Wireless Transmission of JPEG file using GNU Radio and USRP

Sachin Hirve, Saikrishna Gumudavally, Department of Electrical and Computer Engineering,
Cleveland State University

Abstract—**Wireless applications being on the rise with advent of new gadgets, there is increased need of transferring the hardware complexity to software in wireless transmission and reception. It helps in development as well as the multi-standard support for the wireless devices, where software enables making the radio functions hardware independent. Based on this premise, there is a lot of development taking place in the domain of Software Radio using GNU Radio (an Open Source toolkit) and USRP as hardware platform. Along the similar lines, we are planning to implement JPEG file transfer over wireless network using the GNU Radio framework with USRP hardware.**

Index Terms—**Access Protocols (Aloha), Software Defined Radio (SDR), Universal Software Radio Peripheral (USRP), Python.**

I.INTRODUCTION

SOFTWARE defined radio (SDR) is finding an increased importance due to flexibility that it provides to implement radio functions. GNU Radio, an open source software tool, being a driving force to develop new wireless applications, is adding to multiple functionalities from same hardware. Universal Software Radio Peripheral (USRP) is the hardware device which is flexible enough to integrate with GNU radio. GNU radio and USRP together help in implementing Software Defined Radio functionalities. The base concept of SDR is to make radio functions hardware independent. The tasks like modulation, demodulation, encoding etc are implemented in software which eliminates the need of corresponding hardware. In essence, SDR leaves the hardware to do the basic functions like transmission/reception of signal and does all the complex signals processing on the general purpose processor thereby relieving the hardware from signal processing complexities.

II.BACKGROUND

As mentioned above GNU radio and USRP help in realizing SDR. GNU radio provides us with C++ classes which implement various signal processing functions, in other words they replicate hardware signal processing units. While the signal processing blocks are implemented in C++, the main application programming is done in Python. SWIG works as glue between C++ classes and Python language. SWIG is a Linux package that converts the C++ classes into Python compatible classes. This way, GNU Radio framework is able to harness both languages. While C++ provides compact code for signal processing block, Python is preferred because of its flexibility and ease to program. Using Python and C++ in combination, the signal processing units are implemented on general purpose processor by GNU Radio.
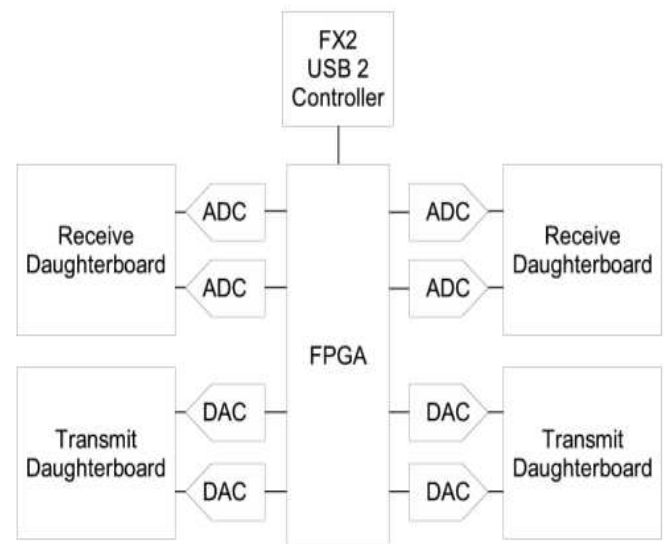

Fig. 1 Block diagram of basic USRP system [1]

USRP is the hardware platform for SDR which is composed of FPGA, ADC/DAC and USB controller. FPGA is used to reduce the data rate on wireless channel to the acceptable data rate to be transmitted over USB2.0 to the host computer [3]. ADC/DAC converts the data

from analog to digital format and vice-versa. And USB is used to transfer data from USRP to host computer for signal processing.

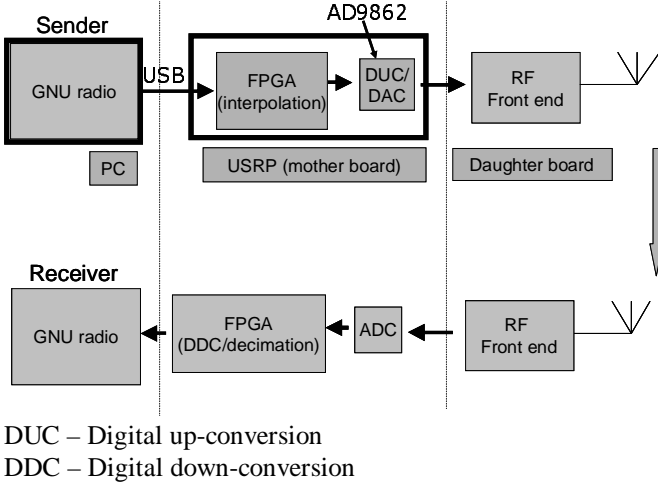DUC – Digital up-conversion
DDC – Digital down-conversion

Fig. 2 Software Radio Architecture [5]

For the project purpose we are using Ubuntu 7.04 version as the development platform. It is found to be the most user- friendly flavor of Linux operating system with ease of installation of GNU Radio package. We will also be using wingware 3.0 as a software development environment.

III. RELATED WORK

There has been a number of application development based on GNU Radio. Some of the applications use USRP as an integral part of the system whereas some of the implementation does not need USRP and basically concentrate on modulation/demodulation schemes rather than wireless communication challenges. We found four main applications which are quite close to our project title.

The first implementation is JPEG file transfer over TCP/IP socket communication between two host computers without using USRP [4]. This implementation concentrated mainly on the modulation/demodulation scheme and JPEG encoding technique. While the main consideration was to test GNU Radio capability to transfer the image file, wireless communication was not addressed in this implementation.

Second implementation used the wireless capability of the GNU Radio and USRP by implementing Aloha protocol to transfer a data file with a back-to-back cable connection between USRPs. While this scenario was quite similar to wireless communication, it failed to consider the interference and noise present in the

environment due to noise-free channel provided by the back-to-back cable connection. Apart from it, it is also having some overheads in the protocol header which is still unaddressed in current implementation [2].

The third implementation is part of the GNU Radio package, in which the data is transmitted on virtual IP/Ethernet port. It resembles the data transmittion over TCP/IP without USRP [5].

The last implementation is also a part of the GNU Radio package, in which the sample data is transmitted on wireless channel using python package consisting of "benchmark_tx.py" and "benchmark_rx.py" files. It resembles the data transmittion over MAC layer in wired network [1].

After studing the above mentioned existing wireless applications using GNU Radio and USRP, we have realised that even though there have been attempts to transfer a JPEG file though GNU Radio without USRP and data transfer through Aloha, there is still a scope for testing the wireless communication for file transfer application. Therefore we propose to implement wireless JPEG file transfer application over USRP as our course project.

IV. PROJECT

A. Project Goal

We have implemented the wireless transmission of JPEG file between two USRP boards using the modules in GNU Radio part of the course project.

B. Project Plan

For the ease of development the project is split into two stages where we initially implemented the file transfer between the USRP's using the loop back cable and later moved to implement wireless transfer of the file. The project is further divided into two parts mainly categorized as sender and receiver implementation.

The radio functions like modulation or demodulation depending on whether it is a sender or a receiver is done by GNU Radio and transmission or reception of packets is done by the USRP at both the ends.

C. Issues faced

From our study of previous implementations resembling our project goal, we decided to follow Aloha-approach. While trying to reproduce the communication as given in documentation, we found some of the

problems hindering our progress in implementing wireless transmission.

- Long header resulting in larger overhead.
- Python compatibility issue
  - o Previous work used older version of Python (earlier than 2.3) having timing module.
  - o Current version (2.5) has no support for timing block.
  - o Therefore, work around for using timing module functionality was needed.
- GNU Radio project version was quite old (2.5). Currently GNU Radio version is 3.1.1 and is not completely compatible with GNU Radio versions earlier than GNU Radio version 3.0.

The above issues resulted in difficulties to setup wireless communication based on Aloha protocol using the existing GNU Radio project [2]. Based on this, we changed our approach for wireless transmission. We took "benchmark_tx.py" and "benchmark_rx.py" as our base framework. These two python modules in existing GNU Radio project are found to be working for wireless communication between two USRP hardwares. This implementation uses one way data flow. We made some changes in this module to enable JPG file transission with loop-back cable. We further made modifications in the existing code and made it two way communication by introducing acknowledgement based reception in transmission and reception.

D. Project Schedule

As per previously defined stages, the project was divided in two parts and the time frame for their completion was set as below.

1. Transferring the JPG file from one USRP to another through loop-back cable connection.
   - March 31, 2008
   - This deadline got extended till April 20th, 2008 due to poor packet reception at receiver end in case of low-frequency daughter boards.
2. Transferring the JPG file over wireless network.
   - April, 30, 2008
   - This deadline was met in time and wireless transmission was implemented with file transfer capability.

E. Procedure

As per previous discussion, "benchmark_tx.py" and "benchmark_rx.py" have been used as frame work to generate packets at the transmitter and receiver respectively. The basic Benchmark_tx just dumps the data i.e. just transmits the data through USRP on wireless channel. Benchmark_rx application is always in listening mode and just listens to the incoming data through USRP. These two python codes take various command line arguments such as type of de/modulation scheme, bit rate and the packet size. The default modulation scheme, bit rate and packet size is GMSK, 500kbps, and 1500 bytes respectively. Other de/modulation techniques like DBPSK and DQPSK can also be used with these Benchmark_tx and Benchmark_rx programs. For our implementation, we chose DBPSK.

At the Sender end, an application written in python, modulates the input JPEG file with one of the existing modulation schemes (DBPSK). Then the modulated signal is transmitted over wireless using the USRP and its RF front-end (1200MHz tranceiver).

The transmitter after transmitting a packet waits for an "ACK" (for transmitted packet) to transmit the next packet. After transmitting a packet, it starts a timer. If it receives a "NACK" or fails to receive any thing before timer expires, it retransmits the packet.

At the receiver end, we will receive the transmitted signals, which will be brought to the base band by the RF front-end and FPGA. The symbol stream received by the receiving host computer is demodulated using DBPSK demodulaton scheme. The received packets at the receiver are checked for errors at two levels, one where the packet header and PN code are checked and the other where CRC of the payload is verified [6]. If the CRC check is successfully completed then an output message "true" is displayed else "false" is displayed along with the packet number. If the packet is successfully received by the receiver then an "ACK" is transmitted else a "NACK" is sent.

Receiver checks a new packet on reception against the latest successfully received packet. If it is a retransmitted packet by sender, it sends an "ACK". Otherwise the new packet is saved in destination file and an "ACK" is sent for an ackmowledgement of the same.

V. EXPERIMENTAL SETUP

We used two host machines and two sets of USRP hardware for this project. The daughter boards for

intended application were 1200 MHz tranceiver boards. 2400 MHz tranceivers were also tried but found to be susceptible to noise which we think is due to wi-fi network in the vicinity of the lab environment. For loop-back file transfer scheme, we found that BasicTx and BasicRx daughterboards give good performance.

Our first implementaion needed the following setup as shown in Fig. 3. Here two USRP boards are connected back-toback through a loop-back cable (SMA-M to SMA-M cable). It uses BasicRx and BasicTx daughterboards.
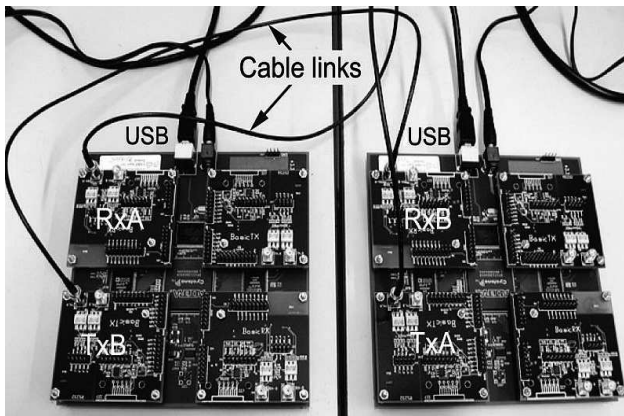

Fig. 3 Loop-back communication setup of USRP system [4]

In the second approach, we used 1200MHz tranceiver daughterboards for wireless communication. The experimental setup is shown in Fig. 4. Both the USRP boards were kept at at least 3 meters of distance to reduce the possible interference caused due to each other.

Fig. 4 Wireless communication setup of USRP system

VI. RESULTS

For loop-back cable scenario, using the setup as shown in Fig. 3, we were able to receive complete JPG file with no errors. It is due to no packet loss and corruptionless environment provided by dedicated wired channel of loop-back cable. Following is the JPG file image (Fig. 5) which we transmitted from one USRP to another in both loop-back and wireless communication scenario.

Fig. 5 Sample image for transmission (13.1KB size)

In the first scenario of loop-back cable communication, received file was error free. The received file is shown in Fig. 6.

Fig. 6 Image received with loop-back cable scenario

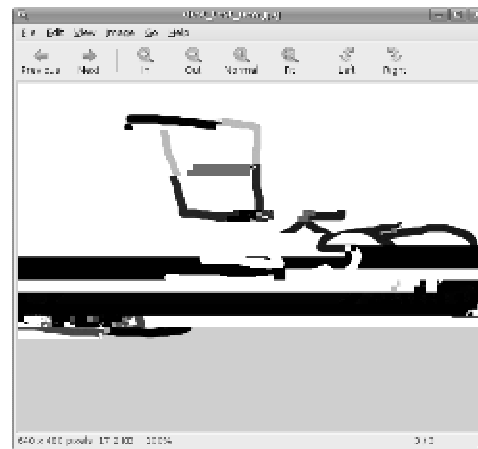In second scenario of wireless file transmission, following was the received file as in Fig. 7.

Fig. 6 Image received with wireless transmission scenario

Here it is apparent that the received file contains some distartion, which can be attributed for lost packets in transmission and reception. Current implementation also needs little more efforts to make it robust against the environmental effects on packet transmission, duplicate packets and possible packet loss. Still this result confirms the successful communication between sender and receiver over wireless channel.

VII. CONCLUSION

As seen from the current implementations of wireless applications on USRP with GNU Radio, it is obvious that USRP in conjunction with GNU Radio is a powerful tool for developing and testing wireless applications. We used some of already tested functions in GNU Radio project for wireless JPEG file transfer. During the course of this project, we got benefited by the knowledge gained and hands-on experience on wireless application development. As demonstrated, we implemented wireless transmission of JPG file. This implementation needs to be made more efficient as the results obtained contain some packet loss affecting the image quality of received JPG file. As future work, we would like to make our implementation more efficient to handle lost and duplicate packets leading to distortion less JPG image at receiving end.

REFERENCES

[1] www.gnuradio.org/trac
[2] www.cs.uni-paderborn.de/en/research-group/research-group-computer-networks/projects/gsr.html
[3] http://www.nd.edu/~jnl/sdr/docs/tutorials/
[4] Ke-Yu, Chen, Zhi-Feng Chen, "GNU Radio," http://www.wu.ece.ufl.edu/projects/softwareRadio/documents/Project%20Report_James%20Chen.pdf
[5] gnuradio.utah.edu/trac/browser/gnuradio/trunk/gnuradio-examples/python/digital/tunnel.py
[6] Gnuradio-3.1.1/guradio-examples/python/digital/readme.txt

[7] DETAILS OF EXPERIMENT

A. Aloha-based approach

As per previous discussion, we chose Aloha-based approach for wireless file transfer application. The source code files were taken from "GNU Radio web page from Paderborn University" and respective modifications were applied to the GNU Radio project in our host machines to be compatible with the source code. With this we tried to set up wireless commuication but could not succeed. We also found some of the problems in this implementaion, which we have listed in "Issues Faced" section of the project report.

This experience motivated us to move to a different approach. We studied the GNU Radio documentation and found "benchmark_tx" and "benchmark_rx" modules in GNU Radio examples to work on wireless channel. These programs confirmed a successful wireless communication. After getting satisfied with this wireless communication, we took these modules as skeleton code for our wireless file transfer implementation.

B. Benchmark_tx & rx based approach

We would like to explain the working of "benchmark_tx" and "benchmark_rx" program modules before explaining details of our approach.
- Benchmark_Tx.py

This transmission module sends the data packets continuously to the connected daughterboard without waiting for any acknoledgement of reception. Transmission frequency, bit rate, packet size and modulation schemes can be defined as command line arguments. This program also accepts file name as argument to send conents of text file over wireless channel.
- Benchmark_Rx.py

This transmission module is always on listening mode. Whenever some data is received through USRP, it is checked for errors at two levels [6]. Firstly, the packet header and PN code are checked and the other where CRC of the payload is verified. Whenver a packet is received, a call back function rx_callback is called and if the CRC check is successfully completed, it displays an output message "true", otherwise "false" is displayed along with the packet number.

We modified both of these programs to enable JPG file transfer. This step was almost like confirming the existing communication functionality and using it to transfer JPG file using loop-back cable between two USRPs.

Once this approach gave us the desired results, we moved to use the same setup for wireless transmission. We found that for wireless transmission, only 1200 MHz and 2400 MHz daughterboards work. After changing our experimental setup to include these daughterboards, we found that the packet loss and packet corruption was higher on wireless channel.

This experience led us to change the functioning of Rx and Tx programs (Tx – Transmitter, Rx – Receiver). We modified these programs as per our following needs.
- Tx should send a packet and waits to receive an "ACK" from receiver.
- Tx starts a timer after sending a packet
- Tx will retransmit a packet when either it receives "NACK" or timer times out.
- When Tx receives any data while waiting for "N/ACK", rx_callback function is called, which is similar to rx_callback in Rx program. If packet is not corrupted, it is checked for "ACK" or "NACK" message. Transmission or same packet or new one is decided on this message.
- Rx should wait for receiving a data packet from Tx. After checking the integrity of received data, an "ACK" mssage is sent to Tx and data is stored in destination file. If packet integrity check is failed, a "NACK" message is sent to Tx.
- In case of duplicate packets arriving at Rx due to loss of "ACK" in the channel, Rx will send the "ACK" message but will not store the data in destination file.