REAL-TIME VEHICLE PERFORMANCE MONITORING WITH DATA INTEGRITY

By

Will Jenkins

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Electrical Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

October 2006

REAL-TIME VEHICLE PERFORMANCE MONITORING
WITH DATA INTEGRITY

By

Will Jenkins

Approved:

Dr. Georgios Lazarou Assistant Professor of Electrical and Computer Engineering (Major Advisor and Director of Thesis)	Dr. Joseph Picone Professor of Electrical and Computer Engineering (Committee Member)
Dr. Julie Baca Research Professor at Center for Advanced Vehicular Systems (Committee Member)	Dr. Nicholas H. Younan Professor of Electrical and Computer Engineering (Graduate Coordinator)
Roger L. King Associate Dean for Research and Graduate Studies	

Name: Will Jenkins

Date of Degree: October 2006

Institution: Mississippi State University

Major Field: Computer Engineering

Major Professor: Dr. Georgios Lazarou

Pages in Study: 46

Title of Study: REAL-TIME VEHICLE PERFORMANCE MONITORING WITH DATA INTEGRITY

Candidate for Degree of Master of Science

A cornerstone of next generation intelligent transportation systems (ITS) is a seamless integration of in-vehicle networking with existing wireless telephony infrastructure. Remote access to on-board diagnostics and performance data is a crucial requirement for ITS.

This thesis investigated the performance benefits of a data integrity buffering technique for an extensible vehicle position and performance tracking system (VPPTS). In support of this investigation, a VPPTS prototype was developed. The prototype used available technologies and interfaces to industry-standard communications channels and is a demonstration of a next-generation intelligent transportation system (ITS). The data integrity buffering technique under investigation was shown to provide quantitative improvements in successful VPPTS data transmissions. The use of this technique addressed important deficiencies in real-time data transmission for these types of systems over wireless networks.

DEDICATION

This work is dedicated to my wife and family for their never-ending support and love during my tenure at Mississippi State University.

ACKNOWLEDGMENTS

I would like to thank Dr. Georgios Lazarou for his guidance and influence throughout the development of the prototype system and the formation of this thesis. I want to thank Dr. Julie Baca for her encouragement and direction for this thesis. I also thank Dr. Joseph Picone for his advice that carried me through this project. I would also like to thank Brock Bland, Marshall Crocker, Ben Rose, Gary Weasel, and Andrew Martin for their excellent contributions towards the development of the Vehicle Position and Performance Tracking System (VPPTS). I want to thank Zach Rowland for giving me this opportunity to work at the Center for Advanced Vehicular Systems (CAVS). I also want to thank my wife, Valerie, for her love, patience, and support.

TABLE OF CONTENTS

Page

DEDICATION ii

ACKNOWEDGEMENTS iii

LIST OF TABLES vi

LIST OF FIGURES vii

CHAPTER

I. INTRODUCTION 1

1.1 Problem Statement and Motivation 3
1.2 Summary of Main Contributions 4
1.3 Organization of the thesis 4

II. VEHICLE TRACKING AND PERFORMANCE MONITORING 5

2.1 Use of GPS to Increase Accuracy 6
2.2 Use of OBD to Monitor Vehicle Performance 8
2.3 Cellular Technologies for Data Transmission 10
2.4 Data caching techniques 11

III. VPPTS PROTOTYPE 12

3.1 Initial Laptop-based Prototype 12
3.1.1 Data Collector 13
3.1.2 Data Server 15
3.2 Multiple Board Embedded System Prototype 18
3.3 Final Single Board Embedded System Prototype 22
3.3.1 Linux System 23
3.3.2 Server Enhancements 25

IV. EXPERIMENTAL RESULTS AND OBSERVATIONS 28

4.1 Laptop-based Prototype System 29
4.2 Multiple-Board Prototype 29
4.2.1 Single-threaded and data buffered directly to a file. 29

CHAPTER Page

4.2.2 Multiple-board, Single-threaded with circular buffer 29
4.3 Single-Board, Multi-threaded Linux Prototype 35

V. CONCLUSIONS AND FUTURE RESEARCH 38

5.1 Thesis Contributions 38
5.2 Future Research 40

REFERENCES 43

LIST OF TABLES

TABLE Page

1 The SAE recognizes four protocols in the J1850 standard, which define how electrical signals will propagate through the vehicles communication bus... 8

2 SAE has also defined networks for heavy-duty vehicles i.e. shuttle buses and tractors.......... 9

3 The vehicle communicator application transmits a data string at least every second. 25

4 The protocols table allows the server to translate the raw data into human readable format. 27

5 Shuttle Buses 903 and 1205 benefited from the circular buffer as indicated by this chart.......... 30

6 Shuttle Bus 1205 had difficulty transmitting all gathered data due in part to the single-threaded application. 32

7 The multi-threaded application enhanced the buffering technique for the test vehicle data strings.......... 35

LIST OF FIGURES

FIGURE Page

1: The VPPTS prototype incorporates existing technologies to provide a real-time performance monitoring system. 5

2: GPS consists of 24 satellites of which at least 5 can be seen from any point on the globe........ 6

3: GPRMC provides the minimal information from NMEA standards for GPS location........ 12

4: The BR-3 interface is used to communicate with the vehicle's on-board diagnostic system, which offers performance data to the data collector software via RS232 port. 13

5: SAE J1962 connector provides access to the diagnostic network.14

6: The data collector polls the vehicle for performance information and receives new GPS coordinates simultaneously........ 15

7: The data flows to the users applet interface after being processed by the server. 16

8: The initial database setup included bus location and performance data and route information........ 17

9: The analog gauges applet displays real-time performance parameters. 18

10: The data buffering technique ensures data integrity. 21

11: The VCOM application uses multiple threads to increase the data gathering rate that was hindered by a single thread. 24

12: Bully's Bus Tracker takes advantage of GIS and JavaScript to provide an interactive mapping application. 26

13: Analysis of transmission time during one full day for shuttle bus 1205 and shuttle bus 903. 31

14: Shuttle bus 1205 shows signs of a poor network with many consecutive timeouts, while shuttle bus 903 was hardly utilized. 34

15: Close-up on an hour in which shuttle bus 1205 shows signs of a poor network with many consecutive timeouts. 34

16: Analysis of transmission time after employing the buffering technique in a multi-threaded application. 37

17: Timeouts do not affect the application from gathering new data. 37

CHAPTER I

INTRODUCTION

The next generation intelligent transportation system (ITS) will rely heavily on several vehicle communication systems, including peer-to-peer and peer-to-base station communications [1]. Peer-to-peer communications provides an ability for information to be relayed down a highway so that a transportation system can adapt and respond to events autonomously in real-time. Seamless integration of in-vehicle networking with existing wireless telephony infrastructure is also cornerstone of next-generation ITS and is a key component of peer-to-base station communications. Such an infrastructure should allow drivers to roam between their cellular phone network and their vehicle network. Data access and synchronization should happen automatically and transparently.

This thesis investigated and addressed data integrity issues in the development of an extensible vehicle performance monitoring system that exploits data transmission capabilities of the Global System for Mobile Communication (GSM) [2] and Code-Division Multiple Access (CDMA) [3] standard, and is based on existing in-vehicle automotive standards. Additionally, it offers the capability to monitor the performance of a vehicle over the Internet. The design is extensible to large metropolitan areas in which millions of vehicles will need to be simultaneously tracked and monitored. Though many systems currently integrate position tracking and wireless networking to allow for remote position tracking, few systems provide the capability to monitor vehicle performance over the web.

The design is based on popular standards for wireless communications, i.e., General Packet Radio Service (GPRS) [4] and Evolution-Data Only (EV-DO) [5]. An in-vehicle standard for diagnostic information specified by the Society for Automotive Engineers (SAE) [6] gathers performance data. The design also exploits Global Positioning System (GPS) technology [7] to provide vehicle location. Data is concatenated and transmitted to a web server using Apache's Tomcat extensions [8] to provide Internet access via a vehicle tracking web site. The data collected from this research will form the basis for a research initiative in prediction and optimization of vehicle performance.

Many systems today combine these technologies to create a unique service. For example, OnStar [9] has grown increasingly popular as a way to deal with roadside emergencies. Homeland security applications are placing a significant demand on fleet operators to account for the location and contents of their vehicles. First responders [10] to emergencies, such as hazardous material spills or natural disasters, have a great need for rapid delivery of information about vehicle content and location, as well as real-time mapping information. Such applications have the ability to exploit next-generation wireless technology that can deliver bi-directional high-speed data connections to moving vehicles, and data warehousing applications that monitor transportation infrastructure. GPS, which began as a military application, has become a viable tool for many commercial and personal applications. One such application has been a vehicle location tracking system (VLTS) [11]. These tracking systems incorporate a GPS receiver and a wireless transceiver that allow a remote unit to track the vehicle's position.

Remote access to on-board diagnostics (OBD) and performance data is a crucial requirement for ITS. The 1990 Clean Air Act [12] required an on-board diagnostic system, which provides an early warning of malfunctions to major engine components such as emission controls. The SAE standardized the networking protocols and information parameters. The Environment Protection Agency (EPA) [13]adopted these standards and mandated these standards and practices to all light-duty automobile manufacturers in 1996. OBD-II is the SAE second-generation of these standards and practices.

Heavy-duty vehicles such as Freightliner, Champion, and Mack also incorporate an on-vehicle network to relay messages between various modules. These messages are similar to their light-duty brethren. In 1986, the SAE also developed the J1708 [14] standard for electronic communication in heavy-duty vehicles to help meet these challenges. The SAE also developed the J1939 [15] standard to compensate for speed restrictions found with J1708. These standards are widely used in the heavy-duty vehicle industry.

1.1 Problem Statement and Motivation

For any of the applications described above, data integrity presents a critical issue in the performance of these systems. Post-processing software analyzes this data for possible vehicle issues and traffic-redirecting solutions. For the analysis to be valid enough data points must be available. Bandwidth availability and signal quality of the wireless network present other problems that can hinder data integrity for any real-time monitoring system. Examples include those which rely on the performance of cellular systems in geographical areas where the signal may be lost or weak. This occurs frequently in any of the application areas for vehicle performance monitoring.

1.2 Summary of Main Contributions

The main contributions of this thesis include:

- Development of a prototype integrated vehicle position and performance tracking system (VPPTS) which provides a testbed for investigating a variety of research issues arising in real-time monitoring and data analysis;
- Use of this testbed for rigorous design and testing of buffering techniques to provide data integrity for real-time monitoring applications;
- Detailed analysis of the performance of these techniques in enhancing data integrity

1.3 Organization of the thesis

The organization of this thesis is as follows:

- Chapter 2 provides a background of the relevant technologies and related research, comparing and contrasting the research prototype system with other VLTS and VPPTS systems.
- Chapter 3 describes the implementation and testing of the VPPTS prototype, including the details of the data integrity buffering techniques.
- Chapter 4 presents an analysis of the performance of the buffering techniques in a variety of laboratory simulated and real world scenarios.
- Chapter 5 concludes and discusses future directions for this research.

CHAPTER II

VEHICLE TRACKING AND PERFORMANCE MONITORING

The VPPTS prototype used for the investigations in this thesis consists of three core underlying technologies, illustrated in Figure 1, position tracking, i.e., GPS, performance monitoring, i.e., OBD-II, and wireless communications, i.e., GPRS and EVDO. Areas of relevant research include: vehicle position tracking, using GPS and

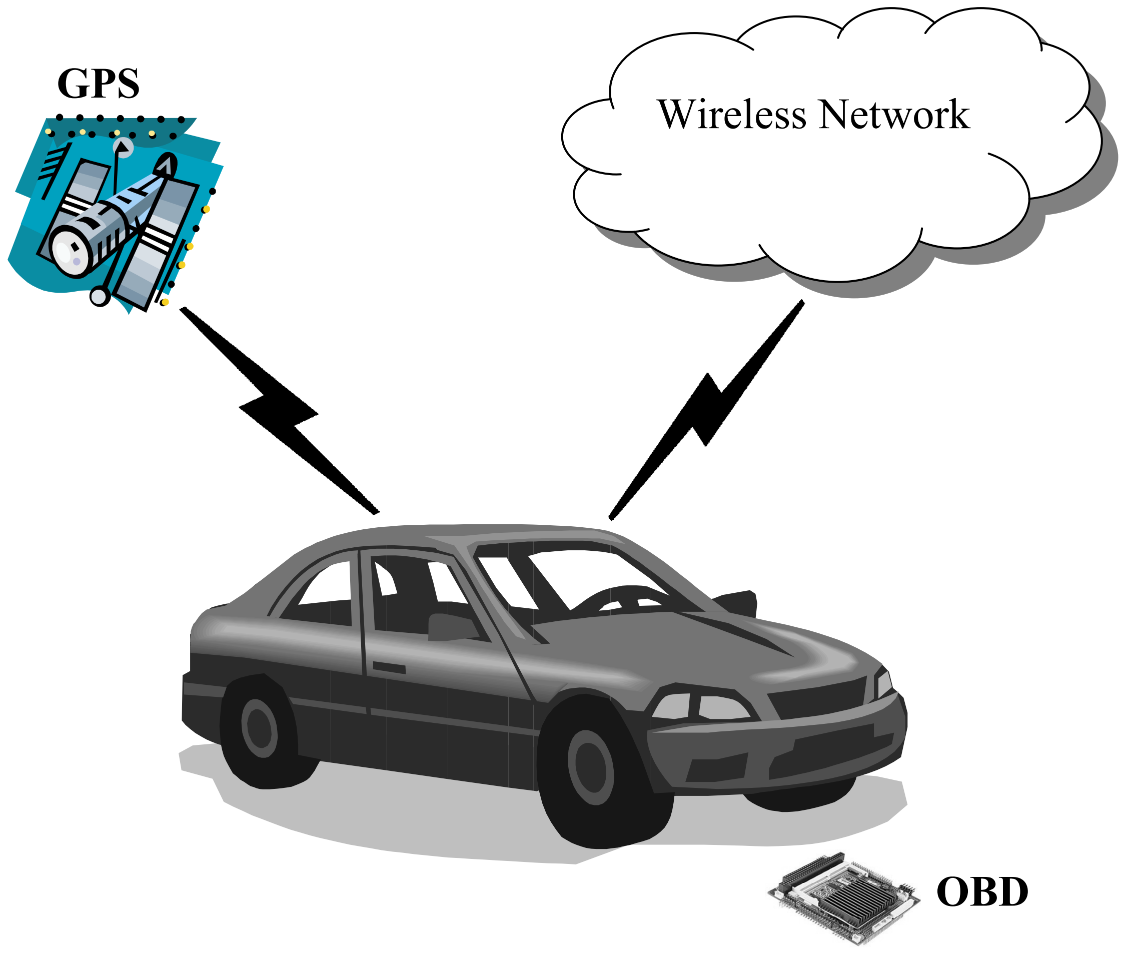


Figure 1: The VPPTS prototype incorporates existing technologies to provide a real-time performance monitoring system.

related technologies, vehicle performance monitoring using sensor data gathered from on-board diagnostics, and vehicle performance data transmission using next generation wireless technologies for real-time monitoring.

2.1 Use of GPS to Increase Accuracy

GPS provides highly accurate position information and can be used for a variety of land, sea, and air applications. GPS, developed by the U.S. Department of Defense (DoD), consists of a constellation of 24 satellites, illustrated in Figure 2, orbiting around 11,000 miles above the Earth's surface [16]. GPS was dedicated solely for military use and has recently been declassified for civilian use. To acquire GPS information, a

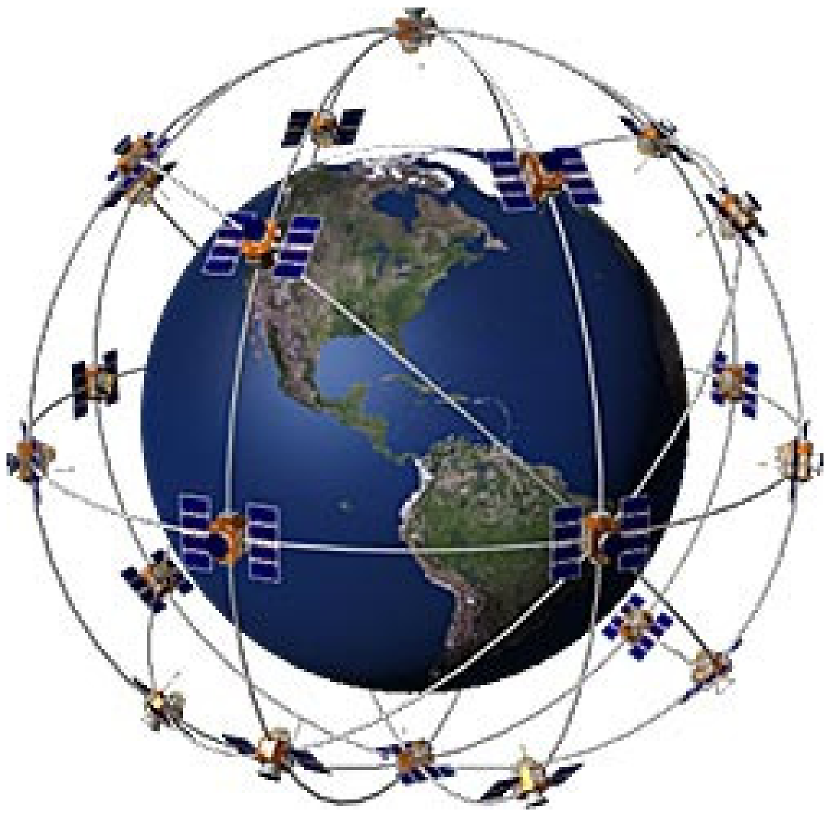


Figure 2: GPS consists of 24 satellites of which at least 5 can be seen from any point on the globe.

wireless receiver capable of the civilian L1 frequency (1575.42 MHz) is required. The GPS receiver measures distances to four or more satellites simultaneously [17]. Using triangulation the receiver can determine its latitude, longitude, and altitude [16].

Providing an accurate, precise location using GPS has been the focus of many research endeavors. Research has shown that accurate GPS signals cannot be received in many locations e.g. urban areas [18]. Also, it has been shown that GPS signals encounter atmospheric interference that can decrease its level of accuracy [19]. To provide these locations with a small degree of error, another signal can be received that is independent of GPS. Using the popular GSM network could help overcome the loss of an accurate GPS signal [20]. Several algorithms that take advantage of the GSM network have been investigated that can assist GPS-based systems and also become the prime means for location.

The use of differential-GPS (DGPS) also increases the accuracy of the GPS signal. This system incorporates the help of base stations to offset the errors found in GPS signals [21]. These base stations are equipped with highly sophisticated (expensive) GPS receivers which provide excellent accuracy and precision. Because the roaming GPS receiver and the DGPS base station are in close proximity (line-of-sight), the signal error will be the same. A roaming GPS receiver calculates its GPS signal with the DGPS base station to help eliminate these errors [22].

Another system to increase the accuracy of a GPS signal is the wide area augmentation system (WAAS) [23]. WAAS, developed by the Federal Aviation Administration (FAA) and the Department of Transportation (DoT), uses a system of satellites and approximately 25 ground stations around the United States. The ground stations transmit a correction message via its satellites to any WAAS-enabled receiver.

2.2 Use of OBD to Monitor Vehicle Performance

Since 1996, OBD systems have been incorporated into vehicles to help manufacturers meet emission standards set forth by the Clean Air Act [12] in 1990 and the Environmental Protection Agency (EPA). The Society of Automotive Engineers (SAE) developed a set of standards and practices that regulated the development of these diagnostic systems. The SAE expanded on that set to create the OBD-II standards. The EPA and the California Air Resources Board (CARB) adopted these standards in 1996 and mandated their installation in all light-duty vehicles. The OBD-II system allows for monitoring of most electrical systems on the vehicle. Monitored items include speed, rpm, ignition voltage, and coolant temperature. This system also informs an engineer when an individual cylinder has a misfire.

Table 1: The SAE recognizes four protocols in the J1850 standard, which define how electrical signals will propagate through the vehicles communication bus.

Protocol	**Signal Type**	**Manufacturer(s)**
SAE J1850 VPW	Variable Pulse Width	GM
SAE J1850 PWM	Pulse Width Modulation	Ford
ISO 9141-2	Two Serial Lines: Half-duplex (L) Full-duplex (K)	European, Asia, and Chrysler
ISO 15765 (CAN)	Single or Dual Wire Serial Lines	Most manufactures are beginning to incorporate CAN

The SAE recognizes at least four communication patterns described in Table 1. The SAE J1850 VPW [24] standard uses a variable pulse width modulation signal. It operates at 10.4k Baud with one signal wire and a ground wire. The SAE J1850 PWM [24] standard uses a pulse width modulation signal. This operates at 41.7k Baud by using a differential transmission scheme. The ISO 9141-2 standard uses two signals (K and L) [24]. One signal travels on a full-duplex wire, and the other operates on a half-duplex wire. Most communications with the OBD-II bus occur on the K signal while the L signal is required for initialization of the bus. The latest standard is based on the controller-area network (CAN) standard (ISO 15765) [25]. This network can provide up to 500 Kbit/s data rates operating on either a differential signal or single-wire configuration.

The SAE has also developed standards for heavy-duty vehicle networks shown in Table 2. In 1986, the J1708 [14] standard was created. Based on the Electronic Industries

Table 2: SAE has also defined networks for heavy-duty vehicles i.e. shuttle buses and tractors.

Protocol	**Signal Type**	**Manufacturer(s)**
SAE J1708	Modified RS-485 network	Most manufactures are incorporating both protocols and slowly phasing out J1708
SAE J1939	CAN	

Alliance (EIA) RS-485 serial data link, J1708 made a multipoint network for communication between vehicle ECUs. In 1998, with the development and popularity of the CAN standard, the SAE established the J1939 [15] set of specifications. The physical-layer specification uses a shielded twisted pair operating at 250 Kb/s. This standard is set to replace J1708 in heavy-duty vehicles. J1708 and J1939 are widely incorporated in today's heavy-duty vehicles such as shuttle buses and cargo trucks.

2.3 Cellular Technologies for Data Transmission

Several cellular technologies are now available for data transmission. Though they vary in reliability, all are vulnerable to drops in connectivity which inevitably lead to data loss at certain junctures. This characteristic common to all cellular technologies makes the focus of this thesis, data integrity, a critical issue for any vehicle performance monitoring system (VPMS) relying on these technologies. It is necessary, therefore, to review the most popular, widely used technologies, each of which were evaluated in the testbed for this thesis. These include GSM/GPRS [4] and CDMA2000/1xEV-DO [26].

GSM has become the world's fastest growing mobile communication standard. It allows for seamless and secure connectivity between networks on a global scale. Digital encoding is used for voice communication, and time division multiple access (TDMA) transmission methods provide a very efficient data rate/information content ratio [27]. While GSM is becoming the standard for person-to-person communication, the circuit-switched network limits data transmission. GPRS was developed to relieve this limitation.

GPRS is a data communication layer built over the GSM wireless transmission link [4]. GPRS uses the remaining capacity leftover from GSM voice communication [28] and has a theoretical max speed of 171.2 Kbps making it a viable choice for wireless data transfer [27]. Using a packet format for data transmission allows for full compatibility with existing Internet services such as HTTP, FTP, email, instant messaging, and more.

Because GSM is based on TDMA, much research indicates that this could hinder the overall capacity the network can handle [29]. The research in developing next generation CDMA networks could provide a solution. CDMA2000 and its data communication layer, 1xEV-DO, provide a broadband-like high data-rate network up to 3.1 Mbps downlink and 1.8 Mbps uplink for next-generation networking solutions and applications.

Many proposed systems exploit the above technologies and enhancements to provide a wireless-based location tracking application. By incorporating GPS for location and using the wireless communication network such as CDMA or GSM, a system can provide real-time location tracking of a vehicle [20], [30], and [31].

2.4 Data Caching Techniques

Data caching and buffering has many applications to real-world problems, though research in this area has been limited. Research on data-caching techniques for a wireless environment has attempted to solve performance and efficiency issues such as accessing web documents [32]. These techniques use cache replacement algorithms based on recency and frequency of access to these web documents. Thus, efficient informational retrieval was the focus of the data caching schemes investigated in that research.

In contrast, this thesis investigated data-caching techniques to address the problem of real-time data transmission in a wireless environment. The technique ensures the integrity of the data while maintaining the real-time performance monitoring. To date, no other methods reported in the literature focus on this critical problem.

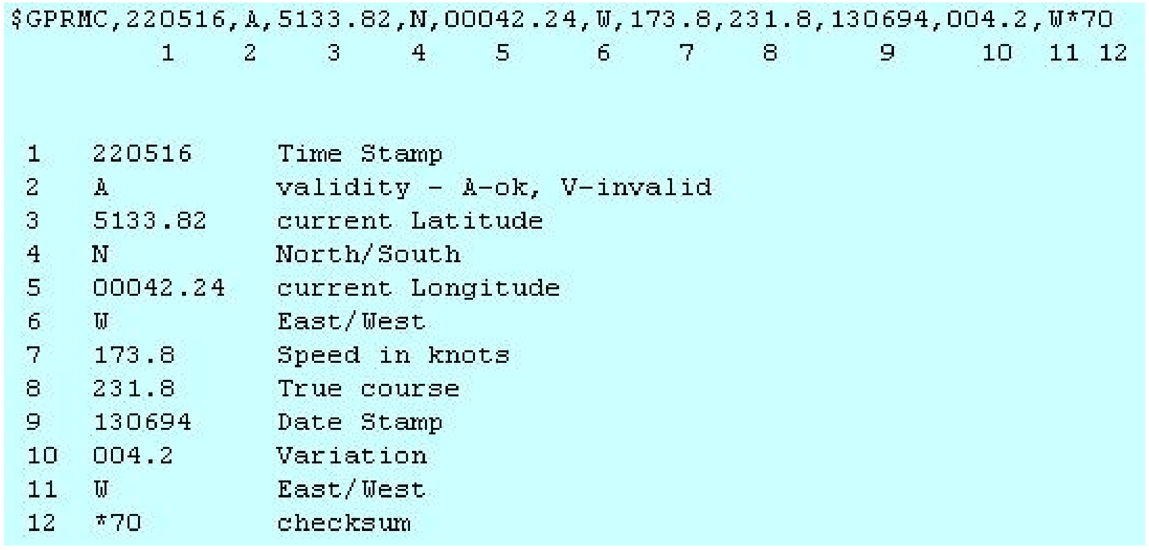
CHAPTER III

VEHICLE POSITION AND PERFORMANCE TRACKING SYSTEM PROTOTYPE

The VPPTS prototype evolved over the course of this thesis investigation through three phases, with the final goal of providing a single board, embedded communications device in the vehicle. The three phases of prototypes can be characterized as: 1) a laptop-based prototype with no embedded attributes, 2) a Windows-based prototype with a multiple board embedded system, and 3) a Linux-based prototype with a single board embedded system.

3.1 Initial Laptop-based Prototype

Off-the-shelf items, acquired to establish proof-of-concept, were used to assemble the initial VPPTS prototype. A Garmin [33] GPS 35-PC receiver collected the recommended minimum data sentence (GPRMC), detailed in Figure 3, from the NMEA



```
$GPRMC,220516,A,5133.82,N,00042.24,W,173.8,231.8,130694,004.2,W*70
         1    2    3     4    5     6    7     8      9     10  11 12


 1   220516     Time Stamp
 2   A          validity - A-ok, V-invalid
 3   5133.82    current Latitude
 4   N          North/South
 5   00042.24   current Longitude
 6   W          East/West
 7   173.8      Speed in knots
 8   231.8      True course
 9   130694     Date Stamp
 10  004.2      Variation
 11  W          East/West
 12  *70        checksum
```

Figure 3: GPRMC provides the minimal information from NMEA standards for GPS location.

standard protocol [34]. OBD-II data was gathered by a BR-3 [35] interface. This interface incorporates a Microchip BR16F84-1.07 microcontroller, which operates on three SAE J1850 protocols [24] i.e., VPW, PWM, and ISO-9141. A Sony Ericsson [36] GC-82 EDGE PC card is used to access the Cingular Wireless [37] GSM/GPRS network. A laptop, equipped with two serial ports (DB9) and a PCMCIA port, acted as a hub through which data is routed. This initial system was designed with these components to support rapid experimentation and data collection.

3.1.1 Data Collector

The data collection software was developed with Microsoft C# using a Visual Basic 6 serial port API. A tomcat web server, in conjunction with a MySQL database

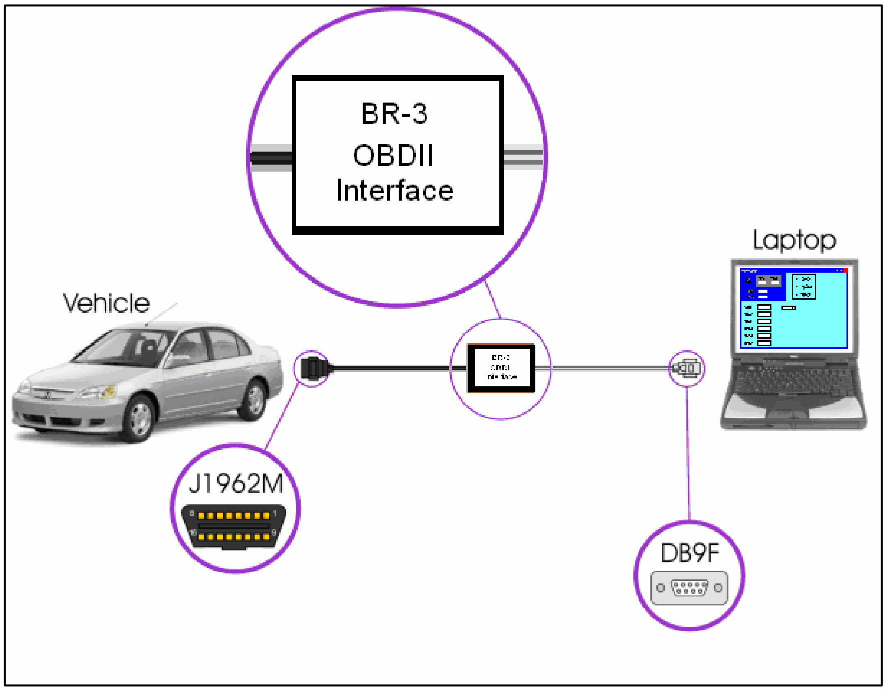


Figure 4: The BR-3 interface is used to communicate with the vehicle's on-board diagnostic system, which offers performance data to the data collector software via RS232 port.

server, functioned as the gateway for users to view the location and performance data of each vehicle. The user interface was developed with JAVA SDK 1.4.2_05.

The data collection software combined GPS coordinates and OBD-II data into a single data-stream that is sent to the server via the GSM/GPRS network. The data is retrieved from the OBD-II system by continuous polling. Transmission of data to the server is triggered by a received event from the GPS device connected to a serial port. This allows for a one second minimum resolution.

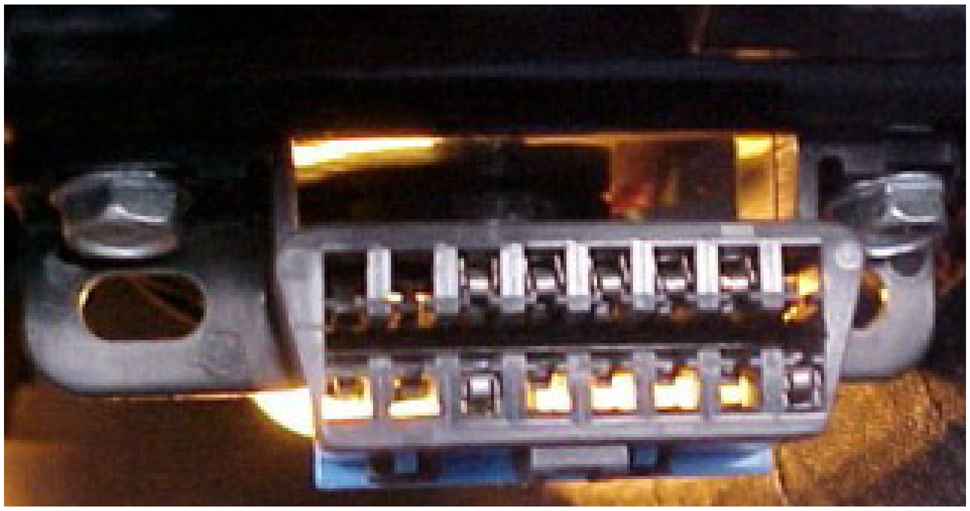


Figure 5: SAE J1962 connector provides access to the diagnostic network.

The BR-3 OBD-II interface, illustrated in Figure 4, connects to a vehicle via the SAE J1962 [38] connector, shown in Figure 5 located within three feet of the steering column. A serial RS232 port on the laptop allows the data collector software to communicate with the BR-3 OBD-II interface. The baud rate between the BR-3 and the laptop is 19,200 Baud with no handshaking. A CRC byte, specified by SAE J1850, is checked to confirm a successful transmission. All three protocols specified by SAE J1850 standard can be accessed with the BR-3. The VPPTS prototype uses generic parameter identifications or PIDs defined in SAE J1979 [39]. These PIDs include vehicle speed, engine RPM, calculated throttle position sensor (TPS), engine load, engine coolant temperature, and air intake pressure. Car manufacturers such as GM and Ford have defined enhanced PIDs that provide additional information for their vehicles.

Figure 6 illustrates the state diagram for the data collector. The BR-3 must be initialized and, depending on the make of the vehicle, a proper protocol must be set. Once these are established, polling for data will commence on a continuous basis. The GPS data is transmitted as character arrays known as sentences. These sentences correspond to the NMEA standard for GPS data. The GPRMC sentence, shown in Figure 3, which contains UTC time, UTC date, longitude, and latitude, is decoded and combined with the current OBD-II data. The data is then sent to the server via GSM/GPRS.

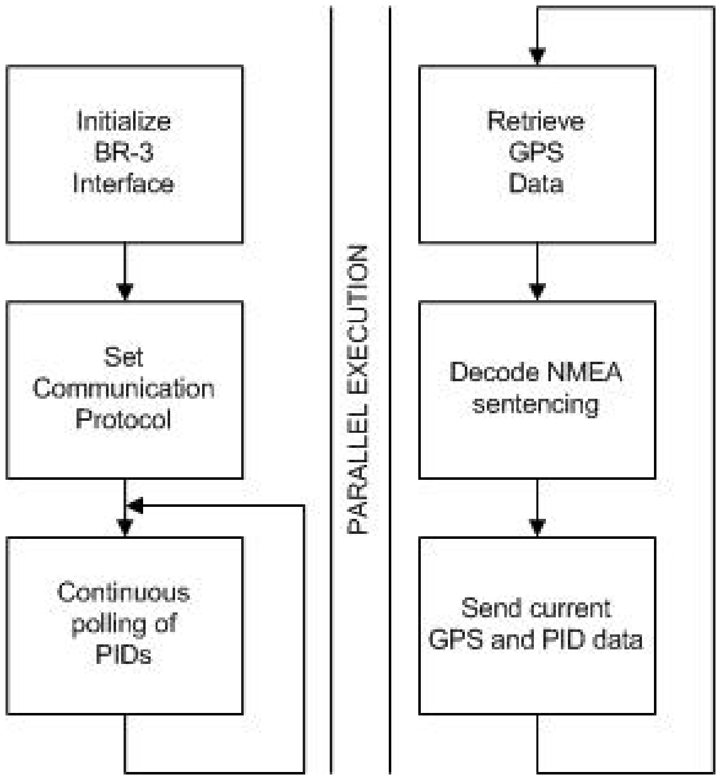

Initialize BR-3 Interface
Set Communication Protocol
Continuous polling of PIDs
PARALLEL EXECUTION
Retrieve GPS Data
Decode NMEA sentencing
Send current GPS and PID data

Figure 6: The data collector polls the vehicle for performance information and receives new GPS coordinates simultaneously.

3.1.2 Data Server

The server software was designed using a dual processor PC, which is used to run the necessary software for the laptop-based prototype system. Tomcat, MySQL, and Apache constitute the software needed to run the data and the applet. Five Tomcat httpservlets maintain the data flow. MySQL was chosen as the database management service, and Apache handles all HTTP page and image requests.

The httpservlets handle all the connections made to the database server (MySQL). Two servlets receive data from the collector through an http post. The data is then transferred to the database for update. The other servlets make queries to the database, package the data into specialized classes, and send the classes to the applet when the data is requested. Figure 7 shows the data flow to and from the server.

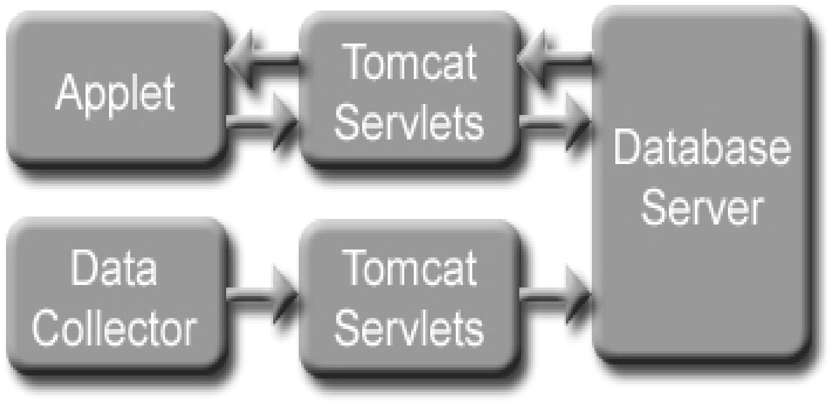


Figure 7: The data flows to the users applet interface after being processed by the server.

Several tables, shown in Figure 8, were used to maintain separation of data within the database that pertain to the campus shuttle bus system. The *stops* table contained a label and GPS coordinates for each shuttle bus stop on all routes. The *routes* table contained a list of the routes and the order at which the stops are traversed. The *shuttle buses* table contained the current location and route information for each shuttle bus. The

	Bus Stop ID	Latitude	Longitude
Stops	The Union	33.4548	88.7905
	Aiken Village	33.464	88.7998
	CAVS	33.4733	88.7933

	Route ID	List of Bus Stops (in order)
Routes	Blue	The Union, Multi-Tenant, ERC, CAVS
	Maroon	The Union, Architecture, Aiken Village, Coliseum, Butler
	Express	The Union, Architecture, Coliseum, Butler

	ID	Bus ID	Route ID	Latitude	Longitude
Buses	1	898	Maroon	33.4539	88.7942
	2	903	Maroon	33.4589	88.7984
	3	1003	Express	33.4549	88.7945

	ID	Bus ID	speed	RPM	TPS	EngineLoad	FuelEconomy	CoolantTemp
Gauges	1	898	12	1456	25	35	8	88
	2	903	14	1543	14	15	10	89
	3	1003	2	945	0	7	13	86

Figure 8: The initial database setup included bus location and performance data and route information.

gauges database contained the telemetry data from each shuttle bus. There is also a table for each shuttle bus that contains all the past telemetry readings for that specific shuttle bus. This data can be stored indefinitely so it can serve as a tool for analysis and simulation of vehicle performance.

A Java applet was developed to display the tracking and performance information to the public via the Internet. The applet displays vehicle location on a digital map. Route information about the vehicles, in this case the campus shuttle bus system, is also available. A user can also view the current shuttle bus gauge data via graphical gauges such as in Figure 9. This implementation allows the public to track a shuttle bus of interest, and allows fleet managers to monitor shuttle bus performance.

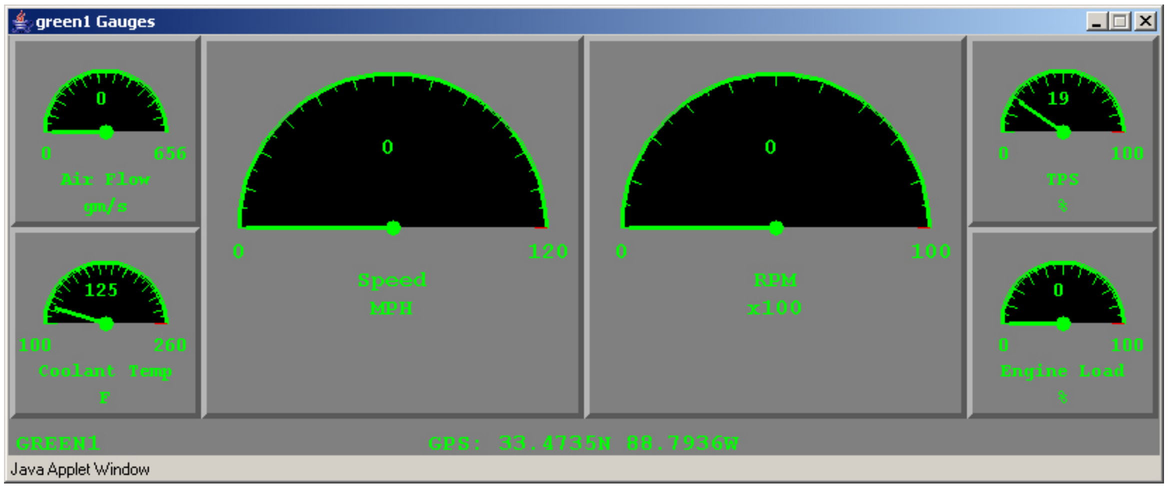


Figure 9: The analog gauges applet displays real-time performance parameters.

3.2 Multiple Board Embedded System Prototype

The initial laptop-based prototype supported preliminary investigations into the feasibility and design of the later prototypes, which were used to gather data for this thesis. To build a deployable prototype, an embedded device with its own operating system (OS) was necessary. The next phase in the evolution towards the goal of such an embedded capability entailed use of multiple boards to encapsulate specific functionalities needed, i.e., CPU, vehicle interface, modem communications interface, and power supply. This section describes the multiple board embedded prototype in detail.

The PC104 architecture [40] was chosen because of its saturation in the embedded industry. Many embedded systems employ the PC104 architecture for its modular nature. The PC104 architecture specifies a form factor which makes the embedded devices compact. It also specifies the bus architecture that is used for communication between each PC104 board. The Industrial Standard Architecture (ISA) [41] was adopted first followed by the more modern Peripheral Control Interface (PCI) [41]. Many PC104 board incorporate both bus communications to support legacy devices. The devices chosen were used in a shuttle bus program at Mississippi State University called *Bullybus* [42].

The main or CPU board was the Kontron [43] MOPSlcd7. This board acts similarly to a motherboard found in most personal computers today. Its specifications consisted of a 300MHz Celeron processor, 256MB RAM, IDE hard drive support, and PC104 (ISA) and PC104+ (PCI) bus support. The specialized peripheral boards connected to the CPU board provide the entire system with the needed functionality. These boards include the Winsystems [44] Cardbus adapter for use with a cellular modem PC Card, the Dearborn [45] DPAIII/PC104 that interfaces with shuttle vehicle networks J1939 [15] and J1708 [14], and the Tri-M Systems [46] V104 vehicle power supply. The latter protects the embedded system from the volatile electrical environment in a vehicle and also provides the necessary power through the PC104 communication bus. The Garmin GPS 35-PC external GPS receiver was also chosen to collect location data.

The embedded system used a customizable Windows XP OS known as Windows XP Embedded [47]. This OS and its associated development tools allow a developer to install and modify the necessary components that are needed for the embedded system. The idea was to use, but enhance the same application code that was used in the laptop-based prototype.

Problems were encountered when the C# code would not communicate with the DPAIII/PC104. A bug existed with the C# driver for the DPAIII/PC104. It was also determined that C# was the wrong programming language for a real-time application. The application was then ported to C++ which had no inherent bug with the DPAIII/PC104.

This C++ application was initially designed to be similar to the C# application. However, it was determined that a connection manager was needed to monitor the mobile connection and transmit the data. The connection manager incorporated Windows API for Remote Access Server (RAS) [48] functions to establish a connection. These functions also allow for monitoring the status of the connection such as connection speed and number of bytes transmitted and received. For transmitting the data, the curl libraries [49] were used. These libraries allow for simple transmission of data using HTTP posts. Curl also provides statistical data for each HTTP connection and transmission.

During initial tests of this second phase prototype, using the GSM/GPRS network, the connection was periodically dropping, and the reconnection procedure could last more than five minutes in duration. This meant that more than five minutes of data was lost and unrecoverable. A buffering technique was needed to prevent this data loss and increase data integrity. An initial simple technique was used to temporarily store this data on the embedded system by writing it to a file. The application would then attempt to send each item in the file when the shuttle bus was switched off.

3.2.1 Data Buffering Technique for Data Integrity

Though this technique provides some buffering capability, it does not offer the robustness that is needed for a real-time system. In order for the data integrity to be ensured, the data must to be transferred to the server as quickly as possible. The buffering technique was revised to incorporate a data cache that allows the transmission of stored data along with new data. This helps to increase the reliability of the system by making sure the data is transmitted to the server sooner rather than later. If a transmission attempt reached a timeout, the data being sent would be stored in the data cache. The next transmission would attempt to send the newest data as well as the data in the cache. The

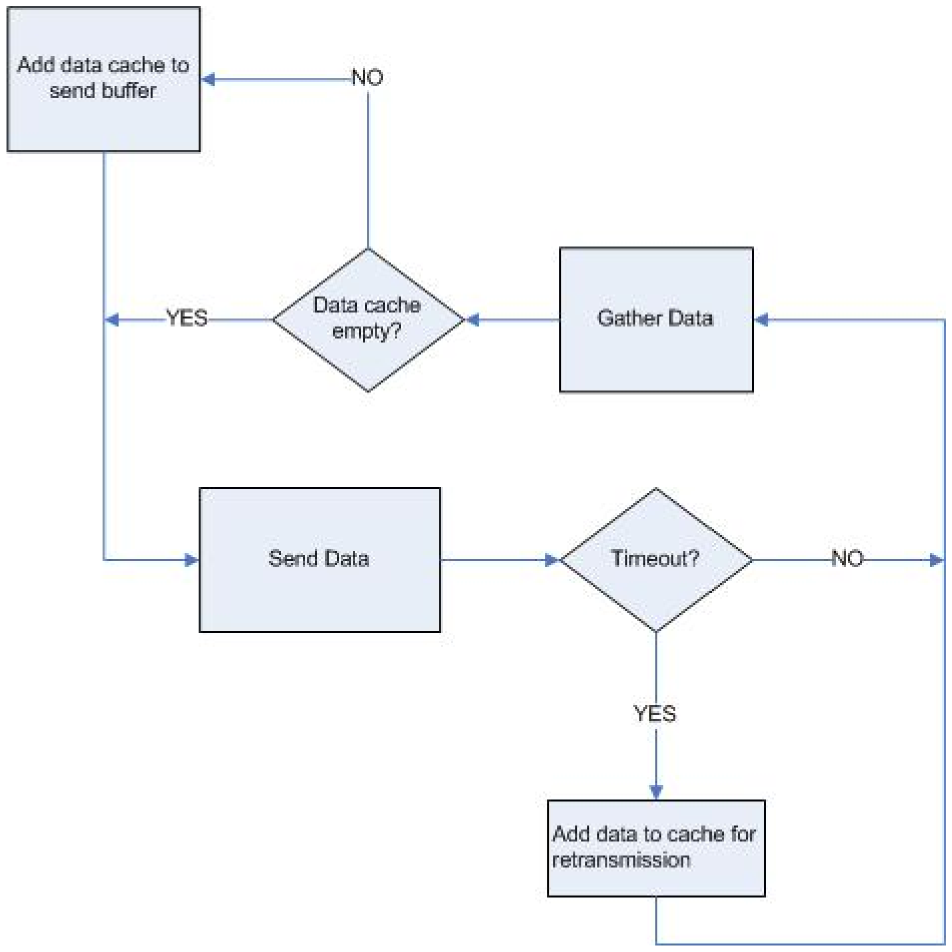

Add data cache to send buffer
NO
YES
Data cache empty?
Gather Data
Send Data
Timeout?
NO
YES
Add data to cache for retransmission

Figure 10: The data buffering technique ensures data integrity.

size of the data cache is configurable from minutes to days or even months worth of data. This technique is illustrated in Figure 10

3.2.2 Limitations of Single-Threaded Buffering Technique

The new buffering techniques relieved some of the transmission problems. Also, testing of the prototype with the new EVDO network yielded fewer timeouts and unwanted interruptions. An EVDO network theoretically offers greater capacity and higher bandwidth than GSM-based networks [50]. Its underlying CDMA2000-based network also offers a soft-handoff between the network towers unlike GSM with a hard-handoff., which can introduce unwanted interruptions with the data transmission [51]. However, even the EVDO network could not eliminate all timeouts and unwanted interruptions. Further, neither the new EVDO network, nor the enhanced buffering techniques could solve a major problem: because the C++ application was single-threaded, the application would halt all data-gathering when attempting to reconnect to the wireless network. This called for a multi-threaded approach with separate threads for gathering and transmitting data to achieve greater robustness.

During the transition to a multi-threaded design, Windows XP Embedded [47] requirements for system resources were shown to be prohibitive. An embedded Linux system requires less system resources. Further, it can be more easily customized to meet the needs of a robust system. Therefore, the multi-threaded application was designed for the Linux environment.

3.3 Final Single Board Embedded System Prototype

Significant data was gathered using the multiple board prototype and many valuable insights gained for this research. However, the final prototype embodied the goal of a single board embedded system. The main advantages of the single board system are presented, of course, by the reduced cost, but of equal importance, the increased ease of development, performance, and maintenance. These latter characteristics made the single board system an enhanced platform for testing the data caching techniques investigated in this thesis.

3.3.1 Linux System

This version of this system uses a single board solution that integrates many features from the previous system. A Micro/sys [52] SBC4495 has a built-in GPS receiver based on the Lassen SQ module [53]. The SBC4495 also incorporates a Cardbus slot for the modems. The SBC4495 uses a ST Microelectronics STPC Atlas processor running at 133MHz and up to 64MB of RAM. The SBC4495 also put PC104 and PC104+ connectors for expandability.

The Dearborn device from the previous system does not support Linux. Two devices were needed to communicate with heavy-duty vehicles i.e. the shuttle busses and light-duty vehicles. The ElmScan5 from ScanTool.net [54] is used to communicate with light-duty vehicles. This is an intelligent device that uses the ELM327 [55] microcontroller to determine what network to communicate. Heavy-duty vehicle networks were monitored with the Autotap [56] HDV100Ax. This device can communicate with both J1939 [15] and J1708 [15] networks.

An embedded Linux distribution based on the AspisOS [57] project was used. This distribution is geared toward wireless embedded systems and optimized for size. Additional modules and configurations were added to customize the distribution. A GPS daemon known as GPSd [58] handles gathering the GPS data. A connection script built around pppd [59] helps keeps the wireless connection alive.

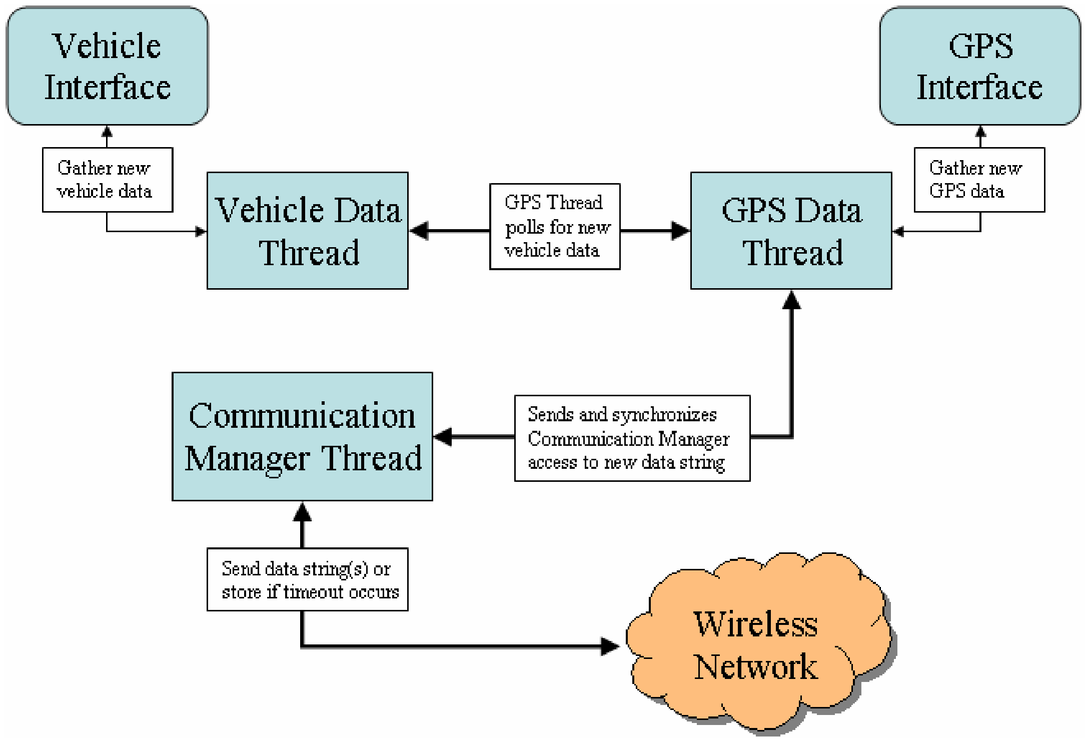

Vehicle Interface
GPS Interface
Gather new vehicle data
Vehicle Data Thread
GPS Thread polls for new vehicle data
GPS Data Thread
Gather new GPS data
Communication Manager Thread
Sends and synchronizes Communication Manager access to new data string
Send data string(s) or store if timeout occurs
Wireless Network

Figure 11: The VCOM application uses multiple threads to increase the data gathering rate that was hindered by a single thread.

The above modules and scripts are external to the new application. The vehicle communicator (VCOM) application was designed with multiple, separate threads to handle 1) gathering data and 2) transmitting data as seen in Figure 11. A vehicle data gathering thread communicates with either the ElmScan5 or HDV100Ax and stores the latest vehicle performance data. A GPS gathering thread uses the libgps [58] library to communicate with GPSd and structures the GPS data. The GPS gathering thread also captures the latest vehicle performance data that was gathered by the vehicle data gathering thread. Semaphores are used to synchronize the communication between threads. Finally, the communication manager thread uses the curl libraries to send the newly gathered data strings, illustrated in Table 3, including at least vehicle ID, date, time, latitude, longitude, speed, RPMs, and throttle position. The data string was designed for expansion beyond these attributes. The parameter values in Table 3 are hexadecimal representations of the actual parameter they are measuring. The server handles the translation from this format to a human-readable format, e.g. RAW speed is 34 and the TRANSLATED speed is 17 kph. These translation factors are gathered from the SAE J1979 [39].

Table 3: The vehicle communicator application transmits a data string at least every second.

Data String Example
vehicleID|date|time|latitude|longitude|1st parameter=protocol;ID,index,value| 2ndparameter=protocol;ID,index,value|3rd parameter=protocol;ID,index,value|…
vehicleId=2|date=240706|utc=115052|gpsN=33.467185|gpsW=-88.795952| data1=J1850;010D,0;4D|data2=J1850;010C,0;17E4|data3=J1850;0111,0;19| data4=J1850;0104,0;17|data5=J1850;0105,0;88

The multi-threaded design allowed for uninterrupted data gathering and storing. The buffering technique was also used to ensure data integrity. The buffering technique takes advantage of this continuous data gathering by storing the data in the data cache while the communication manager thread continues to transmit the gathered data.

3.3.2 Server Enhancements

While the focus of this phase of prototype development was dedicated to the vehicle communicator system, the server also underwent many changes and enhancements. A major goal for the server is to provide a Geographical Information System (GIS) mapping solution [60]. A GIS database can be created to generate user requested maps. This reduced the initial load on the client-side because a large static map was loaded on the client machine and allowing users of small handheld devices to access this information. Using GIS and GPS together, the system can also provide location-based information and attributes for many different applications.

With the move to a GIS-based mapping system, the client application, shown in Figure 12, was also redesigned. The Java applet was replaced with the light weight JavaScript [61]. Many novel mapping applications such as Google Maps [62] and Microsoft's Local Live [63] use JavaScript as their main language for displaying interactive. An interactive playback tool was also developed to aid the user with analyzing performance data for a certain time period. For the playback tool to be reliable

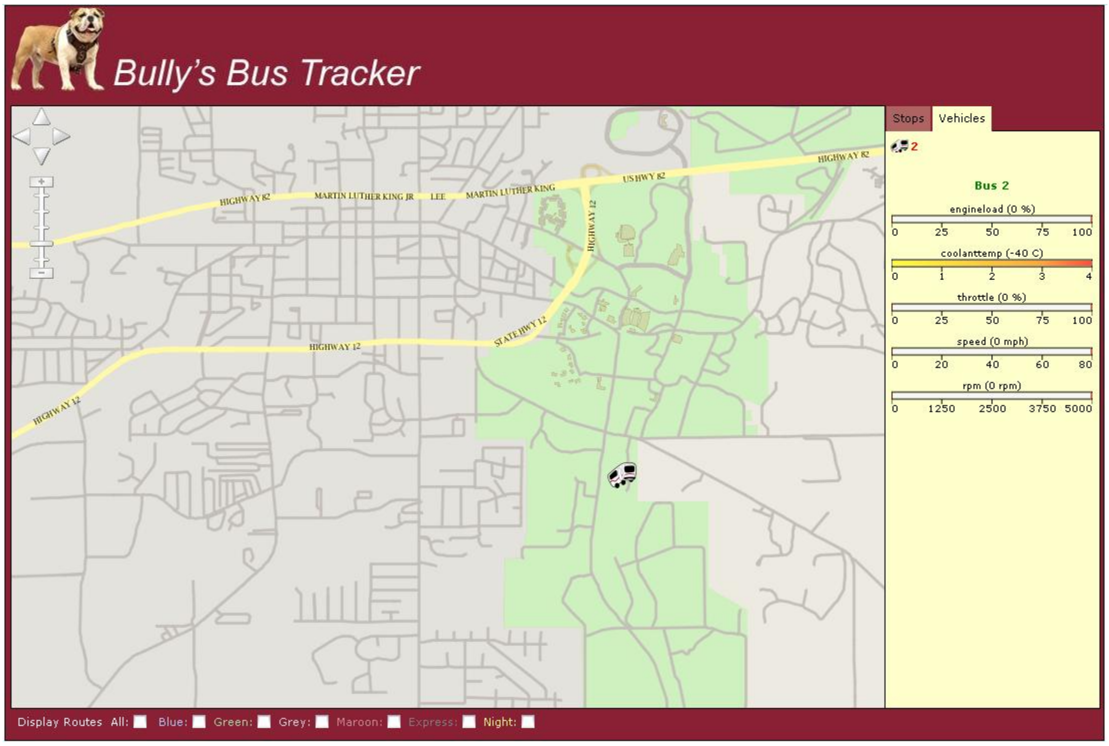


Figure 12: Bully's Bus Tracker takes advantage of GIS and JavaScript to provide an interactive mapping application.

in its analysis, the vehicle data that is being displayed must be available and intact. This, in turn, intensifies the requirement that data integrity be maintained.

The enhanced server design also improved the data storage and data flow from the vehicle communicator to the server and from the server to the client. During the release of the laptop-based prototype and the initial release of the multi-board prototype, the servlet that accepted the incoming data would parse and translate it and store each individual data parameters in the *gauges* table as well as an historical table. This requires the database to include tables with all the vehicle parameters (speed, engine RPM, oil

Table 4: The protocols table allows the server to translate the raw data into human readable format.

Protocol Name	Param ID	Param Name	Param Index	Param Translation	Parm Units
J1708	54	speed	X	0.5	Mph
J1939	0CF00400	Rpm	3	0.125	RPM

pressure, etc.) that could exist. This could potentially reduce the server speed while it searches for the proper location to place new data. One enhancement entailed removing these tables and including a single table that stored the raw data string, shown in Table 3. This allows the servlet handling the incoming data to simply store the raw data. The servlets handling the client application will actually perform the data translation using a protocols table, described in Table 4. This table is an essential part of the playback tool in order for the user to understand how a particular vehicle is operating.

CHAPTER IV

EXPERIMENTAL RESULTS AND OBSERVATIONS

Each of the three prototypes was tested and evaluated using differing approaches of increasing rigor. Observations of the first laptop-based prototype in field settings in a single vehicle were important in determining the design of the later prototypes. The second multi-board prototype was deployed on multiple vehicles, and the system performance was formally monitored and analyzed. The design and development of the final single-board Linux-based prototype incorporated important lessons learned the first two prototypes. Its performance was also formally monitored and analyzed.

Several metrics were used in the experiments to measure the improvements in data integrity offered by the buffering technique investigated in this thesis. First, the number of timeouts detected was compared with the amount of data that was successfully transmitted. This indicated if data was stored and retransmitted successfully using the buffering techniques. Finally, two popular wireless networks were used (GSM/GPRS and CDMA2000/EVDO) during the single-threaded tests, while the CDMA2000/EVDO network was used during the single-board Linux-based prototype. To indicate how these networks might benefit from the buffering techniques, data transmission time was also analyzed. This will indicate if a poor network environment can handle the transmission of the vehicle data. The following sections will discuss the analysis of each prototype system.

4.1 Laptop-based Prototype System

The initial prototype demonstrated the feasibility of transmitting vehicle performance data and vehicle data to a server for real-time monitoring. It also served as an influential design guide for later prototypes. However, no formal experiments were conducted with this prototype. Nonetheless, since it did not incorporate buffering, it also served as a baseline for the results yielded by the later two prototypes.

4.2 Multiple-Board Prototype

4.2.1 Single-threaded and data buffered directly to a file.

This was the first phase of the multiple-board prototype development. Testing for unit and system integration occurred during this phase. A simple buffering technique was used which buffered data strings, similar to Table 3, directly to a file. At the end of day, when the vehicle turned off, the application attempted to send the data that was logged in the file. This technique served the purpose of testing of initial functionality of the multiple-board prototype, but a more robust technique was needed so that the buffered data would be transmitted as soon as possible. This enhanced technique is described in the following section.

4.2.2 Multiple-board, Single-threaded with circular buffer

The circular buffer was first added to the prototype using the single-threaded programming approach. Performance data was collected regularly for this prototype over the course of two months. Data analysis presented in this chapter was conducted for a single day, i.e., 24-hour period, representing typical usage patterns when the university is in session with maximum staff and students utilizing the campus bus system. Further, such a day illustrates the widest range of network load from lowest demand to peak, when maximum timeouts, and associated data loss, may occur. Both GSM/GPRS and CDMA2000/EVDO networks were also analyzed during this phase of testing.

Table 5 presents the performance data for Shuttle buses, 903 and 1205, during this typical 24-hour period. Shuttle bus 903 transmitted data strings using the GSM/GPRS network, while shuttle bus 1205 transmitted data strings using the CDMA2000/EVDO network. The performance data includes the total number of data strings available, the

Table 5: Shuttle Buses 903 and 1205 benefited from the circular buffer as indicated by this chart.

Shuttle Bus	903	1205
Network	GSM/GPRS	CDMA2000/EVDO
Total Data Strings Available	52200	50791
Total Data Strings Gathered	42345	50087
Transmission Attempts of Gathered Data Strings	42345	50087
Timeouts During Transmission	1383	229
Gathered Data Strings Successfully Transmitted	42345	50087
Percent Gathered Data Strings Buffered	2.65%	0.44%
Percent Buffered Data Strings Successfully Transmitted	100%	100%
Percent Data Strings Not Gathered	18.88%	1.39%
Average Successful Transmission Time (seconds)	0.767	0.365

number of transmission attempts of gathered data strings, the total number of timeouts while attempting to transmit, the percentage of gathered data that was buffered, the average successful transmission time, the percentage of buffered data strings that was transmitted successfully, and the percentage of non-gathered data strings. This data string, shown Table 3, in represents one second of gathered data.

Figure 13 shows the successful transmission times of gathered data over a 24-hour period in distinctive groups that divide each hour of operation. In Table 5 the average successful transmission time for the GSM/GPRS shuttle bus (903) was 0.767 seconds while the CDMA2000/EVDO shuttle bus (1205) was only 0.365 seconds. CDMA2000/EVDO shows a 52% increase in performance from GSM/GPRS during this test.

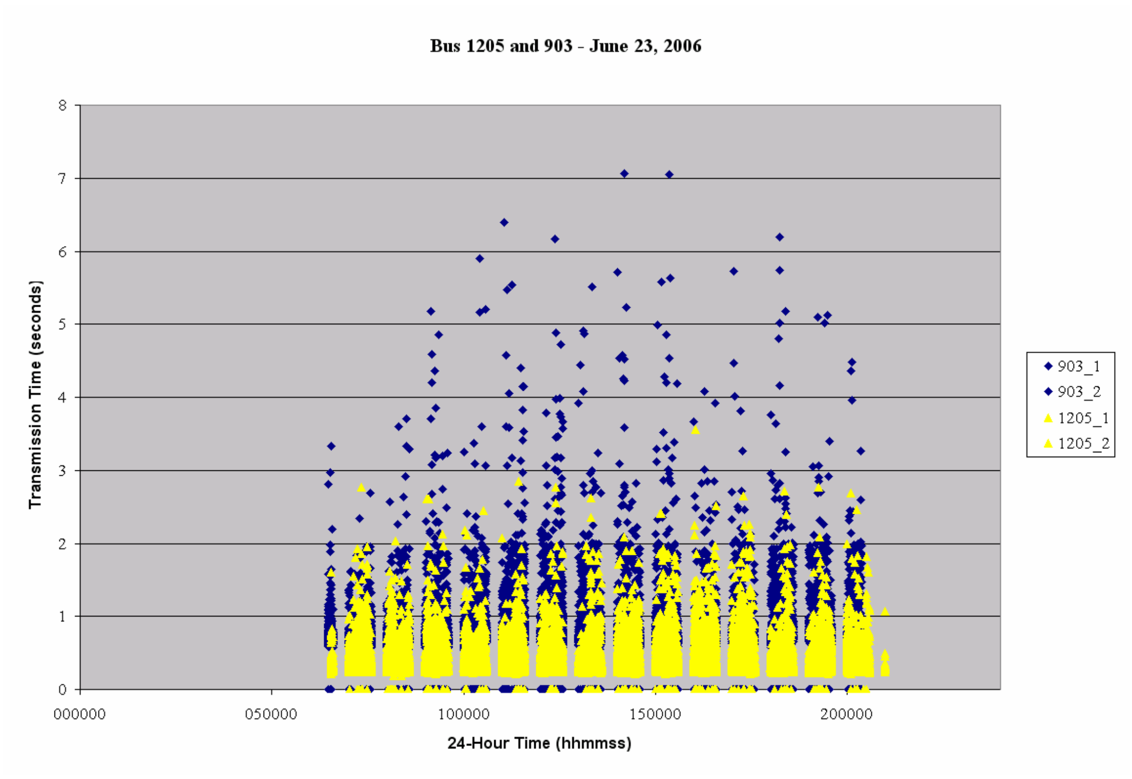


Figure 13: Analysis of transmission time during one full day for shuttle bus 1205 and shuttle bus 903.

Focusing on the percentage of buffered data strings transmitted successfully in Table 5 shows that the circular buffer was able to store all the gathered data strings that could not be transmitted initially. These buffered data strings were successfully retransmitted when the buffered data strings were piggybacked with a newly gathered data string. This shows a definite advantage compared to non-buffered techniques. The buffering technique also attempts to transmit gathered data strings as soon as possible, which the logging-to-a-file buffering technique did not accomplish.

Table 6: Shuttle Bus 1205 had difficulty transmitting all gathered data due in part to the single-threaded application.

Shuttle Bus	**903**	**1205**
Network	GSM/GPRS	CDMA2000/EVDO
Total Data Strings Available	161	30647
Total Data Strings Gathered	140	4969
Transmission Attempts of Gathered Data Strings	140	4696
Timeouts During Transmission	11	1068
Gathered Data Strings Successfully Transmitted	140	4065
Percent Gathered Data Strings Buffered	6.83%	3.48%
Percent Buffered Data Strings Successfully Transmitted	100%	86.56%
Percent Data Strings Not Gathered	13.04%	84.68%
Average Successful Transmission Time (seconds)	0.854	0.420

However, during a timeout, which measures at least two seconds, at least one second of data was not gathered. This is evident by the percentage of non-gathered data strings in Table 5. The single-threaded application is attempting to send but does not gather new data in the process. This problem is magnified if many consecutive timeouts occur. The single-threaded application will attempt to reconnect, but will not gather new data. If a reconnect attempt fails five times, the multiple-board system is restarted to remedy the problem.

Table 6 and Figure 14 reveal how the problem can offset the benefits of the buffering technique and reduce the data integrity of the system. Each hour of data is also grouped together. The shuttle bus 1205 begins in a strong network coverage area, but eventually leaves the coverage area around 0820 hours and is unable to transmit the data. Some of the gathered data is buffered as shown in the percentage of gathered data strings buffered in Table 6. However, excessive timeouts force the single-threaded application to reconnect and halt gathering new data as shown by Figure 15. Only 15% of the available data strings are gathered. Of the gathered data strings, only 87% are transmitted successfully. The next section discusses the results of the single-board Linux-based prototype system using a multi-threaded application, which addresses the problem.

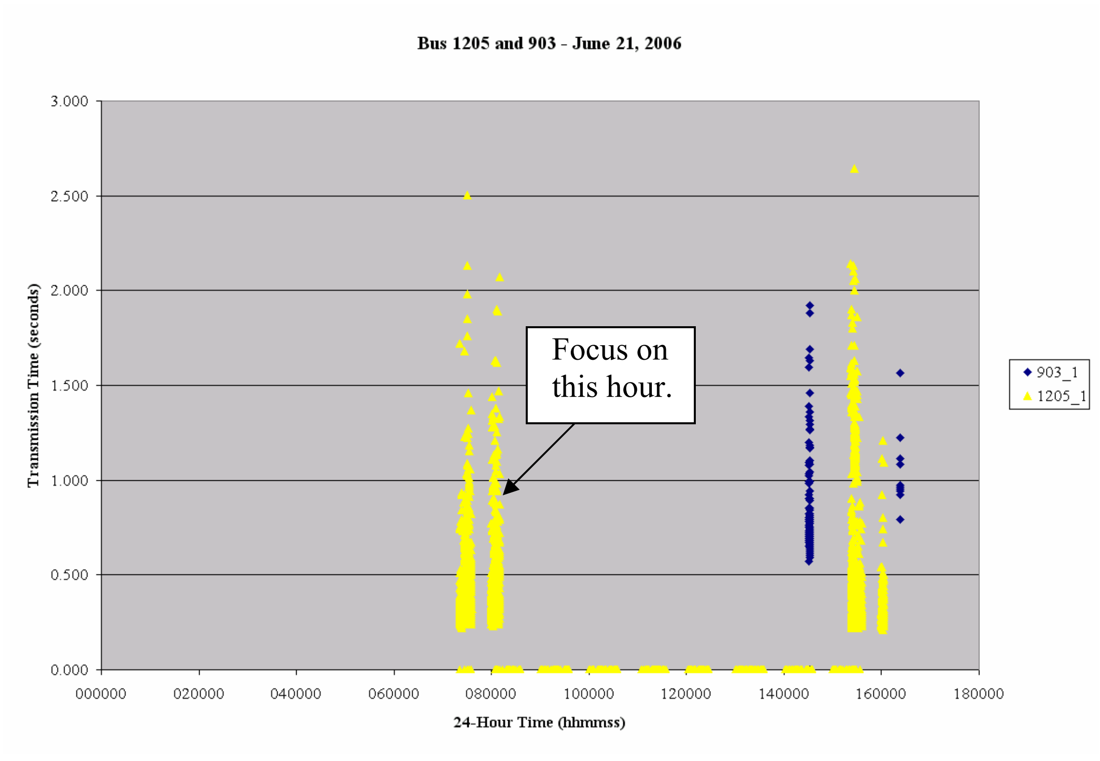


Figure 14: Shuttle bus 1205 shows signs of a poor network with many consecutive timeouts, while shuttle bus 903 was hardly utilized.

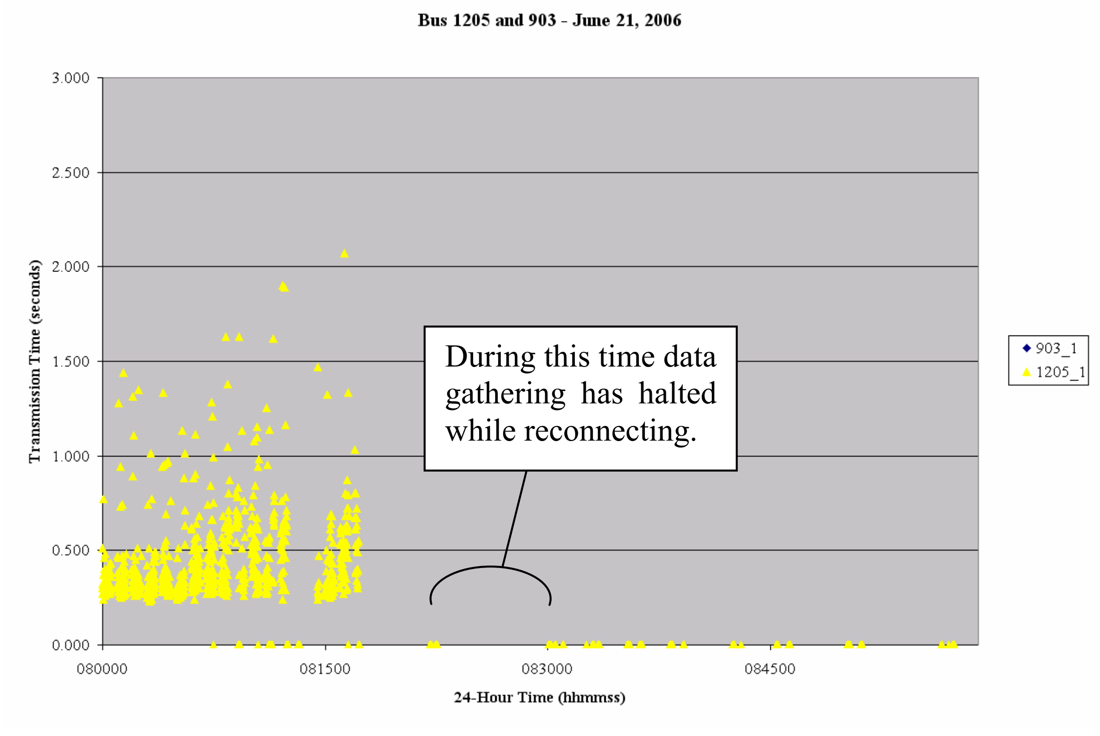


Figure 15: Close-up on an hour in which shuttle bus 1205 shows signs of a poor network with many consecutive timeouts.

4.3 Single-Board, Multi-threaded Linux Prototype

This phase of the prototype system not only saw a complete change in hardware but also in software. The operating system was changed to Linux, and the application incorporated multiple threads. The multi-threaded application continuously gathered new data strings even while the wireless connection was poor and many transmission timeouts occurred. Testing for this phase used only the CDMA2000/EVDO network.

The multi-threaded application, running in a Linux environment, allowed the buffering technique to continue to gather data strings during long transmission times of previously gathered data strings. The single-threaded application described in the previous sections was not able to gather these data strings during these transmission

Table 7: The multi-threaded application enhanced the buffering technique for the test vehicle data strings.

Vehicle	**Test Vehicle**
Total Data Strings Available	2488
Total Data Strings Gathered	2509
Transmission Attempts of Gathered Data Strings	2267
Timeouts During Transmission	48
Gathered Data Successfully Transmitted	2509
Percent Gathered Data Strings Buffered	42.85%
Percent Buffered Data Strings Successfully Transmitted	100%
Percent Data Strings Not Gathered	-0.85%
Average successful transmission time (seconds)	0.433

delays. Table 7 shows that the multi-threaded application actually gathered more than one data string per second. The total number of data strings available is 2488, which is also number of seconds of vehicle operation. The application successfully gathered and transmitted 2509 data strings, which shows that 21 more data strings were gathered and transmitted. Thread timing issues arose from an inaccurate real-time clock on the SBC4495 and led to this over-gathering. This deficiency can be remedied by using the GPS signal as a trigger to gather data or replacing the hardware with a board that incorporates a more accurate real-time clock.

Figure 16 shows the transmission times as the test vehicle is operating. As stated in Table 7, the average transmission time using the CDMA2000/EVDO network was 0.433 seconds. Around the 1840 second of operation the network environment showed signs of instability as many timeouts occurred. However, Table 7 shows that all buffered data was transmitted. This indicated that the multi-threaded application continued to gather new data strings during this time. The transmission timeouts were set to a maximum of 10 seconds. Figure 17 shows eight consecutive timeouts. These timeouts occur because the transmission operation was longer than 10 seconds. The following timeouts occurred because the system attempted to reconnect to the network. When the network was again available the application began to transmit the buffered data strings along with a newly gathered data string. The maximum number of data strings that could be sent was set to 10. Therefore, a total of 120 data strings were buffered during this time.

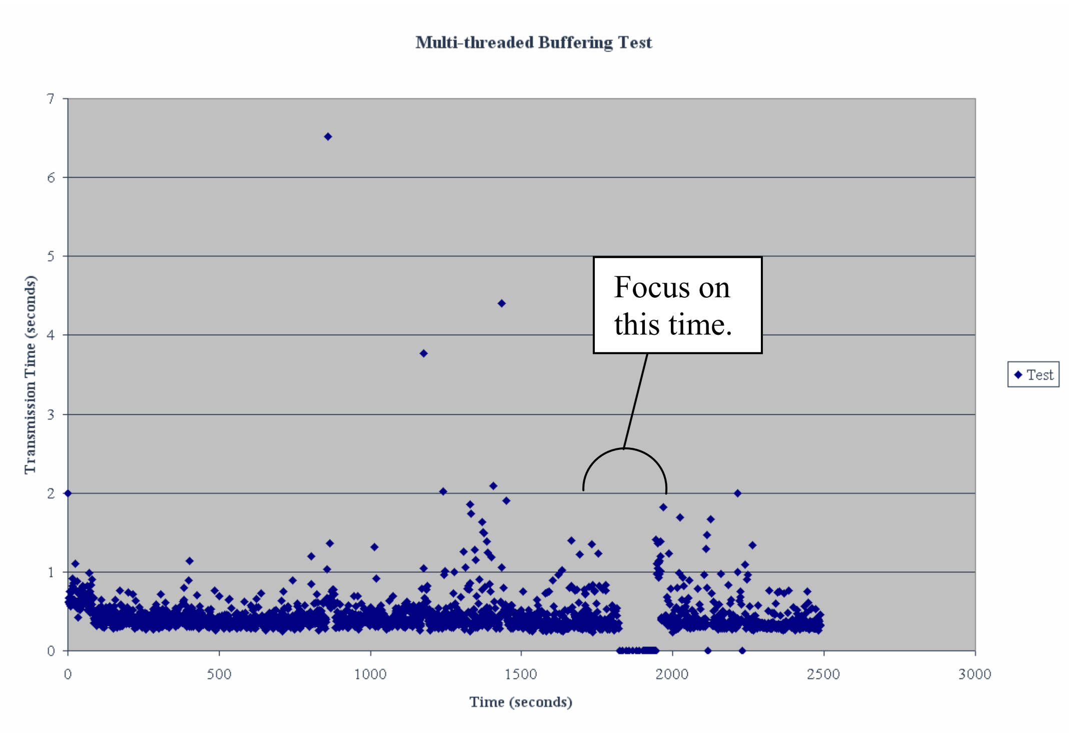


Figure 16: Analysis of transmission time after employing the buffering technique in a multi-threaded application.

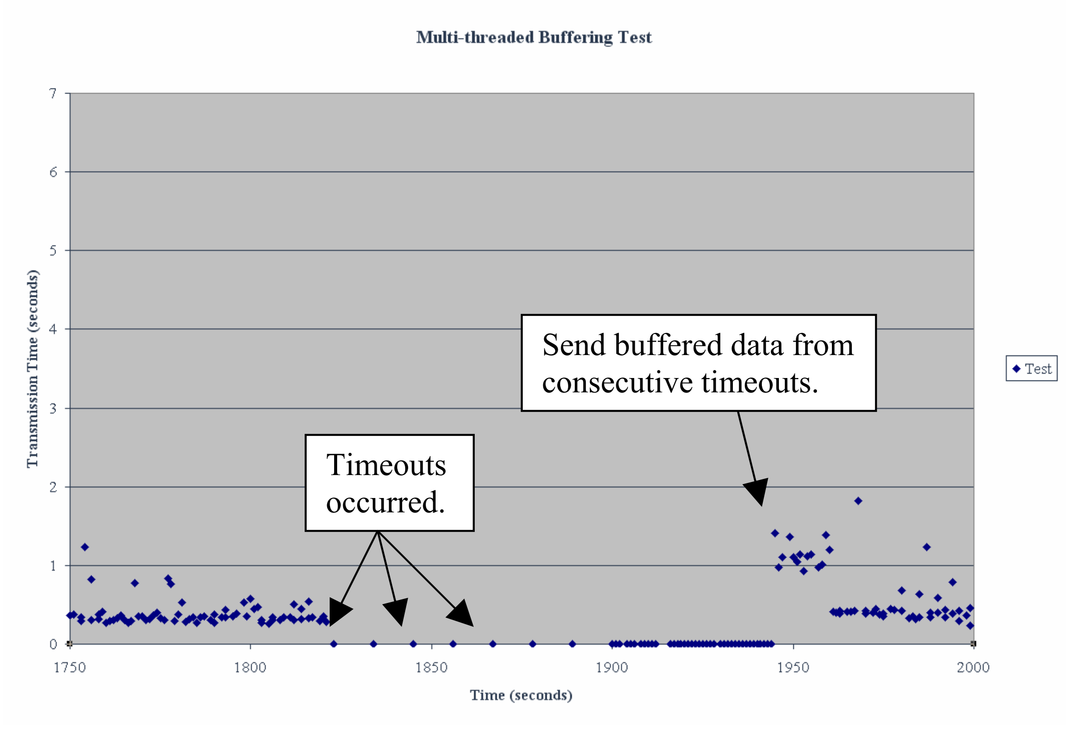


Figure 17: Timeouts do not affect the application from gathering new data.

CHAPTER V

CONCLUSIONS AND FUTURE RESEARCH

This thesis investigated the performance benefits of a data integrity buffering technique for an extensible vehicle position and performance tracking system (VPPTS). In support of this investigation, a VPPTS prototype was developed. The prototype used available technologies and interfaces to industry-standard communications channels and is a demonstration of a next-generation intelligent transportation system (ITS). The data integrity buffering technique under investigation was shown to provide quantitative improvements in successful VPPTS data transmissions. The use of this technique addressed important deficiencies in real-time data transmission for these types of systems over wireless networks.

5.1 Thesis Contributions

This thesis explored critical issues in the development and performance of real-time vehicle position and performance tracking systems. The key contributions of this thesis pertain to the data buffering techniques designed to enhance real-time data transmission and the unique aspects of the prototype system developed to test the thesis hypothesis. These contributions include:

- The VPPTS prototype exploited GPS technology, enabling a vehicle location to be pin-pointed to within a few of meters. An in-vehicle standard for diagnostic information, ODBII and heavy-duty protocols J1939 and J1708, is used to gather performance data. Using a wireless modem, the location and performance information can be made available to a remote site via the Internet. Data is integrated and transmitted to a web server using Apache's Tomcat extensions to provide Internet access via a vehicle tracking web site. Many existing systems offer these technologies with a subscription service. However, this system is designed with an open architecture that can be easily expanded to other applications.

- Integration of a GIS relational database of our region into the system greatly reduces the amount of data that must be downloaded to the client, and vastly increases the system interactivity with the maps. The system has the capability to interact with this information and produce many types of value-added features such as information queries (e.g., "What is the closest fast-food restaurant to the Union bus stop?). More importantly, this will allow the system to work well on low bandwidth devices with small displays, such as cell phones and PDAs.
- Maintaining data integrity is critical for the real-time performance needed by a VPPTS. Despite their advantages, wireless networks introduce instability in the transmission medium. Data is more likely to be lost due to wireless interference and poor signal strength than via a wired connection. To ensure data integrity for this type of transmission, a buffering technique was developed and rigorously tested. This buffering technique addressed the problem, reducing the amount of data lost with wireless transmission compared to a non-buffering system. For the system to take full advantage of the buffering technique, however, a multi-threaded application was designed and tested. This multi-threaded application ensured that data was continually gathered even if transmission was hindered. The

multi-threaded application showed a significant improvement over the single-threaded application that used the same buffering technique.

5.2 Future Research

The prototype system attempts to transmit data at least every second. Future development could prove that delayed transmission increases overall performance. In other words, transmit data every 5 to 10 seconds might reduce the number of transmission timeouts and could possibly reduce the load on the system.

Also the maximum number of data strings that can be transmitted is currently a fixed number. Due to dramatic changes in the wireless network as a vehicle is moving, this number might not be suitable for certain situations. An enhancement to the buffering technique could monitor the network performance and dynamically change the number of data strings that can be transmitted to reduce the number of transmission timeouts and increase the transmission success rate thereby increasing data integrity.

The multi-threaded application utilizes the SBC4495 real-time clock to synchronize the gathering and transmitting of data strings. Most GPS modules act as an accurate atomic clock allowing for an application to use this signal as a precise trigger. The multi-threaded application should use this signal to trigger the gathering a transmission of new data. This would prevent duplicate data strings from being gathered, which could alter performance analysis reports.

The prototype systems rely on a cellular-based network to transmit the data strings. The modular architecture allows for seamless transition between wireless transmission mediums such as cellular, WIFI, WIMAX, etc. Future development should also incorporate an ad hoc vehicular network that will be able to relay messages “down the road” to the destination server.

REFERENCES

[1] L. Figueiredo, I. Jesus, J.A.T. Machado, J.R. Ferreira, J.L. Martins de Carvalho, “Towards the Development of Intelligent Transportation Systems,” *IEEE Intelligent Transportation Systems Proceedings*, Oakland, CA, pp. 25-29, 2001.

[2] M Rahnema. “Overview of the GSM system and protocol architecture,” *IEEE Communications Magazine*, April 1993.

[3] Viterbi, A. “CDMA: Principles of Spread Spectrum,” *Communication Addison-Wesley Wireless Communications Series*, 1995.

[4] J. Cai, D. Goodman, “General Packet Radio in GSM,” *IEEE Communications Magazine*, pp. 122-131, 1997.

[5] Qi Bi; S. Vitebsky, "Performance analysis of 3G-1X EVDO high data rate system," *Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE*, vol. 1, pp. 389-395, 17-21 Mar 2002.

[6] S. Godavarty, S. Broyles and M. Parten, "Interfacing to the On-board Diagnostic System," *Proceedings Vehicular Technology Conference*, vol. 4, pp. 2000-2004, 24-28 Sept. 2000.

[7] T. Yunck, G. Lindal, C. Liu, The role of GPS in precise Earth observation, *Position Location and Navigation Symposium*, pp. 251-258, Dec. 1988.

[8] J. Brittain, I.F. Darwin, *Tomcat: the definitive guide* (O'Reilly, 2003).

[9] OnStar. [online]. Available: http://www.onstar.com/us_english/jsp/index.jsp

[10] First Responders. [online]. Available: http://www.northcom.mil/index.cfm?fuseaction=s.firstresponders

[11] A. Wahab, T. Chong, N. Wah, O. Eng, W. Keong, A Low-Cost Yet Accurate Approach to a Vehicle Location Tracking System, *IEEE ICICS*, Singapore, pp. 461-465, 1997.

[12] U.S. Environmental Protection Agency. “Clean Air Act.” [online]. Available: http://www.epa.gov/air/oaq_caa.html/

[13] U.S. Environmental Protection Agency. “On-Board Diagnostics.” [online]. Available: http://www.epa.gov/obd/basic.htm

[14] SAE J 1708 October 1993, “Serial Communications Between Microcomputer Systems in Heavy-Duty Vehicle Applications,” *2004 SAE Handbook*, SAE International, 2004.

[15] SAE J 1939, “Recommended Practice for a Serial Control and Communications Vehicle Network, *SAE J1939 Standards Collection*,” SAE International, 2004.

[16] Garmin. “What is GPS.” [online]. Available: h*ttp://www.garmin.com/aboutGPS/*index.html

[17] A. El-Rabbany , *Introduction to Gps*, Artech House, Jan 1, 2002.

[18] Youjing Cui; Shuzhi Sam Ge, "Autonomous vehicle positioning with GPS in urban canyon environments," *Robotics and Automation, IEEE Transactions on*, vol.19, pp. 15-25, Feb 2003.

[19] M. May; E. Kreisher; T. Nasuti; C. Sives, "Evaluation of GPS receiver ranging accuracy," *Position Location and Navigation Symposium, 1990. Record. 'The 1990's - A Decade of Excellence in the Navigation Sciences'. IEEE PLANS '90., IEEE*, pp.314-321, 20-23 Mar 1990.

[20] Kan, K.K.H.; Chan, S.K.C.; Ng, J.K.-Y., "A dual-channel location estimation system for providing location services based on the GPS and GSM networks," *Advanced Information Networking and Applications, 2003. AINA 2003. 17th International Conference on*, pp. 7-12, 27-29 March 2003.

[21] S. Chakrabarti; A. Mishra, "A network architecture for global wireless position location services," *Communications, 1999. ICC '99. 1999 IEEE International Conference on*, vol. 3, pp.1779-1783, 1999.

[22] P.K. Enge; R.M. Kalafus; M.F. Ruane, "Differential operation of the Global Positioning System," *Communications Magazine, IEEE*, vol. 26, pp.48-60, Jul 1988.

[23] P. Enge; T. Walter; S. Pullen; K. Changdon; C. Yi-Chung; T. Yeou-Jyh, "Wide area augmentation of the Global Positioning System," *Proceedings of the IEEE* , vol.84, pp.1063-1088, Aug 1996.

[24] SAE J 1850 May 2001, “Class B Data Communication Network Interface,” *2004 SAE Handbook*, SAE International, 2004.

[25] W. Xing; H. Chen; H. Ding, "The application of controller area network on vehicle," *Vehicle Electronics Conference, 1999. (IVEC '99) Proceedings of the IEEE International*, vol. 1, pp.455-458, 1999.

[26] L. Hsu; M.W. Cheng; I. Niva, "Evolution towards simultaneous high-speed packet data and voice services: an overview of cdma2000 1/spl times/EV-DV,"

Telecommunications, 2003. ICT 2003. 10th International Conference on, vol. 2, pp. 1313-1317, 23 Feb.-1 March 2003.

[27] GSMWorld. [online]. Available: http://www.gsmworld.com/technology/faq.shtml

[28] S. Ni, "GPRS Network Planning on the Existing GSM System," *IEEE GLOBECOM*, Nov.-1 Dec. 2000, pp. 1432-1438.

[29] P.W. Baier, "CDMA or TDMA? CDMA for GSM?" *Personal, Indoor and Mobile Radio Communications, 1994. Wireless Networks - Catching the Mobile Future. 5th IEEE International Symposium on*, vol. 4, pp.1280-1284, 18-23 Sep 1994.

[30] M. Minea; G. Stan, "Field tests of a new integrated electronic system for vehicle monitoring, mobile data communications and e-commerce in road transportation," *Telecommunications in Modern Satellite, Cable and Broadcasting Service, 2003. TELSIKS 2003. 6th International Conference on*, vol. 2, pp. 449-452, 1-3 Oct. 2003.

[31] A. Okatan; A. Salih; C. Akpolat; K. Celik, "Micro-controller based vehicle tracing system via use of GPS and GSM," *Recent Advances in Space Technologies, 2003. RAST '03. International Conference on. Proceedings of*, pp. 605-609, 20-22 Nov. 2003.

[32] X. Feng, "Towards universal mobile caching," *International Workshop on Data Engineering for Wireless and Mobile Access. Proceedings of the 4th ACM international workshop on Data engineering for wireless and mobile access*, pp. 73-80, 2005.

[33] Garmin. [online]. Available: http://www.garmin.com/

[34] NMEA 0183 Standard for Interfacing Marine Electronic Devices, Version 2.0, *National Marine Electronics Association*, Mobile, AL, January 1992.

[35] OBDII Automotive Diagnostics. [online]. Available: http://www.obddiagnostics.com/

[36] Sony Ericsson. [online]. Available: http://www.sonyericsson.com/

[37] Cingular Wireless. [online]. Available: http://www.cingular.com/

[38] SAE J 1962 April 2002, "Diagnostic Connector Equivalent to ISO/DIS 15031-3: December 14, 2001," *2004 SAE Handbook*, SAE International, 2004.

[39] SAE J 1979 April 2002, "E/E Diagnostic Test Modes Equivalent to ISO/DIS 15031: April 30, 2002," *2004 SAE Handbook*, SAE International, 2004.

[40] PC104 Consortium. [online]. Available: http://www.pc104.org/

[41] T. Shanely, *Plug and Play PC Architecture*. Addison-Wesley Professional, July 26, 1995.

[42] Bully’s Bus Tracker. [online]. Available: http://www.bullybus.msstate.edu/

[43] Kontron. [online]. Available: http://us.kontron.com/

[44] Winsystems. [online]. Available: http://www.winsystems.com/index.html

[45] Dearborn Group. [online]. Available: http://www.dgtech.com/

[46] Tri-M Systems. [online]. Available: http://www.tri-m.com/

[47] X. Feng, “Towards real-time enabled Microsoft Windows,” *International Conference on Embedded Software Archive. Proceedings of the 5th ACM international conference on embedded software*, pp. 142-146, 2005.

[48] Windows Routing and Remote Access Services. [online]. Available: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/routremtaccserv.asp

[49] cURL. [online]. Available: http://curl.haxx.se/

[50] M. El-Sayed; J. Jaffe, "A view of telecommunications network evolution," *Communications Magazine, IEEE*, vol. 40, pp. 74-81, Dec 2002.

[51] Y. Lin; A. Pang, "Comparing soft and hard handoffs," *Vehicular Technology, IEEE Transactions on*, vol. 49, pp.792-798, May 2000.

[52] Micro/sys. [online]. Available: http://www.embeddedsys.com/index.html

[53] Trimble. [online]. Available: http://www.trimble.com/

[54] ScanTool.net. [online]. Available: http://www.scantool.net/

[55] ELM Electronics. [online] Available: http://www.elmelectronics.com/

[56] Autotap. [online]. Available: http://www.autotap.com/

[57] AspisOS. [online]. Available: http://www.aspisos.org/

[58] GPSd. [online]. Available: http://gpsd.berlios.de/

[59] Pppd. [online]. Available: http://www.samba.org/ppp/

[60] K. English, L. Feaster, *Community geography: GIS in action*, ESRI Press, 2003.

[61] R. Gibson, S. Erle, *Google Maps Hacks: Tips & Tools for Geographic Searching and Remixing*. O’Reilly, Jan 1, 2006.

[62] Google Maps Beta. [online]. Available: http://maps.google.com/

[63] Local Live Beta. [online]. Available: http://local.live.com/