Feasibility of a 802.11 VANET Based Car Accident Alert System

Author: Scott J. Weiner
Faculty Advisor: Gunar Schirner
Institution: Northeastern University
Program: Computer Engineering

***Abstract*—Currently, car to car communication is a very popular area of research because of the numerous application possibilities for driver safety and comfort. More specifically, Vehicular Ad-Hoc Networks (VANETs) are of interest because they have the potential for low latencies, and allow network connections between vehicles to stay active even when they are moving at high speeds. These positive aspects of VANETs allow for the development of time-critical vehicular applications such as a car accident alert system. This paper analyzes the feasibility of such a system by implementing a working prototype using the 802.11g standard, and by viewing latencies obtained by experimental results.**

I. Introduction

A. Application Description

This paper explores one of the many application possibilities associated with VANETs, a car accident alert system. Presently, the process for reporting car accidents is extremely slow. The accident first needs to be physically reported and then announced via a radio broadcast or a portable GPS unit before a driver is alerted. Figure 1 shows how the use of VANETs for this process will allow drivers within a certain range of the accident to immediately receive an alert that could prevent them from crashing as well. Additionally, the driver may choose to take an alternate route; which may result in traffic avoidance. For this project, I intend to explore the feasibility of this application by constructing a working prototype.

In order to provide the basic functionality described above, the car accident alert system will need to have two concurrently executing threads. One thread will first be responsible for detecting if the car has been in an accident. Then, if the car has been in an accident, this thread will wirelessly broadcast a warning message to every car within range. The other thread will listen for these messages, and then promptly warn the driver of impending danger. Finally, in order for the driver to make a decision about a route change, the second thread will provide the relative distance of the accident, as well as an approximation on the severity.

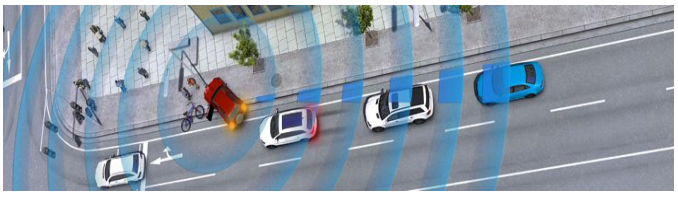


Fig. 1 – Illustration of Car Accident Alert System using Vehicular Ad-Hoc Networking [13].

B. 802.11 Standards and Considerations

In order to provide wireless ad-hoc capability to the system, the 802.11a, 802.11b, and 802.11g standard revisions were considered for the design. The major differences between these revisions are shown in the Table I.

Table I
802.11 Physical Layer Comparison

Standard	***802.11a***	***802.11b***	***802.11g***
Modulation Scheme	OFDM	DSSS	OFDM
Frequency	5.0 GHz	2.4 GHz	2.4 GHz
Potential Throughput	Up to 54Mbps	Up to 11Mbps	Up to 24Mbps

Following Table I, 802.11a has the highest potential throughput of the three standards, and uses the 5 GHz frequency as opposed to the 2.4 GHz band. The advantage of using the 5 GHz frequency is that currently fewer devices operate on it; which lowers the amount of interference. The drawback however, is that a higher frequency will reduce the transmission range. The 802.11b and 802.11g revisions are relatively similar except that 802.11b uses DSSS modulation and 802.11g uses OFDM.

In order to find the appropriate standard revision for the design, a simulation of the car accident alert system was performed using the ns2 network traffic simulator paired with the SUMO road traffic simulator [1]. Our results showed that in general, 802.11a had lower latencies for a single hop VANET because of its higher frequency. However, due to its limited range, 802.11a showed higher latencies than 802.11b or 802.11g in a multi-hop network because packets had to make more hops to arrive at vehicles further away.

Ideally, a multi-hop network would be advantageous over a single hop network because a sent message could potentially reach drivers that are miles away. Although this prototype does not implement a multi-hop VANET, I chose the standard to accommodate one so that the system would be scalable. Therefore, I chose to use 802.11g for two reasons. The first reason is because compared to 802.11a, packets sent using 802.11g will traverse fewer nodes to arrive at destinations further away. This in turn will show reduced latencies in densely populated areas. The second is because the additional range of the 2.4GHz band over the 5.0GHz band will potentially increase the number of drivers that receive the message [1].

II. System Implementation

A. Overview

In order to meet the application goals described in the application description section, I developed the basic architecture depicted in Figure 2.

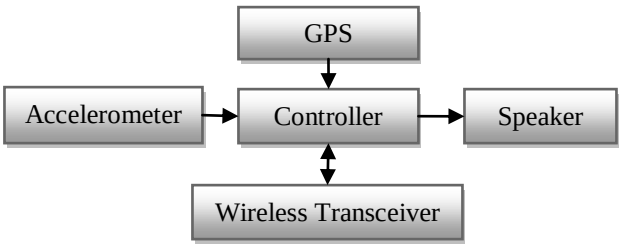


Fig. 2 – Basic System Architecture

In Figure 2, the system begins collecting raw acceleration data through an accelerometer. This data is then streamed into the controller, and subsequently processed in order to determine whether an accident has occurred. If such an event has been detected, the controller queries a GPS in order to obtain the car's current location. After that, the system uses a wireless transmitter to broadcast its location as well as the maximum acceleration experienced during the accident.

Concurrently, the controller also listens for these messages via a wireless receiver. If a message is received, the system obtains its current location from the GPS and use the remote location that is sent in the message to determine the relative distance of the accident. Finally, the controller provides an audio alert via a speaker which presents the driver with the distance and severity of the accident.

B. Hardware Architecture

First, in order to detect an accident, the accelerometer needs to have a high sample rate, and also needs to be capable of detecting very high g forces. This is because car accidents usually generate large g forces over a very short period of time [2]. The Insurance Institute for Highway Safety suggests using a DTS TDAS G5 acquisition system; which designed to sample at 10kHz in a 500g environment [3]. However, since it was infeasible for me to generate such high forces in such a short period of time, I chose to scale the acceleration, and to use an accelerometer that was suitable for in-house testing. Therefore, the device I used for this project was an Analog Devices ADXL345. The ADXL345 is highly customizable, and capable of a $\pm 16g$ range at sample rates of up to 3200 Hz. For this project, a sample rate of 100Hz was used in order to accurately capture an accident event; which is generated over a matter of seconds.

Next, a GPS had to be chosen that could quickly and accurately obtain a vehicle's current location. The speed of acquisition is important because the accident location needs to be broadcasted quickly in order to give drivers as much time as possible to react. In order to provide this quick response, I chose to read the most recent location from a buffer on the GPS rather than to wait for an acquisition of the most current location. However, using this approach with a normal 1Hz GPS at speeds of 100kph could result in errors of up to 60m. Therefore, I chose to employ a Locosys 20031 smart antenna module that uses the MediaTek MT3318 GPS. The update rate on this device is 5Hz which is five times faster than the typical 1Hz update rate that most GPS units provide. Using a 5Hz sampling rate will reduce the maximum error in this situation to around 20m.

Then, to allow for wireless ad-hoc communication between devices, a wireless transceiver module had to be chosen. For this project, I originally chose a Linksys WUSB54GSC wireless USB adapter, but certain implementation issues, discussed in later sections, prevented the use of this device. Instead, a backup Netgear WG111 wireless USB adapter was used. The adapter is dual-band and supports both the 802.11b and 802.11 g revisions.

Finally, in order to build the complex functionality of the controller, I chose a pure software implementation on an Analog Devices ADSP-BF526 Blackfin processor. Furthermore, I chose to use the Analog Devices ADSP-BF526 Easy Board [4] as my design platform, as it provides the additional hardware necessary for each of the interfaces used by the chosen peripherals.

C. Software Architecture and Controller Design

For this project, the majority of the system design falls within the software architecture. Figure 3 depicts the entire software architecture; which includes the application, the software/hardware interfacing, and the data flow through the system.

First, I chose to run the uClinux distribution [5] on top of the Blackfin processor. This operating system was used for a number of reasons, but mainly because uClinux has a wide availability of open source drivers, and provides support for many devices including the ones I have chosen. In turn, this simplifies the interfacing of the hardware devices; which is beneficial due to the time limitations of the project.

Inside of the operating system, all of the data sent to and from external devices travels through a centralized controller. This controller is further broken down into several components to allow for a modular design. First, the accident detector is responsible for processing the raw accelerometer data, and determining whether an accident has occurred. It accomplishes this task by taking a vector magnitude of the x, y, and z acceleration readings, and by comparing it to known car accident thresholds. After an initial accident threshold is exceeded, the detector will listen to the accelerometer for a

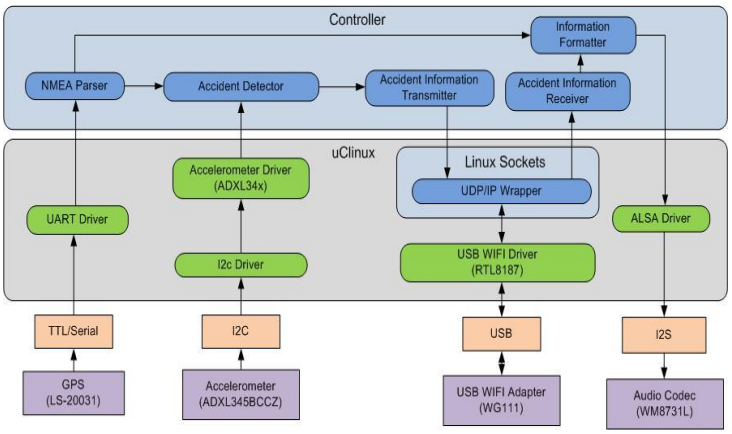


Fig. 3 - Software Architecture

short period of time afterwards in order to determine the maximum G force experienced. The accident thresholds and scaling are shown in Table II.

Table II
Accelerometer Thresholds for Accident Detection

Accident Severity	Actual Maximum G Range Represented [2]	Scaled Maximum G Range
No Accident	0-4g	0-4g
Mild Accident	4-20g	4-8g
Medium Accident	20-40g	8-12g
Severe Accident	40+g	12-16g

In Table II, the scaled maximum G range was split evenly into four categories for testing purposes. This was to allow for a certain type of event to be generated easily and reliably. On the actual device, the g range would be approximately set according to the second column in the table. These values were estimated based on full frontal crash test results from a 2007 Chevrolet Suburban [2]. Finally, after an accident has been detected, the detector passes the maximum G force magnitude to the accident information transmitter for broadcasting.

In addition to receiving the maximum G force experienced, the accident information transmitter also needs to obtain a location from the GPS in order to provide other devices with a reference for calculating the relative distance of the accident. First, in order to process the data sent from the GPS, it must be sent to a NMEA [6] sentence message parser. This parser allows for the latitude and longitude to be extracted from the entire NMEA message; which contains other irrelevant information. When all data has been gathered, the accident information transmitter constructs a datagram, and sends it through the UDP and IP network layers; where it will then be broadcasted via the MAC and physical layers of 802.11g.

Meanwhile, the accident information receiver is solely in charge of detecting these messages and passing them to the information formatter. After it receives the raw data, the formatter is responsible for presenting the information in a way that can easily be interpreted by the driver. This formatting includes calculating the distance from the accident, as well as estimating a severity rating. The distance between the two points is calculated using the Haversine formula [7], that is

$$haversin\left(\frac{d}{R}\right) = haversin(\phi_1 - \phi_2) + \cos(\phi_1)\cos(\phi_2)\, haversin(\Delta\lambda)$$

Where

- haversin is the haversine function, $haversin(\theta) = \sin^2(\theta/2)$
- d is the distance between the two points (along a great circle of the sphere)
- R is the radius of the sphere
- ϕ_1 is the latitude of point 1
- ϕ_2 is the latitude of point 2 and
- $\Delta\lambda$ is the longitude separation

Given this equation, d can be solved for by applying the inverse Haversine or by using the arcsin (inverse of sine) function.

$$d = R\ x\ haversin^{-1}(h) = 2R\ x \arcsin\left(\sqrt{h}\right)$$

Where

$$h = haversin\left(\frac{d}{R}\right)$$

Finally, after the data has been processed, the information formatter sends the data to the audio speaker which is spoken to the driver.

III. *Experimental Results*

A. *Implementation Challenges*

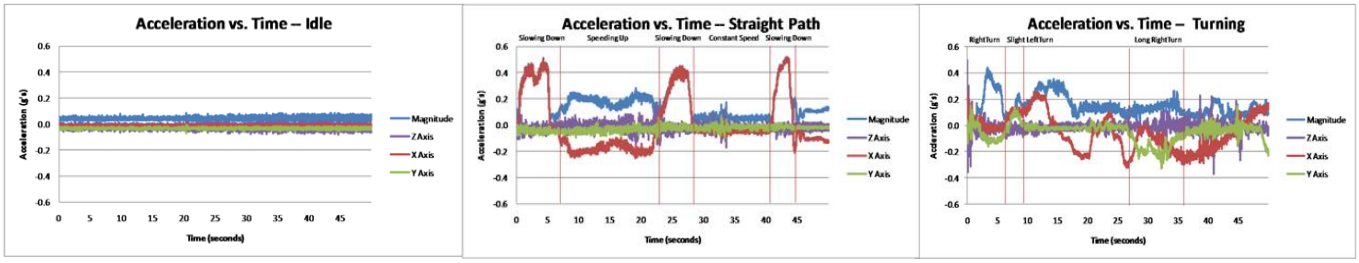
Due to the complexity of the design and the limited time frame for project completion, an implementation challenge arose when building the wireless functionality of the application. First, despite positive documentation [8], the Linksys WUSB54GSC wireless adapter I had originally chosen was unable to be detected by the operating system. Upon further investigation, the root cause was discovered to be in the MUSBHSFC USB 2.0 High-Speed Function Controller [9]. This was concluded by the lack of necessary kernel output required from the driver when the device was inserted. Due to time constraints, a fix could not be implemented, and a backup Netgear WG111 USB adapter was used instead.

However, a major disadvantage came from using the backup adapter. The RTL8187 [10] driver used for the WG111 USB adapter lacked ad-hoc support which was necessary for the project design. Since time restrictions prohibited driver modifications, infrastructure mode was used to provide an upper bound on latencies experienced in a single hop ad-hoc network.

B. *Experiment Setup*

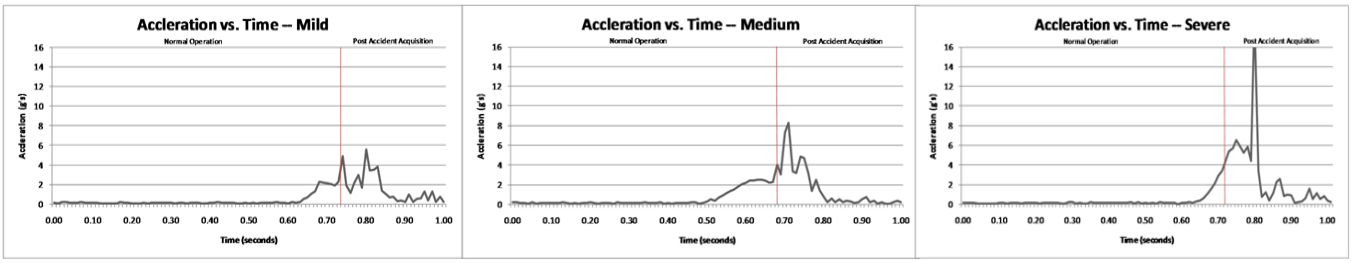
Unfortunately, a full scale experiment in a mobile vehicular setting could not be accomplished. This was due to the implementation challenges describe above, and to the lack of a second device. Therefore, individual experiments were conducted independently in order to construct a logical basis for feasibility. For experiments requiring two communicating devices, one application was run on the ADSP-BF526 EZ Board; while the same application was run on an Intel Core 2 Duo 2.93 GHz PC running Ubuntu Linux 9.10. Each platform was configured to use a static IP address on the same subnet in order to allow broadcast messages to be detected without additional routing. The platforms were then connected to a Zyxel NBG334W wireless router to enable a communication link. Additionally, since the GPS could not obtain a location fix indoors, a software stub was inserted that produced a NMEA GPS sentence stream when called.

In order to demonstrate the feasibility of the system, two types of results had to be obtained from the experiments. First, a correct system implementation had to be shown by



(a) (b) (c)

Fig. 4 – Experiment results for normal car operation



(a) (b) (c)

Fig. 5 – Experiment results for each accident type

demonstrating the functionality described in the application description section. This functionality includes the correct detection of an accident, the proper broadcasting of the accident location along with the severity, the detection of a sent message, and the proper alerting of the driver. In addition to the basic functionality, the driver also needs to be able to receive accident alerts quickly in order to have enough time to react. Therefore, latency was also measured between the time an accident occurred and the time of the accident alert.

C. *Experiment I – Normal Car Operation*

In order to validate that the accident threshold would not be exceeded during normal car operation, accelerometer measurements were taken during normal car activity. To gather this data, the system was mounted on the dashboard of a 2000 Honda Accord. First, I ran the system with an idle accelerometer, and collected acceleration data from the embedded target. This step was to provide a basis for comparison with normal car movement, and car accident data. The accelerometer was positioned so that the x axis was facing forward, and the z axis was pointed upwards. Additionally, 1g was added to the z axis in order to offset the effect of gravity. The results are shown in Figure 4a. From this figure, it is clear that the accelerometer obtains acceleration magnitudes of around 0g when idle. The slight acceleration shown in this figure is due to the accelerometer not being perfectly aligned; which causes some gravity to be seen in the x and y directions.

Next, measurements were taken during two driving tests. These tests included accelerating and breaking in a straight path, and making some normal left and right turns. For these tests, acceleration in the negative x direction meant the car was accelerating forward, and acceleration in the negative y direction meant the car was making a right hand turn. The results are shown in Figures 4b and 4c respectively. From these figures, it is clear that the acceleration does not exceed the accident threshold during normal operation. In fact, these measurements did not come close to the 4g threshold. Therefore, normal car operation would not trigger an accident event.

D. *Experiment II – Accident Simulations*

After the normal car operation measurements were taken, it was then possible to simulate each type of car accident, and to observe the system behavior and latency. The first type of accident, mild, is detected when a maximum of 4-8g's are experienced. For this experiment, the location of the car in the accident was set to be at 32° 17' 58.0806" N, 119° 27' 39.1206" E, and the location of the remote driver was set to be at 32° 17' 58.2"N, 119° 27' 39.2394"E. Additionally, the system clocks on each platform were synchronized, and timestamps were printed in order to show any latency. Finally, the application was run on both systems, and a mild acceleration was applied to the accelerometer. The accelerometer readings of the accident are shown in Figure 5a. The corresponding system output was as follows:

Client (*Accident*) side output:

```
Accident occurred at Sat Apr 24 16:21:00 2010
Accident printed at time Sat Apr 24 16:21:00 2010

****************************************
   CAUTION: NEARBY ACCIDENT

   Distance: 0.00 miles away
   Severity: Mild
****************************************
```

Host (*Remote*) side output:

```
Accident printed at time Sat Apr 24 16:21:00 2010

****************************************
   CAUTION: NEARBY ACCIDENT

   Distance: 0.16 miles away
   Severity: Mild
****************************************
```

Listing 1 – System output of a mild accident simulation

From these results, we can tell that the system correctly calculates the distance, and determines the appropriate category of the accident. Additionally, we can see that the latency is less than one second from when the accident is detected to when it is printed on the remote side. This latency will more than likely give drivers within range sufficient time to react. In order to validate my results, I simulated the exact same scenario using ns2 and SUMO. Network latencies of 802.11g in a single hop network were estimated around 0.002057 seconds [1]. Hence, the network latencies in a single hop VANET will not be a limiting design factor.

Next, the exact same experiment was conducted; only a medium accident was simulated instead of a mild one. This was accomplished by applying slightly more acceleration to the sensor. The accelerometer readings are displayed in Figure 5b.

Note that the post accident acquisition period allows for a medium accident to be detected rather than a mild one. The system output from this experiment is displayed below.

Client (*Accident*) side output:

```
Accident occurred at Sat Apr 24 20:23:40 2010
Accident printed at time Sat Apr 24 20:23:40 2010

**************************************
   CAUTION: NEARBY ACCIDENT

   Distance: 0.00 miles away
   Severity: Medium
**************************************
```

Host (*Remote*) side output:

```
Accident printed at time Sat Apr 24 20:23:40 2010

**************************************
   CAUTION: NEARBY ACCIDENT

   Distance: 0.16 miles away
   Severity: Medium
**************************************
```

Listing 2 – System output of a medium accident simulation

Again, the correct severity level and distance are calculated. Also, as expected, there were no changes in latency because only the accident severity was changed. Finally, the same experiment was conducted a third time, but with a severe accident. The acceleration data is shown in Figure 5c. The corresponding system output was as follows:

Client (*Accident*) side output:

```
Accident occurred at Sat Apr 24 16:20:29 2010
Accident printed at time Sat Apr 24 16:20:29 2010

**************************************
   CAUTION: NEARBY ACCIDENT

  Distance: 0.00 miles away
  Severity: Severe
**************************************
```

Host (*Remote*) side output:

```
Accident printed at time Sat Apr 24 16:20:29 2010

**************************************
   CAUTION: NEARBY ACCIDENT

   Distance: 0.16 miles away
   Severity: Severe
**************************************
```

Listing 3 – System output of a severe accident simulation

Once more, the accident location and severity were correctly calculated, and there was no change in latency.

IV. Conclusions and Future Work

In conclusion, this paper explored the feasibility of a VANET based car accident alert system application. Experimental results showed that latencies for a non-mobile single-hop implementation of the system were sufficient enough to give drivers enough time to react to the message. Although this simple implementation gave desirable results, a real implementation of this application has several other challenges that need to be addressed.

Since the mobility of cars can be very sporadic, connections between them will be constantly changing. Consequently, the physical layer, routing protocol, and topology of the network must be carefully constructed in order to maintain high performance in a constantly changing network. Currently, one IEEE task group is developing the 802.11p revision for wireless access in vehicular environments (WAVE). This revision attempts to provide the minimum set of specifications required in rapidly changing communications environments [11]. Future work will analyze the feasibility of single and multi-hop VANET applications using this protocol, and more rigorous mobile experiments will be conducted.

Additionally, network security is a desired feature when it comes to VANETs. Sensitive information such as vehicle location, time, and internal car sensor data has the potential to be viewed by other drivers. If this data were to be linked to the driver's identity by other network participants, it would allow them to track a particular driver's vehicle [12]. The addition of such a security layer, however, could produce latencies that exceed the design restrictions of VANETs. Thus, various security protocols will be incorporated into future designs in order to further evaluate the feasibility of VANET applications.

References

[1] Abdulla Al-Ali and Scott Weiner, "A Performance Analysis of 802.11 Wireless Standards in a Multi-Hop Vehicular Ad-Hoc Network," Northeastern University, Boston, Technical Report 2010.

[2] General Motors, "New Car Assessment Program Frontal Impact Test," MGA Research Corporation, Burlington, Crash Test Report NCAP-MGA-2006-012, 2006.

[3] Insurance Institute for Highway Safety, Frontal Offset Crashworthiness Evaluation: Offset Barrier Crash Test Protocol, May 2008.

[4] Analog Devices. (2010, May) ADSP-BF526 EZ-Board for the ADSP-

BF522, BF524, BF526 Blackfin Processors. [Online]. http://www.analog.com/en/embedded-processing-dsp/blackfin/bf526-ezbrd/processors/product.html

[5] Analog Devices. (2010, April) Blackfin Koop. [Online]. http://blackfin.uclinux.org/gf/

[6] Klaus Betke, The NMEA 0183 Protocol, August 2001, http://www.cs.put.poznan.pl/wswitala/download/pdf/NMEAdescription.pdf.

[7] J. Noel, T. Montavont, "IEEE 802.11 Handovers Assisted by GPS Information," in *Wireless and Mobile Computing, Networking and Communications, 2006. (WiMob'2006). IEEE International Conference on Wireless*, Montreal, 2006, pp. 166-172.

[8] (2010, March) Linux Wireless Wiki. [Online]. http://wireless.kernel.org/en/users/Drivers/rndis_wlan

[9] Inventra. (2005) MUSBHSFC Datasheet. Datasheet.

[10] (2010, March) Linux Wireless Wiki. [Online]. http://wireless.kernel.org/en/users/Drivers/rtl8187

[11] IEEE. (2005) Status of Project IEEE 802.11 Task Group p. [Online]. http://grouper.ieee.org/groups/802/11/Reports/tgp_update.htm

[12] Elmar Schoch and Christian Maihofer Tim Leinmuller, "Security Requirements and Solution Concepts in Vehicular Ad Hoc Networks," in *The Fourth Annual Conference on Wireless On demand Network Systems and Services*, Obergurgl, 2007.

[13] Car 2 Car Communication Consortium. (2010, May) Car 2 Car Communication Consortium Website. [Online]. http://www.car-to-car.org/

[14] The Insurance Institute for Highway Safety, Frontal Offset Crashworthiness Evaluation: Offset Barrier Crash Test Protocol, May 2008.