GSOC'19: Module for MAC layer development

Rachuri Sri Pramodh

March 15, 2019

Introduction 1

MAC layer and Protocols 2

Suggested Components 2

Proposed Work during GSOC'19 Period 3

Expected Problems 3

Timeline 4

References 4

I. Introduction

GNU Radio gives an open platform for implementing software radios using signal blocks. Currently, GNU Radio's default blocks allow us to only build Physical Layer. Due to this restriction, not more than two devices can talk in the same wireless channel. If the sublayer Medium Access Control (MAC) is implemented, this restriction can be resolved.

GNU Radio provides a great opportunity for students to have hands-on experience in implementation. However, due to the restriction up to the physical layer, experimentation in MAC layers is not possible. This module extends the experience to more areas in communication and networks and at the same time adds the features of random and controlled access.

II. MAC layer and Protocols

MAC layer is responsible for moving data between devices across a shared channel and its protocols make sure that signals of different stations don't collide with each other. Popular and majorly used MAC protocols as Aloha, Slotted Aloha, CDMA, CSMA/CA, CSMA/CD. When Aloha, Slotted Aloha don't give good throughput, CSMA/CD is not possible in Wireless as collision detection can't be done.

III. Suggested Components

- **Block for Packet Buffer:** In general, the packets are not served as they arrive. For storing the packets till the time they can be served, a queue type of buffer must be created. Also, the buffer must pop packet(s) only upon a signal from the MAC controller.
- **Block for Packet Receiving:** A module that keeps listening to the channel, attempts to demodulate the packet if any and outputs only the packets that were sent to its node.
- **Method for Carrier Sensing:** To implement protocols like CSMA/CA, an API has to be made that checks if the channel is busy and reports the same to the MAC controller. Carrier Sensing has to be done independently with the Packet Receiver as the absence of any packet doesn't mean that the channel is free.
- **MAC Controller Block:** Controllers that takes inputs from respective blocks and controls the Packet Buffer. It should include provisions like addressing, error control etc.

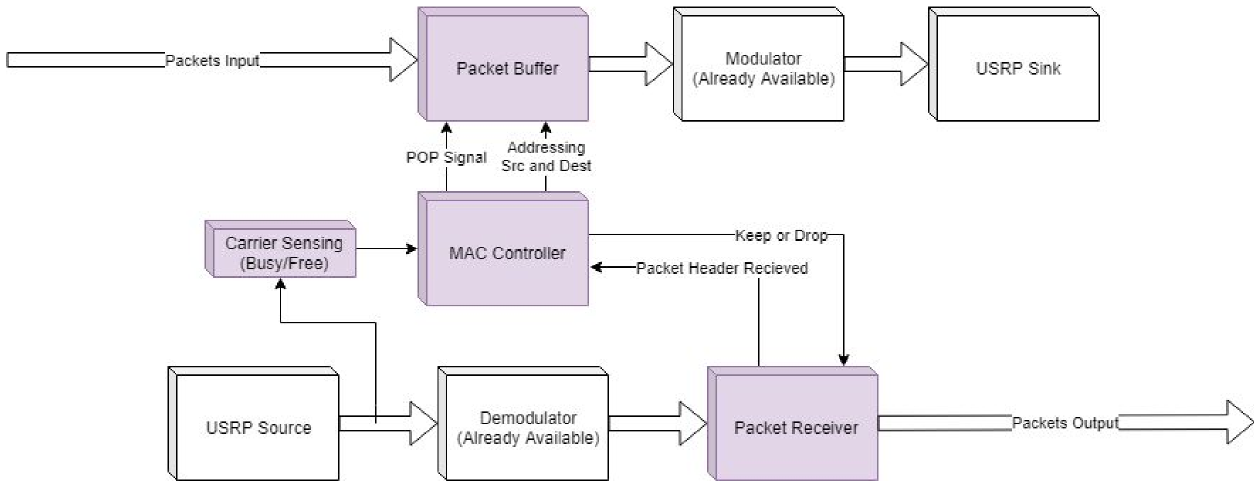

Fig1: An illustration of expected usage of proposed blocks.
Proposed blocks are in purple.

IV. Proposed Work during GSOC'19 Period

- The above-mentioned blocks which are the bare minimum for MAC implementation shall be implemented
- **MAC Controller Blocks:** MAC control blocks for each of the following shall be developed for the following protocols.
    - Simple Aloha
    - Reservation Method
    - CSMA/CA
    - Token Passing Method

Apart from building the blocks suggested above, I am also interested in building the following blocks.

- **Custom MAC Controller Block:** Along with blocks for different MAC protocols, a custom block has to made so that a user can build custom protocols.
- **Throughput Calculator Block:** This block will be useful in understanding the performance of MAC protocols and accessing the capability of a custom made protocol.

All the blocks shall be tested for USRP B200 since I have access to around 10 of them.

V. Expected Problems

- **Timing uncertainty:** As mentioned by Mr Marcus Müller the delay between packet arrivals and calculation of serving time might be pretty random, and also pretty large. The randomness in the delay would be dependent on the machine hosting GNU Radio.
  The delay would surely affect the overall throughput. But however, the queue architecture will make sure that no packet is lost due to the randomness and every packet would be served.
- **Overlap of serving times:** Any overlap of serving times between multiple nodes will result in collision and packet drops. This is a very common problem in random access.

The architecture doesn’t include a method for acknowledgements and so taking a decision to perform retransmission is not possible. This method to send retransmission called ARQ. There are multiple ways it can be implemented but it can be taken care of by higher layers like in TCP (or the main packet source and sink here).
However, I can also include this feature if time permits.

- **Slot time calculation:** Slot time size and margin time size is fixed for every node but to keep themselves in sync, a reference signal is required. For this, the MAC controller will have to listen to the channel for other sender and sync itself.
When a new node arrives into the vicinity of the network, it might timeout before a node sends something first. To resolve this issue, one or more nodes will have to send a garbage packet after a max channel free time.

VI. Timeline

My timeline for the proposal is not yet made. The proposed work shall be divided into 13 weeks (according to GSOC timeline, May 27 to Aug 26 are the dates including evaluations, documentation and cleanup). The timeline will be consisting of dates for implementation, testing, debugging and enhancement.

VII. References

- A Split Architecture for Random Access MAC for SDR Platforms
https://ieeexplore.ieee.org/document/6636826/
- GWN : A framework for packet radio and medium access control in GNU radio
https://iie.fing.edu.uy/publicaciones/2017/GBLRRG17/
- A GNU Radio Testbed for Distributed Polling Service-based Medium Access Control
https://ieeexplore.ieee.org/document/6127723
- Multiple Access Protocols: Performance and Analysis (Telecommunication Networks and Computer Systems) by Raphael Rom
https://www.amazon.com/Multiple-Access-Protocols-Performance-Telecommunication/dp/B01FIZMDUG

My Curriculum Vitae (CV) and Contact details can be found at the following link.
https://github.com/pramodhrachuri/My-Potatos/raw/master/CV_Pramodh.pdf

The latest iteration of this Proposal can be found at the following link.
https://github.com/pramodhrachuri/GSOC_Proposal/raw/master/GSOC'19_Proposal.pdf