Linux Light Bulbs: Enabling Internet Protocol Connectivity for Light Bulb Networks

Stefan Schmid
Disney Research & ETH Zurich
8006 Zurich, Switzerland
stefan.schmid@disneyresearch.com

Theodoros Bourchas
Disney Research & ETH Zurich
8006 Zurich, Switzerland
bourchast@gmail.com

Stefan Mangold
Disney Research Zurich
8006 Zurich, Switzerland
stefan.mangold@disneyresearch.com

Thomas R. Gross
ETH Zurich
8092 Zurich, Switzerland
thomas.gross@inf.ethz.ch

ABSTRACT

Modern light bulbs based on Light Emitting Diodes (LEDs) can be used to create smart indoor environments: LED light bulbs provide a foundation for networking using visible light as communication medium. With Visible Light Communication (VLC), LED light bulbs installed in a room can communicate with each other and other VLC devices (e.g., toys, wearables, clothing). The vision of the Internet of Things requires that light bulbs and VLC devices communicate via the Internet Protocol (IP). This paper explores how the IP stack and other networking protocols can be hosted on Linux-based VLC devices. The VLC link layer for Linux consists of a VLC network driver module on top of a previously developed VLC Medium Access Control (MAC) and Physical (PHY) layers. The network driver provides the necessary interfaces to couple the IP networking protocols and the VLC layers. Performance and interaction between network driver and the existing MAC and PHY layers are analyzed and evaluated for different networking topologies and scenarios. The evaluation results suggest that the selected IP stack and the proposed VLC protocols are flexible enough to inter-operate.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless Communications*

Keywords

Visible Light Communication; Free-Space Optics; Smart Light Bulbs; TCP/IP

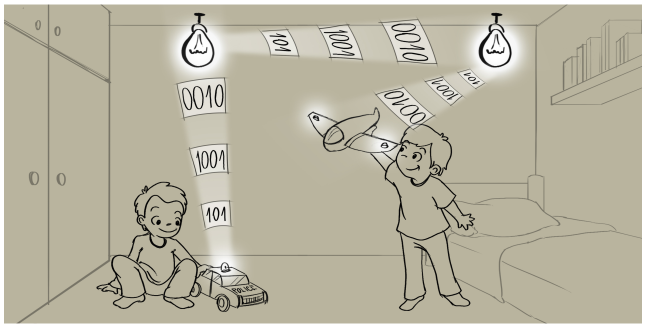

Figure 1: © Disney, concept art: VLC devices and LED light bulbs communicating with each other.

1. INTRODUCTION

Visible Light Communication (VLC) based on Light Emitting Diodes (LEDs) has several attractive properties, and recently developed LED-based light bulbs provide a cost-effective path to bring VLC to every room. But so far, the endpoints in VLC networks often use a simple protocol that is appropriate for the Medium Access Control (MAC) and Physical (PHY) layers but does not directly support networking. As the nodes in a VLC network are required to communicate with nodes in the Internet, a necessary condition for the Internet of Things (IoT), the Internet Protocol (IP) stack must be supported. Also, when bringing the VLC protocols together with an IP stack, many already existing protocols for the IP layer and above can be reused without changes.

This paper describes a prototype realization of an IP-based VLC system. The foundation is a communication firmware: a software-based VLC PHY layer and a listen-before-talk VLC MAC layer protocol with contention executed on an 8-bit microcontroller [1, 2]. This software supports LED-to-LED VLC networks, where nodes communicate with each other over free-space optical line-of-sight channels, achieving a network throughput of up to 1 kb/s at distances of a few meters. Due to the limitations that single LEDs pose to the range of the communication, light bulb prototypes based on consumer LED light bulbs, integrating the aforementioned technology, have been developed [3]. LED light bulbs, with their wide presence in indoor environments, can communicate with each other or with VLC-enabled devices like smartphones, wearables, or toys. They can be used to broadcast beacons enabling indoor localization and can create smart lighting networks. The building blocks to devise a mesh network of light bulbs that route and forward network traffic through their VLC carrier interfaces are therefore available.

This paper presents the following contributions:

- A hardware design of a Linux- and VLC-enabled light bulb: A consumer light bulb acts as a basis and is modified with additional electronics and casing. The hardware design is kept simple and low-cost (Section 2).
- A Linux kernel driver module integrating the VLC protocol's PHY and MAC layers into the Linux networking stack: The VLC firmware is kept on a separate microcontroller and communicates with the Linux platform over a serial interface (Section 3).
- An evaluation of the system for various network topologies: Traffic of the Internet Control Messaging Protocol (ICMP), the User Datagram Protocol (UDP), and the Transmission Control Protocol (TCP) are evaluated and analyzed (Section 4).

2. LIGHT BULB HARDWARE DESIGN

This section describes the modification of the previously introduced VLC light bulb design [3]. Commercial off-the-shelf LED light bulbs are used as a starting point and modified to host a System-on-a-Chip (SoC) running Linux, a VLC controller module, and an additional power supply for the added electronics. The system architecture is further explained in the following section.

2.1 System Architecture

The VLC firmware and protocols [1,2] implement the PHY and MAC layers and enable low-level networking between multiple devices. To make use of higher level network protocols, the VLC controller is extended with a SoC running a Linux distribution for embedded wireless systems [4]. Figure 2 shows the overall system architecture. The microcontroller running the VLC firmware is connected via the Universal Asynchronous Receiver Transmitter (UART) interface to the SoC. On the Linux side, the VLC controller is abstracted as a regular Ethernet interface, implemented as a kernel driver module. Therefore most applications using TCP or UDP sockets will work out of the box and can make use of the VLC link.

Since the VLC firmware is real-time critical, it is kept on a dedicated device, in the same way as Wi-Fi modules or any other networking devices. Another approach is to implement the VLC protocols directly on top of Linux using General-purpose Input/Output (GPIO) [5]. However, such a system works only as long as the operating system workload is kept low; as soon as the Linux host is used for other tasks (which motivated the addition of an operating system to VLC), timings might deviate and consequently influence the behavior of the VLC layers.

2.2 System Components

The system is built from parts of the original light bulb and additional custom-built parts, circuitry and casing. In the following the most important parts are discussed: LED plate, power supply, sensors and amplification circuitry, VLC controller, SoC board, and casing.

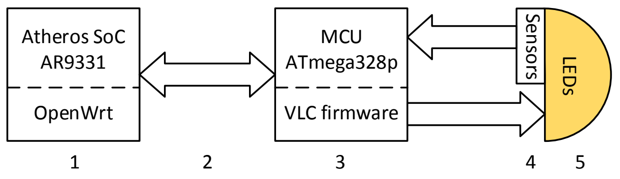

Figure 2: Light bulb architecture: (1) Wi-Fi-enabled SoC module, (2) communication interface between SoC and MCU (UART), (3) MCU running VLC firmware, (4) photodiodes and signal amplification, (5) LED plate.

2.2.1 *LED Plate*

The LED plate included in the original light bulb is reused together with the diffuse bulb that is covering the LEDs. It distributes the light in all the directions and therefore makes it possible to send data to devices all around and not only in a single direction. This diffuse bulb also distributes the light intensity in every direction and makes receiving a clean signal at a distance of several meters challenging (see light sensor section). Furthermore, a heat sink is attached to the LED plate, since the LEDs produce not negligible heat when powered on. The original light bulb uses the socket as a heat sink, but since the LED plate was moved to a different location, it needs to be cooled differently.

2.2.2 *Power Supply*

The bulb built-in power supply outputs around 40 V DC (when loaded) and is used to power on the sixteen LEDs placed in series. It is dimensioned so that it provides just enough current for the LEDs. Therefore another power supply is needed for the additional electronics (VLC and SoC board). Figure 3 (1) shows the employed power supply, consisting of a grid voltage to 9 V AC/DC converter with a custom-built board attached to the bottom. The attached board hosts a voltage regulator providing a low-ripple 3.3 V source providing power for the additional electronics. The low-ripple source is specifically important for the analog amplification circuitry (see sensor section).

2.2.3 *Light Sensors and Amplification Circuit*

Every light bulb is equipped with four photodiodes (Osram SFH213) to sense incoming light from every direction. The incoming signal is DC filtered and amplified. Every photodiode is connected to its own amplifier to keep paths on the board as short as possible and the signal therefore less prone to noise and interference. The amplified signals are fed into the ADC inputs of the VLC controller where they are further processed. Figure 3 (3) shows the PCB. The sensors connect to (3a) respectively to the foreseen locations in the other corners of the PCB. The pin headers (3c) build the socket for the VLC controller described in the next section.

2.2.4 *VLC Controller*

The VLC PHY and MAC layers [1, 2, 6] are software-based and hosted on an 8-bit Atmel microcontroller (ATmega 328p). The microcontroller board connects directly to the amplification PCB. Since the four sensors are individually sensing the incoming light, the VLC firmware can also

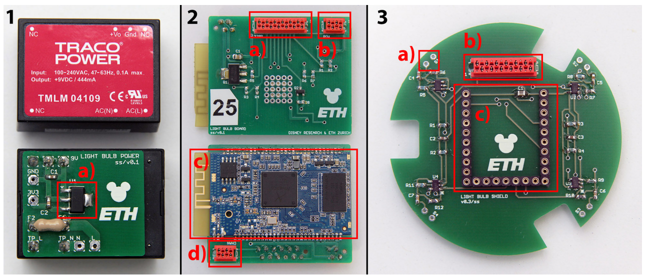

Figure 3: (1) Power supply for electronics with regulator (1a) for low-ripple voltage; (2) SoC (2c) breakout board with connectors to VLC controller board (2a), to the power supply (2b), and to the LEDs (2d); (3) VLC controller and sensor board with connector for sensors (3a), connector to the SoC host board (3b) and pin headers (3c) to mount the VLC microcontroller.

provide information about the originating direction of the signal. The controller uses the serial (UART) interface to communicate with the SoC board running Linux.

2.2.5 *SoC Board*

Each light bulb features a SoC. The module consists of an Atheros (AR9331) SoC with built-in 802.11g/n Wi-Fi including an on-board antenna. It also provides 20 GPIO pins and a serial interface (UART).

The SoC can run OpenWrt [4], an embedded Linux distribution specifically for routers and other networking devices. Since it is a Linux distribution it ships with the entire Linux network stack, providing a complete network- and transport-layer for VLC. A Linux kernel driver module (further explained in the next section) interfaces with the VLC controller and the Linux network stack.

The Wi-Fi interface provided by the SoC is very useful for a testbed environment. The wireless radio channel can be used as a control channel. The devices can be deployed and later configured and reprogrammed remotely. Also, measurements can be started and data can be collected without removing the bulbs again. It is important to know that the objective was not to built light bulbs that interconnect using Wi-Fi. The light bulbs interact with each other using VLC only.

Further, it is possible to use the SoC as a programmer for the VLC controller. This means that a new firmware image can be transferred wirelessly to the bulb and then flashed to the controller. Figure 3 (2) shows the board hosting SoC from both sides. (2c) shows the SoC and (2a,b,d) depict different connectors going to the sensor and VLC controller board, to the power supplies, and LEDs.

2.2.6 *Casing*

The off-the-shelf light bulbs are tightly packed. To create additional space for the modifications, 3D-printed casings are added to the bulb socket providing room for all the electronics. Figure 4 shows the different parts. The bottom part (1) hosts the power supply (1c), sensors (1b), VLC controller board (1a), and on the top the SoC module. It can be screwed directly to the bulb socket (3). On top sits an additional part (2), containing the LED plate (2a) and the heat sink (2b). Additional slits (2c) provide improved air flow. The diffuse bulb is screwed on top of the LED plate

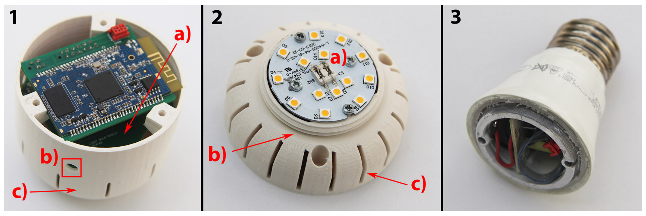

Figure 4: (1) Bottom casing hosting the additional power supply (1c), the light sensors (2b) the VLC controller board (1a) and the SoC board; (2) the top casing housing the LED heat sink (2b) and LEDs (2a), further providing additional air flow through slits (2c); (3) original light bulb socket with integrated power supply for the LEDs.

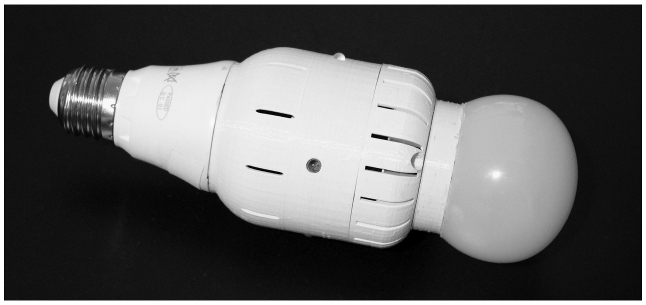

Figure 5: Fully assembled VLC-enabled light bulb.

and held in place by a 3D-printed thread. Figure 5 shows the assembled light bulb.

3. LINUX INTEGRATION

The transparent integration of the VLC communication channel based on the software-defined VLC firmware into the Linux networking stack demands the implementation of a VLC-enabled data link layer. As illustrated in Figure 6, the VLC data link layer consists of three core functional blocks: (1) VLC network driver, (2) VLC firmware MAC protocol, and (3) VLC-UART interface, which interconnects the two aforementioned blocks through UART. In this section, we describe the design and implementation features of the Linux-based VLC network driver.

3.1 Design Considerations

The VLC network driver resides in the kernel space of OpenWrt with root execution privileges. It controls data transfer within the CPU, and handles asynchronous interrupts triggered either by the network device (VLC firmware) through the UART interface or by the higher levels of Linux network stack. It is designed as a loadable kernel module.

The VLC network driver deals with making the MAC layer of the VLC library available to the network protocol suite of OpenWrt, by exposing a VLC-enabled Ethernet-class network interface, which is in charge of sending and receiving data packets. The network interface registers itself within specific kernel data structures, to be invoked when packets are exchanged with the outside world. It responds to asynchronous requests received from the outside (incoming packets from the VLC library), as well as to requests triggered by the kernel for outgoing packets to the

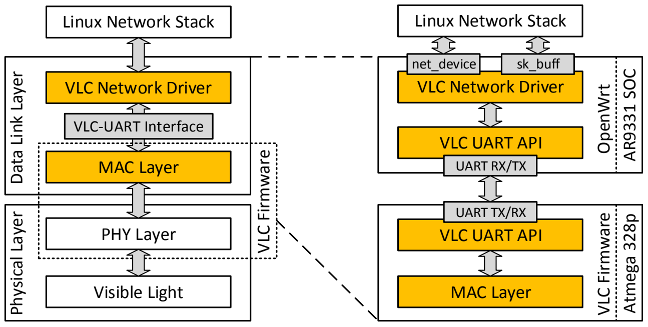

Figure 6: VLC Data Link Layer: VLC network driver, VLC-UART interface, and MAC layer (left). Linux Network Stack interaction with driver and VLC controller is shown on the right.

MAC/PHY layers, without knowing how individual transactions mapped to the actual packets are being transferred.

A core part of the VLC network driver's functionality is the communication of the VLC firmware through the VLC-UART interface. The serial interface is the communication channel between the SoC that hosts the network driver and the microcontroller where the VLC PHY/MAC layers are implemented. It must be ensured that both, the network driver and the VLC firmware, exchange data packets that are compatible with the format that the other end expects to receive.

3.2 Driver Implementation

The interface of the VLC network driver to the Linux network stack consists of two functional blocks, which are inherent in every Linux-based network driver; the `net_device` and the `sk_buff` data structures.

The `net_device` represents the driver module in the upper layers. It is used to initialize and disable the network device, and through this structure a number of device parameters can be set and initialized. Among others, the interrupt number (IRQ) that is mapped to the device is specified, the hardware address and the name of the network interface as it is exposed to user space is set, and a number of operations on the network device are enabled. The `sk_buff` represents the network packet that arrives from the upper networking layers.

The serial communication on the side of the network driver is controlled by means of an interrupt handler. When there is a new data packet (`sk_buff`) in the transmission queue of the driver, a job to forward the packet content over UART to the VLC controller is scheduled and is executed immediately when the processor is available (outside the interrupt context). If the VLC controller is not available to receive, due to concurrently running tasks of the VLC firmware, the data is scheduled to be transmitted later. On the other side, during reception, a data packet is received through consecutive interrupts and pushed to the network stack, wrapped in a `sk_buff` structure.

4. EVALUATION

The performance of the data link layer was evaluated with a single LED as transmitter and receiver. The described light bulbs have been measured to produce better results, also for larger communication ranges, due to the fact of

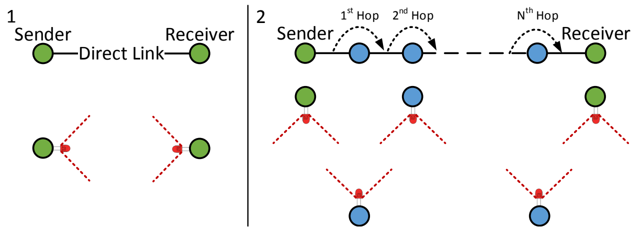

Figure 7: (1) Direct link topology; (2) multi-hop topology, N intermediate nodes between sender and receiver.

higher emitted light intensity and the addition of a photodiode with amplification circuit. Therefore results of the same quality if not better are to be expected. The evaluation section consists of measurements for direct link and multi-hop topologies and different types of network traffic (ICMP, UDP, TCP).

4.1 Direct Link and Multi-hop Topology

In networking terminology, the term "hop" refers to the number of intermediate nodes a data packet passes through. The diagram in Figure 7 (2), shows a path with N hops: in a multi-hop topology, N + 2 nodes participate, one is the transmitter, one is the receiver, and any data packet sent from the transmitter to the receiver must hop over N intermediate nodes. In a direct link topology, only two nodes participate without any hop in between (Figure 7 (1)).

4.2 ICMP Network Traffic

ICMP network traffic is used to measure the Round Trip Time (RTT) of a network packet between a sender and a destination node. In Figure 8 (1), the RTT is evaluated for different ICMP packet payloads and for different distances. It is shown that for distances up to 130 cm and for the same ICMP packet payload, the RTT has the same value but shows high deviations for distances larger than 130 cm. With the LED-to-LED setup, the transmission range is limited to 130 cm, and for larger distances, the two VLC-enabled platforms cannot communicate at all, making the RTT infinite. Therefore RTT for distances larger than 140 cm is not shown. Measurements are also reported for 1-hop and 2-hop topologies (Figure 8 (2)).

As expected, the RTT for multi-hop topologies is significantly higher and the larger the packet payload, the longer the RTT. All measurements were conducted without a "Request to Send/Clear to Send" (RTS/CTS) scheme enabled, since there will be always only one packet in transit. This scheme is a commonly used technique in wireless networks to anticipate the problem of hidden stations. It is also worth mentioning that the maximum VLC payload that can be transmitted is 200 byte, due to memory limitations and low symbol rate of the VLC controller platform.

4.3 UDP Network Traffic

This section describes measurement for UDP datagram traffic. Experiments for different UDP packet sizes, communication distances and network topologies are reported.

Figure 8 (3) shows that the UDP protocol overhead limits the overall throughput, and this effect is more pronounced

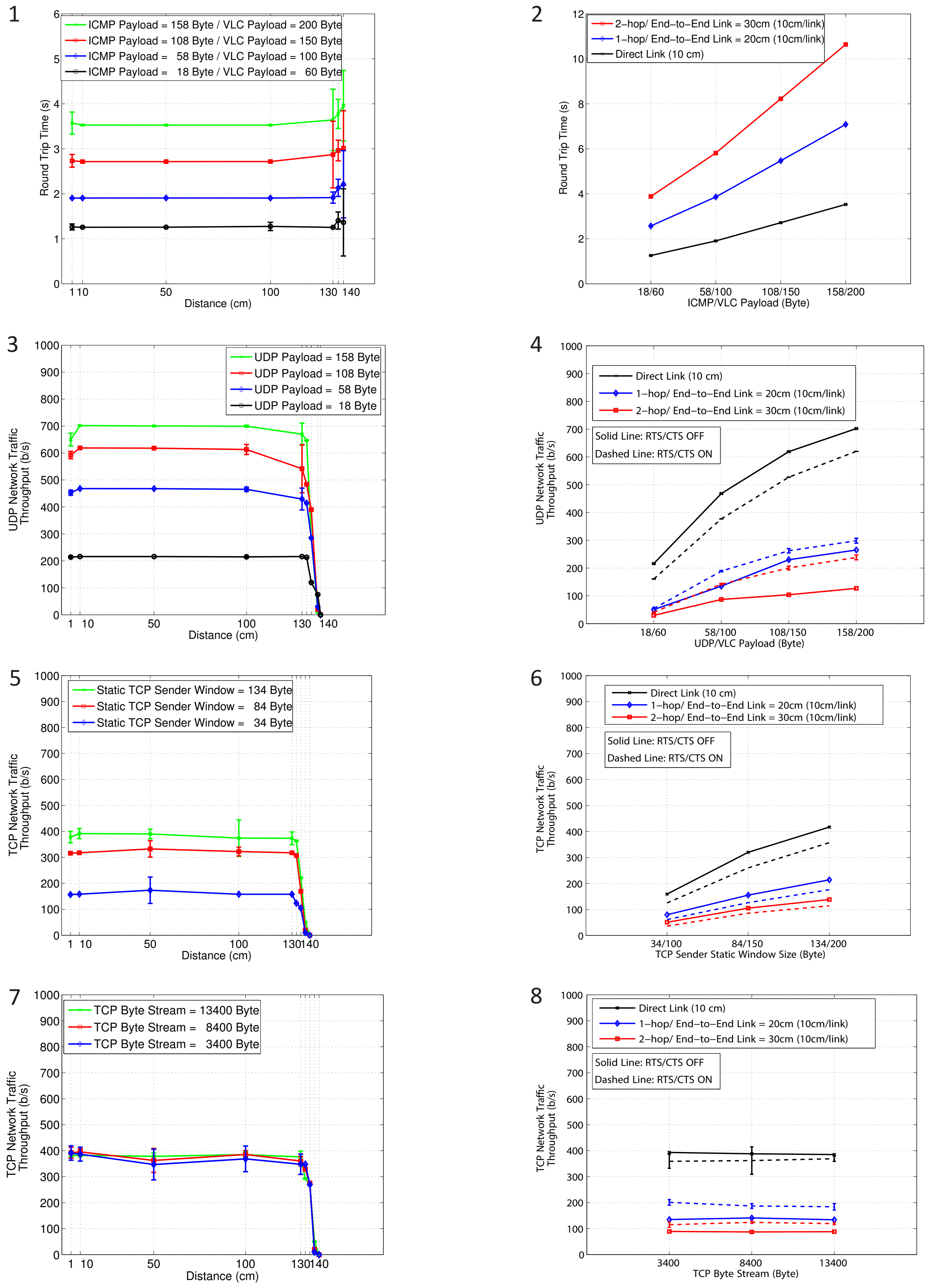

Figure 8: Result graphs for ICMP (1-2), UDP (3-4) and TCP (5-8) measurements.

for small payloads. The throughput drop at 130 cm shows again the limited communication range.

Figure 8 (4) shows UDP traffic for multi-hop scenarios. As it is expected, RTS/CTS in direct link topologies adds more delay and decreases throughput. In 1-hop and 2-hop topologies, a small improvement in the performance with enabled RTS/CTS is visible. Despite the fact that UDP traffic inserts only one packet stream in the network and traffic moves only in one direction at a time, collisions seem to happen and RTS/CTS is reducing the packet collision probability.

4.4 TCP Traffic

Given the bottleneck imposed by the speed of the VLC link, the TCP parameters are examined and adjusted so that they correspond to the capabilities of the VLC link layer. In particular, the policy concerning the TCP sender window is examined for direct link and multi-hop network scenarios. The TCP sender window defines the number of bytes the transport layer can inject into the network layer without having received any acknowledgment (ACK). The sender window size is modified and the network performance is evaluated with respect to the resulting TCP throughput. The main limitation comes from the VLC library, as it cannot handle packets larger than 200 byte. This limitation restricts TCP's Maximum Segment Size (MSS) to 134 byte (200 byte minus the TCP header size). In addition, to limit complexity, a TCP policy is applied so that there is only one TCP stream between two nodes, and only one network packet is in transit in the network at any given moment (TCP window size of one).

Figure 8 (5) shows that larger TCP sender window sizes lead to higher TCP throughput up to the communication range of 130 cm.

Figure 8 (6) shows TCP throughput with a static TCP window, measured for the direct link, 1-hop, and 2-hop topologies. It can be observed that even in multi-hop topologies, the use of RTS/CTS decreases the TCP throughput and this is more eminent when using a larger sender window. This effect is due to the specific nature of the static TCP window and the specific scenario of one TCP stream. Despite the multi-hop topologies, the end-to-end connection keeps the sender waiting for an ACK for every TCP segment transmitted and only then proceeds to the next transmission. Thus, there are no collisions and RTS/CTS adds more delay, decreasing the throughput. For other scenarios, where multiple streams are involved, the use of RTS/CTS is highly recommended. Since this scenario is not very likely for light bulb mesh networks, also experiments with a dynamic TCP setup are reported.

Given the maximum of 134 byte MSS that the VLC library imposes, with a dynamic TCP sender window, we inject multiple TCP segments in the VLC network driver buffer. The network performance for different sizes of TCP streams using a dynamic TCP window in a direct link topology is depicted in Figure 8 (7). As expected, in a direct link with a dynamic TCP sender window, the TCP throughput is between 380-400 b/s, almost the same performance as in the scenario with a static TCP window size of 134 byte.

Figure 8 (8) depicts network performance for multi-hop topologies using a dynamic TCP sender window. Without RTS/CTS there is no mechanism to detect simultaneous transmissions from hidden stations and thus consequently larger delays and lower throughputs are measured. Enabling RTS/CTS decreases the TCP throughput in the direct link topology, while in multi-hop topologies a slight improvement in the network performance is visible.

5. CONCLUSIONS

IP multi-hop networking is an important feature for VLC, as it enables the communication between light bulbs and smart objects, without relying on additional radio technology (ZigBee, Wi-Fi, etc.). This paper describes how an IP stack can operate on an LED-based VLC node, using the OpenWRT light-weight variant of Linux. Endpoints can be light bulbs or other platforms that provide LED-based communication equipped with a compatible IP stack. The initial prototype has not been optimized for performance, and a number of important issues deserve further investigation, but the system provides a proof of concept that indeed the IP stack and the proposed VLC protocols are flexible enough to inter-operate.

The ubiquitous presence of LED-based light bulbs that can be enhanced with VLC functionality, and the availability of low-complex LED-equipped devices that can engage in VLC, unleashes a wide range of opportunities and applications. As we are flooded with light everywhere, VLC-based localization services using the existing lighting infrastructure can be a viable alternative to alternative localization solutions based on radio signals. In addition, controllable LED lighting, identification, and indoor tracking opportunities can be grown using the dedicated VLC channel per light bulb. Communication with light enables a true Internet of Things, as consumer devices, such as toys equipped with LEDs, are transformed into interactive IP communication nodes – an interesting low-complex feature for what we refer to as the Internet of Toys.

6. REFERENCES

[1] S. Schmid, G. Corbellini, S. Mangold, and T. Gross, "LED-to-LED Visible Light Communication Networks," in *MobiHoc, 2013 ACM*, Aug. 2013.

[2] S. Schmid, G. Corbellini, S. Mangold, and T. Gross, "Continuous Synchronization for LED-to-LED Visible Light Communication Networks," in *Optical Wireless Communications (IWOW), 2014 3rd International Workshop in*, pp. 45–49, Sept 2014.

[3] S. Schmid, J. Ziegler, G. Corbellini, T. R. Gross, and S. Mangold, "Using Consumer LED Light Bulbs for Low-cost Visible Light Communication Systems," in *Proceedings of the 1st ACM MobiCom Workshop on Visible Light Communication Systems*, VLCS '14, pp. 9–14, ACM, 2014.

[4] OpenWrt, "OpenWrt," May 2015. https://openwrt.org.

[5] Q. Wang, D. Giustiniano, and D. Puccinelli, "OpenVLC: Software-defined Visible Light Embedded Networks," in *Proceedings of the 1st ACM MobiCom Workshop on Visible Light Communication Systems*, VLCS '14, pp. 15–20, ACM, 2014.

[6] S. Schmid, G. Corbellini, S. Mangold, and T. Gross, "An LED-to-LED Visible Light Communication System with Software-based Synchronization," in *Optical Wireless Communication. Globecom Workshops (GC Wkshps), 2012 IEEE*, pp. 1264–1268, Dec. 2012.