An Open Source Research Platform for Embedded Visible Light Networking

Qing Wang, Domenico Giustiniano, and Daniele Puccinelli

Abstract

Despite the growing interest in visible light communication, a reference networking platform based on commercial off-the-shelf components is not available yet. An open source platform would lower the barriers to entry of VLC network research and help the VLC community gain momentum. We introduce OpenVLC, an open source VLC research platform based on software-defined implementation. Built around a credit-card-sized embedded Linux platform with a simple opto-electronic transceiver front-end, OpenVLC offers a basic physical layer, a set of essential medium access primitives, as well as interoperability with Internet protocols. We investigate the performance of OpenVLC and show examples of how it can be used along with standard network diagnostics tools. Our software-defined implementation can currently reach throughput on the order of the basic rate of the IEEE 802.15.7 standard. We discuss several techniques that researchers and engineers could introduce to improve the performance of OpenVLC and envision several directions that can benefit from OpenVLC by adopting it as a reference platform.

Introduction

The formidable uptake of mobile smart devices is driving an ever increasing demand for wireless data, contributing to the wireless spectrum crunch. As a spectrum-rich alternative to RF, visible light communications (VLC) is attracting the interests of both researchers and engineers. VLC also represents an appealing alternative to RF for networked embedded devices, for instance, in the Internet of Things, wearable computing, indoor localization, and vehicular networks [1–3]. In addition, the adoption of VLC would reduce the health hazards caused by overexposure to RF.

VLC experimental research in networked embedded systems (networked VLC) has yet to gain momentum due to the lack of a low-cost reference platform. The drawbacks of the existing experimental work on VLC platforms include its lack of openness, its failure to provide broad support for common networking protocols, and its focus on high-end platforms [4–6].

Similar to how the introduction of the Berkeley motes spearheaded networked embedded systems research a decade ago, we believe that a general-purpose, low-cost, open VLC platform would pave the way to novel networking research directions. This article takes an initial step toward the adoption of VLC in networked embedded systems and introduces OpenVLC, an open source software-defined networking platform for fast prototyping. OpenVLC runs on a cost-effective yet powerful embedded board, with a unit cost of approximately US$60. The source code and electronic schematic of OpenVLC are available at http://openvlc.org.

In this article, we present the design and evaluation of the open source OpenVLC research platform. We interface an LED-based front-end to an embedded Linux platform and provide a set of software-based primitives, such as signal sampling, symbol detection, coding/decoding, carrier sensing, and communication with the TCP/IP layers of the Linux operating system. We further design and implement a basic medium access control (MAC) protocol running in software and illustrate its performance evaluation. The objective of this first step toward networked VLC is to provide a functional research platform that can easily be extended according to the directions of interest.

In its present form, OpenVLC relies on simple off-the-shelf electronic components and only uses a basic physical layer (PHY), which can be scaled to use more advanced PHYs. Currently, OpenVLC can achieve MAC layer throughput on the order of the basic rate of IEEE 802.15.7 [7], and UDP throughput of 12.5 kb/s operating at distances up to 1 m.

The rest of this article is organized as follows. The system design and implementation of OpenVLC are presented, followed by evaluation at the MAC layer and system level given later. Techniques that could improve the performance of OpenVLC and several research directions that can benefit from OpenVLC are discussed. Closing remarks are presented.

OpenVLC System Design

OpenVLC is a general-purpose software-defined platform for networked VLC. The prototype of OpenVLC is shown in Fig. 1. It is built around

Qing Wang is with IMDEA Networks Institute and University Carlos III of Madrid.

Domenico Giustiniano is with IMDEA Networks Institute.

Daniele Puccinelli is with the University of Applied Sciences of Southern Switzerland.

the BeagleBone Black (BBB) board,[1] a cost-effective, user-friendly, versatile single-board computer with a small form factor. OpenVLC consists of a BBB board, a VLC front-end transceiver, and a software-defined system implementation. The front-end transceiver adopts a single LED together with a few basic electronic components for both transmission and reception. OpenVLC's software components are implemented as a Linux driver that communicates directly with the LED front-end and the Linux networking stack. As a result of this design choice, the VLC communication interface can take advantage of the vast range of Linux tools. The communication between two OpenVLC nodes is illustrated in Fig. 1.

Figure 1. The current prototype of OpenVLC: the front-end transceiver of an OpenVLC node is shown on the left, and an example of inter-node communication is shown on the right.

BIDIRECTIONAL COMMUNICATION

The current version of the OpenVLC front-end transceiver reuses the same LED for both transmitting and receiving light signals. Using LEDs as receivers can reduce the design complexity and increase the resilience to ambient noise (e.g., sunlight and indoor illumination [8]) with no need for additional optical filters [9]. The current design can be extended to use photodiodes as receivers, as discussed later.

The block diagram of the transceiver is shown on the right of Fig. 2. It includes a transconductance amplifier (TCA) for transmission, a transimpedance amplifier (TIA), and an analog-to-digital converter (ADC) for reception, as well as a tristate-output buffer and ancillary circuitry for transmission and reception. A software-defined transmitter (TX)/receiver (RX) switch is used to change the LED operation mode between TX and RX through the GPIO pins:

- In **TX mode**, the tristate buffer is enabled, and encoded signals are first amplified by the TCA and then fed to the forward-biased LED.
- In **RX mode**, the tristate buffer is disabled to avoid current leakages to the TCA circuitry, and the light signal is received by the reverse-biased LED. The small photocurrent is then amplified by the TIA. Finally, an ADC converts the output analog signals to digital signals, which are then sent to the decoder through the serial peripheral interface (SPI).

Through the TX/RX switch and the tristate buffer, OpenVLC can switch the LED between TX mode and RX mode with low latency, such that it can reliably sustain the operation mode for one or more symbol periods. This design offers a basic setup to implement bidirectional communication using a single LED for VLC networks.

SOFTWARE-DEFINED PHY LAYER

The communication stack of OpenVLC is illustrated on the left of Fig. 2. Primitives are implemented to build various PHY and MAC layer protocols in the Linux operating system.

TX, **RX**, and **TX/RX** switching. In TX mode, the BBB outputs the signal to the anode of LED for a symbol period. In RX mode, the small photo-current is amplified by the TIA, and then sampled by the ADC and converted into a digital signal. The BBB samples the output of the ADC at a fixed interval equal to one symbol period. Symbol boundaries are obtained via the
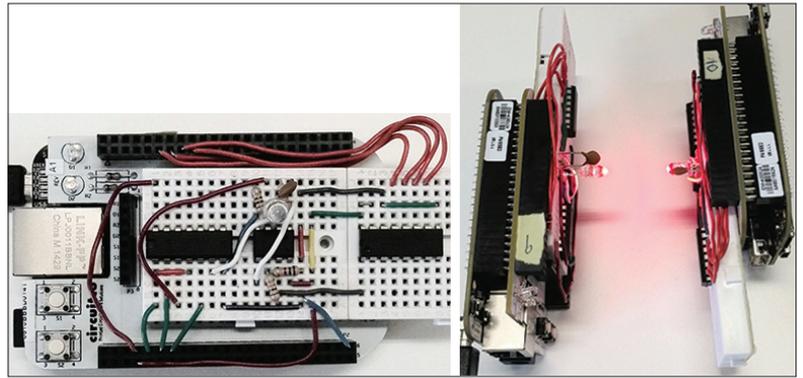
real-time timer of the Linux kernel and handled by our driver. When the timer expires, the TX outputs the signal of the symbol waiting to be transmitted and holds the signal for a symbol period. In turn, the RX samples the output of ADC and stores the value in a sequence that will be decoded by the driver at a later time. The LED switches between TX and RX modes through the software-defined TX/RX switch that runs on the BBB.

Modulation and detection. We adopt intensity modulation for data transmission. Binary information is mapped to the presence (symbol HIGH) or absence (symbol LOW) of the visible light carrier. At the transmitter, we use on-off keying (OOK) modulation and the Manchester Run-Length Limited (RLL) code. Therefore, bit 1 is mapped to symbol sequence LOW–HIGH, and bit 0 is mapped to HIGH–LOW. At the receiver, demodulation is performed with direct detection. Based on the measured voltage, the receiver detects a received signal as a sequence of symbols HIGH and LOW that are then converted to binary data.

Preamble. The PHY layer transmits each frame with a fixed-length preamble, consisting of an alternate sequence of HIGH and LOW starting with a HIGH symbol. The numbers of HIGH and LOW symbols in the preamble are the same. To convert symbols into binary data, an adaptive symbol detection threshold is adopted because the received light intensity is greatly affected by the free path loss attenuation of light transmitted from the TX to the RX. This detection threshold is obtained on a per-frame basis by averaging out the digital samples of the preamble sequence. A special frame delimiter (SFD) field is appended to the end of the preamble.

SOFTWARE-DEFINED MAC LAYER

We define two types of MAC frame: DATA and acknowledgment (ACK). The frame format is shown in Fig. 3. If the frame has no payload (length = 0), it is inferred to be an ACK. Otherwise, it is a DATA frame. Each frame can carry a payload from 0 to MAX (a predefined value) bytes. The destination and source addresses follow the Length field, and each occupies 2 bytes. The 2-byte Protocol field identifies the upper layer protocol encapsulated in the frame pay-

---

[1] http://beagleboard.org/
Products/BeagleBone
+Black

load. Fields from Length to Protocol form the MAC header. A two-byte cyclic redundancy check (CRC) over the MAC header and payload is appended after the payload. The Reed-Solomon (RS) error correcting code over the MAC header, payload, and CRC is appended to the end of each frame.

Carrier sensing. Wireless MAC protocols usually employ carrier sensing to reduce collisions. In our VLC platform, we provide two types of carrier sensing: *basic sensing* and *fast sensing*. Both are implemented in the PHY and can be invoked by the MAC layer. In basic sensing, the platform reads a certain number of continuous symbols. The channel is assessed to be busy if one or more symbols are detected as HIGH symbols; otherwise, it is assessed to be clear.

Unlike basic sensing, fast sensing operates on a per-symbol basis. The channel is assessed to be clear if the symbol is detected as LOW and is assessed to be busy otherwise.

MAC access protocol. We implement a MAC layer protocol based on the primitives discussed above. We employ a contention-based carrier sensing multiple access with collision detection (CSMA/CD) MAC protocol to ensure fair channel access among all VLC nodes and reduce the impact of collisions [8]. When a frame is ready for transmission, the MAC first calls the *basic sensing* block of the PHY layer. The frame is transmitted immediately if the PHY layer reports the channel is clear. If the channel is assessed to

be busy, the MAC starts a backoff counter. The counter is initialized with an integer value randomly drawn from a uniform distribution within the range (0, CW-1]. The contention window CW is initialized as CWmin, where CWmin is the smallest size of the contention window. The PHY keeps sensing the channel, and each time the channel is assessed to be clear, the counter is decremented. The frame is transmitted when the counter reaches zero.

Upon frame transmission, the transmitter can engage in *fast sensing*. This occurs when the transmitter sends a LOW symbol of the Manchester code, as it powers down the LED and is therefore able to switch the LED to RX mode to receive a symbol, as presented earlier. The received symbol is sufficient for *fast sensing*. Afterward, the LED is switched back to TX mode to carry on the transmission. The transmitter alternates between TX and RX mode during data transmission. If the transmitter detects a collision (i.e., the channel is assessed to be busy through *fast sensing* for no less than a predefined interval), the ongoing transmission is immediately interrupted. The illustration of the backoff mechanism, basic sensing, and fast sensing in CSMA/CD is shown in Fig. 3.

After successfully receiving a frame, the receiver sends an ACK to the transmitter. If the transmitter has not received an ACK within the timeout, it retransmits the frame and doubles the CW (until it reaches a pre-defined CWmax threshold that denotes the maximal size of the contention window). The frame is dropped after a predefined number of failed retransmissions.

Interfacing with the Internet layer. We implement the MAC protocol as well as part of the PHY as a new driver of the Linux operating system. The MAC protocol will become transparent to various applications if it can connect with the Internet layer. We implement two primitive functions to receive a packet from the upper layer and the PHY, respectively. The first function is called by the Internet layer to move packets to the MAC layer, where they are enqueued for transmission scheduling. The second one receives packets from the PHY, checks their protocols, and decides whether or not to send them to the Internet layer. By invoking these two functions, any MAC protocol can easily interact with the Internet layer.

EXPERIMENTAL EVALUATION

The experimental evaluation described in this section uses the Debian Linux Distribution with kernel v. 3.8.13 and the Xenomai patch. The details of the electronic devices employed in the current version of OpenVLC can be found in [10]. Unless otherwise specified, each node uses a symbol period of 20 µs and (216,200) Reed-Solomon error correction code. All the experiments are carried out in an indoor office environment in the presence of artificial lighting.

MAC LAYER

We evaluate the saturation throughput of Open-VLC's MAC layer in a two-node scenario, where the two nodes are within each other's field of view (FOV), and one continuously transmits to
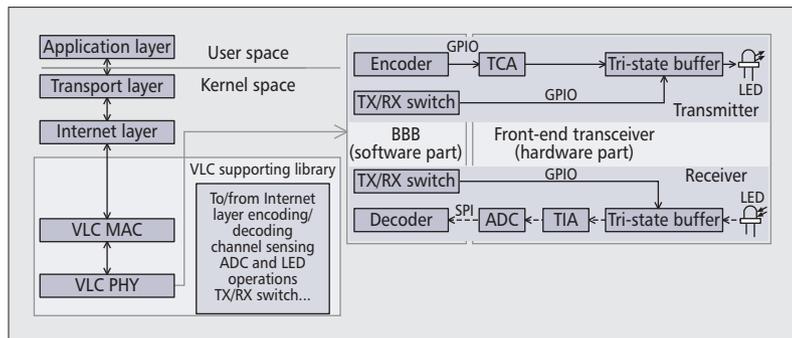
Figure 2. Diagram of the front-end transceiver (right) and the communication stack of OpenVLC in an embedded Linux operating system (left).
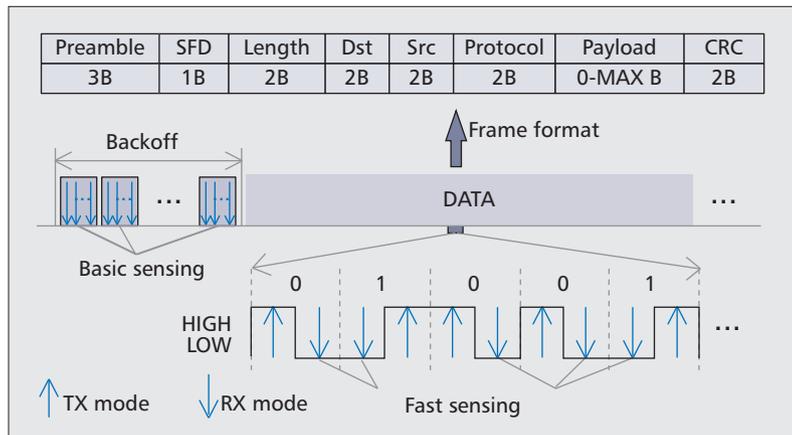
Figure 3. Backoff, basic, and fast sensing of the CSMA/CD protocol. The transmitter uses OOK with Manchester coding to send data. In the frame format: Length > 0 ⇔ DATA; Length = 0 ⇔ ACK.

the other. The throughput as a function of the per-frame payload is shown in Fig. 4, where the two nodes are placed at a distance of 0.6 m, and the payload ranges from 50 to 1000 bytes. We measured a saturation throughput of up to 18 kb/s. The throughput increases as the payload increases, ranging from 6 kb/s when the payload is 50 bytes to 18 kb/s when the payload is increased to 1000 bytes, which also shows the reliability of software synchronization implementation as frame size gets longer.

SYSTEM LEVEL

It is possible to evaluate the performance of OpenVLC using various traditional network measurement tools. In this subsection, we present evaluation results obtained from the well-known network tools `ping` and `iperf` in point-to-point link and three-node scenarios.

The OpenVLC's `ping` performance in a point-to-point link scenario over various `ping` interpacket interval (IPI) settings is shown in Fig. 5 (top left). These results are collected from 1000 `ping` packets where each `ping` packet has 10 bytes of data. From the empirical cumulative distribution function (CDF) of the round-trip time (RTT)), we observe that when the IPI is set to 0.25 s, about 90 percent of the packets incur an RTT below 200 ms. This value drops to 60 percent when the `ping` traffic load increases to IPI = 0.2 s as a result of the longer queuing time.

The network testing tool `iperf` is also used to evaluate the performance of OpenVLC, and the evaluation results of UDP and TCP over a point-to-point link are shown in Fig. 5 (top right). Here, the UDP datagram size is set to 1000 bytes.

The results are collected by running the experiment for 10 min in each scenario, and we plot the results reported by `iperf` every 10 s. We observe that the maximal and median achievable throughputs with UDP are about 17 kb/s and 12.5 kb/s, respectively. As for TCP, the maximal and median achievable throughputs are around 13 kb/s and 9 kb/s, respectively. The throughput drops with TCP with respect to UDP due to the overhead and reliability features of TCP.

Furthermore, we evaluate the performance of OpenVLC in a three-node (point-to-multipoint) network, and the results are shown in Fig. 5 (bottom). In the downlink scenario, one node sends UDP data to the other two nodes (RX1 and RX2). We observe that the median values of the UDP throughput of RX1 and RX2 are around 6 kb/s. In the uplink scenario, two nodes (TX1 and TX2) compete for the shared medium to send data to the third one. The median values of the achieved throughput are also around 6 kb/s, which shows fair access to the medium.

PERFORMANCE ENHANCEMENTS AND FUTURE RESEARCH DIRECTIONS OF OPENVLC

Currently, OpenVLC is designed using basic commercial off-the-shelf components to implement a communication network among LEDs. The achieved data rate is already comparable to
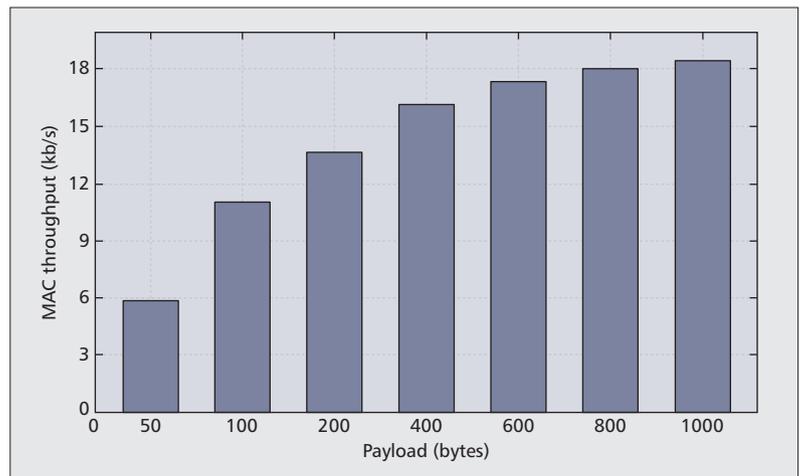
Figure 4. MAC layer throughput as a function of the per-frame payload.

the lowest one specified in IEEE 802.15.7, which specifies a PHY rate of at least 11.67 kb/s [7]. In its present form, OpenVLC already offers a flexible starter kit for VLC research.

While most of the VLC efforts so far have targeted point-to-point systems between resource-rich high-end nodes, to date, resource-poor low-end nodes are an unexplored research area. Exploring networked systems of resource-poor low-end nodes would be instrumental to the adoption of networked VLC and would require a fundamental redesign of the communication stack. The performance of OpenVLC can be improved to reach extend other domains of research investigations, using more powerful hardware and customizing the software implementation to the application scenarios of choice. In this section, we discuss a set of possible enhancements for OpenVLC as well as future research directions based on it.

PERFORMANCE ENHANCEMENTS OF OPENVLC

We begin by reviewing a list of points that could be implemented to boost OpenVLC's performance.

Matched filtering and *timing error recovery* have not yet been implemented in OpenVLC. A matched filter serves to maximize the signal-to-noise ratio and minimize the symbol error probability. Timing error recovery is very useful when the transmitter and receiver are unsynchronized. To support matched filtering, the front-end transceiver hardware needs to be upgraded. To implement timing error recovery, the software-defined PHY of OpenVLC needs to be enhanced to detect the timing error and recover from it. The implementation of matched filtering as well as timing error recovery on OpenVLC would also help to increase the communication range as well as the overall system stability for higher-rate communication.

The *coverage* of an OpenVLC node is currently limited by the output power and FOV of its LEDs. The output power can be increased by using high brightness white LEDs as the optical front-end. For scenarios where one OpenVLC node acts as an access point, hardware should be extended to support multiple-input multiple-output (MIMO) LED communication, with modula-
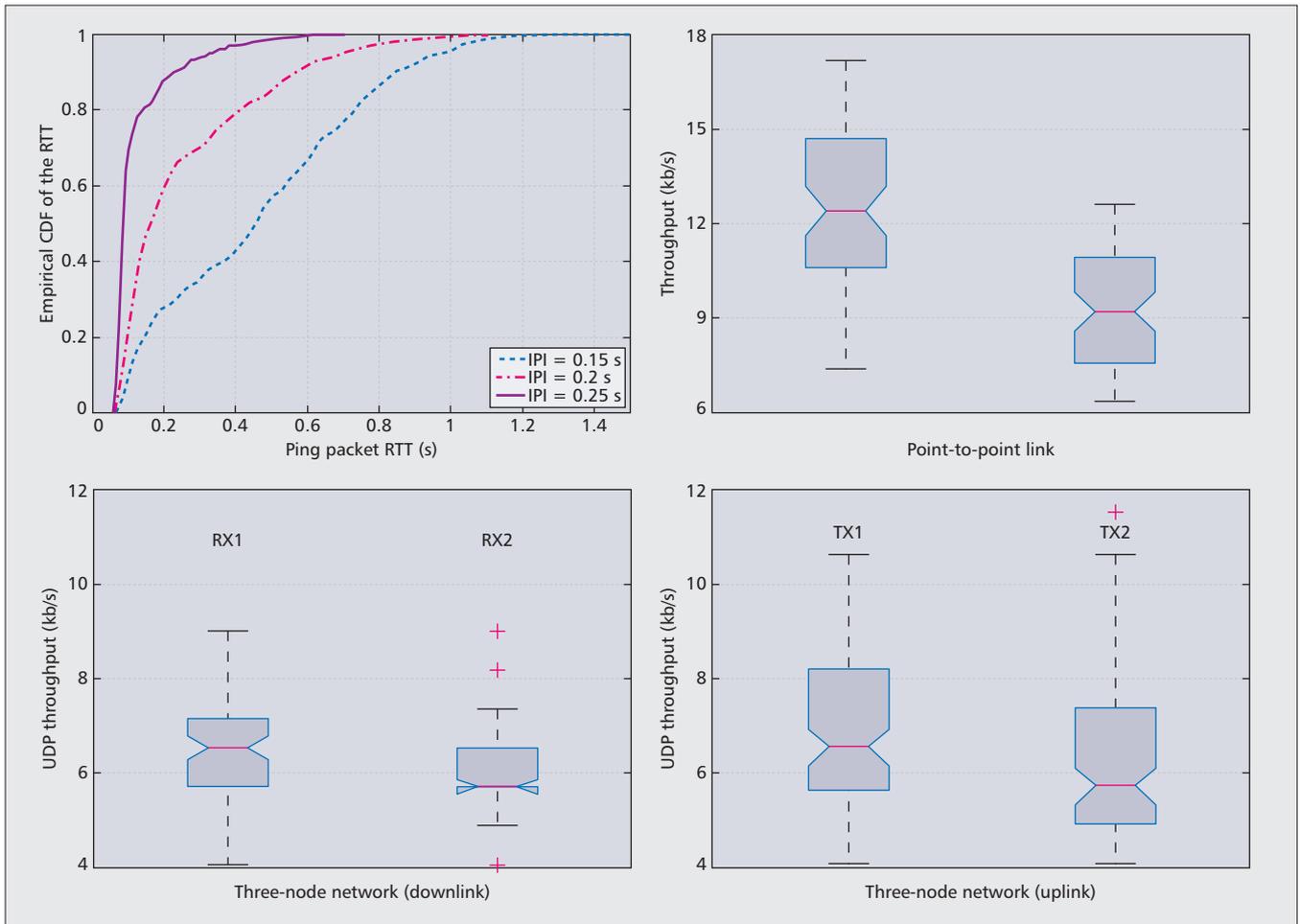
Figure 5. System-level evaluation results using `ping` and `iperf`.

tions such as optical generalized space shift keying (GSSK) [12]. This direction exploits the fact that multiple LEDs are usually required for illumination due to the limited brightness of an individual LED. The software would also need to support the selection and use of different LED-to-LED links.

The current prototype adopts basic OOK modulation, but *advanced modulation schemes* can also be used by adding a digital-to-analog converter (DAC) or exploiting the pulse-width modulation (PWM) pins of the BBB. In this way, the disadvantage of OOK in terms of inefficient bandwidth usage can be circumvented. For OpenVLC, the current bottleneck of the achievable data rate is the speed at which the BBB reads symbols from the ADC (the Linux system we employ fails to provide accurate timing past a certain speed [13]). In turn, the BBB can write symbols to the LED at a much faster speed. To eliminate the current bottleneck, field programmable gate arrays (FPGAs) (e.g., the one used in [14], which can be interfaced with the BBB) or micro-controller units (MCUs) could be employed for PHY implementation. Using such solutions, however, would increase the cost of OpenVLC. A cheaper alternative is to use the programmable real-time units (PRUs) of the BBB for dedicated implementation of time-critical functionalities. The ARM CPU of the BBB

has two PRUs, and each PRU is a low-latency 32-bit micro-controller. To improve the performance of OpenVLC, the PRUs can be used to implement time-sensitive sampling. Because the PRUs can operate at 200 MHz, the performance gain from using them would be significant.

FUTURE RESEARCH DIRECTIONS BASED ON OPENVLC

We now discuss a number of promising research directions that can be pursued based on the OpenVLC.

LED-to-photodiode communication: Extending OpenVLC to support LED-to-photodiode communication is straightforward. It would be very valuable to compare the performance of LED-to-LED and LED-to-photodiode communications in terms of transmission range, maximal achievable throughput, resilience ability to ambient light, and so on.

OpenVLC as an app: Recent research has explored the feasibility of implementing the PHY and MAC layer of ZigBee and WiFi as downloadable pieces of software (e.g., apps for smartphones) [15]. This approach would streamline the testing and deployment of modifications to existing protocols and, in principle, new protocols as well. With the PRUs of the BBB, it is possible to develop a software on the MAC/PHY proto-

Performance enhancements and future research directions	Benefits for the system	Need hardware changes?	Need software changes?	Difficulty	Has been implemented?
Matched-filtering and timing error recovery	–Stability –Throughput –Communication distance	Yes	Yes	Medium	No
Communication coverage (high brightness LED and MIMO)	–Communication coverage	Yes	Yes	Medium	No
Advanced modulation scheme	–Throughput	No/yes	Yes	Medium	No
FPGA for the PHY	–Throughput	Yes	Yes	Hard	No
MCU for the PHY	–Throughput	Yes	Yes	Medium	No
PRUs for the PHY (kernel space)	–Throughput	No	Yes	Hard	No
PRUs for the PHY (user space)	–Throughput	No	Yes	Medium	No
LED-to-photodiode communication	–To be verified	Yes	Yes	Easy	No
OpenVLC as an app	–Fast testing and deployment	No	Yes	Hard	No
Intra-frame bidirectional transmissions	–Throughput	No	Yes	Medium	Yes [11]
Integration with RF communication	–Stability –Communication flexibility	Yes	Yes	Hard	No

Table 1. Summary of the possible performance enhancements and research directions of the OpenVLC platform.

cols of OpenVLC within the user space of Linux without sacrificing the achievable data rate.

Enabling intra-frame bidirectional transmissions: A basic choice for the PHY layer of a VLC system is OOK modulation with Manchester run-length limited (RLL) line code. RLL line codes are used to prevent flickering. With OOK modulation and RLL line codes, a transmitter normally does not need to emit light when it transmits a LOW symbol. As presented earlier, the transmitter can then switch the LED to RX mode to receive a symbol. Furthermore, if the receiver has detected a HIGH symbol in the current symbol slot, and the HIGH symbol is the first part of a modulated bit, the receiver can switch the LED to TX mode to transmit a symbol during the next symbol slot. Therefore, the receiver can *embed data* into the current frame it is receiving. This technique enables intra-frame bidirectional transmissions that can increase the system throughput to a significant extent. We have successfully implemented this technique using OpenVLC; the details can be found in [11].

Integration with RF communication: In order to provide backward compatibility with previous embedded systems, one may think of designing hybrid communication networks that are built on top of both visible light and RF communication. This may allow us to exploit the advantage of both technologies, and use them in the most appropriate channel and network conditions.

A summary of these research directions together with the performance enhancements of OpenVLC is given in Table 1.

CONCLUSION

In this article, we present the design, implementation, and performance evaluation of OpenVLC, an open source platform designed to enable VLC research in the field of networked embedded systems. OpenVLC's paramount goal is to demystify VLC and lower the barriers to entry to VLC research for embedded systems researchers. Much as the Berkeley motes demystified low-power wireless a decade ago and paved the way to a decade's worth of rich and active research in wireless sensor networks, we believe that an open reference platform may open up the unexplored area of networked VLC for embedded devices. OpenVLC leverages the recent diffusion of powerful but cost-effective embedded Linux platforms to provide a reference platform that can be used jointly with a vast array of Linux tools. OpenVLC also shows how a handful of commercial off-the-shelf components can suffice as a starter kit for VLC research. Going forward, we hope that OpenVLC can serve as a bridge between the VLC community and the wireless embedded systems community. We envision that research groups in embedded systems with no prior VLC experience can use OpenVLC to explore the realm of visible light, while research groups with a solid VLC background can easily expand OpenVLC and enrich its set of functionalities, for instance, with more sophisticated hardware and more advanced PHY designs.

We envision that research groups in embedded systems with no prior VLC experience can use OpenVLC to explore the realm of visible light, while research groups with a solid VLC background can easily expand OpenVLC and enrich its set of functionalities.

REFERENCES

[1] X. Zhou and A. T. Campbell, "Visible Light Networking and Sensing," *Proc. ACM Wkshp. HotWireless*, 2014, pp. 55–60.
[2] L. Li *et al.*, "Epsilon: A Visible Light Based Positioning System," *Proc. USENIX NSDI*, 2014, pp. 331–43.
[3] C. B. Liu, B. Sadeghi, and E. W. Knightly, "Enabling Vehicular Visible Light Communication (V2LC) Networks," *Proc. ACM VANET*, 2011, pp. 41—50.
[4] H. Elgala, R. Mesleh, and H. Haas, "Indoor Optical Wireless Communication: Potential and State of the Art," *IEEE Commun. Mag.*, vol. 49, no. 9, 2011, pp. 56–62.
[5] L. Grobe *et al.*, "High-Speed Visible Light Communication Systems," *IEEE Commun. Mag.*, vol. 51, no. 12, 2013, pp. 60–66.
[6] G. Cossu *et al.*, "3.4 Gbit/s Visible Optical Wireless Transmission based on RGB LED," *Opt. Express*, vol. 20, no. 26, 2012, pp. B501–B506.
[7] IEEE Std 802.15.7-2011, "IEEE Standard for Local and Metropolitan Area Networks — Part 15.7: Short-Range Wireless Optical Communication Using Visible Light," 2011, pp. 1–309.
[8] D. Giustiniano, N. Tippenhauer, and S. Mangold, "Low-Complexity Visible Light Networking with LED-to-LED Communication," *Proc. IFIP Wireless Days*, 2012, pp. 1–8.
[9] H. Chun *et al.*, "Demonstration of a Bidirectional Visible Light Communication with an Overall Sum-Rate of 110 Mb/s Using LEDs as Emitter and Detector," *IEEE Photonics Conf.*, 2014, pp. 132–33.
[10] Q. Wang, D. Giustiniano, and D. Puccinelli, "OpenVLC: Software-Defined Visible Light Embedded Networks," *Proc. ACM Wksp. Visible Light Commun. Systems*, 2014, pp. 1–6.
[11] Q. Wang and D. Giustiniano, "Communication Networks of Visible Light Emitting Diodes with Intra-Frame Bidirectional Transmission," *Proc. ACM CoNEXT*, 2014, pp. 21–28.
[12] W. Popoola, E. Poves, and H. Haas, "Error Performance of Generalised Space Shift Keying for Indoor Visible Light Communications," *IEEE Trans. Commun.*, vol. 61, no. 5, 2013, pp. 1968–76.
[13] J. H. Brown and B. Martin, "How Fast Is Fast Enough? Choosing between Xenomai and Linux for Real-Time Applications," *Proc. Real-Time Linux Wksp.*, 2010, pp. 25–27.
[14] S. Videv and H. Haas, "Practical Space Shift Keying VLC System," *Proc. IEEE WCNC*, 2014, pp. 405–09.
[15] Y. Park *et al.*, "Software Radio on Smartphones: Feasible?," *Proc. ACM HotMobile*, 2014, pp. 1–6.

BIOGRAPHIES

QING WANG (qing.wang@imdea.org) is currently a Ph.D. student with the IMDEA Networks Institute as well as University Carlos III of Madrid. He received his Bachelor's and Master's degrees from the University of Electronic Science and Technology of China, Chengdu, in 2008 and 2011, respectively. He also received a Master's degree from University Carlos III of Madrid in 2012. His interests include visible light communication, device-to-device communication, stochastic optimization, and performance evaluation.

DOMENICO GIUSTINIANO (domenico.giustiniano@imdea.org) is a research assistant professor at IMDEA Networks Institute. He was formerly a senior researcher and lecturer at ETH Zurich and a post-doctoral researcher at Disney Research Zurich and Telefonica Research Barcelona. In 2008, he was awarded a Ph.D. in telecommunication engineering from the University of Rome Tor Vergata. He devotes most of his current research to emerging areas in the field of wireless networking and pervasive wireless systems, including visible light communication networks and mobile indoor localization.

DANIELE PUCCINELLI (daniele.puccinelli@supsi.ch) is a senior research scientist at the University of Applied Sciences and Arts of Southern Switzerland. He holds a Ph.D. in electrical engineering from the University of Notre Dame. His research interests include networked embedded systems, low-power wireless networking, pervasive computing, and information technology for energy efficiency.