Beyond 5G: Investigating current and future wireless communication technologies

Daanish Suhail, Kshitij Shah, Troy Madden, Jack Willard, Pryce Tharpe

Faculty Advisors: Profs. Chih-Chun Wang, David Love, and James Krogmeier

Introduction

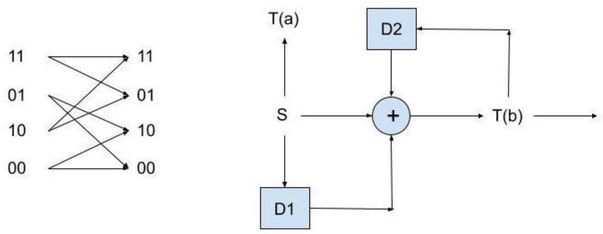
The Beyond5G team specialized in investigating new wireless technologies. The team was made of two primary section. New members were introduced to the team by completing a short series of lessons covering digital communication and then worked to make progress in two projects, performing symbol synchronization and detecting HD radio (NRSC-5) using a USRP device.

Returning members of the team focused in the area of further developments of error correcting codes. Specifically, members of this subteam worked on writing effective implementations for both the Viterbi and BCJR decoding methods for convolutional codes.

Error Correcting Codes

Within the study of error-correcting codes, this subteam had a group of areas to focus on. Namely, the development and implementation of a basic Viterbi decoder and the implementation of a BCJR decoder for convolutional codes, and a first foray into understanding the turbo coding methods.

Within the previous semesters, the subteam was introduced to the convolutional codes which makes use of a circuit and a trellis in order to encode bits in a way which makes continuous use of a source stream and an output stream.



The Viterbi algorithm allows us to have a basic method for traversing along a trellis and successfully decoding a codeword which may have encoded bits in the form of a continuous stream.

The algorithm works by utilizing probabilities. In previous studies, we learned about how we can decode codewords using codeword distances. This method relies on considering the codeword with the maximum probabilities. Notably, each group of bits is considered as its own probabilistic event and the sequence of bits which are most likely (high probability) are chosen.

The BCJR decoder differs in that it utilizes a series of passes in order to achieve the same effect of decoding the final codeword. Specifically, the algorithm utilizes two passes along the trellis. Each pass must then move along either forward or backwards along the trellis. With each pass, we compute the necessary possible probabilities and on the reverse, we then apply a similar process to obtain the requisite eliminations for obtaining the final result. Currently the subteam is still working on developing a working implementation for this.

Lastly, the subteam has began study into the study of factor graphs which act as a greater structure for which previously study decoders and codes can exist. These bipartite graphs describe some of the underlying methods. Understanding the application for these graphs is where we hope to further our studies in the future.

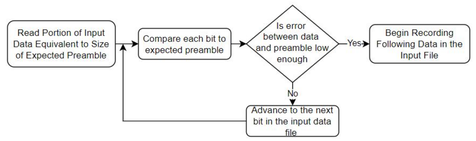
Symbol Synchronization

Symbol synchronization plays a key role in the exchange of data over computer networks in modern society. Preambles are the primary component in this process. Not only do these strings typically containing hundreds of bits notify the machine that data is about to be sent, but can also contain important information about the data as well. This works to ensure data the machine does not interpret noise as data and prevents reduces data loss in processing.

It can be seen how correct recognition of a preamble is important for network communication despite the presence of noise caused by a variety of external factors.

The team developed a goal to increase understanding of symbol synchronization as well as implement a program of its own to implement this. The team first completed background research into matched filters and autocorrelation functions to understand how they work and are implemented.

Using this information the team developed a model in Python to simulate preamble detection as it would theoretically be implemented at the hardware description language level. This is done so by iterating through a test file composed of random bits and a preamble placed somewhere within it. Then a series of NumPy commands are followed to evaluate the accuracy of the substring of bit being evaluated compared to the reference preamble and begin recording the string of data that follows.



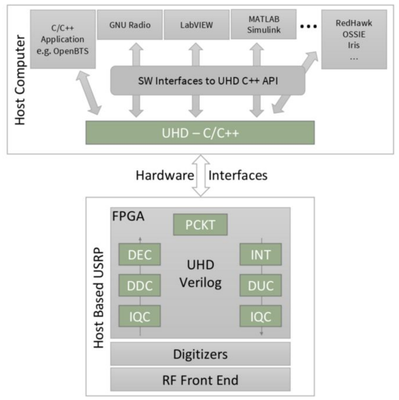
The team developed a threshold for accuracy based on test cases developed with bitstreams of varying lengths and amounts of noise to optimize the program to its fullest extent.

Using NumPy arrays for evaluating accuracy-

- Two NumPy arrays are created as a bitstream is evaluated: One that holds the preamble, another that holds the current stream of bits that is being compared to the preamble (same size).
- To calculate the accuracy of the currently evaluated index in the bitstream, the arrays are subtracted from each other. With this, elements that evaluate to 0 indicate accuracy between the preamble and the compared bitstream, whereas elements that evaluate to -1 or 1 indicate inaccuracy.
- The resulting array is summed using np.sum and the result is divided by the preamble size, giving us the percentage of incorrect bits compared to the preamble. If this score is beneath a calculated inaccuracy threshold, we can determine the location of the preamble and return the bitstream that follows.

USRP

A Universal Software Radio Peripheral (USRP) is a software-defined radio (SDR) that allows for modulating and demodulating radio signals. To interface with the USRP, we installed Linux on a virtual machine to run the USRP Hardware Driver (UHD) and GNU Radio. The UHD allows us to connect with the USRP using C/C++, while GNU radio is an API companion for the UHD that we can use to interface with the device in Python.



UHD components. Retrieved from Ettus Research.

Our goals with the USRP was to set up the desired software to communicate with the device, use it to demodulate FM and AM radio signals, and then detect HD Radio. We were able to achieve communication with the USRP and are currently moving forward with the process of receiving radio signals. Our progress benefited from our efforts by developing an understanding of USRPs, their accompanying software, and how we can efficiently move forward with our research. For the future, we are using our present knowledge as a basis for us to detect and read radio signals, and then go beyond.

Hardware: USRP N210
Software: Virtualbox, Ubuntu 22.04, UHD v4.5.0.0, GNU 3.10.1.1

```
-- UHD Device 0
--------------------------------------------------
Device Address:
    serial: F2A017
    addr: 192.168.10.4
    name:
    type: usrp2
```

For NRSC-5, the team identified an open-source repository created by "Theori" that will allow us to detect and decode HD radio with our SDR.
Source: https://github.com/theori-io/nrsc5

Conclusion

As evidenced by the progress made in our projects, the Beyond5G team has been successful in making further mastery in understanding of critical communications technologies. These technologies and methods are similar to that of what is being currently used in the 5G-6G standards and we hope to develop these ideas further.

As the semester comes to a close, the symbol synchronization and USRP subteam plans to deliver a working Python script using matched filters to accurately identify a preamble from a bitstream and use the USRP to detect NRSC-5 HD radio.

Similarly, the error-correcting codes team hopes to further develop implementations of BCJR, study the turbo code, as well as their potential successors which still have room for development, especially from an implementation perspective.