Exploring Smartphones As WAVE Devices

Jeman Park, Jihye Kim, Seungho Kuk, Yongtae Park and Hyogon Kim
Korea University
Seoul, South Korea
{parkjeman, kjihye, shkuk, ytpark, hyogon}@korea.ac.kr

***Abstract*—In this paper, we explore the possibility of using smartphones as WAVE devices. When it is expected to take more than a decade from now to deploy the WAVE technology in full scale, smartphones running the WAVE protocol stack can bring the benefit of the technology earlier. In particular, we design and implement the lower layers of the WAVE stack in software defined radio (SDR) on smartphone platform and test its vehicle-to-vehicle (V2V) performance in the real driving scenarios. We also investigate the performance in the vehicle-to-pedestrian (V2P) communication context.**

Keywords—Wireless Access in Vehicular Environment(WAVE), smartphone, software defined radio (SDR), V2V, V2P

I. INTRODUCTION

The IEEE Wireless Access in Vehicular Environment (WAVE) technology [1], [2], [3], [4] is finally around the corner. The U.S. Department of Transportation successfully completed a massive deployment test [5], and announced its plan to move forward with the vehicle-to-vehicle (V2V) technology for light vehicles in 2014 [6]. Upon deployment, future vehicles equipped with the WAVE devices operating in the 5.850 – 5.925 GHz dedicated short-range communications (DSRC) frequency band are expected to reduce the crashes not involving impaired drivers or mechanical failure by 80 percent [5]. However, such safety benefits will remain unrealized until a critical mass of vehicles on the road use the technology, which could take more than a decade. Achieving the critical mass should come first to enable the 'network effect' [7]. This lack of day-1 benefit could hamper the deployment by disincentivizing involved parties. Nevertheless, given the importance and attractiveness of safety-related vehicular networking applications, we need to strive in various ways to facilitate early realization of WAVE technology.

As part of the efforts to catalyze the oncoming of the WAVE technology, we explore in this paper the capabilities of existing technologies to showcase WAVE services to drivers at scale now as we wait for the WAVE deployment to begin and eventually reach the critical mass. To this end, we envision that smartphones can act as the platform to quickly realize the WAVE technology, especially with their already large user base and several commercialized traffic-related applications. Unfortunately, the first technical roadblock to support WAVE on smartphones is the fact that none of today's smartphones natively support the IEEE 802.11p protocol. Thus in this paper, we explore if they can support WAVE communication if the radio front were provided. In order to test the hypothetical scenario, we implement the physical (PHY) layer for the WAVE, namely IEEE 802.11p [4][1] in software defined radio (SDR). Then we provide the radio front externally through smartphone's USB port so that the IEEE 802.11p SDR can use for WAVE communication. With a prototype implementation, we test WAVE communication with between prototype devices, as well as with a commercial WAVE on-board unit (OBU). Below, we discuss the details of the implementation and the test results.

II. WAVE STACK FOR SAFETY COMMUNICATION

The IEEE WAVE takes a dual stack approach (Fig. 1, left). For non-safety communication, it uses TCP/IP stack over which applications can run services on one of six Service Channels (SCHs). Safety-related messages, on the other hand, use a dedicated protocol called WAVE Short Message Protocol (WSMP) that is defined by the IEEE 1609.3 standard [2]. WSMP takes the place of TCP/IP in the protocol hierarchy, and can use the Control Channel (CCH) in addition to the aforementioned six SCHs. For both cases, however, the lower layers are common, and they are provided by the IEEE 802.11p and the IEEE 1609.4 protocols that supplement the IEEE 802.11p for multi-channel operation [3]. For instance, when a vehicle periodically transmits Basic Safety Messages (BSMs) [8] to its neighbors, it encapsulates the BSM in WSMP and pushes it to the lower layers. The security credential is attached to the message, which is addressed by the IEEE 1609.2 standard [1].

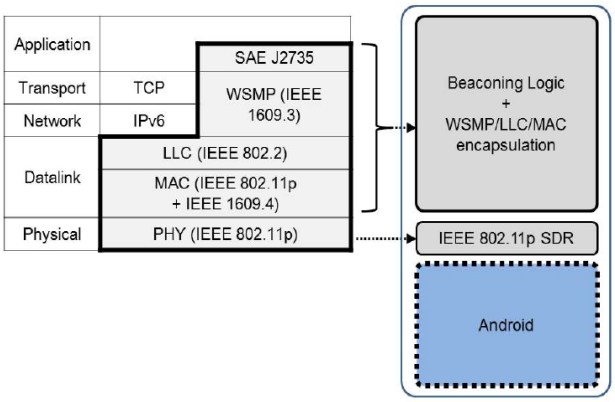

Fig. 1. WAVE protocol stack and corresponding software implementation

To implement the stack on the smartphone platform, we implement two modules of software, both on the user space (Fig. 1, right). The first includes the BSM generation application, encapsulations for WSMP, Logical Link Control (LLC), and medium access control (MAC) layer. The second module is the SDR of the IEEE 802.11p protocol. The MAC PDU (MPDU) generated by the first module is given to the

978-1-4799-8091-8/15/$31.00 ©2015 IEEE

SDR that performs signal processing on it and transmits through the external radio front in the IEEE 802.11p protocol. Below, we discuss the design and the implementation of the IEEE 802.11p SDR in more detail.

III. WAVE IN SOFTWARE DEFINED RADIO

The IEEE 802.11p PHY layer is essentially a half-clocked 802.11a PHY. The only major difference other than its licensed frequency band is that IEEE 802.11p uses 10MHz bandwidth instead of 20MHz. We make use of the publicly available IEEE 802.11a SDR called SORA [9] and adjust it for the 10MHz band operation. Also, since the code was originally developed for Intel CPU architecture, we ported it to the ARM architecture used on the test smartphones. Since we found that using SIMD instruction sets is crucial to the signal processing performance, we use the ARM NEON extension in our implementation.

It is well known that among the eight PHY data rates available in IEEE 802.11p, 6Mbps is the most stable mainly because 3Mbps is too slow to finish packet transmission within the channel coherence time in the vehicular communication environment [10]. So for our implementation, we use 6Mbps as our transmission rate. For this rate, the modulation is QPSK, and the channel coding ratio is 1/2. We use two threads for the SDR, where a separate thread is assigned to the Viterbi decoder, as it is known to be the heaviest workload in the baseband processing on PC testbeds [9], [11].

A. Approach: WAVE SDR as app

We implement our smartphone SDRs as an app instead of a device driver in Android kernel. There are a few reasons. First, it is to harness the superb software distribution infrastructure there is for smartphone platforms already, i.e., app stores. It makes SDR lend itself easily to creation, maintenance, and distribution. If necessary, for instance, we can update or improve the SDR app in any parts and reload it to the app store. As with any other app, the app store infrastructure automatically handles the distribution to users' smartphones of updated versions. As a matter of fact, we registered our WAVE SDR in Google Play store, which can be downloaded to an Android smartphone and run with an external radio through USB port. We believe this app-based approach can help expedite the deployment of WAVE SDR on smartphone platforms. Second, it is to eliminate OS dependency. Considering how the OS is updated in smartphones today, this concept is important in two aspects: delay until the next OS update, and more seriously, whether the SDR will be included at all therein. The detachment of the SDR from the OS can bring us the degree of freedom in these aspects.

B. Implementation details

Fig. 2 shows the implemented software architecture. To put this software architecture in perspective, we place it against the standard Android software framework in the backdrop. The two modules of our software are internally connected by JAVA Native Interface (JNI). As we depicted in Fig. 1, the higher-layer module contains the beaconing logic that determines the WSMP parameters (DSRC channel, Tx power, Rx sensitivity) as well as the beaconing frequency that can be used for congestion control. The lower-layer module contains the IEEE 802.11p SDR. It is implemented as an embedded library, and performs the baseband processing, packetization, and CRC computation among others. The SDR also manages the accesses to the radio front end. The native library in the SDR implements the USRP Hardware Driver (UHD) and the USB driver using libUSB in order to access and control the radio front end. When the SDR processing logic calls the USRP Driver, the USRP Driver calls the USB Driver to talk to the external radio front. Since the SDR is self-contained with necessary libraries, it does not require any additional support from Android or the Linux kernel to run on smartphones, except that we must give the SDR app the root privilege when prompted by the app.

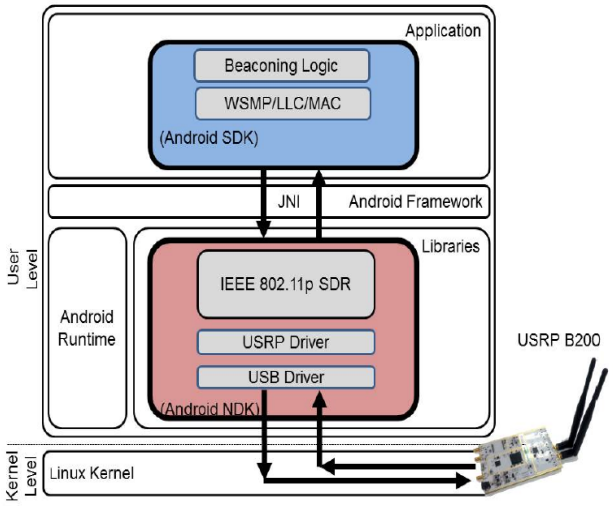

Fig. 2. WAVE software implementations on Android platform

For the external radio front, we use USRP B210 [12]. It provides continuous RF coverage from 70MHz to 6GHz, which includes the DSRC band. Note that it does not perform digital signal processing, but only serves as a rudimentary radio front that performs frequency up/down-sampling and analog-digital and digital-analog conversion (ADC/DAC). The SDR module running on the smartphone application processor (AP) performs all baseband signal processing for the physical (PHY) layer. The USRP B210 is connected to the smartphone through the USB On-The-Go (OTG), which allows the smartphone to act as a host to the USRP device. To communicate with USRP, SDR application uses Android Native Development Kit (NDK). We ported the libUSB library to Android so that Android native library may use USB. We also ported the USRP Hardware Driver (UHD) for Android native library.

To use USRP B210, the SDR module has firmware and FPGA images, so the path to image files to use the external radio front USRP B210 [12] is automatically set up upon app installation, and the images are automatically loaded on USRP B210 when the application opens USB connection. The FPGA image performs two tasks. First, it buffers the digital samples generated by the ADC so that the smartphone can read them out through the USB interface. Second, it sends the digital samples from the smartphone to the DAC so that analog transmission can take place. The USB interface version that the

smartphone supports is 3.0, and the USRP B210 also supports USB 3.0. For OTG interface, however, smartphones today support only USB 2.0, and that limits us to USB 2.0 for our smartphone supports is 3.0, and the USRP B210 also supports USB 3.0. For OTG interface, however, smartphones today support only USB 2.0, and that limits us to USB 2.0 for our experiments at the maximum of 32 MB/s bandwidth, considering overheads while the maximum raw throughput of USB 2.0 is 480Mbps. The ADC and DAC in USRP B210 can process up to 61.44 mega samples per second (MS/s) [12]. Last but not least, USRP B210 is powered by the smartphone battery through the USB OTG connection in our prototype. Although B210 can be powered by a separate source, we use this test configuration to explore the possibility of an external radio module that can be clicked on in case the smartphones will not support an in-device shared radio front end. Since USRP B210 is a multipurpose device, a tailored radio front end for the smartphone SDRs will require less power.

C. Packet processing performance

For real-time MAC/PHY processing, we must meet the timing requirements in encoding, decoding, response generation (e.g. ACK and CTS), and event trigger timing (e.g. CCA) [9]. The encoding and the decoding parts should be done in real-time in order to get the full transmission rate. As to ACK and CTS, they are not used in the WAVE safety communication because safety messages are broadcast. In this section, we perform IEEE 802.11p packet transmission and reception using the SDR. We use four different smartphone platforms in the test: Galaxy Nexus, S2, S4, and S5. Then we let the prototype above communicate with a commercial OBU. Fig. 3 shows the test setup.

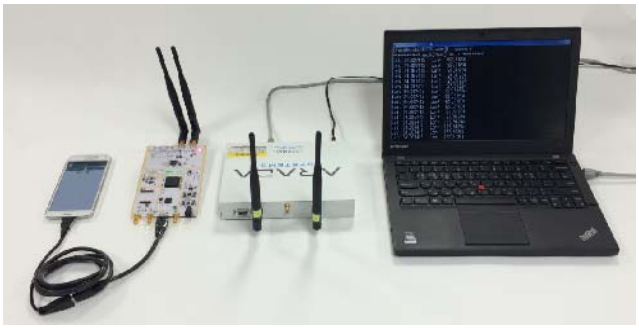

Fig. 3. Smartphone WAVE device communicating with a commercial WAVE OBU

Fig. 4 shows total encoding and decoding times of the implemented SDR, when processing a IEEE 802.11p frame carrying 1KB of application payload. We first notice that the decoding times are shorter than encoding times. Note that in terms of SDR execution mechanism, encoding is not necessarily the reverse of decoding, which leads to a difference in processing time. The decoding operations run almost continuously, as we do not know when a packet will arrive from another node. But encoding is executed only when we have a packet to send. It affects the performance of some components in the given hardware platform. For instance, the cache miss ratio is higher in encoding. In decoding, however, lookup tables, fetched sample blocks and states loaded in cache are frequently accessed, keeping the cache warm. In case we use the USB I/O to connect to the external radio, getting the USB ready for use can incur latency. For decoding, on the other hand, continuous operation amortizes the cost.

3Mbps Req. (ms)	Decoding	Encoding	Android version
	2.667	**2.667**	
Nexus	4.741	5.717	Icecream Sandwich
S2	4.594	6.812	Gingerbread
S4	2.078	2.291	Jellybean
S5	1.659	2.196	KitKat

(a) Total execution times per packet for 3Mbps

6Mbps Req. (ms)	Decoding	Encoding	Android version
	1.333	**1.333**	
S5	0.743	1.213	KitKat

(b) Total execution times per packet for 6Mbps

Fig. 4. Packet processing performance of the prototype

From Fig. 4(a), we also notice that both encoding and decoding times are within required time limit on recent smartphones (S4 and S5). Fig. 4(b) shows the times for 6Mbps rate. As we mentioned above, this is the most stable PHY data rate on the road environment. Here also, for both encoding and decoding, the times are within the required limit for S5. The execution time is approximately halved compared with the 3Mbps case, because 6Mbps in IEEE 802.11p uses QPSK instead of BPSK. Each symbol carries 2 bits instead of 1, so the number of symbols to process given the same packet size of 1KB is halved. Notice that the time requirement is also halved due to the doubled data rate.

In essence, the results we obtained in Fig. 4 means that the IEEE 802.11p SDR on today's smartphones can deal with real-time transmission and reception of WAVE packets.

D. Power consumption

Here, we measure the power consumption of the smartphone-based WAVE communication, using Monsoon power monitor. Between transmission and reception, the latter is more costly due to the continuous processing as it does not know when a packet will arrive. So we measure the reception case here. Fig. 5 shows the result.

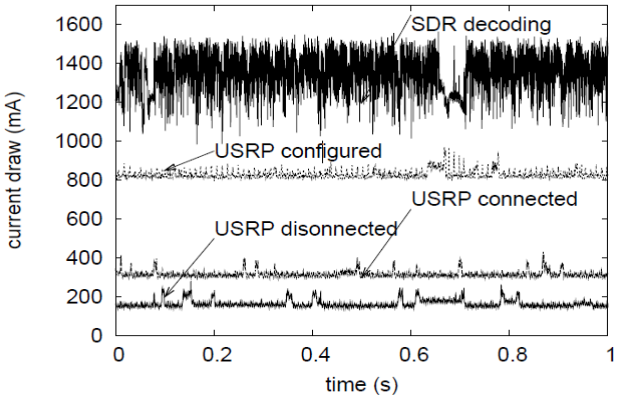

Fig. 5. Current draw for WAVE operation

When the USRP B210 is disconnected, the current draw caused by the idling SDR app is approximately 200mA. Next, when we connect the external radio front, it boots the draw to approximately 430mA. By configuring the USRP B210 for communication, it exceeds 800mA. Finally, when the WAVE beacons begin to arrive from the commercial WAVE device and full decoding logic is engaged, the additional current draw exceeds 500mA. So in total the IEEE 802.11p draws 1360mA when it receives WAVE beacons. Considering the battery capacity of 2800mAh in Galaxy S5, our system will completely drain the battery in just over 2 hours. But in case we use WAVE applications in vehicles, there is usually a stable power source that the smartphone can use, such as cigar jack. So this operating lifetime may not be a critical one. Moreover, if the radio front is internally provided inside the smartphone or if it can be made into a small form factor external module (not a general board likes USRP), the current draw could be further reduced. A final note is that such USB power supply interface for the smartphone provided by the vehicle can encase a 5GHz radio front dedicated for WAVE communication. In this case, we could bring the benefit of the technology even today, with a large deployment base which is the smartphones.

IV. APPLICATIONS

In this section, we test the prototype device in vehicular environment. We conduct two experiments. In the first, we equip two vehicles each with a prototype smartphone WAVE device and let them communicate, while varying distance, BSM size, and messaging frequency. We observe the packet delivery ratio (PDR) in each test case. On the transmitter side, the Tx gain for the USRP B210 board is 90dB, which is measured to be 4.8dBm on spectrum analyzer, to which we attach an amplifier (RFbay LNA-8G) and an antenna to add 4.1dB and 4.5dB, respectively. So, the final Tx power is 13.4dBm. On the receiver side, the Rx sensitivity is set to -87.8dBm. Note that these parameters are far from what is prescribed for WAVE systems. In particular, the Tx power is significantly smaller than the default WAVE device Tx power of 23dBm. Therefore, it will lead to shorter communication distance than is required in WAVE applications. However, the focus of this paper is in demonstrating the capabilities of today's smartphones to perform signal processing for WAVE communication, and to run WAVE applications. We believe that the development of a dedicated or shared 5GHz radio front that fits the WAVE communication requirements will follow if the software WAVE approach proves viable. In the second experiment, we let a pedestrian carry the smartphone-based WAVE system, while placing a commercial WAVE OBU on a nearby vehicle. With this configuration, we test the scenario where vehicle-to-pedestrian (V2P) communication is used to prevent car accidents involving pedestrians.

A. Vehicle-to-vehicle (V2V) communication

In the first experiment for V2V communication, we park the two vehicles each equipped with the smartphone-based WAVE device in a parking lot. For the base scenario, we let the one device on the first vehicle transmit BSMs at 20Hz, where the BSM message size is 41 bytes. We do not include the security credential [1] in this experiment. After encapsulations in WSMP, LLC, and MAC layers, the MPDU size is 93 bytes. Then we vary the messaging frequency, BSM size, and the inter-vehicle distance. For a total of 8,000 BSMs transmitted in each case, we measure the PDR at the device on the second vehicle. Fig. 6 shows the measurement results. Although the communication distance is not large, the PDR can exceed 70% in the closest distance and the minimal message size and messaging frequency. It shows that the software implementation has a potential for real V2V communication.

Distance	10m	25m	37.5m	50m
Avg.	66.55	61.98	37.55	0

(a) PDR in distance, f = 20Hz, W = 41B

Packet Freq.	5Hz	10Hz	20Hz	30Hz	50Hz	100Hz
Avg.	73.32	63.38	66.55	53.42	62.77	58.45

(b) PDR in messaging frequency, d = 10m, W = 41B

WSM size (MPDU size)	21B (73B)	31B (83B	41B (93B)	51B (103B)	61B (113B)
Avg.	71.08	69.90	66.55	65.73	63.67

(c) PDR in message size, d = 10m, f = 20Hz

Fig. 6. PDR in static V2V experiment (%)

Next, we drive two vehicles each equipped with the smartphone WAVE device. Due to the small communication distance we saw (Fig. 6(a)), we keep the inter-vehicle distance as small as possible, by tailgating when safe or by running in parallel in adjacent lanes. In the first experiment in the mobile case, we measure the PDR between the two smartphone-based WAVE devices while varying inter-vehicle distance d and BSM size W, with the messaging frequency fixed at 20Hz. We drove the vehicles on an urban highway (Jayuro, Paju City, Korea). Since we cannot precisely control the distance while driving on the real highway, we postprocess the communication log where the receiver application records the GPS coordinate of each successful reception of a BSM. Since the BSM itself carries the GPS coordinate of the transmitter, we can classify the PDR performance in several bins of distances between the GPS coordinates of the transmitter and the receiver as in Fig. 7(a). We first observe that the PDR has visibly worsened compared with the static case. For instance, with W = 41B, the PDR was 66.6% at d = 10m for the static case whereas we can only get 18.6% for the mobile case. This is likely due to the increased fading and Doppler effect in mobility. Second, we observe that the BSM size variation for each distance is not as consistent a factor to affect the PDR as in the static case.

In the second mobile experiment, we fix the BSM size at 41B, and the messaging frequency at 20Hz. We investigate if other concurrently running app on the smartphone affects the WAVE SDR communication performance. This is because the WAVE SDR may run short of CPU cycles due to other apps that might also run when driving. For this purpose, we let the smartphone either run a sat-nav app or turn it off while the V2V communication is ongoing. Fig. 7(b) shows the result. We do not observe a clear winner in the experiment, although we see the PDR at the closest distance is better when the sat-nav app is not running.

WSM size (MPDU size)	21B (73B)	31B (83B)	41B (93B)	51B (103B)	61B (113B)
0-5m	52.07	49.66	57.95	63.61	44.97
5-10m	13.39	22.02	18.62	13.98	11.43
10-15m	1.77	4.50	0.55	0.97	1.86
15-20m	0.22	0.72	0.55	1.16	1.86
20m-	0.00	0.00	0.00	2.05	0.31
Avg.	14.83	17.63	8.46	19.10	8.26

(a) PDR in distance, f = 20Hz

	Sat-nav Off	Sat-nav On
0-5m	57.95	49.36
5-10m	18.62	30.76
10-15m	0.55	11.68
15-20m	0.55	0.14
20m-	0.00	0.00
Avg.	8.46	28.54

(b) PDR in distance, f = 20Hz

Fig. 7. PDR in mobile V2V experiment (%)

B. Vehicle-to-pedestrian (V2P) communication

In this scenario, we assume that a pedestrian holds or carries a smartphone, and the smartphone beacons using WAVE technology (which may be automatically turned on when the user looks at the screen or listens to music while walking). Passing vehicles with WAVE OBU can notice the beacons and use the pedestrian location information to duely warn the driver when there is a risk of collision. Conversely, the WAVE app on the smartphone may warn the pedestrian of the incoming vehicle. For safety, we perform the experiment in non-mobile setting. The Tx power at the OBU is 27.5dBm, which includes the 4.5dB antenna gain. The Rx sensitivity of the OBU is -92dBm. Fig. 8 summarizes the results. In the figure, the area of a blob represents the standard deviation of the GPS values communicated through the WAVE protocols. Interestingly, the GPS precision on the commercial OBU device is not clearly superior to that of the smartphone. Considering the PDRs we observe in Section IV-A along with the enhanced Tx power and Rx sensitivity with the use of commercial OBU devices, the PDR is less of an issue than the GPS precision. Since detecting the near- or expected collision between vehicles and pedestrians without excessive false positives or false negatives requires distance measurement precision, GPS technology should be further enhanced for this type of application.

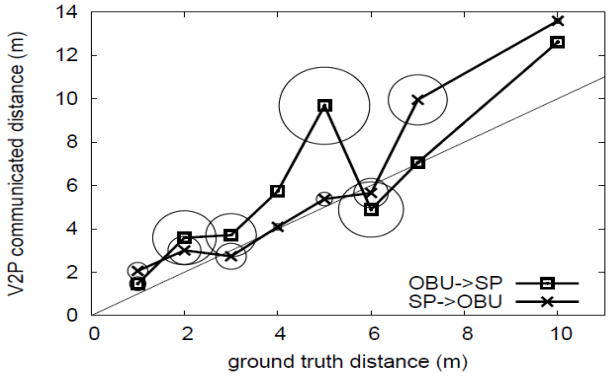

Fig. 8. Precision of GPS coordinates communicated through vehicle-to-pedestrian (V2P) communication

V. CONCLUSION

Although today's smartphones do not natively support the IEEE 802.11p protocol, they are well prepared to run it in software defined radio. This paper demonstrates the potential by implementing it as a downloadable app, and shows it can communicate with a commercial WAVE device. If only a radio front operating on 5GHz band becomes available for SDR, it can be used to facilitate the introduction of the WAVE technology that is expected to take more than a decade.

ACKNOWLEDGMENT

This work was supported by the Ministry of Education, Science and Technology of Korea through the Mid-career Researcher Program under NRF Grant 2011-0028892.

REFERENCES

[1] IEEE, IEEE Standard for Wireless Access in Vehicular Environments (WAVE) — Security Services for Applications and Management Messages, IEEE Std 1609.2-2013, 2013

[2] IEEE 1609 WG, IEEE Standard for Wireless Access in Vehicular Environments (WAVE) — Networking Services, IEEE Std 1609.3-2010, 2010.

[3] IEEE 1609 WG, IEEE Standard for Wireless Access in Vehicular Environments (WAVE) — Multi-channel Operation, IEEE Std 1609.4-2010, 2010.

[4] IEEE 802.11 WG, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, Amendment 6: Wireless Access in Vehicular Environments, IEEE Std 802.11p-2010, 2010.

[5] Natl. Highway Transportation Safety Agency (NHTSA). Improving Safety and Mobility Through Connected Vehicle Technology U.S. Department of Transportation Safety Pilot, 2013. Available at http://www.its.dot.gov/safety pilot/pdf/safetypilot nhtsa factsheet.pdf.

[6] National Highway Traffic Safety Administration. U.S. Department of Transportation Announces Decision to Move Forward with Vehicle-to-Vehicle Communication Technology for Light Vehicles, February 2014.

[7] H. Hartenstein and K. P. Laberteaux. A Tutorial Survey on Vehicular Ad Hoc Networks. *IEEE Communications Magazine*, June 2008.

[8] SAE International, Dedicated Short Range Communications (DSRC) Message Set Dictionary, SAE J2735, November 2009.

[9] K. Tan, H. Liu, J. Zhang, Y. Zhang and J. Fang. Sora: High-Performance Software Radio Using General- Purpose Multi-Core Processors. *CACM*, 54(1), January 2011.

[10] F. Bai, D. D. Stancil, and H. Krishnan. Toward understanding characteristics of dedicated short range communications (DSRC) from a perspective of vehicular network engineers. In *Proceedings of ACM Mobicom*, 2010.

[11] J. Kim, S. Hyeon, , and S. Choi. Impelementation of an SDR System Using Graphics Processing Unit. *IEEE Communications Magazine*, March 2010.

[12] Ettus Research. USRP B210 Information Sheet. Available at https://www.ettus.com/product/details/UB210-KIT.