Characterization and performance evaluation of 802.11p NICs

Characterization and Performance Evaluation of IEEE 802.11p NICs

Francesco Raviglione
Politecnico di Torino
Torino, Italy
francesco.raviglione@studenti.polito.it

Marco Malinverno
Politecnico di Torino
Torino, Italy
marco.malinverno@polito.it

Claudio Casetti
Politecnico di Torino
Torino, Italy
claudio.casetti@polito.it

ABSTRACT

The automotive industry is scrambling to equip high- and middle-segment vehicles with communication capabilities that will enable the commercialization of connected vehicles in the near future. Although both IEEE and 3GPP are developing new solutions, it is likely that IEEE 802.11p will be the protocol of choice. In this paper, we develop an open-source testing framework for IEEE 802.11p cards and characterize the performance of Unex DHXA-222 cards in terms of throughput and packet loss, especially when different traffic classes, hence access categories, are selected.

CCS CONCEPTS

• **Networks** → **Network measurement**; • **Hardware** → **Wireless devices**; • **Computer systems organization** → *Embedded systems.*

KEYWORDS

802.11p, Vehicular networks, OpenWrt

ACM Reference Format:
Francesco Raviglione, Marco Malinverno, and Claudio Casetti. 2019. Characterization and Performance Evaluation of IEEE 802.11p NICs. In *Proceedings of 1st ACM Workshop on Technologies, mOdels, and Protocols for Cooperative Connected Cars (TOP-Cars) (TOP-Cars'19).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The emergence of many projects on self-driven vehicles has renewed the interest in technologies for connected vehicles as well. We are thus witnessing on-going standardization efforts both within IEEE, seeking to extend the now decade-old IEEE 802.11-based V2X (Vehicle-to-everything) standards, as well as within 3GPP, aiming at leveraging the upcoming 5G technology. The current IEEE 802.11p wireless access in vehicular environments (WAVE) technology for V2X applications is based on IEEE 802.11-2016 [7] and it is designed to operate in the 5.9 GHz band. IEEE is considering the introduction of a new amendment, termed IEEE 802.11bd [12], to cater for emerging use cases for V2X communication technology and strengthen its foothold on 802.11-based technology for V2X applications. However, before any IEEE 802.11bd card hits the market, legacy IEEE 802.11p is likely to be the hardware of choice for car manufacturer seeking to equip their latest models with WAVE communication capabilities. This explains the need to a clear characterization of IEEE 802.11p cards, starting from a controlled lab environment, which is the topic of this paper. In particular, we chose to focus as much as possible on cards that support open-source drivers, i.e., the *ath9k* Linux driver and an open source software platform such as OpenWrt. For this purpose, we have made our source code available through GitHub.

The main contributions of this paper include an open source framework for testing of V2X solutions using Unex DHXA-222 and 802.11p, the performance evaluation of UDP throughput and packet loss observed between two cards in a laboratory environment, the characterization of the throughput achievable when data belonging to different Access Categories (AC) are exchanged between the cards, and the analysis of the received power sensitivity.

2 RELATED WORK

Vehicular communication platforms are getting more and more integrated into the automotive industry. The challenge of equipping vehicles with standard-compliant OBUs, able to communicate in the 5.8-5.9 GHz frequency band, was undertaken by major car manufacturers all over the world. The adoption of 802.11p-compliant equipment was recently announced by the Volkswagen Group [5] and by Toyota and Lexus [4]. There are some commercial ready-to-use solutions available on the market that are able to support vehicular communication and the 802.11p protocol. In the awareness that this will not be a complete and exhaustive list, some available solution are listed here, taking as reference, and updating, the list in [9].

- ARADA Systems, focusing on solutions for connected vehicles, has produced the LocoMate series, providing hardware for both RSUs and OBUs.

TOP-Cars'19, July 2019, Catania, Italy
2019. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

- Cohda Wireless developed MK5, which provides a complete solution for OBUs and RSUs. This system implements both the IEEE and ETSI stacks.
- Kapsch TrafficCom, an austrian group developing solutions for connected vehicles, targeted DSRC 5.9 GHz devices, such as the Onboard unit KVE-3320 V2X ECU, supporting all the V2X standards. This company also has a group developing V2X software.
- Marvell is instead developing a family of automotive wireless transceivers (Marvell 88W8987xA), integrating IEEE 802.11ac and 802.11p for V2X and ADAS applications.

Being these commercial devices, their cost is high and the customization possibility is often null. For these reasons, open solutions are typically preferred in the academic world, since they have the advantage of being more accessible and customizable for research and testing purposes. Two of the main companies developing hardware that can be used to build custom V2X solutions are Unex and PC Engines. Unex produces several WNICs which can be used in the field of vehicular communications, including the Unex DHXA-222 MIMO 2x2 card, which can be used to transmit over the 5.8/5.9 GHz frequency band and which is supported by the *ath9k* Linux driver [10]. PC Engines sells embedded boards targeted at networking applications and supporting any compatible WNIC, such as Unex cards.

There is a large number of published studies that propose combinations of off-the-shelf hardware and open-source software to build working 802.11p solutions. The combination between PC Engines' ALIX and Unex's DCMA-86P2 has been used by [15], [9] and [13] to develop 802.11p working solutions. In [15], Qin *et.al.*, focused on the packet loss rate and latency under different speeds and distances. They demonstrated that their devices can communicate up to a distance of 300 m, results that are in contrast with [9], in which the authors claim to reach 1.2 km with almost the same hardware setup. Kamal *et al.* [13], instead, only validated their model through DSRC spectrum analysis. Another important part of the literature proposes and evaluates V2X-dedicated software frameworks, such as the OpenWrt 10.03 Backfire modified in the GCDC challenge, the CVIS OS, the C2X platform, and other several patches enabling services in the automotive domain [14], [11], [16].

To the best of our knowledge, *this is the first work proposing an implementation of a working 802.11p solution* using PC Engines APU1D and Unex DHXA-222 with patched ath9k drivers and the latest LTS Linux kernel (4.14.63), *focusing on the measurements of packet loss and reachable throughput under different conditions, and the assessment of the 4 MAC layer traffic classes functionality.*

3 TESTBED DESCRIPTION

The implementation and integration work was performed on PC Engines APU1D boards, which are a flexible, up-to-date and customizable platform for networking systems and running embedded Linux. These boards mount a dual-core AMD G-series T40E x86 CPU with 64 bits support and 2 GB of DDR3-1066 DRAM. As storage, we used a SATA III Transcend *MSA370* MLC NAND Flash SSD for each board, with a capacity of 16 GB each. Two mini PCI express slots are provided, allowing the deployment of any compatible WLAN card and making customization easier. In our case we used a Unex DHXA-222 WNIC on each board. These cards provide support to ITS frequencies, with a declared maximum output power of 17 dBm. Since they are based on the Atheros *AR9462* chipset, they are supported by the *ath9k* Linux driver. This chipset represents one of the main characteristics that made us choose these cards, since chipsets supported by *ath9k* have been successfully tested and used in many research works, including the development of the OpenC2X embedded platform [14].

As a software platform, we used the latest (at the time of this writing) version of OpenWrt, i.e., release 18.06.1, with Linux kernel 4.14.63. Using newer versions of the software platform allows the user to access new packages and applications as they are introduced, together with newer versions of the existing ones. This, together with suitable hardware (faster, with more storage, possibly based on SSD as in the APU boards, and more RAM), may also lead to the integration of build systems directly on ITS systems. In particular, by choosing OpenWrt, this operation can be performed by simply configuring its build system to integrate *gcc* and other utilities.

In our work, we focused on open source solutions, highlighting their importance in research activities, as source code is available and new features can be coded and tested with ease before they possibly reach commercial systems. Another important advantage of open source is the fact that, as new features are coded, they can be shared with the research and developers community, helping other groups to develop their ideas and collectively improve existing solutions for vehicular applications; at the moment, *there are not many open source solutions concerning 802.11p and V2X communications.*

The next subsections will describe our setup more in details, covering physical, MAC and higher layers.

3.1 Physical layer

The physical layer is actually managed in hardware by the Unex WLAN cards, as opposed to SDR solutions in which PHY layer algorithms can be implemented in software.

In order to enable the usage of the DSRC channels, in the 5.9 GHz frequency band, the *ath9k* driver needed a patch for the support of both the Outside Context of BSS (OCB) mode and the selection of 10 MHz-wide channels. For this purpose, we used the kernel patches provided with the OpenC2X embedded platform, developed by the *CCS Labs* group at the University of Paderborn [14]. These patches were developed with kernel version 3.18 in mind, together with LEDE 17.01, an older branch of OpenWrt. With very few modifications, they were ported to OpenWrt 18.06.1, effectively enabling the use of the ITS channels (with IEEE numbering from 172 to 184) [1], together with an an-hoc patch for the Italian (IT) and German (DE) regulatory database flags. With the now built-in (at least inside OpenWrt) capability of the `iw` Linux wireless configuration tool to select OCB mode and 10 MHz-wide channels, we were able to use any of the 802.11p channels as required by the standard.

3.2 MAC layer

For what concerns the MAC layer, we focused on the EDCA functionality and its performance, since it is one of the main enhancements required by the IEEE 802.11 standard [6].

We patched the *mac80211* Linux wireless subsystem, providing a framework for the development of soft-MAC drivers (in which the MAC Layer Management Entity - MLME - is implemented in software), such as *ath9k*. The patch was again provided by the OpenC2X platform and it was ported to OpenWrt 18.06.1, enabling the direct selection of the EDCA traffic classes by means of calls to `setsockopt()` [1].

A tool for measuring the traffic transmitted over different Access Categories (ACs) was, however, missing: this led us to the creation of an ad-hoc patch for *iPerf 2.0.12*, which is the tool of choice as far as network measurements are concerned and which can be included, as a package, inside OpenWrt. In this patch, we introduced an additional command-line option to generate traffic in any of the 4 traffic classes (BK, BE, VI or VO), as provided for in [6].

3.3 Higher layers

In order to create applications running on top of the operating system and hardware platform, we wrote a C library enabling the use of raw sockets to send packets according to a specific protocol over the wireless medium [2]. The whole library was created to simplify the use of raw sockets also to less experienced users, providing functions to create headers and populate payloads, and with modularity in mind: at the moment it only supports UDP but it has been built in such a way that other protocols, such as WSMP (WAVE Short Message Protocol [8]), could be implemented by expanding the library without modifying most of the existing functions and without touching the Linux kernel structure, which may be subject to updates. The library also provides two functions for injecting errors inside IP and UDP packets, for research and testing purposes.

The development of this library also shows how the approach of using raw sockets could be an efficient way to transmit data using non-IP protocols, such as WSMP, without the need of adding blocks directly inside the kernel space.

4 PERFORMANCE EVALUATION

Before collecting quantitative data, we analyzed the frequency spectrum used by the boards in order to check whether it was standard compliant, using a MetaGeek Wi-Spy DBx 3 USB spectrum analyzer, coupled with the open-source Kismet Spectrum-Tools. Since there was no basic support for the DSRC spectrum at 5.9 GHz, we patched it, in order to add the possibility to analyze it. The patched version is available at [3].

We were able to verify the correct 10 MHz-wide channel usage, together with the absence of interference in the lab in which the tests took place and in which the maximum collected background signal level, between 5.850 and 5.925 GHz, was -92 dBm, which can be interpreted as pure noise.

4.1 Throughput and packet loss measurements

To measure packet loss and the reachable throughput we used iPerf, a widely known cross-platform tool, over UDP. The tests were carried out by selecting different physical data rates (among 3, 6 or 12 Mbit/s) and by varying the quantity of offered traffic and the payload size, from 16 B to 1470 B, which is the iPerf default value. The goal was to find which were the measured values under almost "ideal" conditions, with the boards placed near to each other. Unless stated otherwise, the transmission power was set to 3 dBm *txpower*, using 5 dBi antennas and a distance of ~16 cm between enclosures. Each test, yielding a single data point, lasted for 60 seconds.

The curves for a physical rate of 3 Mbit/s are depicted in Figure 1, where the top plot show the measured throughput as a function of the offered traffic, while the bottom plot shows the packet loss percentage. As can be expected, the reachable throughput depends on the payload size: as a higher size is used, the measured throughput is higher. We were able, with a 1470 B payload size, to reach a throughput of 2.62 Mbit/s for a 3 Mbit/s physical bit rate, 4.86 Mbit/s for 6 Mbit/s and 8.42 Mbit/s for 12 Mbit/s.

The packet loss tends to increase when offering too much traffic with respect to the network capacity. This is due to packets dropped in the buffers inside the kernel, which are often full, and not to losses over the air. Indeed, when offering less than the maximum reachable value (e.g., when offering less than 2.62 Mbit/s with a data rate of 3 Mbit/s and a payload

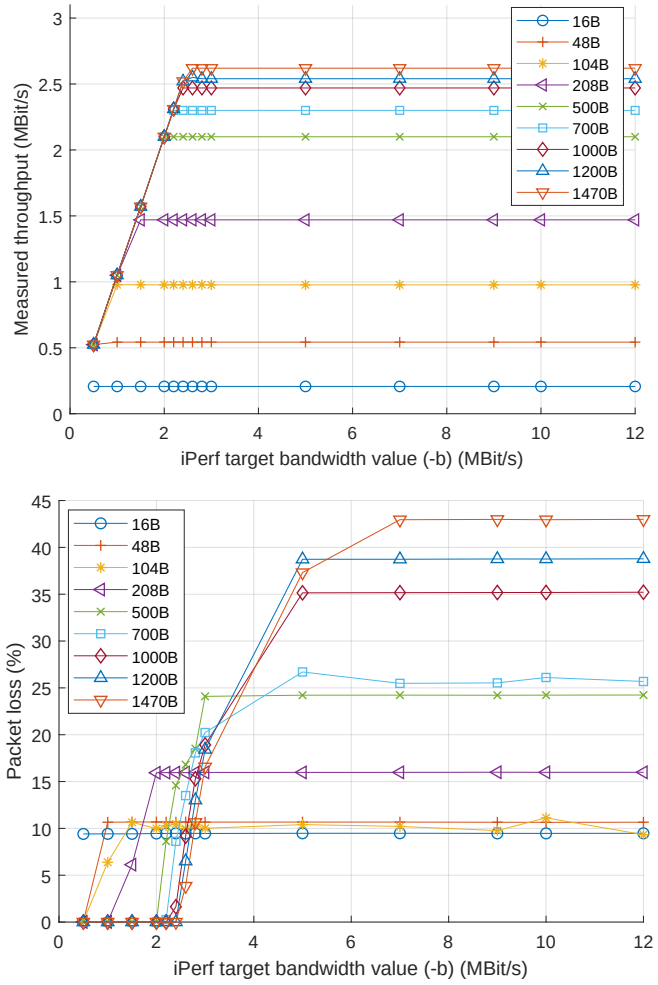

Figure 1: Throughput (top) and Packet loss (bottom) measurements for 3 Mbit/s of physical layer data rate, with different payload size and offered traffic values

size of 1470 B), a loss of 0% is almost always observed, with the exception of the 16 B case, which is, in any case, a very low, non-optimized value.

The reasons for such packet loss values are to be sought in the interaction between the application and the lower layers, mediated by buffers at the socket level. We investigated how the buffers are influencing the transmissions in cases like the aforementioned ones. Although we refrain from showing specific results for reasons of space, we were able to characterize how iPerf manages the transmission of packets. Also, by patching the iPerf UDP send loop, we could plot the evolution of the UDP buffer, showing how it gets full when the offered traffic is high compared to the selected physical data rate.

It is finally worth mentioning that the undocumented interaction between application and lower layer buffers is the reason why we have not been able to reliably measure the latency of the cards under test.

4.2 Traffic classes and access categories

An important part of the performance evaluation was related to transmissions at different EDCA Access Categories (ACs), by using the patched iPerf version presented in Section 3. The tests were performed by launching an iPerf client and server on each of the two available boards, with the client on one board transmitting to the server on the other board, and vice versa. The clients were configured to transmit over different ACs, with one board transmitting all the data under measurement and the other generating interfering traffic. We then collected the reachable throughput returned by iPerf, and estimated the connection stability, computed as the maximum percentage variation in the throughput values reported by iPerf every 2 seconds. Each test, representing a single data point, was set to last for 60 seconds. We then obtained plots for each board and for each physical data rate, considering the values mentioned in the previous subsection. The plots for a data rate of 3 Mbit/s and for one of the boards are reported in Figures 2 and 3.

Figure 2 reports, for each AC used by the board, the interfering traffic AC on the x axis (from the lowest to the highest priority) and the reachable throughput, in presence of such an interfering data flow, on the y axis.

Figure 3 shows instead the maximum percentage throughput variation, as described before, on the y axis, with the interfering traffic AC on the x axis. A low percentage value means that the connection was quite stable, while a high value indicates some instability.

When a board is transmitting using a higher priority AC, such as AC_VI or AC_VO, it can always reach a higher throughput value even in presence of another traffic stream at a lower priority AC, such as AC_BK or AC_BE, as expected. The connection stability is also improved when transmitting higher priority traffic, no matter the traffic class used by the other board, generating the interfering traffic. It has some small non linearities with respect to the increasing priority AC used by the board, but more or less it is again showing how, using a higher priority AC, it is possible to achieve not only a better throughput, but also less oscillations in the measured values. Using AC_VI and AC_VO, it also never happens that a board is unable to communicate over a full report interval (2 seconds).

Another important consideration is related to the fact that, when the tested and the interfering flows are using the same traffic class, the channel usage is fair. This can be seen in Figure 2 by drawing an horizontal line that could be ideally

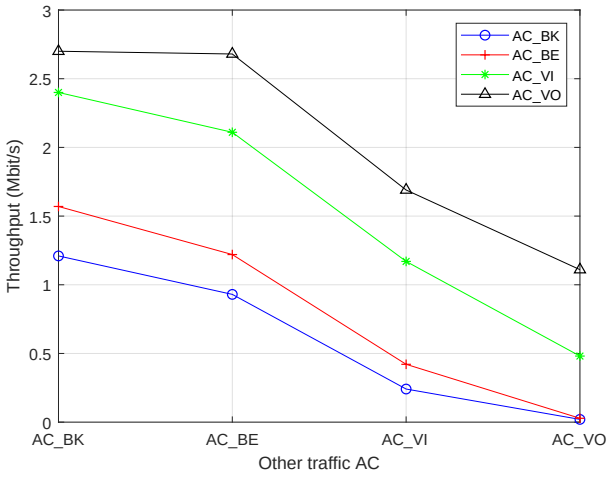

Figure 2: Reachable throughput (Mbit/s) when using different ACs

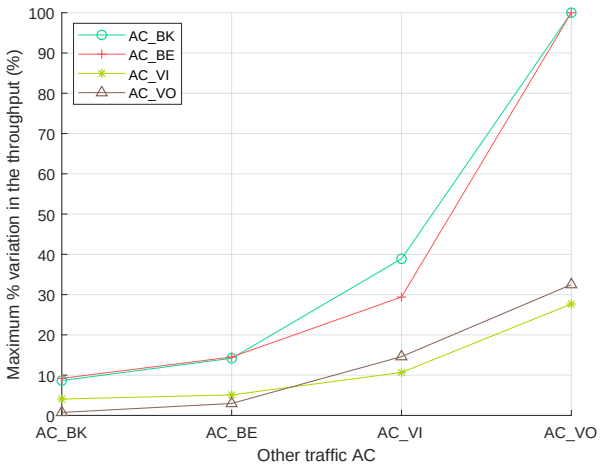

Figure 3: Maximum % variation in the throughput during the test, when using different ACs

traced around 1.2 Mbit/s for 3 Mbit/s. The same consideration holds for the other physical data rates.

In Figure 4, throughput and stability measurements are depicted, when testing over a a single traffic flow, generated by one board and received by the second board, running an iPerf server. We varied the AC used by the client and the physical data rate, again between the values (3, 6 and 12 Mbit/s) mentioned before. The throughput measurements were performed trying to offer a higher amount of traffic with respect to each of the physical data rates, in order to test the maximum reachable throughput. The stability of the measured throughput values turned out to be quite high, being the variation always less than 0.7%, since only one traffic flow was present. This actually shows that the WNICs we used, together with our platform, can maintain a stable connection.

As can be seen, the throughput values increase as the priority is increased, thanks to shorter AIFS (which are all different when working in OCB mode [6]) and smaller contention window sizes. We could also verify that not specifying any traffic class is equivalent to the selection of the AC_BE Access Category.

4.3 Received power and connectivity measurements

Part of the performance evaluation work also related to indoor throughput, packet loss and connection stability measurements, with a single data stream over a chosen AC (AC_BE, in our case). The results were obtained by deploying two iPerf client/server instances on the boards, putting them at increasing distances and finally correlating the achievable throughput with the average received signal level.

Part of the results, in terms of throughput, are shown in Figure 5, for decreasing received power values on the x axis.

As for the received power, we considered an average signal level value returned by the driver. In this regard, a few remarks are in order:

- Oscillations in the reported values of the order of 1 dBm were always present, so these values have to be always taken with at least a ±1 dBm precision.
- There was a particular situation in which the same average received signal level (-88 dBm) was detected in two slightly different positions, with almost the same ±1 dBm oscillations. It is likely that the true received power was a little higher in one point with respect the other, looking at how better results, in terms of throughput, could be achieved. In order to discriminate the two points and considering that probably the true power was a little higher in one of the two, we assigned it a value of *-87.5 dBm*: which explain the meaning of this odd point on the x-axis in Figure 5.

Taking into account that these tests are all static ones, we were able to show that the connection stability and the reachable throughput remain quite high until the received power is above *-87 dBm*; below this value, we observed an abrupt drop until the connection could no longer be established. When the signal was weaker, small distance variations were sufficient to cause variations in the throughput and in the received power average values.

5 CONCLUSIONS AND FUTURE WORK

This paper studied the performance of IEEE 802.11p cards for vehicular communication with the aim to provide a baseline characterization in a controlled environment. For this reason,

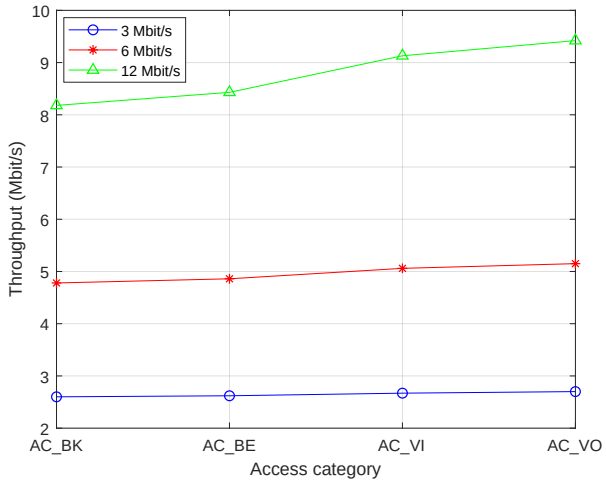

Figure 4: Measured throughput with a single stream, at different ACs and physical data rates; payload length: 1470 B, offered traffic: 10 Mbit/s

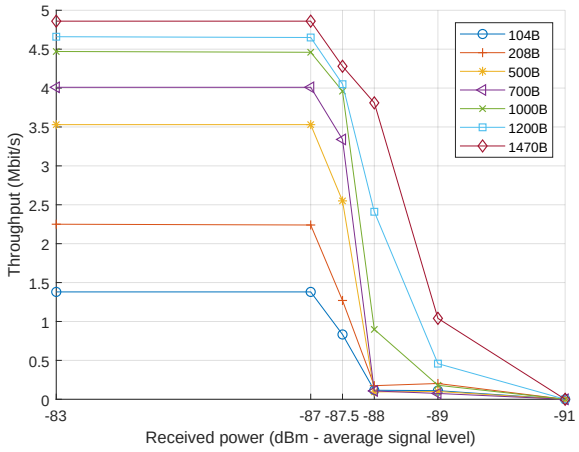

Figure 5: Throughput measurements for different values of received power; physical data rate: 6 Mbit/s, payload length: 1470 B, *txpower*: 10 dBm, offered traffic: 10 Mbit/s

we conducted several lab tests to establish a benchmark, in terms of spectral analysis, throughput, connection stability and received power. The performance of data flows belonging to difference Access Categories was also investigated. We developed an open-source framework for conducting such tests, and made it available on GitHub.

One issue that we identified is related to the fact that multicast mode only allows the AC_BE queue to be selected. This is probably due to the introduction of the so-called *intermediate software queues* inside the most recent versions of the Linux kernel, not taking into account the requirements of the vehicular networks. This could potentially require an important patching work and will be investigated in the future, along with real vehicular deployments in road settings.

REFERENCES

[1] GitHub - openwrt-V2X patch [online]. https://github.com/francescoraves483/OpenWrt-V2X.

[2] GitHub - Rawsock library [online]. https://github.com/francescoraves483/Rawsock_lib.

[3] GitHub - Spectools-dsrc patch [online]. https://github.com/francescoraves483/spectools-dsrc.

[4] Toyota and Lexus to launch technology to connect vehicles and infrastructure in the U.S. in 2021 [online]. https://corporatenews.pressroom.toyota.com/releases/toyota+and+lexus+to+launch+technology+connect+vehicles+infrastructure+in+u+s+2021.htm.

[5] Volkswagen group assumes pioneering role in rapid road safety improvement [online]. https://www.volkswagenag.com/en/news/2018/02/volkswagen_group_rapid_road_safety.html.

[6] IEEE standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements - part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)* (Dec 2016), 1–3534.

[7] IEEE standard for wireless access in vehicular environments–security services for applications and management messages. *IEEE Std 1609.2-2016 (Revision of IEEE Std 1609.2-2013)* (March 2016), 1–240.

[8] IEEE standard for wireless access in vehicular environments (WAVE) – networking services. *IEEE Std 1609.3-2016 (Revision of IEEE Std 1609.3-2010)* (April 2016), 1–60.

[9] Agafonovs, N., Strazdins, G., and Greitans, M. Accessible, customizable, high-performance IEEE 802.11p vehicular communication solution. In *2012 The 11th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)* (June 2012), pp. 127–132.

[10] de Jongh, J., van de Sluis, J., Heuven, D., Voronov, A., and Passchier, I. IEEE 802.11p [CTU-IIG] on PCEngines APU1D running Voyage.

[11] Grau, G. P., Pusceddu, D., Rea, S., Brickley, O., Koubek, M., and Pesch, D. Vehicle-2-Vehicle communication channel evaluation using the CVIS platform. In *2010 7th International Symposium on Communication Systems, Networks Digital Signal Processing (CSNDSP 2010)* (July 2010), pp. 449–453.

[12] IEEE. IEEE 802 PARs & ICAIDs under consideration [online]. http://www.ieee802.org/PARs.shtml.

[13] Kamal, F., Lou, E., and Zhao, V. Design and validation of a small-scale 5.9 GHz DSRC system for vehicular communication. In *2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)* (April 2012), pp. 1–4.

[14] Laux, S., Pannu, G. S., Schneider, S., Tiemann, J., Klingler, F., Sommer, C., and Dressler, F. Demo: OpenC2X — an open source experimental and prototyping platform supporting ETSI ITS-G5. In *2016 IEEE Vehicular Networking Conference (VNC)* (Dec 2016), pp. 1–2.

[15] Qin, Z., Meng, Z., Zhang, X., Xiang, B., and Zhang, L. Performance evaluation of 802.11p WAVE system on embedded board. In *The International Conference on Information Networking 2014 (ICOIN2014)* (Feb 2014), pp. 356–360.

[16] Vivek, N., Sowjanya, P., Sunny, B., and Srikanth, S. V. Implementation of IEEE 1609 WAVE/DSRC stack in Linux. In *2017 IEEE Region 10 Symposium (TENSYMP)* (July 2017), pp. 1–5.