Round Trip Latency Timing Analysis

Saptarshi Hazra

KTH Royal Institute of Technology

1. Introduction

The goal of the project is to enable real time operation of 802.15.4e TSCH protocol in a software defined radio architecture. The round trip latency measurement is essential to enable ACK and other timing critical operations of the protocol.

The hardware I am using is the LimeSDR-USB which uses a USB 3.0 interface for communicating with the host computer. It supports MIMO operations with 2 RX and TX channels operating simultaneously. The maximum sampling rate supported by the ADC and DAC of LMS7002M is 160 Mhz, but the USB 3.0 restricts it to 61.44 MSPS[1] if all the RX and TX channels are used simultaneously.

However, in order to process data at such high sampling rates I would need huge computing resources. I have been able to do measurements at 20 MHz sampling rate, the results are very encouraging.

2. Method

The project uses the Wime Project[2] implementation of 802.15.4 MAC and PHY layers in GNU Radio. For the purpose of measurement of round trip latency, a loop back experimentation setup was implemented. On the GNU Radio side, the RTT Time Measurement block generates messages and notes down the time of the message ($t1$) and indexes it into a dictionary by defining a key. It is then processed and modulated by the 802.15.4 MAC and PHY respectively and sent through the OSMOCOM transreceiver to the LimeSDR. The RX and TX ports of the FPGA which connects to LMS7002M chip has been shorted and hence the original sent message loopbacks through the FPGA and comes back to the GNU Radio and is demodulated and processed by the PHY and MAC blocks respectively and is ultimately received by the RTT block. It notes down the time($t2$) and searches in the dictionary if it

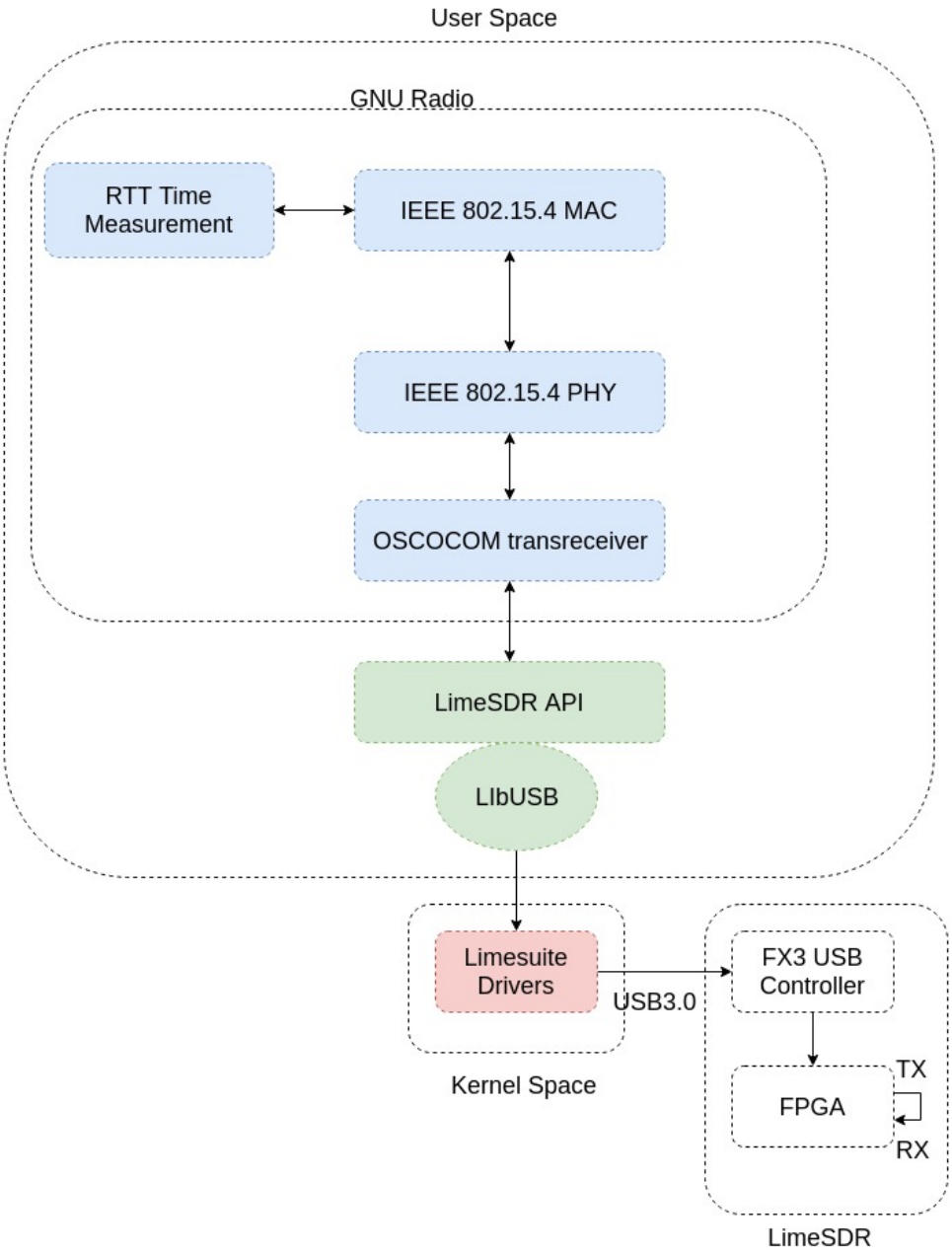

Figure 1: Loop back Experiment setup

corresponds to any valid key. If the valid entry is present then *t2-t1* gives the round trip latency of the message.

3. Results and Analysis

To ensure minimum delays the process was given the maximum priority settings. The delays were measured at different sampling rates supported by the LimeSDR. Two different host computers were used for timing measurements.

Name	**CPU**	**RAM**
Setup_1	i5-4300U @ 1.90GHz	4GB
Setup_2	i7-7500U @ 2.70GHz	16GB

Table 1: Test Setups

Message Size	**Sampling Rate**	**RTT Mean**	**RTT Standard Deviation**	**Setup**
20 bytes	4 Mhz	2911 μs	460 μs	Setup_1
120 bytes	4 Mhz	6241 μs	478 μs	Setup_1
120 bytes	20 Mhz	3542 μs	1238 μs	Setup_1
120 bytes	20 Mhz	1183 μs	103 μs	Setup_2

Table 2: Round Trip Timing Measurements

Taking into consideration, USB transfer time at 20 MSPS with $1Sample = 1.5Bytes$, we get the USB transfer time to be around 1.03 ms. The observed value of 1183 in Setup 2 shows that the GNU Radio Processing and User Space to Kernel Space adds around 180μs. So if I am able to measure at higher sample rates I would be able to get lower latencies. But, so far increasing the speed higher than 20 MSPS overloads the CPU and the process is not able to cope up with processing this data.

Another thing to note from the results is that on Setup 1 increasing the sample rate increases the jitter primarily caused by lack of computing resources on the host computer.

4. Challenges

Theoretically at 61.44 MSPS, the USB round trip transfer time would come down to 336 μs for a 127 bytes packet (Worst case scenario) which

would enable me to use the 1ms ACK window specified by 802.15.4e for TSCH protocol.

The results show that the RTT time decreases significantly with better computing resources. The key challenge is to identify how to handle the increased data rate so that the performance can be enabled on smaller computing resources.

As the IQ samples on the TX and RX stream are different. There is a need for methods to identify the RX stream so as to do proper loop back experimentation. I also need to take a look into the USB MTU size and how it affects the timings as reported by an earlier paper[3].

5. What's next

I propose to do the following :

- Do a detailed analysis of the timing delays at the host computer.
- Look into methods to enable faster data rate handling and processing.
- Implementing the TSCH protocol
- Do a interoperability tests of the SDR setup with Zoul Firefly.

6. References:

[1] Myriad rf discourse forum: Max bandwidth of dual rx, "https://discourse.myriadrf.org/t/max-bandwidth-of-dual-rx/307/5, 2016.

[2] Wime project, https://www.wime-project.net/, 2018.

[3] G. Nychis, T. Hottelier, Z. Yang, S. Seshan, P. Steenkiste, Enabling mac protocol implementations on software-defined radios, in: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, NSDI'09, USENIX Association, Berkeley, CA, USA, 2009, pp. 91–105.