where's My Droid." Accessed January 9, 2012 from <http://wheresmydroid.com/>
- [58] Stuart Barnes (2003), "Location-Based Services: The State of the Art," *e-Service Journal*, Vol. 2 pp. 59-70.
- [59] Bharat Rao and Louis Minakakis (2003), "Evolution of mobile location-based services," *Commun. ACM*, Vol. 46 pp. 61-65.
- [60] M. Sunay, "Evaluation of Location Determination Technologies Towards Satisfying the FCC E-911 Ruling - Next Generation Wireless Networks." Vol. 598, S. Tekinay, Ed., ed: Springer Netherlands, 2002, pp. 157-192.
- [61] S. Soliman, P. Agashe, I. Fernandez, A. Vayanos, P. Gaal, and M. Oljaca (2000), "gpsOne: a hybrid position location system," in *Spread Spectrum Techniques and Applications, 2000 IEEE Sixth International Symposium on*, pp. 330-335 Vol. 1, Sep 2000.
- [62] J. Ashjaee "GPS: The Challenge of a Single Chip." *GPS World*, Vol. 12, p. 24. Accessed at: <http://www.gpsworld.com/gnss-system/receiver-design/gps-the-challenge-a-single-chip-4218>
- [63] Richard B. Langley (2001), "Satellite Navigation: GPS Modernization and R&D in the Academic Sector," in *National Sector Team for Space Annual Meeting*, St-Hubert, QC, July 3, 2001.
- [64] George Chia Liu (2004), "GPS RTK positioning via Internet-based 3G CDMA2000/1X wireless technology," *GPS Solutions*, Vol. 7 pp. 222-229.

- [65] Frost & Sullivan. (2008). *E911 and LBS - Addressing the New Location Accuracy Gap*.
- [66] F. van Diggelen "Indoor GPS: The no-chip challenge." *GPS World*, Vol. 12, p. 50. Accessed at: <http://sfx.fcla.edu/usf?sid=google>
- [67] F. van Diggelen and Charles Abraham, "Indoor GPS Technology," presented at the CTIA Wireless Conference, Dallas, TX, 2001.
- [68] F. van Diggelen (2002), "Indoor GPS theory & implementation," in *Position Location and Navigation Symposium, 2002 IEEE*, pp. 240-247, 2002.
- [69] F. van Diggelen. (2009). "The Smartphone Revolution." *GPS World*. Accessed: December 1, 2009 from <http://www.gpsworld.com/wireless/smartphone-revolution-9183>
- [70] Larry D. Vittorini and Brent Robinson (November 1, 2003). "Optimizing Indoor GPS Performance - Receiver Frequency Standards." *GPS World*, Vol. 14, pp. 40-48.
- [71] Gérard Lachapelle (2004), "GNSS Indoor Location Technologies," in *The 2004 International Symposium on GNSS/GPS*, Sydney, Australia, December 6, 2004.
- [72] Gérard Lachapelle (2007), "Pedestrian navigation with high sensitivity GPS receivers and MEMS," *Personal and Ubiquitous Computing*, Vol. 11 pp. 481-488.
- [73] Jason Zhang, Kefei Zhang, and Ron Grenfell (2004), "On the relativistic Doppler Effects and high accuracy velocity determination using GPS," in *The 2004 International Symposium on GNSS/GPS*, Sydney, Australia, December 6, 2004.
- [74] Jason Zhang, Kefei Zhang, Ron Grenfell, and Rod Deakin (2006), "Short Note: On the Relativistic Doppler Effect for Precise Velocity Determination using GPS," *Journal of Geodesy*, Vol. 80 pp. 104-110.
- [75] Stephane Beauregard (2006), "A Helmet-Mounted Pedestrian Dead Reckoning System," *Applied Wearable Computing (IFAWC), 2006 3rd International Forum on*, pp. 1-11.
- [76] Javier DeSalas and F. van Diggelen (February 16, 2011). "Single-Shot Position - Cell Phone Location without Ephemeris." *GPS World*, Vol. 22, p. 28.
- [77] W. Ballantyne, Gregory Turetzky, Gary Slimak, and John Shewfelt "PowerDown-Achieving Low Energy-Per-Fix in Cell Phones." *GPS World*, Vol. 17, p. 24. Accessed at: <http://www.gpsworld.com/gps/powerdown-achieving-low-energy-per-fix-cell-phones-1091>

- [78] Henrik Blunck, Mikkel Kjærgaard, and Thomas Toftegaard, "Sensing and Classifying Impairments of GPS Reception on Mobile Devices - Pervasive Computing." Vol. 6696, K. Lyons, J. Hightower, and E. Huang, Eds., ed: Springer Berlin / Heidelberg, 2011, pp. 350-367.
- [79] Hassan A. Karimi, "Indoor Navigation - Universal Navigation on Smartphones," ed: Springer US, 2011, pp. 59-73.
- [80] William Kearns, James Fozard, Vilis Nams, and Jeffrey Craighead (2011), "Wireless telesurveillance system for detecting dementia," *Gerontechnology*, Vol. 10,p. 90.
- [81] Casas Roberto, David Cuartielles, Alvaro Marco, Hector J. Gracia, and Jorge Falco (2007), "Hidden Issues in Deploying an Indoor Location System," *IEEE Pervasive Computing*, Vol. 6 pp. 62-69.
- [82] Mikkel Baun Kjaergaard, Georg Treu, Peter Ruppel, and Axel Küpper, "Efficient indoor proximity and separation detection for location fingerprinting," presented at the Proceedings of the 1st international conference on MOBILE Wireless MiddleWARE, Operating Systems, and Applications, Innsbruck, Austria, 2007.
- [83] Fang Shih-Hau and Lin Tsung-Nan (2008), "Indoor Location System Based on Discriminant-Adaptive Neural Network in IEEE 802.11 Environments," *Neural Networks, IEEE Transactions on*, Vol. 19 pp. 1973-1978.
- [84] J. Hightower and G. Borriello (2001), "Location systems for ubiquitous computing," *Computer*, Vol. 34 pp. 57-66.
- [85] J. Hightower and G. Borriello (2001). "A Survey and Taxonomy of Location Systems for Ubiquitous Computing," Technical Report UW-CSE 01-08-03, August 24, 2001.
- [86] M. Mintz-Habib, A. Rawat, H. Schulzrinne, and X. Wu (2005), "A VoIP emergency services architecture and prototype," in *Computer Communications and Networks, 2005. ICCCN 2005. Proceedings. 14th International Conference on*, pp. 523-528, 17-19 Oct. 2005.
- [87] R. José and N. Davies, "Scalable and Flexible Location-Based Services for Ubiquitous Information Access - Handheld and Ubiquitous Computing." Vol. 1707, H.-W. Gellersen, Ed., ed: Springer Berlin / Heidelberg, 1999, pp. 52-66.
- [88] Internet Engineering Task Force, "Request for Comments (RFC) 2608 - Service Location Protocol, Version 2," ed, 1999.
- [89] Maximilian Zundt, Girija Deo, Mirko Naumann, and Markus Ludwig (2005), "Realizing Peer-to-Peer Location-Based Services in Mobile Networks," in *2nd Workshop on Positioning, Navigation, and Communication*, pp. 175-182, March 17, 2005.

- [90] J. Taheri and A. Y. Zomaya (2004), "The Use of a Hopfield Neural Network in Solving the Mobility Management Problem," in *Pervasive Services, 2004. ICPS 2004. IEEE/ACS International Conference on*, pp. 141-150, 19-23 July 2004.
- [91] M. Spanoudakis, A. Batistakis, I. Priggouris, A. Ioannidis, S. Hadjiefthymiades, and L. Merakos (2003), "Extensible platform for location based services provisioning," in *Web Information Systems Engineering Workshops, 2003. Proceedings. Fourth International Conference on*, pp. 72-79, 13 Dec. 2003.
- [92] A. Kupper, G. Treu, and C. Linnhoff-Popien (2006), "TraX: a device-centric middleware framework for location-based services," *Communications Magazine, IEEE*, Vol. 44 pp. 114-120.
- [93] Axel Küpper and Georg Treu (2006), "Efficient proximity and separation detection among mobile targets for supporting location-based community services," *SIGMOBILE Mob. Comput. Commun. Rev.*, Vol. 10 pp. 1-12.
- [94] Georg Treu, Axel Küpper, and Thomas Wilder (2008), "Extending the LBS-framework TraX: Efficient proximity detection with dead reckoning," *Computer Communications*, Vol. 31 pp. 1040-1051.
- [95] World Wide Web Consortium (W3C), "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)," ed, 2007.
- [96] A. Leonhardi, U. Kubach, K. Rothermel, and A. Fritz (1999), "Virtual information towers-a metaphor for intuitive, location-aware information access in a mobile environment," in *Wearable Computers, 1999. Digest of Papers. The Third International Symposium on*, pp. 15-20, 1999.
- [97] A. Leonhardi and K. Rothermel (2002), "Architecture of a large-scale location service," in *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on*, pp. 465-466, 2002.
- [98] J. Nord (2002), "An Architecture for Location Aware Applications," in *35th Hawaii International Conference on System Sciences*, pp. 293-293, January 7, 2002.
- [99] Bing-Fei Wu, Ying-Han Chen, Chao-Jung Chen, Chih-Chung Kao, and Po-Chia Huang, "An Efficient Web-Based Tracking System through Reduction of Redundant Connections - Advances in Neural Network Research and Applications." Vol. 67, Z. Zeng and J. Wang, Eds., ed: Springer Berlin Heidelberg, 2010, pp. 671-677.
- [100] Paolo Bellavista, Antonio Corradi, and Carlo Giannelli (2008), "The PoSIM middleware for translucent and context-aware integrated management of heterogeneous positioning systems," *Computer Communications*, Vol. 31 pp. 1078-1090.

- [101] Xiaoyan Chen, Ying Chen, and Fangyan Rao, "An efficient spatial publish/subscribe system for intelligent location-based services," presented at the Proceedings of the 2nd international workshop on Distributed event-based systems, San Diego, California, 2003.
- [102] Y. Chen, X. Y. Chen, F. Y. Rao, X. L. Yu, Y. Li, and D. Liu (2004), "LORE: An infrastructure to support location-aware services," *IBM Journal of Research and Development*, Vol. 48 pp. 601-615.
- [103] Ganesh Ananthanarayanan, Maya Haridasan, Iqbal Mohomed, Doug Terry, and Chandramohan A. Thekkath, "StarTrack: a framework for enabling track-based applications," presented at the 7th international conference on Mobile systems, applications, and services, Krakow, Poland, 2009.
- [104] Internet Engineering Task Force, "Request for Comments (RFC) 3261 - SIP: Session Initiation Protocol," ed, 1999.
- [105] Gunther Pospischil, Johannes Stadler, and Igor Miladinovic (2001). "A Location-based Push Architecture using SIP," Forschungszentrum Telekommunikation Wien (FTW), Institut für Nachrichtentechnik und Hochfrequenztechnik, and Institut für Kommunikationsnetze.
- [106] Stefan Berger, Henning Schulzrinne, Stylianos Sidiroglou, and Xiaotao Wu, "Ubiquitous computing using SIP," presented at the Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video, Monterey, CA, USA, 2003.
- [107] Z. Shah, R. A. Malaney, and N. T. Dao (2007), "An Architecture for Location Tracking Using SIP," in *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pp. 124-128, 26-30 Nov. 2007.
- [108] Shah Zawar (2007), "Reliability Issues in a SIP based Location Tracking Architecture," in *International Conference on Wireless Broadband and Ultra Wideband Communications*, Sydney, Australia, p. 77, August 27, 2007.
- [109] YanHao Wu (2005), "SIP-based Location Service Provision," Magister Science Thesis, Department of Computer Science, University of the Western Cape.
- [110] Sun Microsystems, Inc., "Java Specification Request (JSR) 180: SIP API for J2ME," ed, 2011.
- [111] Sun Microsystems, Inc., "Java Specification Request (JSR) 248: Mobile Service Architecture," ed, 2008.
- [112] Sun Microsystems, Inc., "Java Specification Request (JSR) 249: Mobile Service Architecture 2," ed, 2009.



- [113] Zhang Jianjun, Zhang Gong, and Liu Ling (2007), "GeoGrid: A Scalable Location Service Network," in *Distributed Computing Systems, 2007. ICDCS '07. 27th International Conference on*, pp. 60-60, 25-27 June 2007.
- [114] A. J. Perez, M. A. Labrador, and S. J. Barbeau (2010), "G-Sense: a scalable architecture for global sensing and monitoring," *Network, IEEE*, Vol. 24 pp. 57-64.
- [115] Jakob Langdal, Kari Schougaard, Mikkel Kjærgaard, and Thomas Toftkjær, "PerPos: A Translucent Positioning Middleware Supporting Adaptation of Internal Positioning Processes - Middleware 2010." Vol. 6452, I. Gupta and C. Mascolo, Eds., ed: Springer Berlin / Heidelberg, 2010, pp. 232-251.
- [116] David Rutledge (May 1, 2010). "Innovation: Accuracy versus Precision." *GPS World*, Vol. 21, p. 90. Accessed at: <http://www.gpsworld.com/gnss-system/algorithms-methods/innovation-accuracy-versus-precision-9889>
- [117] S. J. Barbeau, M. A. Labrador, P. L. Winters, R. Perez, and N. L. Georggi (2006), "A general architecture in support of interactive, multimedia, location-based mobile applications," *Communications Magazine, IEEE*, Vol. 44 pp. 156-163.
- [118] Sean Barbeau, Rafael Perez, Miguel Labrador, Alfredo Perez, Philip Winters, and Nevine Georggi (2011), "A Location-Aware Framework for Intelligent Real-Time Mobile Applications," *Pervasive Computing, IEEE*, Vol. 10 pp. 58-67.
- [119] S. Barbeau, M. A. Labrador, A. Perez, P. Winters, N. Georggi, D. Aguilar, and R. Perez (2008), "Dynamic Management of Real-Time Location Data on GPS-Enabled Mobile Phones," in *Mobile Ubiquitous Computing, Systems, Services and Technologies, 2008. UBICOMM '08. The Second International Conference on*, pp. 343-348, Sept. 29 2008-Oct. 4 2008.
- [120] Sean J. Barbeau, Nevine L. Georggi, and Philip L. Winters (2010), "Global Positioning System Integrated with Personalized Real-Time Transit Information from Automatic Vehicle Location," *Transportation Research Record: Journal of the Transportation Research Board*, pp. 168-176.
- [121] Marcy E. Gordon, S. Barbeau, and M. A. Labrador (2011), "Location Data Signing - Protecting the Integrity and Authenticity of Positioning System Data," in *ITS World Congress*, Orlando, Florida, USA, October 16, 2011.
- [122] Sean J. Barbeau, Philip Winters, Rafael Perez, Miguel Labrador, and Nevine Georggi, "Optimizing Performance of Location-Aware Applications Using State Machines," Patent #8036679, Patent Application #60/977140, U.S. Patent and Trademark Office, 2011.
- [123] Sean J. Barbeau, Philip L. Winters, Rafael Perez, Miguel Labrador, and Nevine Georggi, "Travel Assistant Device," Patent #8138907, Patent Application #11/464079, U.S. Patent and Trademark Office, 2012.

- [124] Sean J. Barbeau, Philip L. Winters, Rafael Perez, Miguel Labrador, and Nevine Georggi, "Wireless emergency-reporting system," Patent #8045954, Patent Application #11/465931, U.S. Patent and Trademark Office, 2011.
- [125] Sean J. Barbeau, Philip L. Winters, Rafael Perez, Miguel Labrador, Nevine Georggi, and Sasha Dos-Santos, "On-Demand Emergency Notification System Using GPS-equipped Devices," Patent #8145183, Patent Application #11/534763, U.S. Patent and Trademark Office, 2012.
- [126] S. Barbeau, P. Winters, R. Perez, M. A. Labrador, N. Georggi, and Dmytro Bilov, "Method of providing a destination alert to a transit system rider," Patent #8169342, Patent Application #12/234778, U.S. Patent and Trademark Office, 2012.
- [127] Sean J. Barbeau, Philip L. Winters, Rafael Perez, Miguel Labrador, and Nevine Georggi, "System and Method for Determining Critical Points in Location Data Generated by Location-Based Applications," Patent Application #12/196673, U.S. Patent and Trademark Office, 2008.
- [128] Sean J. Barbeau, Philip L. Winters, Rafael Perez, Miguel Labrador, Nevine Georggi, and Alfredo Perez, "Architecture and Two-Layered Protocol for Real-Time Location-Aware Applications," Patent Application #13/082094, U.S. Patent and Trademark Office, 2011.
- [129] Sean J. Barbeau, Philip L. Winters, Rafael Perez, Miguel Labrador, and Nevine Georggi, "Adaptive Location Data Buffering for Location-Aware Applications," Patent Application #13/082722, U.S. Patent and Trademark Office, 2011.
- [130] S. J. Barbeau, P. L. Winters, N. L. Georggi, M. A. Labrador, and R. Perez (2010), "Travel assistance device: utilising global positioning system-enabled mobile phones to aid transit riders with special needs," *Intelligent Transport Systems, IET*, Vol. 4 pp. 12-23.
- [131] Sean J. Barbeau, Miguel A. Labrador, Nevine Labib Georggi, Philip L. Winters, and Rafael A. Perez (2009), "TRAC-IT: Software Architecture Supporting Simultaneous Travel Behavior Data Collection and Real-Time Location-Based Services for GPS-Enabled Mobile Phones," in *Transportation Research Board 88th Annual Meeting*, Washington, D.C., USA, p. 21, January 9, 2009.
- [132] Sean J. Barbeau, Nevine Labib Georggi, Philip L. Winters, Miguel A. Labrador, and Rafael A. Perez (2008), "TRAC-IT: A Smart User Interface for a Real-Time Location-Aware Multimodal Survey Tool," in *15th World Congress on Intelligent Transport Systems*, New York, New York, p. 12, November 16, 2008.
- [133] Oracle. "Glassfish >> Community." Accessed February 25, 2012 from <http://glassfish.java.net/>



- [134] Google, Inc. "Android Location API." Accessed March 17, 2012 from <http://developer.android.com/reference/android/location/package-summary.html>
- [135] Motorola, Inc. (2007). "iDEN Java ME Developer Guide."
- [136] T. Vincenty (1975), "Direct and Inverse Solutions of Geodesics on the Ellipsoid with Application of Nested Equations," *Survey Review, Ministry of Overseas Development*, Vol. XXII pp. 88-93.
- [137] R. Potlapally Nachiketh (2006), "A Study of the Energy Consumption Characteristics of Cryptographic Algorithms and Security Protocols," *IEEE Transactions on Mobile Computing*, Vol. 5 pp. 128-143.
- [138] Santi Jarusombat and Surin Kittitornkun (2006), "Digital Signature on Mobile Devices based on Location," in *Communications and Information Technologies, 2006. ISCIT '06. International Symposium on*, pp. 866-870, Oct. 18 2006-Sept. 20 2006.
- [139] Xuan Zuguang, Du Zhenjun, and Chen Rong (2009), "Comparison Research on Digital Signature Algorithms in Mobile Web Services," in *Management and Service Science, 2009. MASS '09. International Conference on*, pp. 1-4, 20-22 Sept. 2009.
- [140] Sun Microsystems, Inc., "Java Specification Request (JSR) 118: Mobile Information Device Profile 2.0," ed, 2010.
- [141] Sun Microsystems, Inc., "Java Specification Request (JSR) 172: J2ME Web Services Specification," ed, 2008.
- [142] K. K. Leung, T. E. Klein, C. F. Mooney, and M. Haner (2004), "Methods to improve TCP throughput in wireless networks with high delay variability [3G network example]," in *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, pp. 3015-3019 Vol. 4, 26-29 Sept. 2004.
- [143] Hala Elaarag (2002), "Improving TCP performance over mobile networks," *ACM Comput. Surv.*, Vol. 34 pp. 357-374.
- [144] J. W. Jung, R. Mudumbai, D. Montgomery, and Kahng Hyun-Kook (2003), "Performance evaluation of two layered mobility management using mobile IP and session initiation protocol," in *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, pp. 1190-1194 Vol. 3, 1-5 Dec. 2003.
- [145] Sun Microsystems, Inc., "Java Specification Request (JSR) 271: Mobile Information Device Profile 3," ed, 2009.
- [146] P. Prasithsangaree and P. Krishnamurthy (2003), "Analysis of energy consumption of RC4 and AES algorithms in wireless LANs," in *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, pp. 1445-1449 Vol. 3, 1-5 Dec. 2003.

- [147] The Legion of BouncyCastle. "Java Cryptography APIs." Accessed May 16, 2012 from <http://www.bouncycastle.org/java.html>
- [148] Philip L. Winters, Sean J. Barbeau, and Nevine L. Georggi (2008). "Smart Phone Application to Influence Travel Behavior (TRAC-IT Phase 3)," National Center for Transit Research.
- [149] S. Schönfelder and K.W. Axhausen, *Urban rhythms and travel behaviour: spatial and temporal phenomena of daily travel*. Surrey, England: Ashgate Publishing, Ltd., 2010.
- [150] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási (2010), "Limits of Predictability in Human Mobility," *Science*, Vol. 327 pp. 1018-1021.
- [151] Kay W. Axhausen, Andrea Zimmermann, Stefan Schönfelder, Guido Rindsfuser, and Thomas Haupt (2002), "Observing the rhythms of daily life: A six-week travel diary," *Transportation*, Vol. 29 pp. 95-124.
- [152] Robert Schlich and Kay Axhausen (2003), "Habitual travel behaviour: Evidence from a six-week travel diary," *Transportation*, Vol. 30 pp. 13-36.
- [153] Gary Langer, "Poll: Traffic in the United States," ed. ABC News: ABC News, 2005.
- [154] Marta C. Gonzalez, Cesar A. Hidalgo, and Albert-Laszlo Barabasi (2008), "Understanding individual human mobility patterns," *Nature*, Vol. 453 pp. 779-782.
- [155] T. H. Witte and A. M. Wilson (2004), "Accuracy of non-differential GPS for the determination of speed over ground," *Journal of Biomechanics*, Vol. 37 pp. 1891-1898.
- [156] W. Peukert (1897), "Über die Abhängigkeit der Kapazität von der Entladestromstärke bei Bleiakumulatoren," *Elektrotechnische Zeitschrift*, Vol. 20 pp. 20-21.
- [157] Dennis Doerffel and Suleiman Abu Sharkh (2006), "A critical review of using the Peukert equation for determining the remaining capacity of lead-acid and lithium-ion batteries," *Journal of Power Sources*, Vol. 155 pp. 395-400.
- [158] Sun Microsystems, Inc., "Java Specification Request (JSR) 224: Java API for XML-Based Web Services (JAX-WS) 2.0," ed, 2011.
- [159] Richard W. Bohannon (1997), "Comfortable and maximum walking speed of adults aged 20—79 years: reference values and determinants," *Age and Ageing*, Vol. 26 pp. 15-19.

- [160] E. Murakami, D. P. Wagner, and D. M. Neumeister (1997), "Using Global Positioning Systems and Personal Digital Assistants for Personal Travel Surveys in the United States," in *International Conference on Transport Survey Quality and Innovation*, Grainau, Germany, May 24, 1997.
- [161] E. Murakami and D. P. Wagner (1999), "Can using global positioning system (GPS) improve trip reporting?," *Transportation Research Part C: Emerging Technologies*, Vol. 7 pp. 149-165.
- [162] Timothy Forrest and David Pearson (2005), "Comparison of Trip Determination Methods in Household Travel Surveys Enhanced by a Global Positioning System," *Transportation Research Record: Journal of the Transportation Research Board*, Vol. 1917 pp. 63-71.
- [163] P. A. Gonzalez, J. S. Weinstein, S. J. Barbeau, M. A. Labrador, P. L. Winters, N. L. Georggi, and R. Perez (2010), "Automating mode detection for travel behaviour analysis by using global positioning systems enabled mobile phones and neural networks," *Intelligent Transport Systems, IET*, Vol. 4 pp. 37-49.
- [164] Narin Persad-Maharaj, Sean J. Barbeau, Miguel A. Labrador, Philip L. Winters, Rafael A. Perez, and Nevine Labib Georggi (2008), "Real-Time Travel Path Prediction Using GPS-Enabled Mobile Phones," in *15th World Congress on Intelligent Transport Systems*, New York, New York, p. 12, November 16, 2008.
- [165] National Institute on Disability and Rehabilitation Research (1997). "Survey of Income and Program Participation (SIPP)."
- [166] American Public Transportation Association (APTA) (2004). "2004 Public Transportation Factbook."
- [167] Google, Inc. "General Transit Feed Specification Reference." Accessed February 24, 2012 from <http://developers.google.com/transit/gtfs/reference>
- [168] Front Seat Management, LLC. "City-Go-Round." Accessed March 1, 2012 from <http://www.citygoround.org/agencies/>
- [169] McKay Moore Sohlberg, Stephen Fickas, Pei-Fang Hung, and Andrew Fortier (2007), "A comparison of four prompt modes for route finding for community travellers with severe cognitive impairments," *Brain Injury*, Vol. 21 pp. 531-538.
- [170] John Lee Brent, John D. Lee, Brent Caven, Steven Haake, and Timothy L. Brown (2001), "Speech-based Interaction with In-vehicle Computers: The Effect of Speech-based E-mail on Drivers' Attention to the Roadway," *Human Factors*, Vol. 43 pp. 631-640.
- [171] Marvin C. McCallum, John L. Campbell, Joel B. Richman, James L. Brown, and Emily Wiese (2004), "Speech Recognition and In-Vehicle Telematics Devices: Potential Reductions in Driver Distraction," *International Journal of Speech Technology*, Vol. 7 pp. 25-33.

- [172] Avi Parush (2005), "Speech-Based Interaction in Multitask Conditions: Impact of Prompt Modality," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, Vol. 47 pp. 591-597.
- [173] Aaron Antrim. "Transit Data Feeder." Accessed February 24, 2012 from <http://code.google.com/p/transitdatafeeder/>
- [174] Joachim Pfeiffer. "Google Transit Data Feed Open Source Project." Accessed February 24, 2012 from <http://code.google.com/p/googletransitdatafeed/>
- [175] National Rural Transit Assistance Program. "GTFS Builder." Accessed February 24, 2012 from <http://www.nationalrtap.org/public/WebApps/GTFSBuilder.aspx>
- [176] Trillium Solutions. "Trillium Transit Internet Solutions." Accessed February 24, 2012 from <http://www.trilliumtransit.com/>
- [177] Khoa Tran, Edward L. Hillsman, S. Barbeau, and M. A. Labrador (2011), "GO-Sync- A Framework to Synchronize Crowd-Sourced Mapping Contributions from Online Communities and Transit Agency Bus Stop Inventories," in *ITS World Congress*, Orlando, Florida, USA, October 16, 2011.
- [178] Arica J. Bolechala, Raymond G. Miltenberger, Sean J. Barbeau, and Marcy E. Gordon (2011), "Evaluating the Effectiveness of the Travel Assistance Device on the Bus Riding Behavior of Individuals with Disabilities," in *Transportation Research Board 90th Annual Meeting*, Washington, D.C., USA, p. 16p, January 23, 2011.
- [179] Megan Geuss. (2011). "Why Your Smartphone Battery Sucks." *PCWorld*. Accessed: May 18, 2011 from [http://www.pcworld.com/article/228189/why\\_your\\_smartphone\\_battery\\_sucks.html](http://www.pcworld.com/article/228189/why_your_smartphone_battery_sucks.html)
- [180] I. M. Taylor and M. A. Labrador (2011), "Improving the energy consumption in mobile phones by filtering noisy GPS fixes with modified Kalman filters," in *Wireless Communications and Networking Conference (WCNC), 2011 IEEE*, pp. 2006-2011, 28-31 March 2011.
- [181] S. Barbeau. "Change I25460da7: Adds command "force\_platform\_scheduling" to GPS provider." Accessed April 25, 2012 from <http://android-review.googlesource.com/#/c/34230/>
- [182] C. Laoudias, C. Desiniotis, J. Pajunen, S. Nousiainen, C. Panayiotou, and J. G. Markoulidakis (2008), "Ubiquitous Terminal Assisted Positioning Prototype," in *Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE*, pp. 3261-3266, March 31 2008-April 3 2008.

- [183] C. Laoudias, C. G. Panayiotou, C. Desiniotis, J. G. Markoulidakis, J. Pajunen, and S. Nousiainen (2008), "Part one: The Statistical Terminal Assisted Mobile Positioning methodology and architecture," *Computer Communications*, Vol. 31 pp. 1126-1137.
- [184] J. G. Markoulidakis, C. Dessiniotis, and D. Nikolaidis (2008), "Part two: Kalman filtering options for error minimization in statistical terminal assisted mobile positioning," *Computer Communications*, Vol. 31 pp. 1138-1147.
- [185] J.G. Markoulidakis (2010), "Received signal strength based mobile terminal positioning error analysis and optimization," *Computer Communications*, Vol. 33 pp. 1227-1234.



## APPENDIX A. REPRINT PERMISSIONS

Rightslink® by Copyright Clearance Center - Google Chrome  
https://s100.copyright.com/AppDispatchServlet#formTop

Copyright Clearance Center RightsLink®  
Home Create Account Help

**IEEE**  
Requesting permission to reuse content from an IEEE publication

**Title:** Dynamic Management of Real-Time Location Data on GPS-Enabled Mobile Phones  
**Conference Proceedings:** Mobile Ubiquitous Computing, Systems, Services and Technologies, 2008. UBICOMM '08. The Second International Conference on  
**Author:** Barbeau, S.; Labrador, N.A.; Perez, A.; Winters, P.; Georggi, N.; Aguilar, D.; Perez, R.  
**Publisher:** IEEE  
**Date:** Sept. 29 2008-Oct. 4 2008  
Copyright © 2008, IEEE

User ID  
Password  
 Enable Auto Login  
LOGIN  
[Forgot Password/User ID?](#)  
If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials. Already a RightsLink user or want to [learn more?](#)

**Thesis / Dissertation Reuse**

**The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:**

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*


- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications\\_standards/publications/rights/rights\\_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink.


If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

## APPENDIX A (CONTINUED)

Rightslink® by Copyright Clearance Center - Google Chrome  
https://s100.copyright.com/AppDispatchServlet#formTop

 **RightsLink®** [Home](#) [Create Account](#) [Help](#)

 **IEEE**  
Requesting permission to reuse content from an IEEE publication

**Title:** A location-aware framework for intelligent real-time mobile applications  
**Author:** Barbeau, S.J.; Perez, R.A.; Labrador, M.A.; Perez, A.J.; Winters, P.L.; Georggi, N.L.  
**Publication:** IEEE Pervasive Computing Magazine  
**Publisher:** IEEE  
**Date:** July-September 2011  
Copyright © 2011, IEEE

User ID  
Password  
 Enable Auto Login  
[LOGIN](#)  
[Forgot Password/User ID?](#)  
If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials. Already a RightsLink user or want to [learn more?](#)

**Thesis / Dissertation Reuse**

**The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:**

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

- 1) In the case of textual materia (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new colective works for resale or redistribution, please go to [http://www.ieee.org/publications\\_standards/publications/rights/rights\\_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

[BACK](#) [CLOSE WINDOW](#)

## APPENDIX A (CONTINUED)

### Barbeau, Sean

---

**From:** Marc Anderson <manderson@cambridge.org>  
**Sent:** Tuesday, March 13, 2012 4:57 PM  
**To:** Barbeau, Sean  
**Subject:** Cambridge University Press - Reprint Permission - Dissertation by Sean Barbeau - University of South Florida

Cambridge University Press  
Reprint Permission

March 13, 2012

Sean Barbeau  
4202 E. Fowler Ave. CUT100  
University of South Florida  
Tampa, FL 33620  
email = [barbeau@cutr.usf.edu](mailto:barbeau@cutr.usf.edu)  
tel = 813-974-7208

#### Reference

Positional Accuracy of Assisted GPS Data from High-Sensitivity GPS-enabled Mobile Phones Paul A. Zandbergen and Sean J. Barbeau *Journal of Navigation*, Volume 64, Issue 03 (July 2011), pp. 381-399  
Selection: Figure 1, Table 1, Figure 2, Figure 3, Figure 4, and text from Section 4 "Results and Discussion"

#### Reprint Description

Title: Dissertation: A Location-Aware Architecture Supporting Intelligent Real-Time Mobile Applications for Location-Based Services, by Sean J. Barbeau  
Institution: University of South Florida  
Format: print  
Availability: May 2, 2012

Thank you for your request which has been forwarded. The Cambridge University Press North American Branch in New York responds to permission requests involving publishers and institutions based in North America.

#### Rights/Acknowledgement

Permission is granted for nonexclusive rights to reprint figures from your journal article as described above. This permission requires full acknowledgement:

Positional Accuracy of Assisted GPS Data from High-Sensitivity GPS-enabled Mobile Phones, by Paul A. Zandbergen and Sean J. Barbeau *Journal of Navigation*, Volume 64, Issue 03 (July 2011), pp. 381-399 Copyright © 2011 The Royal Institute of Navigation. Reprinted with the permission of Cambridge University Press

This permission is subject to the following:



## APPENDIX A (CONTINUED)

- a.) the material to be reprinted must be original to you as the Cambridge University Press journal author and must not involve material obtained from a third party; any material copyrighted by or credited in the journal article to another source may require further clearance by you from the original source
- b.) you must forward to Cambridge University Press all reprint/reproduction requests that you may receive involving the journal material published by Cambridge University Press
- c.) please also inform your co-author and ensure that they do not object

Sincerely,

---

Marc P. Anderson  
Rights and Permissions Manager  
Cambridge University Press  
32 Avenue of the Americas  
New York, N.Y. 10013-2473

tel.: 212-924-3900; 212-337-5048 (direct)  
email: [manderson@cambridge.org](mailto:manderson@cambridge.org)

---

Cambridge Academic and Professional Books Cambridge Books Online Cambridge Journals Online Cambridge English Language Learning \_\_\_\_\_

From: /General Management/UK/cambridge

---

From: Permissions request <[barbeau@cutr.usf.edu](mailto:barbeau@cutr.usf.edu)>  
To: Web Email <[lnicol@cambridge.org](mailto:lnicol@cambridge.org)>  
Date: 12/03/2012 14:44  
Subject: Permission Request

3/12/2012

Sean Barbeau  
4202 E. Fowler Ave. CUT100  
University of South Florida  
Tampa, FL 33620

email = [barbeau@cutr.usf.edu](mailto:barbeau@cutr.usf.edu)  
tel = 813-974-7208

title = journal, The Journal of Navigation pubyear = 2011 book = author bookauthor = Paul A. Zandbergen and Sean J. Barbeau isbn = journalvol = Vol. 64 No. 3

details = I would like to reproduce material from a paper I authored in my Ph.D. dissertation in Computer Science and Engineering. The material I would like to reproduce in my dissertation is from:

## APPENDIX A (CONTINUED)

Paul A. Zandbergen and Sean J. Barbeau. "Positional Accuracy of Assisted GPS Data from High-Sensitivity GPS-enabled Mobile Phones," *The Journal of Navigation*, volume 64, issue 03, pp. 381-399. July 2011.  
doi:10.1017/S 037346331100005.

Specifically, in my dissertation I would like to include the data and results from experiments examining the accuracy of GPS data from mobile phones, including Figure 1, Table 1, Figure 2, Figure 3, Figure 4, and text from Section 4 "Results and Discussion".

request = reprint

language =

by = author

authorname = Sean J. Barbeau


workingtitle = A Location-Aware Architecture Supporting Intelligent Real-Time Mobile Applications for Location-Based

Services publisher = University of South Florida market = world other = format = print extract = 10 printrun = 8 pubdate =

May 2, 2012 sellprice = 0

## APPENDIX A (CONTINUED)

4/14/12 Google Maps and Google Earth Content Rules & Guidelines - Maps Help



---

### Google Maps and Google Earth Content Rules & Guidelines

Thank you for your interest in using content such as maps or satellite images from **Google Maps** or **Google Earth** (referred to in these guidelines as **"Content"**). The tool below will ask you up to four questions about the Content you plan to use and how you will use it and then display the relevant usage requirements and guidelines.

Unless mentioned in your results, Google does not need to provide you explicit permission to move forward with your project and no contact with Google is necessary so long as you follow the requirements mentioned.

Which Content are you interested in using?

- Google Maps
- Google Earth
- Street View
- SketchUp or Panoramio
- Product Logos
- Other

How do you plan to use this Content in your project?

- Print for distribution
- Print for private use
- Digital (website, mobile app, or software)
- Media (television, film, or online video)
- Other
- Not applicable - need help with using the product

What medium will you be printing our Content in?

- Advertisement
- Newspaper or Magazine Article
- Academic Paper or Book

support.google.com/maps/feedback/tatic.py?hl=en&ts=1342631&page=ts.cs

1/4

## APPENDIX A (CONTINUED)

4/14/12 Google Maps and Google Earth Content Rules & Guidelines - Maps Help

- Professional Use (i.e. Proposal/Analysis)
- Fiction or Non-Fiction Book
- Guidebook
- Item for Resale
- Other

Please review the following rules & guidelines relevant to your project based on your responses:

**Showing Attribution**

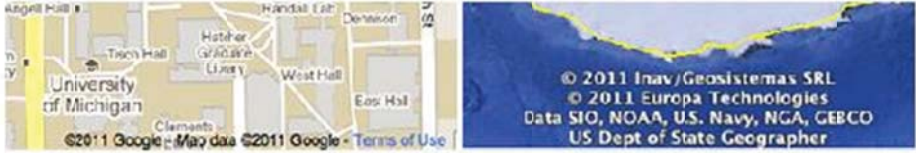
All uses of Google Maps and Google Earth and its Content must provide attribution to Google and our suppliers. Google does not approve of any use of Content without proper attribution. Depending on the region, the Content provider may be Google alone or Google and one or more 3rd party providers.

*Requirements:*

- Attribute Google (e.g. © 2011 Google) and third-party suppliers (e.g. © 2011 Tele Atlas)
- Make attribution readable to the average reader or viewer (e.g. avoid micro-sized letters)
- For Print: Display attribution within or immediately adjacent to the visual
- For Online: Attribution is automatically added within the API and cannot not obscured.
- For TV/Video: Display attribution the entire duration the Content is shown, only showing attribution briefly at the start, end, or credits is not allowed

*Where to Find the Attribution:*

- Attribution is in the bottom right of Google Maps and in the bottom center of Google Earth
- Please note suppliers of Content can change between zoom levels as well as among regions



*Additional Information:*

- Attribution is in the bottom right of Google Maps and in the bottom center of Google Earth
- For screenshots, the Google or Google Maps logo is not required but attribution must always be present. However, the reverse is not allowed - only including Google logo is not proper attribution, particularly when 3rd party suppliers were used for the Content.
- Google logos cannot be used in-line (e.g. "These maps from [Google logo].")

**Understanding Fair Use**

Your project may fall under 'fair use'. Fair use is a concept under copyright law in the United States that, generally speaking permits you to use a copyrighted work in certain ways without obtaining a

support.google.com/maps/feedback/tatic.py?hl=en&ts=1342631&page=ts.cs

2/4

## APPENDIX A (CONTINUED)

4/14/12 Google Maps and Google Earth Content Rules & Guidelines - Maps Help

license from the copyright holder. There are a variety of factors that affect whether or not your use of Content would be considered a fair use, including:

- the purpose and character of your use
- the nature of the copyrighted work
- the amount of the copyrighted material used
- the effect of your use upon the potential market for the copyrighted work

For example, there are differences between use in a for-fee service and use in a work of scholarship, or the use of a single map screenshot and the use of detailed map images for an entire country. There are similar, although generally more limited, concepts in other countries' copyright laws, including a concept known as "fair dealing" in a number of countries.

Please do not request that we interpret whether your use of Content is a fair use. Google cannot tell you if your use of Content from our products would be a fair use or would be considered fair dealing; these are legal analyses that depend on all of the specific facts of your proposed use. We suggest you speak with an attorney if you have questions regarding fair use of copyrighted works and go through our permissions wizard.

### Ensuring Print Reflects Online

When using Maps/Earth Content in print, any images used must reflect how they would look on online. For example, you are not allowed to make any changes (e.g. delete, blur, etc.) to our products that would make them look genuinely different. This includes, but is not limited to, adding clouds or other natural elements, altered user-interfaces, and modification that do not appear in the actual product.

However, Google offers a [Styled Maps API](#) which allows you to edit the colors of individual map components as well as toggle visibility for each component (i.e. change water to purple and make roads invisible).

### Keeping it 'Google'

Google Maps and Earth are geography exploration tools and are not to be used to extract Content for derivative uses that do not relate to the products. For example, tracing Google Maps or Google Earth imagery for an alternative use is not allowed. Also, whenever you wish to use our Content in print, your use should make use of a distinctive aspect of our products. Distinctive aspects of the products include:

Google Maps:

- Satellite Imagery w labels ([example](#), [not allowed](#))
- My Maps ([more info](#))
- Maps with search results ([example](#))
- Driving, walking, biking, or transit directions ([more info](#))

Google Earth (see [Media Features](#)):

- Views of terrain
- 3D buildings and trees

support.google.com/maps/faq/tatic.py?hl=en&ts=1342631&page=ets.cs 3/4



## APPENDIX A (CONTINUED)

4/14/12 Google Maps and Google Earth Content Rules & Guidelines - Maps Help

- ◆ Paths and polygons
- ◆ Historical images
- ◆ Ocean view and other layers

Any use that shows Google Maps or Google Earth in context, such as a product tutorial or news article about a feature, is OK provided it shows the entire Google Maps webpage or Google Earth application.

**Printing High-Resolution Imagery**

We cannot provide high-resolution or vector screen captures of Google Maps. For high-resolution imagery from Google Earth, you must use [Google Earth Pro](#). Stitching printed images together to create a larger map is also not permitted.

**Tell us how we're doing** - Answer five short questions about your help center experience

---

Maps - Contacting Us - Help for other Google products -  
©2012 Google  
- Home - Privacy Policy - Terms of Service

## APPENDIX A (CONTINUED)

**Barbeau, Sean**

---

**From:** Bond, Anna <ABond@theiet.org>  
**Sent:** Wednesday, April 25, 2012 11:26 AM  
**To:** Barbeau, Sean  
**Subject:** RE: Enquiry from IET Web Site - Permission to reprint material for dissertation

Dear Sean J. Barbeau,

Permission to reproduce as requested is given, provided that the source of the material including the author, title, date, and publisher is acknowledged.

A reproduction fee is not due to the IET on this occasion.

Thank you for your enquiry. Please do not hesitate to contact me should you require any further information.

Kind regards,

Anna Bond  
Editorial Assistant  
The IET

[www.theiet.org](http://www.theiet.org)

T: +44 (0)1438 767269

F: +44 (0)1438 767317

The Institution of Engineering and Technology, Michael Faraday House, Six Hills Way, Stevenage, SG1 2AY, United Kingdom

-----Original Message-----

**From:** [webmaster@theiet.org](mailto:webmaster@theiet.org) [<mailto:webmaster@theiet.org>]  
**Sent:** Wednesday, April 25, 2012 4:15 PM  
**To:** Bond, Anna  
**Subject:** Enquiry from IET Web Site - Permission to reprint material for dissertation

A visitor to the IET Web site has used your contact form to send the following enquiry.

**Name:** Sean J. Barbeau

**Subject:** Permission to reprint material for dissertation

Their email address: [barbeau@cutr.usf.edu](mailto:barbeau@cutr.usf.edu)

**Comments:**

## APPENDIX A (CONTINUED)

I am a lead author on the paper:

Sean J. Barbeau, Miguel A. Labrador, Nevine L. Georggi, Philip L. Winters, Rafael A. Perez. "The Travel Assistance Device: Utilizing GPS-enabled Mobile Phones to Aid Transit Riders with Special Needs," Institution of Engineering and Technology (IET) Intelligent Transportation Systems, 2010, Vol. 4, ss. 1, pp. 12-23. doi: 10.1049/iet-its2009.0028. © The Institution of Engineering and Technology 2010.

I would like to request written permission to reprint portions of this paper, including figures and tables, in my Ph.D. dissertation in Computer Science & Engineering at the University of South Florida. I need to receive permission within the next two weeks.

Please contact me with any questions.

Thanks,  
Sean

Sean J. Barbeau, M.S. Comp.Sci.  
Research Associate  
Center for Urban Transportation Research University of South Florida  
4202 E. Fowler Avenue, CUT100 | Tampa, FL 33620-5375  
813.974.7208 | 813.974.5168 (fax) | [barbeau@cutr.usf.edu](mailto:barbeau@cutr.usf.edu)

The Institution of Engineering and Technology is registered as a Charity in England and Wales (No. 211014) and Scotland (No. SC038698). The information transmitted is intended only for the person or entity to which it is addressed and may contain confidential and/or privileged material. Any review, retransmission, dissemination or other use of, or taking of any action in reliance upon, this information by persons or entities other than the intended recipient is prohibited. If you received this in error, please contact the sender and delete the material from any computer. The views expressed in this message are personal and not necessarily those of the IET unless explicitly stated.

---

The Institution of Engineering and Technology is registered as a Charity in England and Wales (No. 211014) and Scotland (No. SC038698). The information transmitted is intended only for the person or entity to which it is addressed and may contain confidential and/or privileged material. Any review, retransmission, dissemination or other use of, or taking of any action in reliance upon, this information by persons or entities other than the intended recipient is prohibited. If you received this in error, please contact the sender and delete the material from any computer. The views expressed in this message are personal and not necessarily those of the IET unless explicitly stated.



## APPENDIX A (CONTINUED)

**Barbeau, Sean**

---

**From:** Barber, Phyllis <PBARBER@nas.edu>  
**Sent:** Wednesday, April 25, 2012 2:28 PM  
**To:** Barbeau, Sean  
**Subject:** RE: Permission to reprint portion of TRR paper I authored in my dissertation

Dear Mr. Barbeau:

The papers mentioned in your e-mail below were not published by the Transportation Research Board; therefore, permission to reprint these papers in your dissertation is not needed from TRB. Copyright belongs to the authors.

However, because each paper may have incorporated comments from the TRB peer review, we ask that the authors as a professional courtesy acknowledge that the papers were peer-reviewed by TRB and presented at the TRB annual meeting.

Please let me know if you have any additional questions.

---

**From:** Barbeau, Sean [<mailto:barbeau@ctr.usf.edu>]  
**Sent:** Wednesday, April 25, 2012 11:32 AM  
**To:** Barber, Phyllis  
**Cc:** Barbeau, Sean  
**Subject:** RE: Permission to reprint portion of TRR paper I authored in my dissertation

Phyllis,

I have authored several other papers that have been included in the TRB conference proceedings overlap the contents of my dissertation. My initial impression was that TRB did not own the copyright on these papers, and therefore I did not need to request permission.

However, after talking to some folks here, I'd like to err on the side of caution and request reprint permission for material for these papers as well, including:

- Sean J. Barbeau, Nevine L. Georggi, Philip L. Winters, Marcy E. Gordon. "From Idealism to Realism: Lessons Learned from Development of Standards-Based Software for Advanced Public Transportation Systems," Proceedings of the National Academy of Sciences' Transportation Research Board 90th Annual Meeting, January 24, 2011. Paper #11-2254.
- Arica J. Bolechala, Raymond G. Miltenberger, Sean J. Barbeau, Marcy E. Gordon. "Evaluating Effectiveness of Travel Assistance Device on Bus Riding Behavior of Individuals with Disabilities," Proceedings of the National Academy of Sciences' Transportation Research Board 90th Annual Meeting, January 24, 2011. Paper #11-1418.
- Nevine L. Georggi, Sean J. Barbeau, Marcy E. Gordon, Philip L. Winters. "Evaluating the Deployment of a Mobile Navigation Device at four Transit Agencies in Florida," Proceedings of the National Academy of Sciences' Transportation Research Board 90th Annual Meeting, January 24, 2011. Paper #11-2213.
- Sean J. Barbeau, Miguel A. Labrador, Nevine L. Georggi, Philip L. Winters, Rafael A. Perez. "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, 2009.

Thanks,

## APPENDIX A (CONTINUED)

Sean

---

**From:** Barber, Phyllis [<mailto:PBABBER@nas.edu>]  
**Sent:** Tuesday, February 28, 2012 10:12 AM  
**To:** Barbeau, Sean  
**Subject:** RE: Permission to reprint portion of TRR paper I authored in my dissertation

Dear Mr. Barbeau:

The Transportation Research Board grants permission to you to reproduce graphs and descriptions of experiments regarding energy consumption on mobile devices from your paper, "Global Positioning System Integrated with Personalized Real-Time Transit Information from Automatic Vehicle Location," coauthored with N. Georggi and P. Winters, in your Ph.D. dissertation, as identified in your request of February 25, 2012, subject to the following conditions:

1. The citation should include the following information:

From Barbeau, S., N. Georggi, and P. Winters. Global Positioning System Integrated with Personalized Real-Time Transit Information from Automatic Vehicle Location. In *Transportation Research Record: Journal of the Transportation Research Board*, No. 2143, graphs and selected portions of text. Reproduced with permission of the Transportation Research Board.

2. None of this material may be presented to imply endorsement by TRB of a product, method, practice, or policy.

Every success with your Ph.D. dissertation. Please let me know if you have any questions.

Sincerely,

Javy Awan  
Director of Publications  
Transportation Research Board

---

**From:** Barbeau, Sean [<mailto:barbeau@cutr.usf.edu>]  
**Sent:** Saturday, February 25, 2012 3:37 PM  
**To:** Barber, Phyllis  
**Subject:** Permission to reprint portion of TRR paper I authored in my dissertation

Dear Sir or Madam,

I would like to obtain written permission to reprint portions of a TRR paper I authored in my Ph.D. dissertation. Specifically, I'd like to use the graphs and descriptions of experiments regarding energy consumption on mobile devices. The publication is:

Sean J. Barbeau, Nevine L. Georggi, Philip L. Winters. "Global Positioning System Integrated with Personalized Real-Time Transit Information from Automatic Vehicle Location," *Transportation Research Record: Journal of the Transportation Research Board*, Transit 2010 Vol 1, No. 2143, pp. 168-176, October 2010. DOI 10.3141/2143-21.  
<http://trb.metapress.com/content/r27623155x844065/?p=a84b0e1814da40b2ad321445c727c946&p=0>

## APPENDIX A (CONTINUED)

Please let me know how I can go about obtaining written permission.

Thanks,  
Sean

**Sean J. Barbeau, M.S. Comp.Sci.**  
Research Associate  
Center for Urban Transportation Research  
University of South Florida  
4202 E. Fowler Avenue, CUT100 | Tampa, FL 33620-5375  
813.974.7208 [2D barcode](#) | 813.974.5168 (fax) | [barbeau@cutr.usf.edu](mailto:barbeau@cutr.usf.edu)

[USF Location-Aware Information Systems Lab](#) - <http://www.locationaware.usf.edu/>

Subscribe to USF's Location-Aware Information Systems listserv to be notified by email of new research reports, industry news, etc. and to discuss location-aware technology issues and experiences.

## APPENDIX A (CONTINUED)

**Barbeau, Sean**

---

**From:** Charlie Tennyson <CTennyson@itsa.org>  
**Sent:** Monday, April 30, 2012 9:34 AM  
**To:** Barbeau, Sean  
**Subject:** RE: Permission to reprint material from authored papers from ITS World Congress proceedings

Good morning Sean,

As the original author, we have absolutely no problem with you reprinting or reusing any of the work you've submitted to past conferences.

Thanks for contacting us about this, and good luck with your dissertation!

Charlie

Charlie Tennyson  
Member Services Manager  
Intelligent Transportation Society of America  
1100 17th St NW Ste 1200  
Washington, DC 20036  
Office: 202-721-4207

**[ITS America 22<sup>nd</sup> Annual Meeting & Exposition](#)**  
*Smart Transportation: A Future We Can Afford*  
May 22 - 23, 2012 • Gaylord National Resort and Convention Center  
National Harbor, Maryland (outside Washington, DC)  
*Mark your calendar and make plans to attend!*

ITS America is the leading advocate for technologies that improve the safety, security and efficiency of the nation's transportation system. Follow us on [Twitter](#) and join our [Facebook](#) and [Linked In](#) groups!

 Please consider the environment, and only print this e-mail if necessary.

---

**From:** Barbeau, Sean [<mailto:barbeau@ctr.usf.edu>]  
**Sent:** Wednesday, April 25, 2012 11:23 AM  
**To:** Information  
**Subject:** Permission to reprint material from authored papers from ITS World Congress proceedings

I would like to request written permission to reprint material from several papers I have authored that have appeared in the ITS World Congress conference proceedings in my Ph.D. dissertation in Computer Science & Engineering.

I need to receive permission in the next two weeks.

The papers I would like to reprint material from are:

## APPENDIX A (CONTINUED)

- Marcy Gordon, Sean J. Barbeau, Miguel Labrador. "Location Data Signing – Protecting the Integrity and Authenticity of Positioning System Data," *Proceedings of the 2011 ITS World Congress*, Orlando, FL, October 20, 2011.
- Sean J. Barbeau, Miguel A. Labrador, Philip L. Winters, Rafael Perez, Nevine Labib Georggi. "[The Travel Assistant Device: Utilizing GPS-Enabled Mobile Phones to Aid Transit Riders with Special Needs](#)," 15th World Congress on Intelligent Transportation Systems, New York, New York, November 16-20, 2008.
- Narin Persad-Maharaj, Sean J. Barbeau, Miguel A. Labrador, Philip L. Winters, Rafael Perez, Nevine Labib Georggi. "[Real-time Travel Path Prediction using GPS-enabled Mobile Phones](#)," 15th World Congress on Intelligent Transportation Systems, New York, New York, November 16-20, 2008.
- Sean J. Barbeau, Miguel A. Labrador, Philip L. Winters, Rafael Perez, Nevine Labib Georggi. "[Trac-It – A 'Smart' User Interface For A Real-Time, Location-Aware, Multimodal Transportation Survey](#)," 15th World Congress on Intelligent Transportation Systems, New York, New York, November 16-20, 2008.

Thanks,  
Sean

**Sean J. Barbeau, M.S. Comp.Sci.**  
Research Associate  
Center for Urban Transportation Research  
University of South Florida  
4202 E. Fowler Avenue, CUT100 | Tampa, FL 33620-5375  
813.974.7208 [2D barcode](#) | 813.974.5168 (fax) | [barbeau@cutr.usf.edu](mailto:barbeau@cutr.usf.edu)

[USF Location-Aware Information Systems Lab](#) - <http://www.locationaware.usf.edu/>

Subscribe to USF's Location-Aware Information Systems listserv to be notified by email of new research reports, industry news, etc. and to discuss location-aware technology issues and experiences.

## **ABOUT THE AUTHOR**

Sean J. Barbeau received his B.S. and M.S. in Computer Science from the University of South Florida (USF) and joined the research faculty of the Center for Urban Transportation Research at USF in 2004. He has served as the Principal Investigator for many research projects investigating intelligent software systems and mobile applications for GPS-enabled mobile phones. Mr. Barbeau's research interests include intelligent location-based services, lightweight data communication frameworks for mobile devices, and mobile application optimization to conserve battery life. He served as a member of the international Expert Group that developed the Java Micro Edition Location API 2.0. While a Ph.D. candidate, he produced over 40 peer-reviewed papers and conference presentations on the topics of intelligent location-based services and mobile applications. Mr. Barbeau has 6 issued U.S. patents and another 11 patents pending on location-aware technology. He is a founding faculty member of the USF Location-Aware Information Systems Laboratory.