Demo: PULS: Processor-Supported Ultra-Low Latency Scheduling

Simon Yau, Ping-Chun Hsieh, Rajarshi Bhattacharyya, Kartic Bhargav K. R.,
Srinivas Shakkottai, I-Hong Hou, and P. R. Kumar
Texas A&M University, College Station
{symoyau, lleyfede, rajarshibh, kbhargav, sshakkot, ihou, prk}@tamu.edu

ABSTRACT

Ultra-low per-packet latency has become an essential system requirement as well as a critical challenge for wireless networks. While there is a rich literature on real-time wireless scheduling, it is still unclear what the minimum achievable latency is and what level of throughput can be obtained in practice. This demo presents PULS, a processor-supported software-defined wireless testbed that supports ultra-low-latency scheduling protocols. We will demonstrate that PULS provides strict per-packet latency guarantees as low as 1 millisecond with realistic throughput for wireless networks.

CCS CONCEPTS

• **Networks → Programmable networks**; *Wireless access networks*;

KEYWORDS

MAC Scheduling, Ultra-low latency, Software Defined Radio

ACM Reference Format:
Simon Yau, Ping-Chun Hsieh, Rajarshi Bhattacharyya, Kartic Bhargav K. R., Srinivas Shakkottai, I-Hong Hou, and P. R. Kumar. 2018. Demo: PULS: Processor-Supported Ultra-Low Latency Scheduling. In *Proceedings of ACM MobiHoc conference (MobiHoc '18)*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

The Medium Access Control (MAC) layer presents novel challenges to wireless networks. There has been a growing emphasis on stricter latency requirements for wireless networks to support next generation applications such as Virtual Reality (VR), Augmented Reality (AR), factory Internet of Things (IoT), and tactile Internet. Such applications necessitate very quick response times, and the MAC layer will have a crucial role to play in reducing the response times of nodes on the edge.

This trend has led to the development of several theoretical frameworks which aim at reducing the latency for scheduling multiple nodes with varying timing requirements [1]. However, most of these works do not include experimental results to validate the performance of their proposed protocols, which becomes a major hindrance to their widespread adoption. While many testbeds have been developed, to the best of our knowledge, none have included the features required for the verification of protocols involving millisecond (ms) resolution latency with realistic throughputs.

Perceiving this need for a testbed to examine the performance of such protocols, we developed PULS [3], a processor-supported software-defined wireless platform for the experimentation of ultra-low latency scheduling protocols. PULS utilizes a powerful host machine coupled with a Field Programmable Gate Array (FPGA) software-defined radio (SDR) connected via PCI-e to obtain millisecond resolution for scheduling on a per-packet basis. PULS includes features for varying the intensity and pattern of incoming traffic, as well as changing policies on the fly using an easy to use Graphical User Interface (GUI). PULS enables us to observe the performance of various scheduling protocols under different traffic conditions.

2 DESIGN OF PULS

In PULS, each wireless node is a FPGA-based SDR, which consists of a USRP-2953R that is connected through PCI-e to a multi-core Windows laptop as the Host machine. The USRP-2953R is programmed through LabVIEW Communication Systems Design Suite [2] by National Instrument (NI). We start with the NI's 802.11 Application Framework (AF), which provides the basic PHY and MAC functionality of the random-access-based 802.11 protocol like interframe space timing, carrier sensing and MAC layer ACKs, but without wireless scheduling features.

To achieve flexible MAC implementation, we employ a Host-FPGA separation: (i) high-level functionality such as packet scheduling and packet dropping, are located on the Host machine for flexiblity, and (ii) low-level and time-critical functions such as packet encoding/decoding, carrier sensing, and ACK processing are implemented on the FPGA to meet the required latency requirements. More details on the architecture of PULS can be found in [3].

2.1 Toolbox: Real-Time Wireless Scheduling

In PULS, scheduling is performed and repeated on a per-packet basis and is achieved by enforcing a sequence of mechanisms, all of which are implemented on the Host. We highlight the major real-time wireless mechanisms implemented on the proposed testbed as shown in Figure 1.

- **Prepend packet deadines**: Each real-time packet is prepended upon arrival with a Host timestamp, which indicates its absolute deadline. The time resolution of the timestamp can be as small as 1 microsecond. Note that the packets of the same flow are allowed to have different relative deadlines if required.

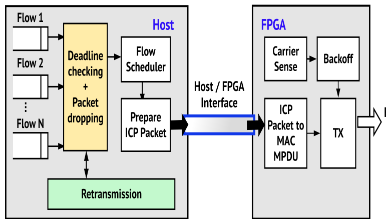

Figure 1: High-level design of real-time wireless scheduling on PULS.

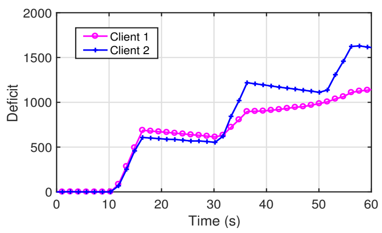

Figure 2: Deficit evolution for (2,5,4,6)

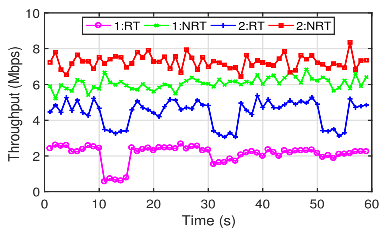

Figure 3: Throughput evolution for (2,5,4,6)

- **Deadline checking and packet dropping**: This is a generic module for dropping unwanted packets and checking the deadline of each packet. For example, for real-time wireless scheduling, a packet will be dropped when it misses its deadline or it reaches its retransmission limit.
- **Update per-flow states**: Each scheduling policy requires continual update of state variables, such as deficits and queue lengths. To update state variables, the Host machine may also read register values in FPGA, such as the number of ACK and ACK timeout, through the PCI-e interface.
- **Flow scheduler**: This is a generic module with per-flow state variables as input and a flow identifier as output. Moreover, it can support scheduling policy change on-the-fly. Since the scheduler is located on the Host machine, it can support scheduling algorithms with high complexity.
- **Retransmissions**: Retransmission is achieved by creating a separate retransmission queue to store the duplicate of the current scheduled packet. This queue is assigned strictly higher priority than all the other normal flow queues, and therefore there is at most one packet in the retransmission queue at any time. The packet in the retransmission queue will get removed if it misses the deadline.
- **Prepare ICP packets**: Based on the ICP format, the scheduled packet is prepended with the corresponding ICP header, which consists of the configuration of bandwidth, target power level, modulation and coding scheme, source and destination addresses.

3 EXPERIMENTAL RESULTS

We run a series of experiments to demonstrate the ability of the platform to compare the performance of different scheduling policies under different arrival processes, also showing that the platform can switch scheduling policies during the course of an experiment. We consider four policies in this context, Largest Deficit First (LDF), Longest Queue First (LQF), Round Robin (RR) and Random, and enable packet dropping in all of them to ensure a fair comparison.

The setup for the experiments includes a single Access Point (AP) and two clients. Each client hosts two flows, a real time flow where each packet is associated with a deadline and a non real time flow. Both flows are downlink in nature. Packet size is fixed at 1500 bytes and we use IEEE 802.11a with MCS of 7 and a theoretical data rate of 54 Mbps, although we do not utilize all the features available in the standard. Generation of packets is done every 5 ms and the number of packets is uniformly distributed between 0 and K_{RT} for real time flows and between 0 and K_{NRT} for non real time flows.

Since we want to show the ability of the platform to switch policies on the fly, we define a sequence for activating the scheduling policies. Each policy is enabled for a fixed interval of time. We start the experiment with LDF and then switch to the other policies according to the schedule, with LDF interleaved between any two consecutive switches. The sequence which we have considered is as follows: LDF→LQF→LDF→Random→LDF→RR→LDF and the corresponding timestamps are (0, 10, 15, 30, 35, 50, 55) in seconds.

The delivery ratio, deadlines and packet arrival details are as follows. Client 1 has a delivery ratio of 0.97 with a corresponding deadline of 2 ms for real time packets. K_{RT} is set to 2 and K_{NRT} is set to 5. Client 2 has a delivery ratio of 0.98 and a deadline of 3 ms, with K_{RT} set to 4 and K_{NRT} set to 6. This has been represented as (2,5,4,6) in Figures 2 and 3, where we can see that LDF is the only policy which is able to stabilize the evolution of the deficits, while also resulting in the best timely throughput.

The results described clearly exhibit the ability of the platform to compare the performance of different scheduling algorithms under varied arrival processes and to dynamically change scheduling policies during the course of an experiment.

4 DEMO DESCRIPTION

In this demo, we plan to set up a wireless network of three PULS nodes (one AP and two clients). We plan to conduct the following experiments: (i) Let the AP schedule transmissions for two clients, each of which has one real-time and one non-real-time flow, and show the throughput, queue length, and deficit values under different arrival patterns. (ii) Demonstrate the above performance while changing the scheduling policy and other parameters on the fly. The additional facilities needed are 1 monitor and access to at least 8 power sockets. The space required is about 4 feet by 3 feet, and the setup time ia about 30 minutes.

ACKNOWLEDGMENTS

This material is based upon work partially supported by NSF under Contract Nos. CCF-1619085, CNS-1646449, CNS-1719384, CNS-1149458, AST-1443891, Science & Technology Center Grant CCF-0939370, Office of Naval Research under Contract N00014-18-1-2048, and the U.S. Army Research Office under Contract No. W911NF-15-1-0279.

REFERENCES

[1] I-Hong Hou. 2014. Scheduling heterogeneous real-time traffic over fading wireless channels. *IEEE/ACM Transactions on Networking (TON)* 22, 5 (2014), 1631–1644.
[2] National Instruments. 2018. LabVIEW Communications System Design Suite. http://www.ni.com/labview-communications/. (2018).
[3] Simon Yau, Ping-Chun Hsieh, Rajarshi Bhattacharyya, Kartic Bhargav K.R., Srinivas Shakkottai, I-Hong Hou, and P.R. Kumar. 2018. PULS: Processor-Supported Ultra-Low Latency Scheduling. In *Proc. of ACM MobiHoc*.