Master Thesis

What is my research?

Software Radio Timing Characterization

1. Introduction

1.1 Need

- With ever increasing number of devices and people getting connected to the cellular networks and internet.
- There is a need for rapid development of communication protocols to cope up with the increasing demand for data rate.
- Now different kinds of devices have diverse communication needs, hence there is a need for viable platforms which offer flexibility in terms of communication protocol design and evaluation.
- (Context) Mention the need of SDR for 5G-Coral, where there is a need for flexibility in the deployment platform.

1.2 SDR

- Most widely used approach today is simulation, but modeling real world usage is complex and simulation falls short.
- Software Radio allows us to design and develop the systems which can be tested in real world use cases.
- The SDR platform can help in design of flexible and rapid prototyping platforms. Now the question arrives what is a SDR?

1.2.1 Problem Area

- But Software Radio has unique limitations in:
  - Time Delay.
  - Jitter.

1.3 Implications

- Most modern wireless communication systems share the medium with other devices, hence they have unique MAC mechanisms to manage access.
- Most of these protocols have stringent requirements on the round trip time so as to efficiently use the spectrum.
- But time delay and jitter of SDR makes it difficult to implement these protocols in this platform.

1.3.1 Schmid et .al

- One example of this problem is demonstrated by Schmid et.al where he benchmarks a USRP2 based 802.15.4 system and finds that these timing delays are significant and can lead to blind spots in carrier sensing based systems. He attributes most of these delays to bus communication delays.
- He suggests use of TDMA based protocols would be a good fit for SDR systems. But here the jitter makes the system difficult to synchronize and stick to schedule.

1.4 GAP

- Hence, to develop protocols for these platforms the MAC developer should be aware of the system delays and how system parameters impact these delays.
- This project tries to fill that gap by taking a closer look at these delays for a SDR based system, and testing how various system parameters affect these delays.

1.5 Project Context

- Internet of Things is a rapidly developing application scenario with diverse number of devices.
- There is a need for rapid evolvement and flexibility in IOT systems.
- Mention 5G-Coral Project.
- So in this project, I decided on concentrate on the understanding of one of these LPWAN protocols i.e. 802.15.4.
- The platform used in this project is the new SDR platform, LimeSDR, which has offers significant capability and supports all the frequency channels of interest, at a fraction of the cost of much more expensive SDR platforms like USRP.

2. Background

- Essential Concepts
- LimeSDR.
- 802.15.4 and Wime Project.
- Tools
  - USBMon
  - pidstat

2.1 Essential Concepts

- MAC and PHY layers.
- Software Radio.
- GNU Radio.

2.1.1 MAC and PHY layers.

- Most modern communication systems are based on OSI abstract model.
- PHY Layer: Handling Modulation, line encoding and converting digital representation to electrical signals.
- MAC Layer: Handling access to a common communication medium. Popular protocols involve CSMA and TDMA. Brief account of each.

2.1.2 Software Radio

- Main points.
  - Flexibility: Move from H/W to S/W.
- Architecture.
  - Host-PHY.
  - NIC-PHY.
- Highlight different application scenarios: RFID, Heterogeneous Platforms.
- Explain the Host-PHY architecture in more details. (As its popular, and the architecture I am using.)

2.1.3 GNU Radio.

- Digital signal processing framework easily integrable with hardware platforms.
- It helps to design systems with visually in a modular manner, supports stream interface.
- Scheduler handles how the blocks are executed and when to schedule a particular block.
- Each block has an associated event handler, input and output buffers and signaling mechanism.
- Overview of the GNU Radio Scheduler.(Need to add the points for this.)

2.2 LimeSDR(Overview)

- Low Cost SDR , 100KHz to 3.6 GHz, 160MHz ADC and DAC, USB 3.0 connection.
- General board specifications
  - LMS7002M
  - Altera Cyclone IV EP4CE40F23
  - Cypress USB 3.0 CYUSB3014-BZXC
  - Programmable Clock Generator (Si5351C)

2.2.1 LMS7002M

- Overview: 2*2 MIMO FPRF chip.
  - Control Interface: SPI.
  - Data interface: 12 bit data interface(JESD).
- Brief: Digital and Analog Chains.

2.2.2 Hardware Architecture

- Picture of the overall architecture:
  - Control Path.
    - NIOS CPU
  - Data Path.
    - TX Path.
    - RX Path.
  - USB Slave Interface

2.2.3 Software Architecture

- Organization:
  - Stream Channel and FIFOs: Interface to GNU Radio.
  - Streamer: Handles the USB Packets on TX and RX side.
  - Dataport: To communicate with the appropiate hardware, which in this case is USB so communicates with libusb.

2.3.1 802.15.4

- Low Rate data specification of PHY and MAC layers.
- Explain the OQPSK PHY.
- Explain the MAC format. Method of medium access (CSMA, TDMA).

2.3.2 Base System- Wime Project-802.15.4

- Implementation of PHY.
- Implementation of MAC.
- Rime Stack.
- USRP Platform.

2.3 Evaluation Tools

- **2.3.1** USBMon: Description and method for measurement.
- **2.3.2** Pidstat: Tool and data description. Method of measurement.

3. Literature Study

- Papers to discuss:
  - Nychis et. Al “Enabling MAC protocol implementation of Software Radio”
  - Bastian Bloessl et.al “GNU Radio based 802.15.4 testbed.”
  - “Timings Matter:Standard Compliant IEEE 802.11 Channel Access for a Fully Software-based SDR Architecture”
  - Nguyen B.Truong-“Investigating Latency in GNU Software Radio with USRP Embedded Series SDR Platform”.
  - Discuss about SORA.
  - Discuss about CRAN.

4. Method.

- Explanation of various experiments and the methods used in measurement.

4.1.1 Loopback Setup

- Why loopback setup was chosen?
  - To ignore the over the air time, interference and make it simpler to measure using a single SDR platform.
  - Another reason is if there the RX and TX are on different computers, for time delay measurements, they need to be synchronized.
- Explain the modification in the Wime Project to accommodate the LimeSDR.
  - tx_eob (why?)

4.1.1 System Description.

- System Flowgraph, hardware and software description.

4.2.1 RTT Measurement.

- Periodic Message Source.
- How the system measures RTT?
- Rationale of doing the timing measurements at those points (t1 and t8).
  - All the MAC timings are typically defined with reference to the first element or the last element. But its never first element with last element.
- Change in PHY layer to accommodate change in sampling rates.
- Variation of RTT with sampling rate.

4.2.2 Rationale: Experiment 1

- How varying message sizes and sampling rates helps me in answering my research question?
  - **The comparative study will help me have a rough map of computing resources and key protocol features like B/W and MTU size.**
  - This experiment helps me answer what kind of protocols are good for implementation with this setup?
    - High Sampling Rate → High B/W → High Data rate protocols.
    - Large Message Sizes → Higher Data Rate (Argue why it is the case)

4.2.3 Experiment 1: RTT vs Message Size, Sampling Rate.

- Input Metrics.
  - Message Size.
  - Sampling Rate.
- Output Metrics.
  - Latency.
  - Jitter.
- Experimental Design.
  - Number of packets sent: 10000(Need to explain why?). How the data was collected?: Running the process at the maximum priority level to minimize context switches
- Analysis Method (?)

4.3.1 Rationale for doing component timing analysis.

- Closer look at the component timings helps me figure out the timing bottlenecks.
  - Longer Software Transfers → Need for more host processing/ memory.
  - Longer Kernel Time → Need to handle task scheduling in the proper way.
  - Longer Bus Transfer → Need for better bus technology.
- Implementation decisions:
  - Where to insert timing probes? What information does it give us?
  - What methods should be used in online and offline measurements?
  - How to figure out t4 and t5 from the usbmon measurements (online vs offline) ?
  - (Analysis)Problems with figuring the relationship between individual data points

4.3.2 Experiment 2 Design and Analysis

- Output Metrics:
  - Component timings on RX and TX paths.
- Input Metrics:
  - Sampling Rate, Message Size
- Experimental Design:
  - Same as experiment 1
- Analysis Method (?)

4.4 Experimental 3:Setup:RTT vs Batchsize

- Rationale: Helps in understanding how bus transfers affects the RTT time (which is the most significant according to Schmidt et al).
  - How to say this comes from the results of Experiment 2?
- Output Metric: RTT and USB Transfer Times.
- Input Metric: Batchsize.
- Experiment Design: Number of packets, how the data was recorded.
- Analysis: Explain the Queuing Time in relation to the bus transfer delay, and how it affects the latency and jitter.

4.5.1 Experiment 4

- Follows the analysis of the experiment 3, since lower batch sizes gives lower latency and jitter, see how lower packet sizes affect the system .
- Check how the performance is related to the host computer resources.

4.5.2 Exp 4: Change in FPGA hardware.

- To accommodate the changes in number of samples in one packet, there is a need to change the TX and RX paths in FPGA.
- In Depth understanding of the TX and RX paths of the FPGA.
- Explanation of Modifications Made.

4.5.3 Exp 4: FX3 Firmware

- DMA Channel Modifications.

4.5.4 Exp 4: Software

- Changes in Streamer class.
- Changes in FPGA data packet structure.

4.5.5 Experimental Setup: RTT, Host Computer Resources vs USB Transfer Size, GNU Radio Block Buffer

- Experiment Design.
  - Input Metrics:
    - USB Transfer Size.
    - GNU Radio Block Buffer Size.
  - Output Metric:
    - RTT
      - Latency
      - Jitter
  - Experiment Design: 1000 secs, Measurement setup, python experiment generation
- Analysis(?)

5.Results and Analysis

5.1.1 Exp 1

- Box Plot of the RTT with sampling rates, message sizes.
- Appendix 1: Summarize the percentile results in Table.

5.1.2 Analysis #1

- Higher Sampling Rates, leads to lower RTT, because of lower queuing time (Refer to Methods.)
- But since the host processor now needs to handle a large amount of data, there is a increase in jitters and number of outliers.
- With my new t1 and t8 probes, the RTT for different message sizes are more or less equal.

5.2.1 Experiment 2

- Tabulate the different methods using min-max method.
- This will help in showing the variation of the component timings and help in better understanding of what is causing the major bottlenecks.
- Results Brief:
  - GNU Radio processing takes up most of the time.
  - Followed by the USB time.
- Highlight the 4MHz result as this going to the one I am using in the next part of analysis.

5.2.2 Analysis 2

- Question to be answered: Why is GNU Radio taking so much time in the receive chain.
- Need to figure out how different USB Transfer sizes affect the measurement.

5.3.1 Experiment 3

- RTT vs Batchsize: Lower Batchsizes leads to lower latency and jitter.

5.4Experiment 4:

- RTT vs USB Transfer Sizes, GNU Radio Buffer Sizes.
- Analysis

6 Conclusion and Future Work

6.2 Future Work

- Enabler of future work, maybe protocols designed with this knowledge needs to be evaluated.
- GNU Radio Buffer sizes balanced for optimum throughput.