Software-defined Radio Based Measurement Platform for Wireless Networks

I-Chun Chao* Kang B. Lee† Richard Candell† Frederick Proctor† Chien-Chung Shen‡ Shinn-Yan Lin**
*Department of Electrical Engineering, National Taiwan University, Taiwan
†Networked Control Systems Group, National Institute of Standards and Technology, USA
‡Department of Computer and Information Sciences, University of Delaware, USA
**Telecommunication Laboratories, Chunghwa Telecom Company Ltd., Taiwan

***Abstract*—End-to-end latency is critical to many distributed applications and services that are based on computer networks. There has been a dramatic push to adopt wireless networking technologies and protocols (such as WiFi, ZigBee, WirelessHART, Bluetooth, ISA100.11a, *etc.*) into time-critical applications. Examples of such applications include industrial automation, telecommunications, power utility, and financial services. While performance measurement of wired networks has been extensively studied, measuring and quantifying the performance of wireless networks face new challenges and demand different approaches and techniques. In this paper, we describe the design of a measurement platform based on the technologies of software-defined radio (SDR) and IEEE 1588 Precision Time Protocol (PTP) for evaluating the performance of wireless networks.**

***Key words*—**Software-defined Radio, IEEE 1588 Precision Time Protocol, end-to-end latency, hardware time-stamp, out-of-band method

I. Introduction

End-to-end latency is critical to many distributed applications and services that are based on computer networks. Examples of such applications include industrial automation, telecommunications, power utility, and financial services. In particular, the capability of precise timing and time synchronization is of paramount importance for industrial control networks [1]. In addition, for applications such as Voice over Internet Protocol (VoIP) and stream videos, correct timing behaviors are extremely important to their (perceived) performance. As traditional applications can tolerate tens of milliseconds of end-to-end latency, modern real-time precision control and algorithmic trading are sensitive to latency of microseconds or even sub-microsecond. Given the growing demands for lower latency, higher time resolution, and more accurate timing, it becomes essential to measure end-to-end latency with such precision and accuracy.

At the same time, there has been a dramatic push to adopt wireless networking technologies and protocols (such as WiFi, ZigBee, WirelessHART, Bluetooth, ISA100.11a, *etc.*) into industrial control networks [2], [3]. There are two main reasons for such a development. One is the cabling (both material and labor) cost. For instance, a reasonable power utility management system typically includes thousands of measured relay nodes, and the use of any wired technology such as Ethernet will incur high wiring and installation costs [4]. In addition to the cabling cost, there are applications and scenarios that render any wired configuration and deployment infeasible, such as mobile nodes and nodes operating in hazardous environments. In such cases, wireless technologies and autonomous deployment are the only feasible options.

While performance measurement of wired networks has been extensively studied, measuring and quantifying the performance of wireless networks face new challenges and demand different approaches and techniques. For instance, disturbances or noise affecting timing precision incurred during wireless communications should be mitigated as much as possible. One major component of the disturbances is incurred in either the Medium Access Control (MAC) or the Physical (PHY) layer of the protocol stack. It has been argued that so long as the performance metrics, such as one-way delay and jitter, can be precisely measured in the lower protocol layers, performance evaluation at the higher layers, such as quality of service and real-time constraints, can be facilitated [5].

In addition, although the activity of performance measurement for both wired networks and wireless networks shares certain common concerns (*e.g.*, real-time response and determinism), wireless networks impose additional challenges (*e.g.*, multi-path fading and Inter-Symbol Interference (ISI)). Therefore, how to precisely measure the arrival time (or the departure time) of a packet becomes the fundamental issue of getting precise time information in wireless networks.

There exist efforts such as [4] that used hybrid networks to achieve time synchronization between wired and wireless networks, and [6] that applied Network Time Protocol (NTP) to measure the time information between nodes in wireless networks. These efforts are based on the technique of software time-stampers. In comparison to hardware time-stampers [7], these solutions result in either imprecise synchronization performance or coarse accuracy on time-related parameters (*e.g.*, jitter and latency).

In this paper, we describe the design of a measurement platform based on the technologies of software-defined radio (SDR) and IEEE 1588 Precision Time Protocol (PTP) for measuring and evaluating the performance of wireless networks (including wireless sensor networks). By evaluating the performance metrics described in the paper, application behaviors based on robust time synchronization could be better quantified and evaluated.

We proceed in the next section to describe the GNU

Radio software-defined radio platform and the accompanying Universal Software Radio Peripheral (USRP). The SDR-based measurement platform is described in Section III. Section IV describes the software timestamping mechanism implemented in the IEEE 802.11 WiFi protocol and the IEEE 802.15.4 ZigBee protocol, respectively. Section V depicts the hardware platform used to prototype the timestamping mechanism. Demonstrations of running the wireless protocols of WiFi and ZigBee to generate timestamps are also presented. The proposed implementation of a hardware Time-Stamping Unit (TSU) within the Field Programmable Gate Array (FPGA) of USRP is described in Section VI[1]. Section VII concludes the paper with future research activities.

II. Overview of GNU Radio and USRP

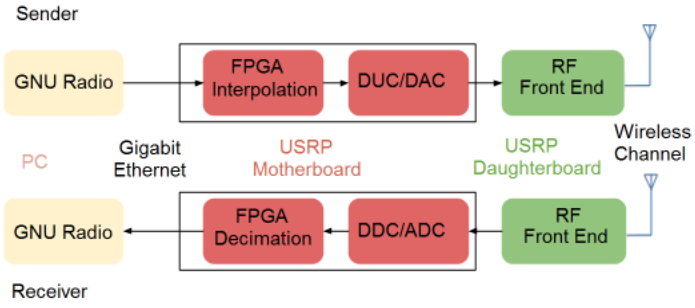

Fig. 1. Wireless communications with GNU Radio and USRP

GNU Radio is a collection of open source software which includes most wireless protocols and necessary modules for radio engineering and signal processing. The physical wave signals transmitted and received are defined by software and implemented on USRPs. Fig. 1 depicts how USRP and GNU Radio work together. The GNU Radio software library only executes on the personal computer (PC), and the USRP motherboard consists of some functionality in hardware, such as signal interpolation/decimation, Analog-to-Digital Converter (ADC)/Digital-to-Analog Converter (DAC), as well as Digital Up Converter (DUC)/Digital Down Converter (DDC). Furthermore, for transmission and reception in different frequency bands, different choices of daughterboards become necessary. For example, if we are conducting an experiment for 802.11 in the 2.4 GHz band, a specific daughterboard for operating in 2.3-2.9 GHz is required. Fig. 2 depicts the roles that both USRPs and GNU Radio play in a Transmission Control Protocol/Internet Protocol (TCP/IP) stack.

[1]Implementation of the hardware TSU has not been completed yet. We present a high level design with specific FPGA modules identified where the hardware TSU could be implemented.

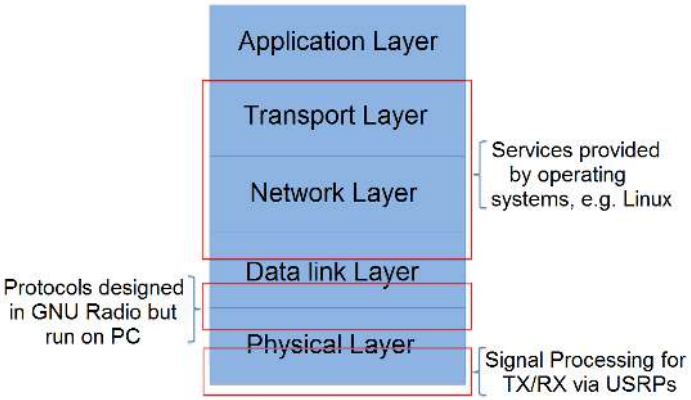

Fig. 2. The roles of GNU Radio and USRP in the TCP/IP stack

SDR is the other critical technology used in our proposed platform. In the following, we introduce the specific SDR platform, USRP and GNU Radio, we propose to use in our design. USRP is a platform for developing software radios, which has been developed in both computer-hosted form and embedded form. For this project, we choose the computer-hosted form. USRPs are controlled with open source drivers USRP Hardware Driver (UHD) and connected to a PC (for computer-hosted form) with either a Universal Serial Bus (USB) or a Gigabit Ethernet link so that radio protocols or algorithms can be designed and executed on a PC while the data are transmitted and received by the USRPs. USRPs are usually developed with the GNU Radio software suite to design complex software-defined radio systems. We selected one of the X series, USRP X310, as the platform for our testbed, which provides higher dynamic range and bandwidth, as well as a Multiple Input Multiple Output (MIMO) expansion port.

III. Measurement Platform

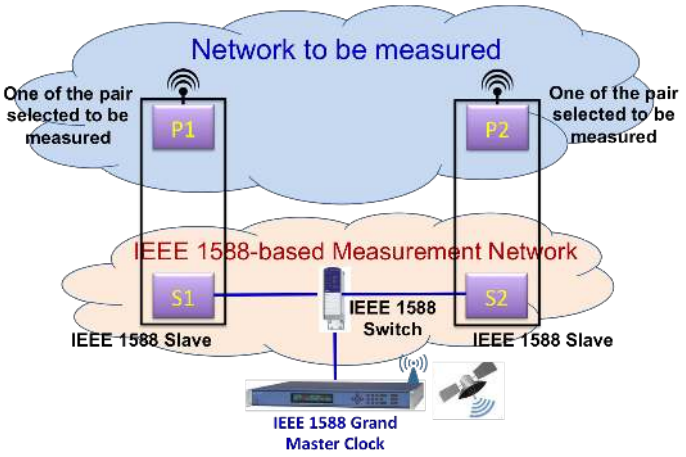

Fig. 3. Architecture of measurement platform

In this section, we describe the architecture of the proposed measurement platform and how we evaluate the results. The section is divided into (1) system architecture, (2) the design

of an Adapting Gateway (AG) (a wireless component using USRP and a wired component using syn1588® Peripheral Component Interconnect Express (PCIe) Network Interface Card (NIC), (3) integration of the wireless and the wired portions, (4) use of wireless communication protocols, and (5) performance measurements.

A. System Architecture

To evaluate the performance of wireless networks we focus on time-related metrics, such as one-way delay and jitter. To obtain precise measurement of such information, it is critical to obtain the precise timestamp of packet arrivals and departures in the appropriate protocol layer(s). Fig. 3 presents the architecture which also depicts how the measurements are performed.

The idea depicted in Fig. 3 is to evaluate any two nodes in a wireless network (the upper network) with a condition that the two nodes are synchronized by using a synchronization network (the lower network). In the architecture, there are two networks and each plays a different role. The upper network is the target wireless network to be measured, termed WNUM (Wireless Network Under Measurement). Any pair of measured nodes (*e.g.*, P1 and P2 in the Fig. 3) in the WNUM may be selected and the performance such as latency, jitter, and one-way delay between the nodes may be obtained. The lower network is a PTP-based synchronization network responsible for synchronizing the clocks on the selected pair of nodes in the WNUM to a grand master traceable to, for instance, GPS (Global Positioning System) time.

In the PTP-based synchronization network, slaves (S1 and S2) selected to perform synchronization are actually part of their respective Adapting Gateways, which adapt an Ethernet network to a specific wireless network. For example, a gateway for measuring the performance of a ZigBee network is an Ethernet-ZigBee AG. The reason why measured nodes and slave nodes need to be integrated into one device is our proposed use of IEEE 1588 to synchronize S1 and S2. IEEE 1588 can guarantee the synchronization performance within sub-microsecond or less over Ethernet, so the arrival time and the departure time of each packet delivered from one end of the selected pair to the other end can be precisely measured. This specific AG for measuring wireless networks will be designed as depicted in Fig. 4. Inside the AG, there is a clock, which should be with high enough quality such as Oven-Controlled Crystal Oscillator (OCXO). This clock, located on the syn1588 PCIe NIC, will be synchronized to a grand master using IEEE 1588 via the syn1588 PCIe NIC, and, in the meantime, be a time source for drawing precise timestamps through USRP. In Fig. 4, we only consider one direction, either in a transmission or a reception. If the measurement requirements demand both directions, due to the asymmetric propagations of wireless communications, two antennas with daughterboards on each USRP become necessary.

B. Adapting Gateway (AG)

The functionality of AG can be divided into two components: one module to connect a wireless network (WNUM) and one to connect an Ethernet network (an PTP-based synchronization network).

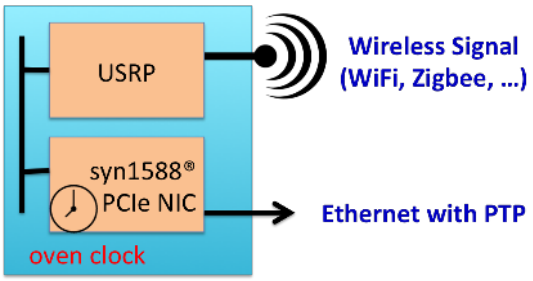

Fig. 4. Adapting gateway (AG)

1) Design of AG at wireless portion by using USRP: A USRP X310 is connected to a PC and equipped with two SBX 400-4400 MHz Rx/Tx daughterboards for receiving and transmitting wireless signal in the 400-4400 MHz band. Popular wireless technologies such as WiFi and ZigBee operate in this band. By programming new FPGA hardware modules for generating hardware timestamps while converting baseband signals to digital format, the customized USRP X310 also functions as a precise hardware time-stamper for the events of the packet arrival and departure in a wireless network.

2) Design of AG at wire portion by using syn1588 PCIe NIC: Since the clock on an AG directly affects the quality of the timestamp that USRP generates, synchronization between the node pair selected for measurement is extremely important. We use PTP to meet the necessary synchronization between the clocks on the chosen node pair in the WNUM. One syn1588 PCIe NIC plays the role of synchronizing to a grand master with PTP. Based on the results of our preliminary work, synchronization can achieve a precision on the order of hundred-nanosecond or better.

3) Integration of wireless and wired portions: However, two issues arise when a wired portion and a wireless portion are integrated. One issue is how USRP X310 and syn1588 PCIe NIC can share the same clock in an AG, and the other is how to coordinate these two components in a consistent manner. We propose to develop a software-based solution using Inter-Process Communication (IPC). There will be four processes running in Linux on the host computer to access both USRP X310 and syn1588 PCIe NIC, respectively. The first process, written with GNU Radio in Python, implements certain targeted protocols, *e.g.*, 802.11, ZigBee, etc. The second process accesses syn1588 PCIe NIC so that the internal OCXO can be precisely synchronized to a grandmaster. The third process draws timestamp information from the USRP X310. The last process will be an application program for accessing timestamp information on the two nodes of the selected pair in WNUM, so as to generate the performance

indices of latency, jitter, and one-way delay to evaluate the WNUM.

IV. Software Timestamping in GNU Radio Wireless Protocols

In this section, we identify the specific blocks in the signal flow graphs where the software timestamping mechanism is implemented for WiFi and ZigBee, respectively. All these flow graphs were created with the GNU Radio Companion (GRC), a graphical user interface to GNU Radio.

A. IEEE 802.11 WiFi

Figs. 5 and 6 are the signal flow graphs

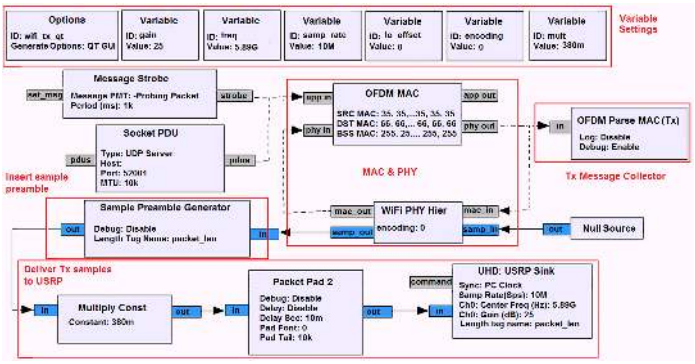

Fig. 5. Flow graph of IEEE 802.11 Tx Path

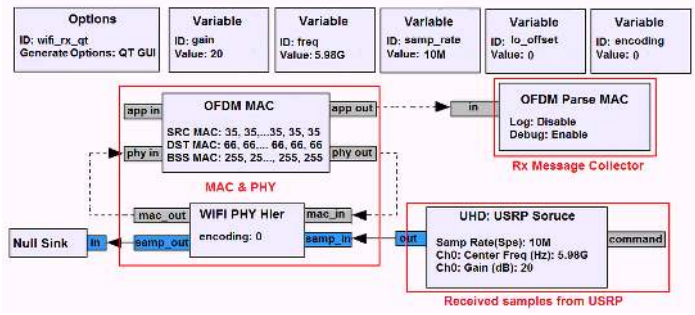

Fig. 6. Flow graph of IEEE 802.11 Rx path

B. IEEE 802.15.4 ZigBee

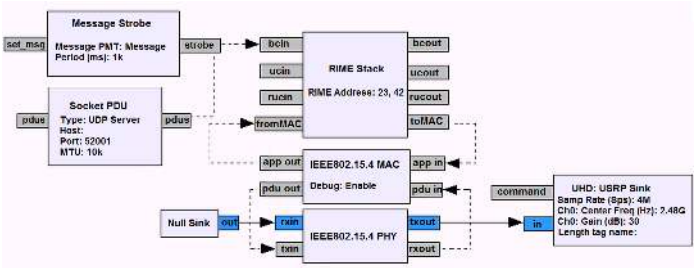

Fig. 7. Flow graph of IEEE 802.15.4 (ZigBee) Tx path

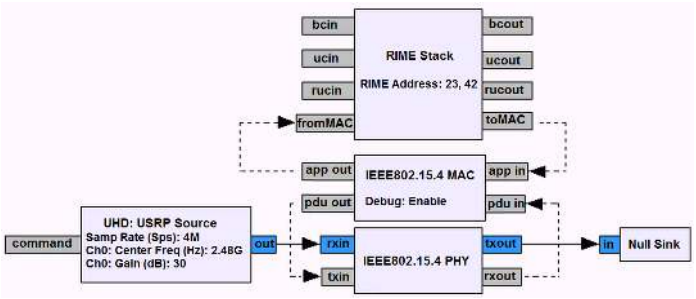

Fig. 8. Flow graph of IEEE 802.15.4 (ZigBee) Rx path

V. Timestamping Testbed and Demonstration

Fig. 9 and Fig. 10 depict the execution traces of timestamp generation while communicating via the IEEE 802.11 WiFi protocol and the IEEE 802.15.4 ZigBee protocol, respectively. For WiFi Tx, timestamps are generated in the block of "OFDM Parse MAC" of both Tx and Rx nodes. For ZigBee, timestamps are generated in the block titled "IEEE802.15.4 MAC" in both the Tx and Rx nodes.

VI. Hardware TSU in the USRP's FPGA

Conceptually, hardware TSUs should be implemented, along the signal path, as close to the physical network interface (to the connection wire for wired networks or to the antenna for wireless networks) as possible to mitigate timing uncertainty. In the context of USRP, this is depicted by the two red arrows in Fig. 11, one before the digital upconversion (DUC) step in the transmitter chain and the other after the digital downconversion (DDC) step in the receiver chain.

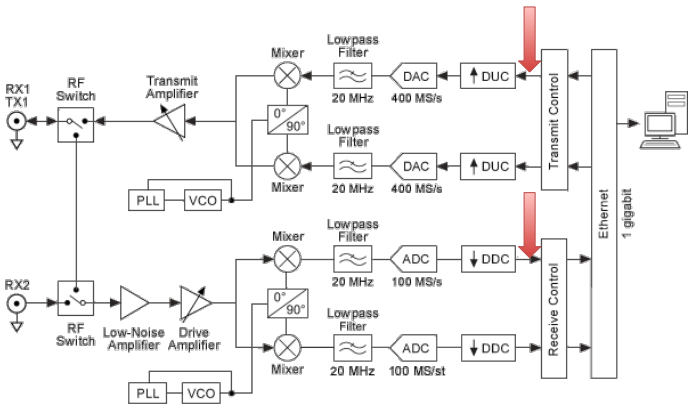

Fig. 11. Conceptual location of hardware TSU in USRP [8]

Since both DUC and DDC are implemented in the FPGA on the USRP, the two red arrows in Fig. 12 correspond to the two in Fig. 11 and depict the locations of the hardware TSUs inside the FPGA of the USRP X310. Specifically, we propose to develop (1) the *Tx TSU module* in-between the two existing FPGA modules `new_tx_control` and `duc_chain`, and (2) the *Rx TSU module* in-between the two existing FPGA

```
isen@machine1:~$
Generating: "/home/isen/nist_work/src/gr-ieee802-11/examples/wifi_tx_qt.py"

Executing: "/home/isen/nist_work/src/gr-ieee802-11/examples/wifi_tx_qt.py"

linux; GNU C++ version 4.8.2; Boost_105400; UHD_003.008.001-0-unknown

Using Volk machine: avx_32_mmx_orc
OFDM MAPPER: encoding: 0
set_min_output_buffer on block 10 to 48000
set_min_output_buffer on block 14 to 100000
set_min_output_buffer on block 15 to 10000
set_min_output_buffer on block 17 to 48000
-- X300 initialization sequence...
-- Determining maximum frame size... 1472 bytes.
-- Setup basic communication...
-- Loading values from EEPROM...
-- Setup RF frontend clocking...
-- Radio 1x clock:200
-- Initialize Radio control...
-- Performing register loopback test... pass
-- Performing register loopback test... pass
-- Initializing clock and time using internal sources... done
set_min_output_buffer on block 31 to 100000
0-Probing Packet (1425151953961 ms)
1-Probing Packet (1425151954961 ms)
2-Probing Packet (1425151955962 ms)
3-Probing Packet (1425151956962 ms)
4-Probing Packet (1425151957962 ms)
5-Probing Packet (1425151958962 ms)
6-Probing Packet (1425151959962 ms)
7-Probing Packet (1425151960962 ms)
8-Probing Packet (1425151961962 ms)
9-Probing Packet (1425151962962 ms)
10-Probing Packet (1425151963962 ms)
11-Probing Packet (1425151964962 ms)
```

```
isen@machine2:~$
Generating: "/home/isen/nist_work/src/gr-ieee802-11/examples/wifi_rx_qt.py"

Executing: "/home/isen/nist_work/src/gr-ieee802-11/examples/wifi_rx_qt.py"

linux; GNU C++ version 4.8.2; Boost_105400; UHD_003.008.001-0-unknown

Using Volk machine: avx_32_mmx_orc
OFDM MAPPER: encoding: 0
set_min_output_buffer on block 10 to 48000
set_min_output_buffer on block 14 to 100000
set_min_output_buffer on block 15 to 10000
set_min_output_buffer on block 17 to 48000
-- X300 initialization sequence...
-- Determining maximum frame size... 1472 bytes.
-- Setup basic communication...
-- Loading values from EEPROM...
-- Setup RF frontend clocking...
-- Radio 1x clock:200
-- Initialize Radio control...
-- Performing register loopback test... pass
-- Performing register loopback test... pass
-- Initializing clock and time using internal sources... done
gr::log :WARN: gr uhd usrp source0 - Sensor 'lo_locked' failed to lock within timeout on c
hannel 0.
0-Probing Packet (1425151953876 ms)
1-Probing Packet (1425151954875 ms)
2-Probing Packet (1425151955873 ms)
3-Probing Packet (1425151956875 ms)
4-Probing Packet (1425151957873 ms)
5-Probing Packet (1425151958874 ms)
6-Probing Packet (1425151959873 ms)
7-Probing Packet (1425151960874 ms)
8-Probing Packet (1425151961876 ms)
9-Probing Packet (1425151962874 ms)
10-Probing Packet (1425151963875 ms)
11-Probing Packet (1425151964874 ms)
```

Fig. 9. Timestamping demonstration over IEEE 802.11 (WiFi)

modules `new_rx_framer` and `ddc_chain_x300` to generate the Tx and Rx timestamps, respectively.

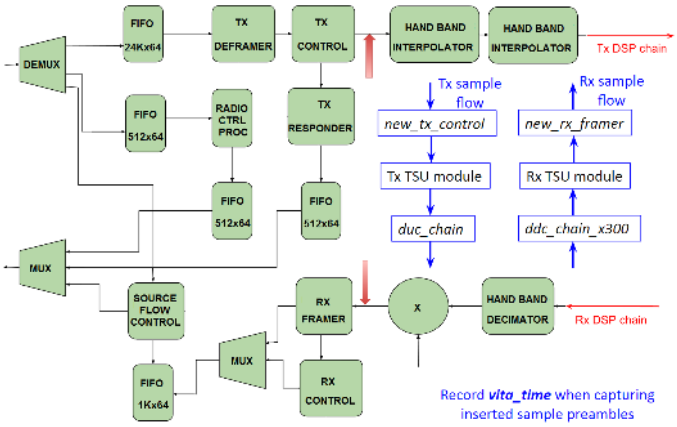

Fig. 12. Proposed location of hardware TSU in USRP's FPGA

To know the exact time at which to generate timestamps, we need a "triggering" mechanism that causes the *Tx TSU module* to generate a timestamp when a packet is about to be transmitted. Similarly, by recognizing the triggering condition for an incoming packet, the *Rx TSU module* generates the arrival timestamp. In our implementation, a *Sample preamble* is added, by the Sample Preamble Generator block in Fig. 5, before the Orthogonal Frequency-Division Multiplexing (OFDM) preamble of each outgoing packet, as depicted in Fig. 13. By recognizing the Sample preamble of an incoming packet, an Rx timestamp is generated.

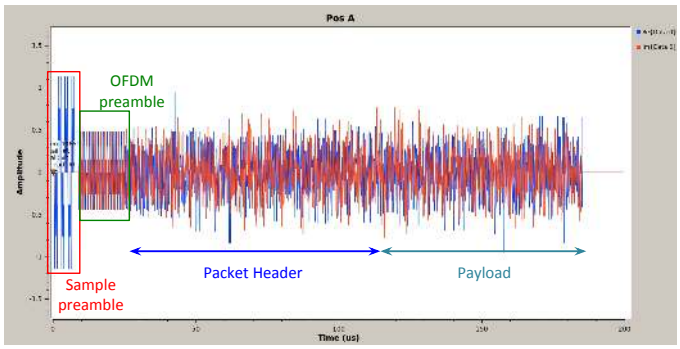

Fig. 13. Adding sample preamble before OFDM preamble

VII. Conclusion

In this paper, we describe a measurement platform based on the technologies of software-defined radio (SDR) and IEEE 1588 Precision Time Protocol (PTP) for measuring and evaluating the performance of wireless networks (including wireless sensor networks). By evaluating the performance metrics described in the paper, application behaviors based on robust time synchronization could be better quantified and evaluated. Work is in progress to complete the implementaion of the hardware timestmper inside the FPGA of USRP X310.

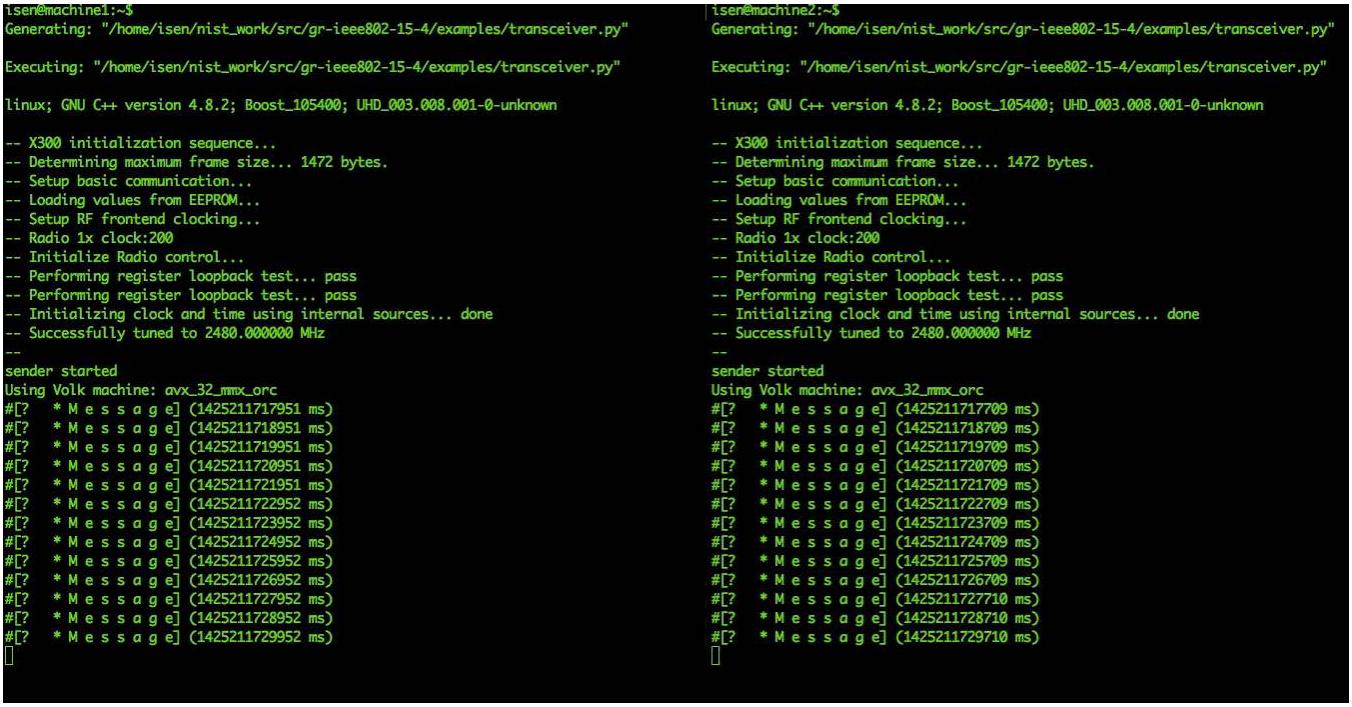

Fig. 10. Timestamping demonstration over IEEE 802.15.4 (ZigBee)

ACKNOWLEDGMENT

The work is supported in part by National Institute of Standards and Technology (NIST) Cooperative Agreement #70NANB14H087.

DISCLAIMER

No approval or endorsement of any commercial product by the National Institute of Standards and Technology is intended or implied. Certain commercial equipment, instruments, or materials are identified in this paper in order to facilitate understanding. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose. This publication was prepared by United States Government employees as part of their official duties and is, therefore, a work of the U.S. Government and not subject to copyright.

REFERENCES

[1] B. Galloway and G. P. Hancke, "Introduction to industrial control networks," *IEEE Comm. Surveys & Tutorials*, vol. 15, no. 2, pp. 860–880, Second Quarter 2013.

[2] A. Willig, "Recent and emerging topics in wireless industrial communications: A selection," *IEEE Transactions on Industrial Informatics*, vol. 4, no. 2, pp. 102–124, May 2008.

[3] J. R. Moyne and D. M. Tilbury, "The emergence of industrial control networks for manufacturing control, diagnostics, and safety data," *Proceedings of IEEE*, vol. 95, no. 1, pp. 29–47, January 2007.

[4] A. Mahmood and F. Ring, "Clock synchronization for IEEE 802.11 based wired-wireless hybrid networks using PTP," in *Proc. IEEE Int. Symp. Precision Clock Synchronization Meas., Control Commun.*, San Francisco, California, September 23-28 2012, pp. 1–6.

[5] A. Hernandez and E. Magana, "One-way delay measurement and characterization," in *Proc. 3rd International Conference on Networking and Services (ICNS)*, Athens, Greece, 2007, pp. 114–120.

[6] K. Stangherlin, R. C. Filho, W. Lautenschlager, V. Guadagnin, L. Balbinot, R. Balbinot, and V. Roesler, "One-way delay measurement in wired and wireless mobile full-mesh networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, Cancun, Mexico, 2011, pp. 1044–1049.

[7] A. Mahmood and G. Gaderer, "Timestamping for IEEE 1588 based clock synchronization in wireless LAN," in *Proc. IEEE Int. Symp. Precision Clock Synchronization Meas., Control Commun.*, Brescia, Italy, 2009, pp. 1–6.

[8] "What Is NI USRP Hardware?" National Instruments, Tech. Rep., April 1 2015. [Online]. Available: http://www.ni.com/white-paper/12985/en/