

OpenVLC1.2: Achieving Higher Throughput in Low-End Visible Light Communication Networks

Ander Galisteo^{†‡} Diego Juara[†] Qing Wang[‡] Domenico Giustiniano[†]

[†]IMDEA Networks Institute, Madrid, Spain

[‡]Universidad Carlos III de Madrid, Spain

[‡]KU Leuven, Leuven, Belgium

Abstract—In this paper, we introduce the OpenVLC1.2 research platform for low-end visible light communication (VLC) networks. The platform builds on top of previous versions that has attracted dozens of users from the VLC research community. We maintain its main advantages such as the support for communication with TCP/IP layers, software-based and programmable MAC and PHY layers and low-cost front-end. In this new version, we make an effort to increase the network throughput to 100 kb/s, i.e. by a factor of 8 with respect to previous versions, without adding any hardware cost. This benefit comes from our exploitation of the Programmable Real-time Units (PRUs) of the BeagleBone Black board, together with our low-resource consumption frame detection technique.

I. INTRODUCTION

Visible light is present everywhere and is gaining significant interest as a medium to connect objects to the Internet. Visible Light Communication (VLC), enabled by modulating the LED light, is being investigated by researchers for next-generation networks [1], [2], V2X communication [3], etc. VLC has several advantages over RF communication, e.g. wider bandwidth, use of the same transmission source for both illumination and communication, low-energy front-end. Most of VLC research is based on resource-rich platforms, e.g., WARP [4], USRP [5], and aims at increasing data rates in isolated cells with point-to-point scenarios. These systems have cost around \$1000s. Recently, a range of applications have also been developed with low-end VLC platforms: toy-to-toy communication [6], human sensing [7], mobile interaction [8], indoor localization [9], [10], and passive VLC [11], [12]. These recent low-end platforms (with prices less than \$100 per transceiver) sit at the opposite extreme of the high-end VLC systems: trading throughput (kb/s) for cost and simplicity.

To solve the lack of an open-source and flexible platform for low-end VLC research, we introduced OpenVLC at the VLCS workshop [13]. It is the first open-source platform for low-end VLC networks, targeting at applications for the Internet of Things (IoT). Since its release, the platform has been used in several research works and made available to the community through open calls. OpenVLC is now being used in more than 20 research groups worldwide for either research or teaching.

To guarantee the highest flexibility and reconfigurability, we implemented a minimal set of functionalities in the hardware. The MAC and PHY layers were mainly implemented in the software (Linux driver), allowing for quick and flexible testing

of new VLC protocols and applications. However, this solution limited the throughput to around 12 kb/s (UDP throughput). There are two *challenges* that have not been solved by our previous platforms:

- How to modulate the LED light at a higher speed without increase in the cost?
- How to demodulate the incoming visible light signals as fast as possible with a low-end platform?

In this paper, we introduce the new version of our platform, OpenVLC1.2, that makes an effort to increase the throughput by addressing the above challenges without adding any hardware cost to the platform. We also improve its stability and optimize the platform by including feedback from our users. Since the platform runs in low-end hardware, we also design a new technique for computation- and memory-limited fast frame detection. Through a novel implementation, we can now modulate the LED light and perform sampling at a rate of several hundred kHz and achieve a UDP throughput of above 100 kb/s that could satisfy the needs of a range of IoT applications with only off-the-shelf low-end hardware.

The rest of this paper is organized as follows. Sec. II introduces the background on OpenVLC and related hardware, followed by the new system architecture and design presented in Sec. III and Sec. IV, respectively. Some preliminary results are reported in Sec. V. Finally, discussions and conclusion are drawn in Sec. VI.

II. BACKGROUND

A. Earlier versions of OpenVLC

The previous versions of OpenVLC mainly consisted of three parts: the BeagleBone Black board [14], cape, and driver.

- *BeagleBone Black (BBB)*: a low-cost embedded platform (≈ 55 USD) that runs Linux Operating System, equipped with a AM335x 1 GHz CPU, two Programmable Real-time Units (PRUs), and 65 GPIOs for quick prototyping.
- *OpenVLC cape*: the front-end transceiver that is attached directly to the BBB. The cape is equipped with one high-power LED, one low-power LED and one photodiode.
- *OpenVLC driver*: the implementation of MAC and PHY layers in Linux kernel. The key primitives are sampling, symbol detection, coding/decoding, channel contention, carrier sensing and Internet protocol interoperability.

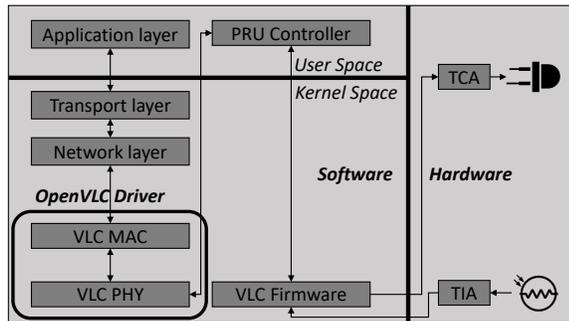


Fig. 1. The diagram of OpenVLC1.2.

B. PRUs of the BBB

The PRUs are exploited in this work to control the hardware more precisely and faster. A PRU is a fast processor operating at 200 MHz with 32-bit registers. It has single-cycle input/output access to a set of BBB pins and peripherals, and can access the memory shared by the CPU of BBB. The PRUs are programmed in assembly with a limited set of instructions. This makes programming them difficult, but allows to have a tight timing control over the hardware.

III. OPENVLC1.2: NEW SYSTEM ARCHITECTURE

Based on its predecessors, the architecture of OpenVLC1.2 has been redesigned in order to increase the network throughput. Meanwhile, OpenVLC1.2 maintains the advantages of being flexible and open-source by supporting the interface with TCP/IP layers, implementing the software-based and programmable MAC and PHY layers and being equipped with a low-cost front-end. It will be also made available to the research community.

The system architecture of OpenVLC1.2 is shown in Fig. 1. The hardware (OpenVLC1.2 cape) harnesses the LED and PD, together with ancillary circuits to transmit and receive visible light signals, respectively. The software is responsible for modulating the LED light (while transmitting) or sampling the incoming signals (while receiving), both implemented in the OpenVLC1.2 firmware. The software also implements the MAC and PHY layers in the OpenVLC1.2 driver.

Comparing OpenVLC1.2 to its predecessors, there are four main differences in the design:

- An advanced OpenVLC cape (hardware).
- A redesigned system architecture (mainly in software).
- A firmware implemented in the PRUs for signal sampling and data transmission (software).
- A technique for computation- and memory-limited frame detection (software).

A. Hardware

The new hardware (OpenVLC1.2 cape) is shown in Fig. 2. Improvements are detailed below.

1) *TX circuit*: We improve the TX circuit mainly to support a higher transmission rate and a larger communication range.

- *Increase the transmission rate*: we use a PRU at the TX to modulate the LED light at higher speeds. Moreover, a



Fig. 2. The OpenVLC1.2 cape.

TABLE I
FRAME FORMAT

Fields	Length (bytes)
Preamble	3 (24 symbols)
SFD	1
Frame length	2
Dst. address	4
Src. address	4
Protocol	2
Payload	0-MAX
CRC	2

MOSFET gate driver transistor that controls the current flowing to the LED is added to support a faster switch.

- *To expand the communication range*: a new DC-DC component that supports higher power has been introduced. With it, the front-end can now support up to 2W power LEDs.

2) *RX circuit*: In the previous versions of the OpenVLC, a bottleneck was the RX's sampling rate. In OpenVLC1.2, this is solved partly by introducing a new faster photodiode (PD). This PD does not have its own amplifying circuit. Thus, we add an external amplifier to the RX. The PD's position in the cape is also adjusted for a better detection of visible light.

B. Software

To boost the data rate in OpenVLC1.2, we exploit the BBB's PRUs to modulate the LED light and sample incoming signals. Accordingly, we redesign the original software architecture by disassembling it into three parts: *the driver (in the Linux kernel)*, *the firmware (in the PRUs)*, and *the PRU controller (in the user space)*. Time-sensitive operations are implemented in the PRUs (OpenVLC firmware) that controls the GPIOs to modulate LED light and performs sampling of incoming signals. The proposed technique for computation- and memory-limited frame detection also resides in the firmware (the details are presented in Sec. IV). The OpenVLC driver implements the MAC protocol and non-time sensitive PHY operations. This maintains the advantages of software-based flexibility and programmability. Communication between the driver and the firmware is handled by the *PRU controller* in user space.

C. Data flow

When the MAC layer receives data from the upper layers, it creates frames to encapsulate the data. Each frame starts with a frame header that contains the following fields: preamble, Start Frame Delimiter (SFD), frame length, destination address, source address, and protocol as shown in Table I.

The preamble consists of 24 alternating HIGH and LOW symbols. After that, SFD is appended to avoid false positives. The next field denotes the length of frame in bytes, followed by the destination and source addresses. The protocol is used to specify the protocol of the data in payload. Finally, Circular Redundancy Check (CRC) is used to detect symbol errors.

After encoding the frame in the PHY layer, the bit stream is sent to the PRU controller in user space who then forwards it to the firmware in the PRU. The PRU then controls the GPIOs to modulate the LED light for data transmission.

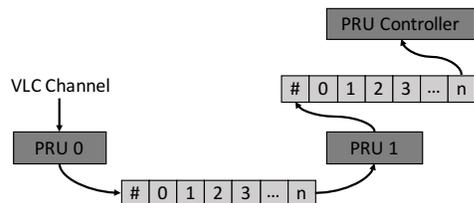


Fig. 4. Memory sharing between the PRUs, and between PRU₁ and the PRU controller ('#' stores the physical address of the latest updated data).

2) *Signal reception*: While receiving, PRU₀ captures data from ADC and stores it to a shared memory with PRU₁. This memory is circular and continuously updated. The address of the latest measurement is placed at the beginning of the shared memory as illustrated in Fig. 4. Based on this information, PRU₁ reads the data and processes it. Once a frame is totally decoded, PRU₁ sends it to the PRU controller through a shared memory, similar to the one used in the TX. This is also illustrated in Fig. 4.

V. PRELIMINARY EVALUATIONS

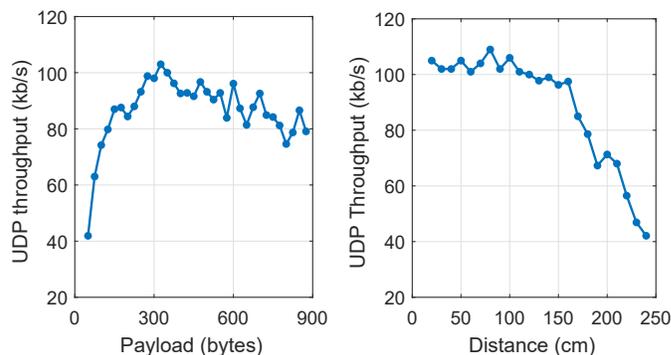
In this section we evaluate the performance of OpenVLC1.2.

Setup. We use two OpenVLC1.2 nodes, one as a transmitter and the other as a receiver. For the optical devices, we use the SENCART 2W LED and the SFH-206 (PD). The sampling frequency at the receiver is 200 MHz. The speed of modulating the LED light at the transmitter is set to the same frequency. Since OpenVLC1.2 provides a new network interface that can be easily accessed by upper layer applications, we use the tool *iperf* to evaluate the UDP performance of OpenVLC1.2.

Throughput vs. payload. We place the receiver at a distance of 20 cm from the transmitter. We vary the UDP payload from 50 to 900 bytes. The evaluation results are shown in Fig. 5(a). We can observe that the UDP throughput first increases then decreases with the increasing of payload. If the payload is small, the throughput is low due to the relative large overhead in the frame headers. When the payload is large, the probability of wrongly decoding a frame also increases, which degrades the achievable system throughput. Nevertheless, we can see that the maximum UDP throughput achieved by OpenVLC1.2 can reach 100 kb/s when the payload is 300 bytes, increasing the throughput over its predecessors by around 8x.

Throughput vs. distance. In this scenario, we fix the payload to 300 bytes. We vary the distance between the transmitter and the receiver to evaluate the performance of OpenVLC1.2 with respect to distance. The results are shown in Fig. 5(b). We can see that the UDP throughput remains around 100 kb/s at distances up to 150 cm. After that, the signal strength becomes weaker and the UDP throughput starts to decrease.

Limitations. The throughput and transmission distance in OpenVLC1.2 can be still largely improved with better front-end design. Nevertheless, the achieved 100 kb/s UDP throughput in OpenVLC1.2 is fast enough for several IoT applications.



(a) UDP throughput vs. payload (b) UDP throughput vs. distance

Fig. 5. Preliminary evaluations of the OpenVLC1.2 platform

VI. CONCLUSION

In this paper, we presented the design and preliminary performance evaluation of OpenVLC1.2, an open-source platform designed to lower the barriers to VLC research for the Internet of Things. To the best of our knowledge, OpenVLC1.2 is the first open-source platform that achieves a UDP throughput of 100 kb/s using only low-end hardware. Apart from being used for research and teaching as its predecessors, OpenVLC1.2 can enable applications in real world. Going forward, we will continue to improve the performance of OpenVLC to benefit the research community.

ACKNOWLEDGMENT

This work has been funded in part by the Madrid Regional Government through TIGRE5-CM (S2013/ICE-2919) and in part by the “La Caixa international PhD program” fellowship.

REFERENCES

- [1] “pureLiFi,” <https://purelifi.com/>, 2018.
- [2] J. Zhang, X. Zhang, and G. Wu, “Dancing with light: Predictive in-frame rate selection,” in *Proc. IEEE INFOCOM*, 2015, pp. 1–9.
- [3] C. B. Liu, B. Sadeghi, and E. W. Knightly, “Enabling vehicular visible light communication (V2LC) networks,” in *Proc. VANET*, 2011.
- [4] “WARP Project,” <http://warpproject.org>, 2018.
- [5] “Universal Software Radio Peripheral,” <https://www.ettus.com/>, 2018.
- [6] N. O. Tippenhauer, D. Giustiniano, and S. Mangold, “Toys communicating with leds: Enabling toy cars interaction,” in *IEEE CCNC*, 2012.
- [7] T. Li, C. An, Z. Tian, A. T. Campbell, and X. Zhou, “Human sensing using visible light communication,” in *ACM MobiCom*, 2015.
- [8] C. Zhang, J. Tabor, J. Zhang, and X. Zhang, “Extending mobile interaction through near-field visible light sensing,” in *ACM MobiCom*, 2015.
- [9] Y. Kuo, P. Pannuto, K. Hsiao, and P. Dutta, “Luxapose: Indoor positioning with mobile phones and visible light,” in *ACM MobiCom*, 2014.
- [10] C. Zhang and X. Zhang, “Litell: Robust indoor localization using unmodified light fixtures,” in *ACM MobiCom*, 2016.
- [11] Q. Wang, M. Zuniga, and D. Giustiniano, “Passive communication with ambient light,” in *ACM CoNEXT*, 2016.
- [12] X. Xu, Y. Shen, J. Yang, C. Xu, G. Shen, G. Chen, and Y. Ni, “PassiveVLC: Enabling Practical Visible Light Backscatter Communication for Battery-free IoT Applications,” in *ACM MobiCom*, 2017.
- [13] Q. Wang, D. Giustiniano, and D. Puccinelli, “OpenVLC: Software-Defined Visible Light Embedded Networks,” in *ACM VLCS*, 2014.
- [14] “BeagleBone Black,” <http://beagleboard.org/Products/BeagleBone+Black>, 2018.