GNU Radio Project Proposal: Forward Error Correction

Kayla Comer
kmcomer2002@gmail.com
`https://github.com/kaylacomer`
`https://www.linkedin.com/in/kayla-comer/`

Google Summer of Code
March 23, 2024

1 Introduction

As data is transmitted across a channel, the signal is accompanied by background noise that may lead to corrupted data at the receiver. Forward error correction (FEC) mitigates data loss by detecting and correcting bit errors when possible.

In GNU Radio, the gr-fec framework implements FEC through pairs of encoder/decoder deployment blocks. gr-fec contains various blocks to support streaming and asynchonous deployments with options for tagging data (including timing and state information), handling puncturing and depuncturing, and converting data types [1, 2]. Deployments are compatible with a few FEC techniques, including convolutional code (CC), consultative committee for space data systems (CCSDS), low density parity check (LDPC), turbo product encoder (TPC), polar, and repetition. However, some FEC open-source implementations include methods that perform better in some applications.

The AFF3CT toolbox contains several additional FEC schemes, including Bose, Ray-Chaudhuri, and Hocquenghem codes (BCH); Reed-Solomon codes (RS); Turbo Parallel; and Repeat and Accumulate (RA) [3]. AFF3CT can be used as a library for software-defined radios (SDRs). AFF3CT uses an open-source MIT license, so its libraries can be directly used (with due acknowledgement) within GNU Radio.

This project addresses FEC implementation in GNU Radio by incorporating additional methods and evaluating existing methods.

First, I will use AFF3CT to add BCH, RS, Turbo Parallel, and RA error correction techniques to gr-fec to improve GNU Radio's performance for certain applications. AFF3CT's codecs are written in C++. For each technique, I will create GNU Radio encoder and decoder blocks in C++.

Next, I will compare the current implementation of polar, LDPC, CC, TPC, and repetition codecs in gr-fec to their counterparts in AFF3CT. gr-fec's current encoders and decoders are written in Python, while AFF3CT's library is in C++. I will create new blocks in GNU Radio using AFF3CT's library.

AFF3CT has a simulation tool to generate BER/FER curves, and gr-fec contains a BER curve generator. AFF3CT also has a tool to plot and compare BER/FER data uploaded as text files. I will use both tools with both implementations to evaluate codecs' performance.

2 Deliverables

Deliverable 1: Add blocks (written in C++) to gr-fec for Turbo Parallel, BCH, RS, and RA using AFF3CT library

Deliverable 2: Add AFF3CT-library-based blocks (C++) for polar, LDPC, TPC, and repetition

Deliverable 3: Simulate new blocks' performance with GNU Radio flowgraphs and BER curve generator

Deliverable 4: Simulate error correction implementations in AFF3CT. Use the BER/FER comparator tool

3 Disclosures

I will use the GNU General Public License Version 3 (GPLv3). I acknowledge and understand GNU Radio's three strikes rule. Cyberspectrum is the best spectrum.

4 Schedule

My project is 350 hours, and I will spend 12 weeks working on the project for 30 hours a week. I plan to take off a week around U.S. Independence Day.

- **May 1 - 26** – Community bonding period. Finish GNU Radio tutorials; read about forward error correction applications. Identify each method's strongest application
- **May 27** – Coding period begins
- **May 27 - 31** – Add Turbo Parallel block in gr-fec
- **June 3 - 7** – Test Turbo Parallel block. Write QA codes (or use existing codes)
- **June 10 - 21** – Add and test BCH, RS, and RA blocks

- **June 24 - 28** – Document new blocks. More testing / bug fixes
- **July 1 - 5** – Personal vacation period
- **July 8 - 12** – Simulate new blocks in GNU Radio with BER curve generator
- **July 12** – Midterm evaluation due
- **July 15 - August 2** – Add, test, and document AFF3CT-based polar, LDPC, TPC, and repetition blocks
- **August 5 - 9** – Simulate new and existing blocks in GNU Radio with BER curve generator
- **August 12 - 20** – Simulate FEC implementations in AFF3CT. Compare using BER/FER comparator tool
- **August 21 - 23** – Wrap up loose ends. Document GNU Radio and AFF3CT simulations and results. Submit final work product and mentor evaluation.
- **August 26** – Contributor's final evaluation deadline

5 Personal background

I completed my BS in Electrical Engineering from Purdue University in December 2023, and I plan to start my PhD program in the field of networks and communications in the fall. I am originally from the U.S., and I will live with my parents in the state of Indiana over the summer. At Purdue, I had a communications lab that used GNU Radio and an SDR. We covered some fundamental analog modulation schemes (FM, AM, SSB) and briefly studied digital modulation techniques (BPSK, QPSK, PAM).

During spring 2024, I am an intern at Argonne National Lab, contributing to a project involving vehicle-to-infrastructure communications for connected/automated vehicle applications.

I have not participated in an open-source project before, but I am excited to contribute. I am comfortable coding in C/C++ and Python, and I am familiar with Git. I mainly use MacOS but have access to a Linux machine. I have also used Doxygen.

References

[1] *GNU Radio Manual and C++ API Reference: Forward Error Correction.* URL: `https://www.gnuradio.org/doc/doxygen/page_fec.html`.

[2] Tracie R. Perez and Thomas Rondeau. "Open-source Forward Error Correction using GNU Radio". In: *AIAA SPACE 2015 Conference and Exposition*. DOI: `10.2514/6.2015-4655`. eprint: `https://arc.aiaa.org/doi/pdf/10.2514/6.2015-4655`. URL: `https://arc.aiaa.org/doi/abs/10.2514/6.2015-4655`.

[3] A. Cassagne et al. "AFF3CT: A Fast Forward Error Correction Toolbox!" In: *Elsevier SoftwareX* 10 (Oct. 2019), p. 100345. ISSN: 2352-7110. DOI: `https://doi.org/10.1016/j.softx.2019.100345`. URL: `http://www.sciencedirect.com/science/article/pii/S2352711019300457`.