Incorporation of Rate ½ Convolutional Encoder-Decoder into the GNU Radio Benchmark Script

Anjan N. Rayamajhi

Advisor: Dr. Harlan B. Russell

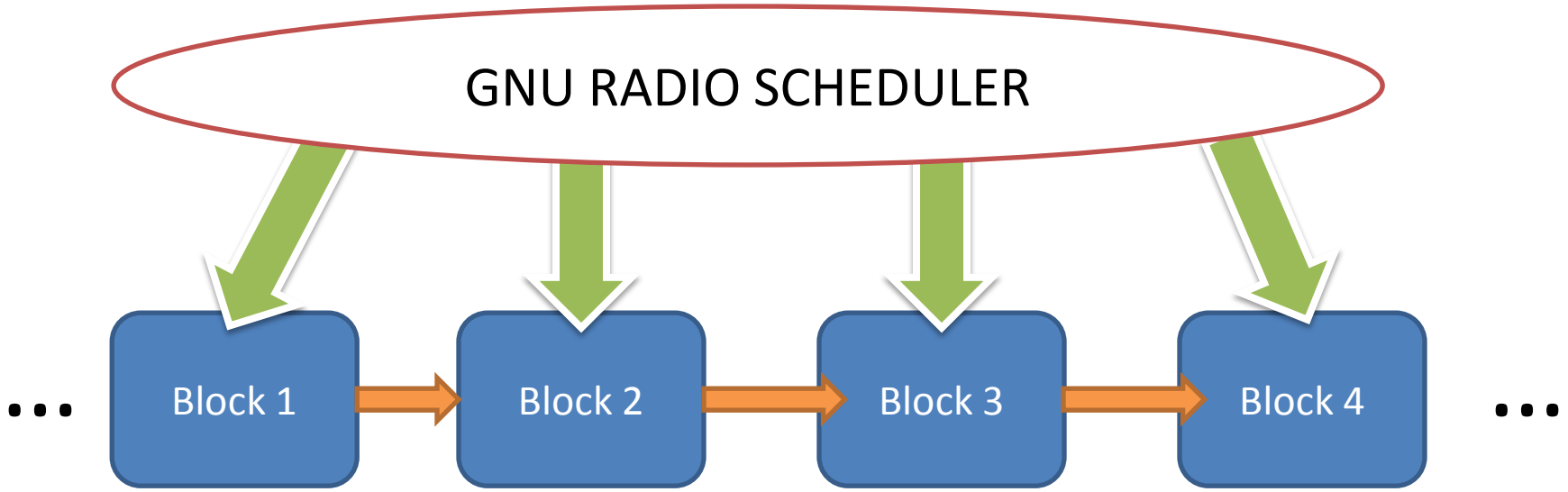
Clemson University

Introduction

- Original benchmark script uses stream based architecture of GNU Radio and maintains continuous flow of packets from Transmitter to Receiver without using error correction codes
- Our goal is to exchange packets between two nodes adding encoding and decoding techniques and using stream tags for defining packet boundaries
- Long term goal is to develop a testbed for Ad-hoc Networks

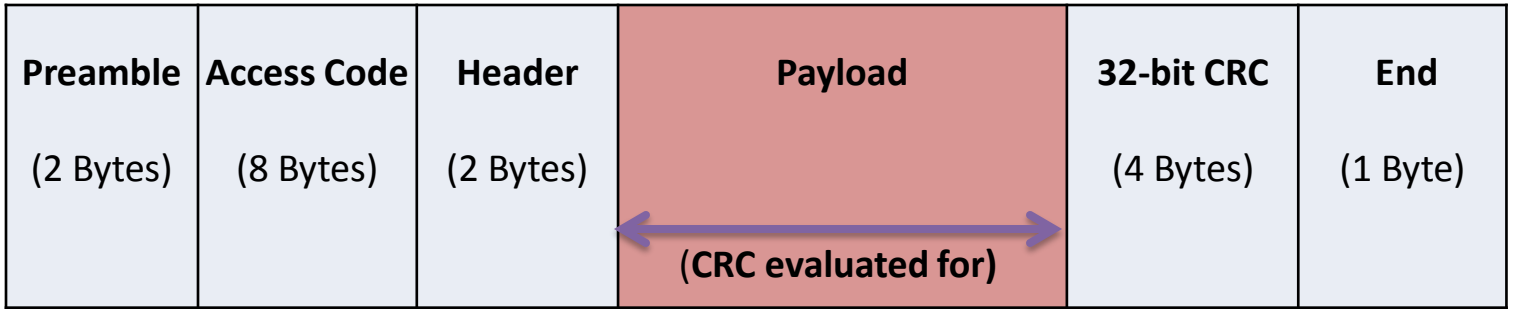
GNU Radio Scheduler

- GNU Radio Scheduler uses Thread per Block processes
- Each block executed once certain number of input items available as the output from preceding blocks



Original Benchmark Script

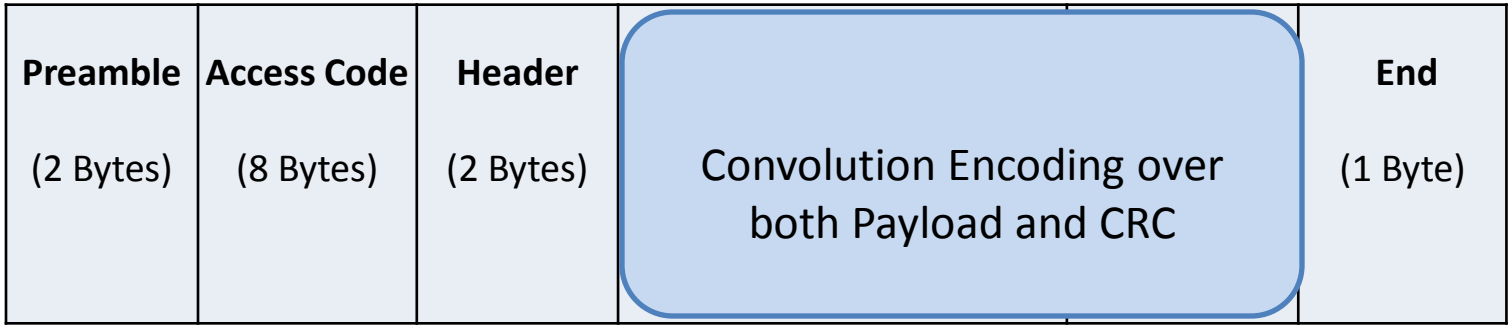
- Benchmark script : example script available in *gnuradio_source_dir/gr-digital/examples/narrowband*
- Packet based data communication but packet boundaries defined using python string



Packet format used in original Benchmark Script

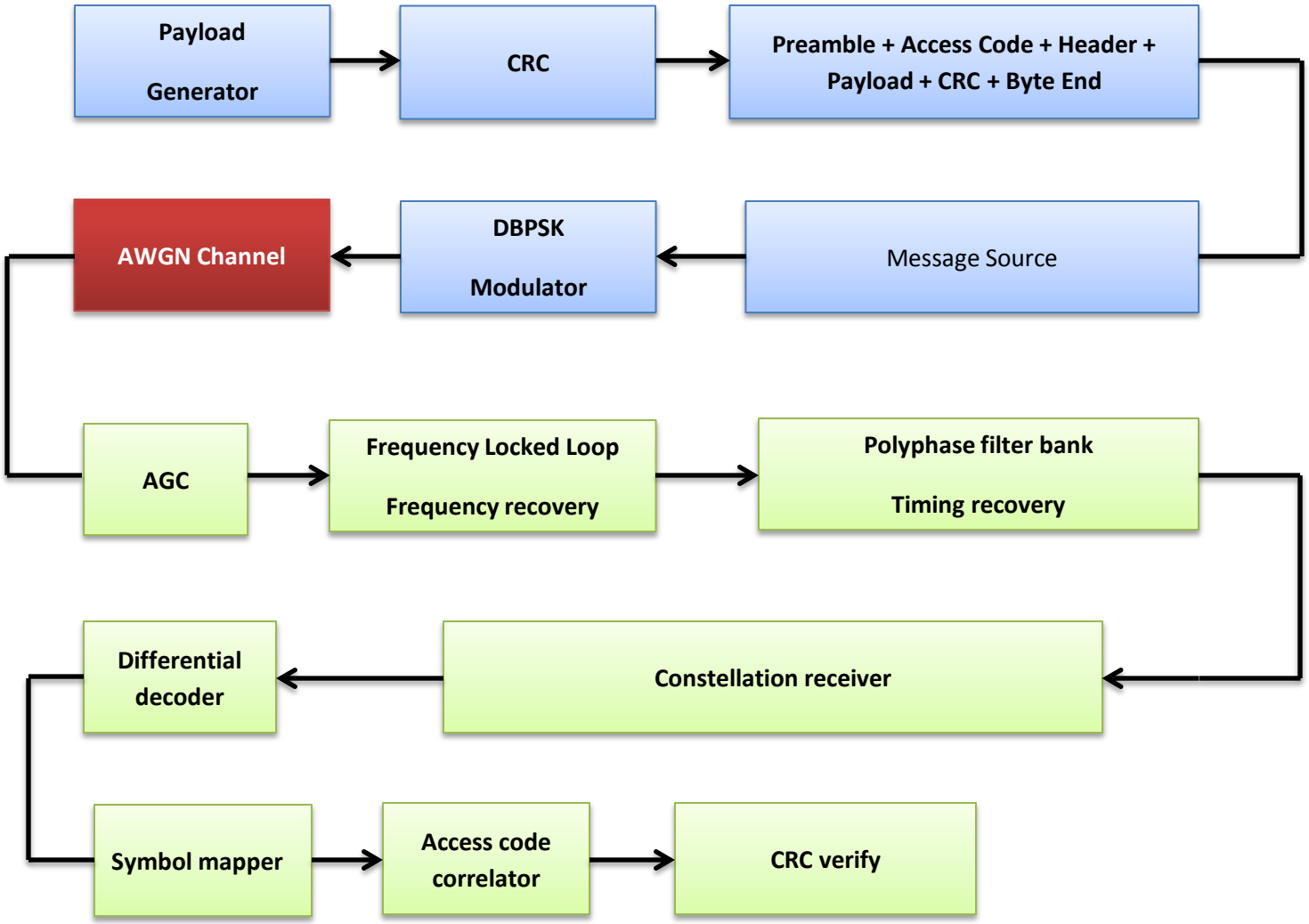
Modified System

- Need for Error Correction Coding of data unavoidable in wireless communication
- rate ½ Consultative Committee for Space Data Systems (CCSDS) [171,133] convolution encoder and Viterbi Decoder



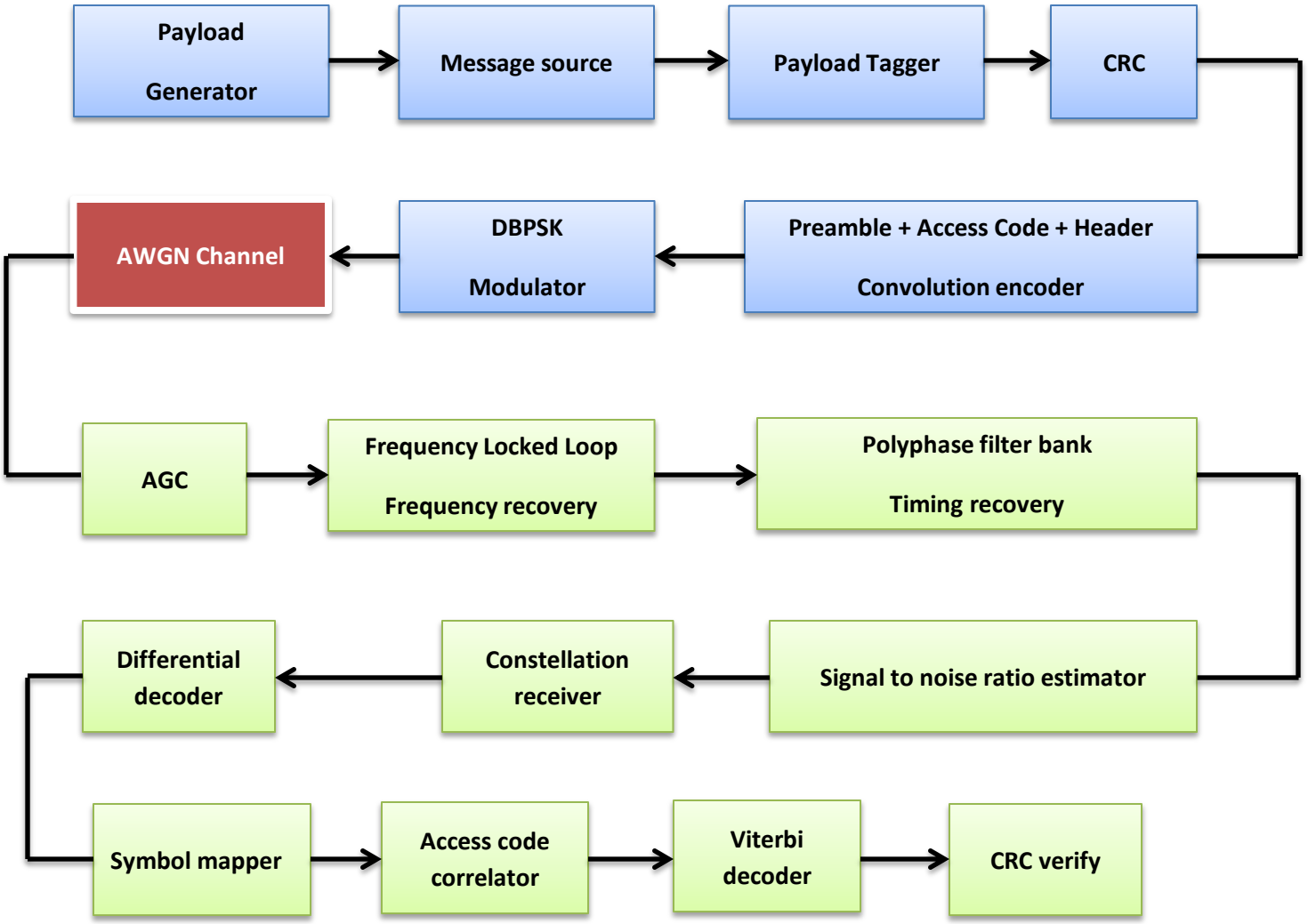
Packet format in the modified system

From Original to Modified Benchmark Script



Benchmark Script Block Diagram

From Original to Modified Benchmark Script

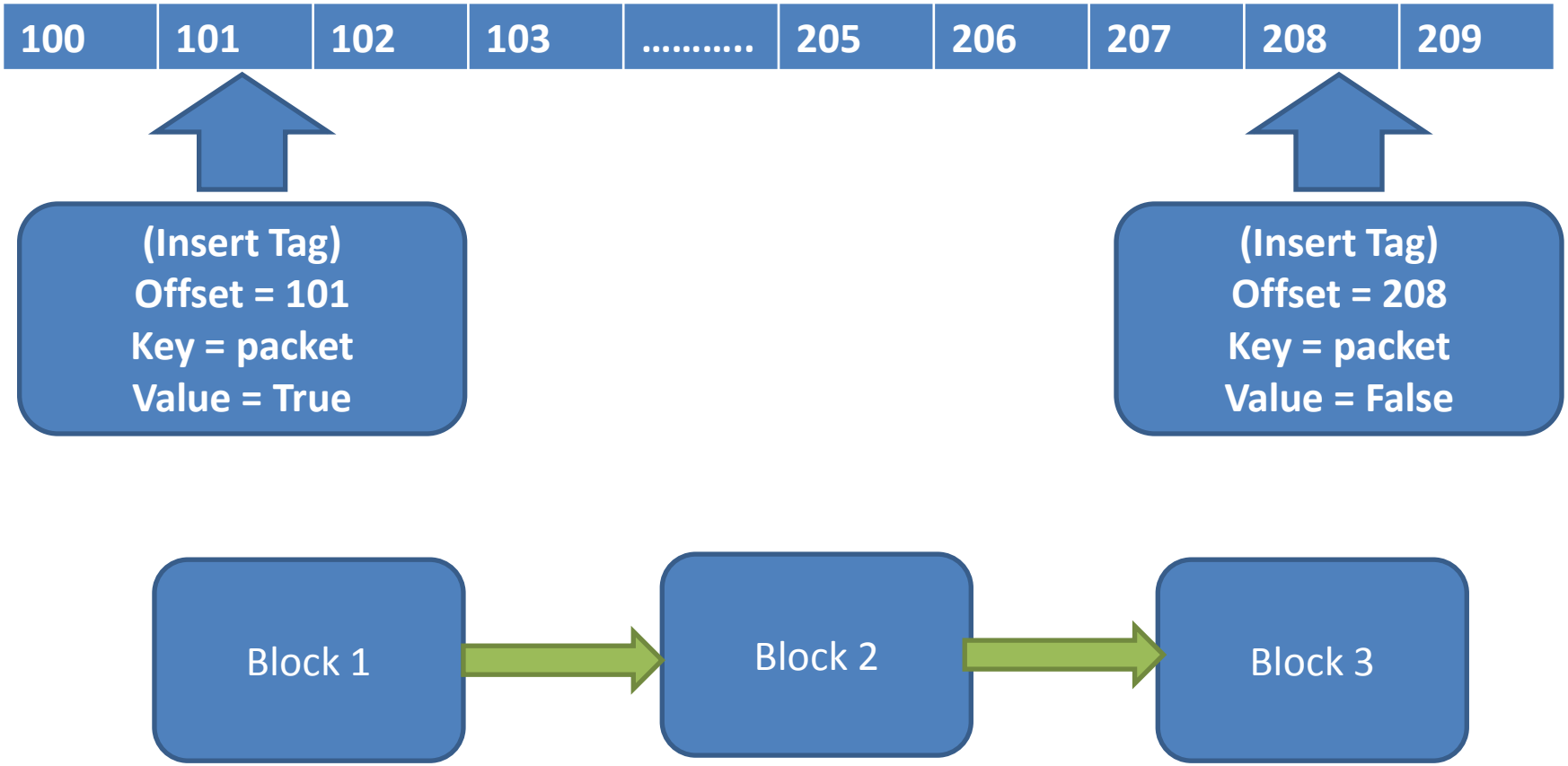


Modified System Block Diagram

Encoded System Design

- Python strings and python's convolve module for encoding appeared relatively slow for our need
- C++ based encoding technique was necessary
- Stream tags used for determining packet boundaries
- Stream tags are polymorphic data types that can be attached to specific item in a stream of data

Stream Tags

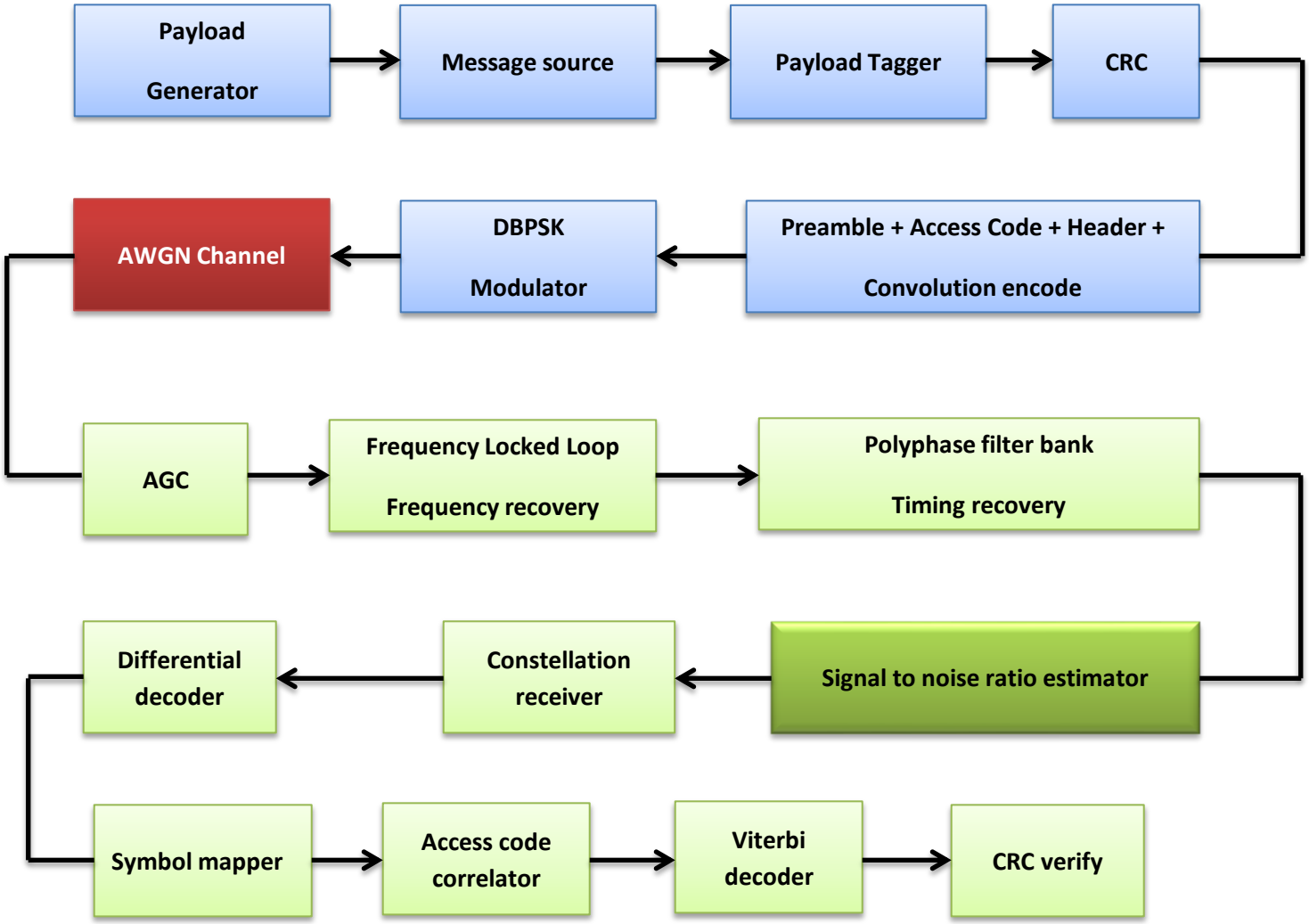


Tags added to Block 1 can be read from any downstream blocks (depends on Tag Propagation Policy though)

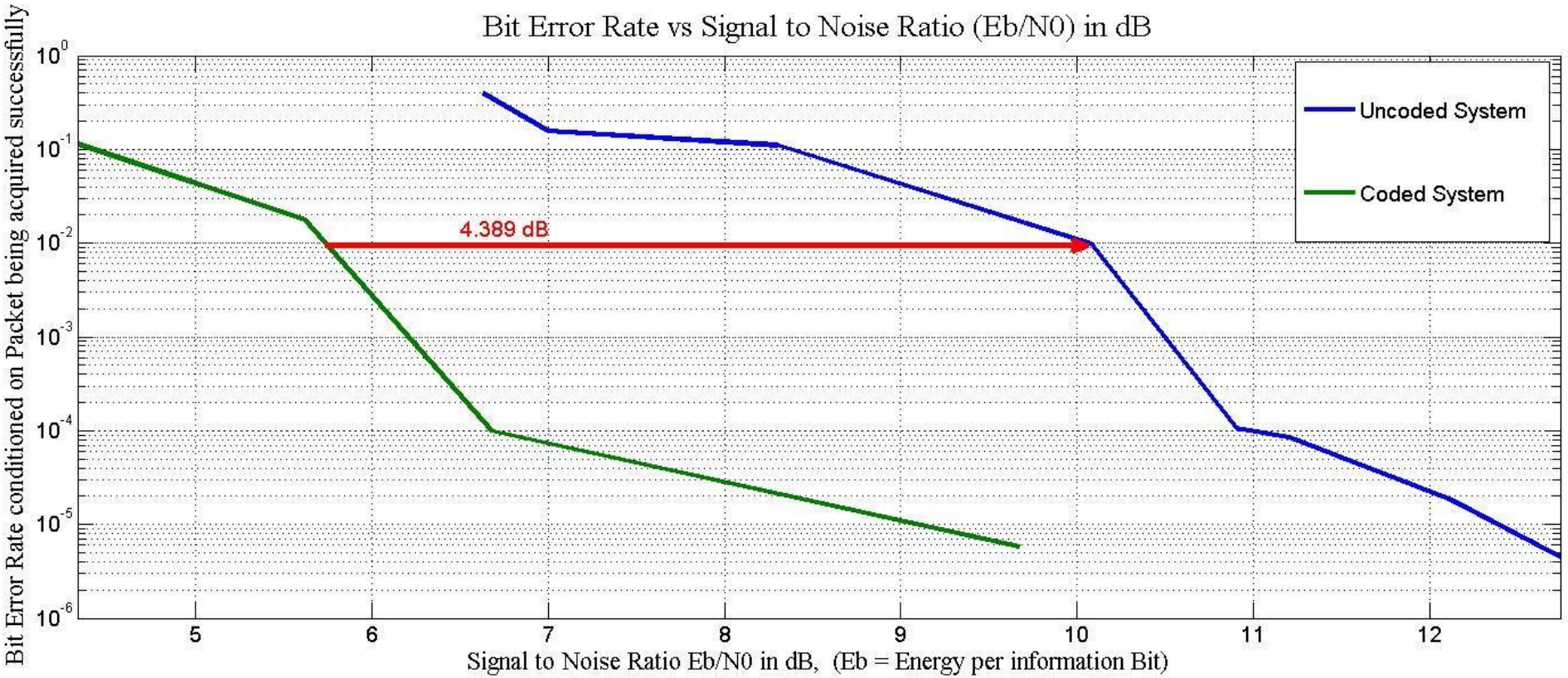
Results

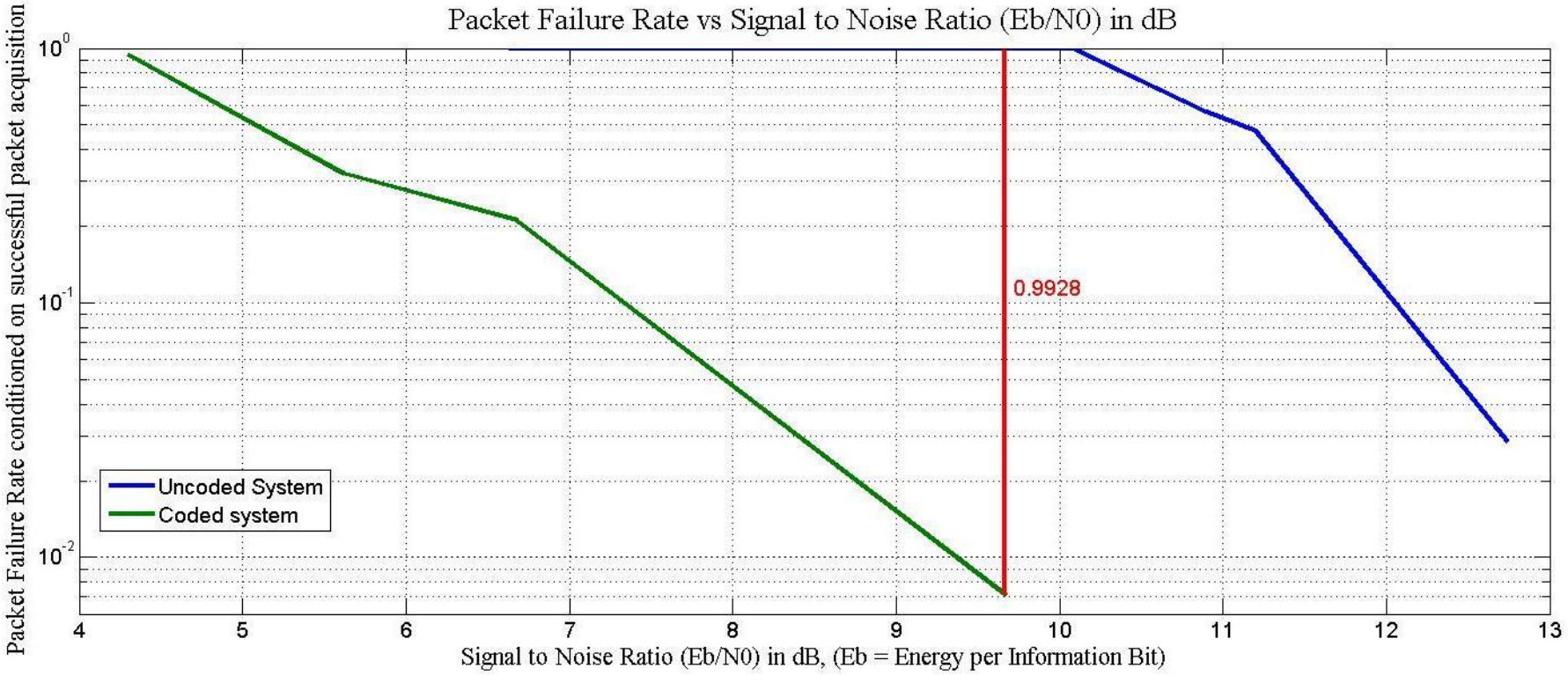
- Simulated with an AWGN Channel for uncoded and coded system
- Performance Measures:
  - Bit Error Rate conditioned on successful packet acquisition
  - Packet Failure Rate conditioned on successful packet acquisition
  - Packet Acquisition Rate

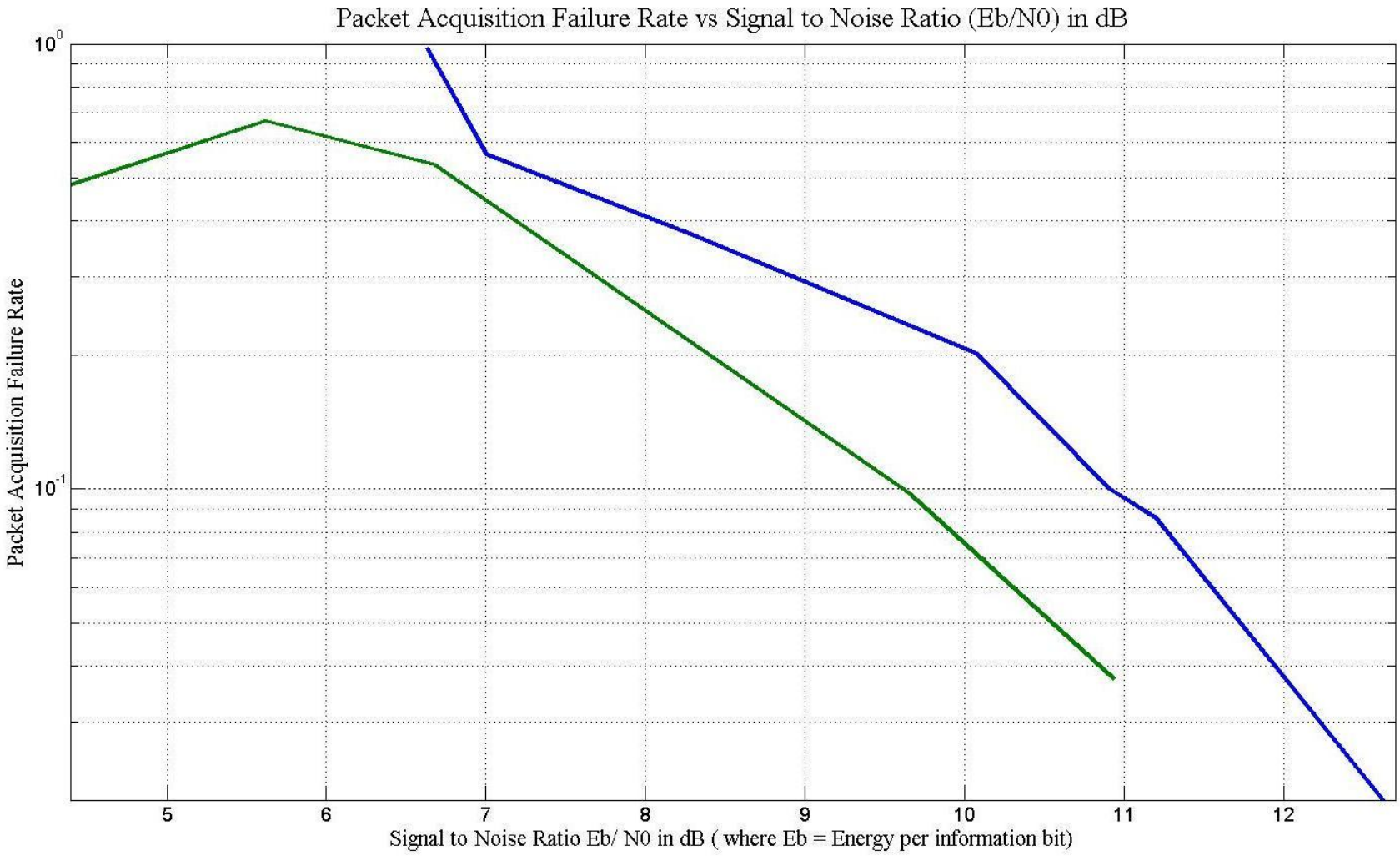
SNR Estimator



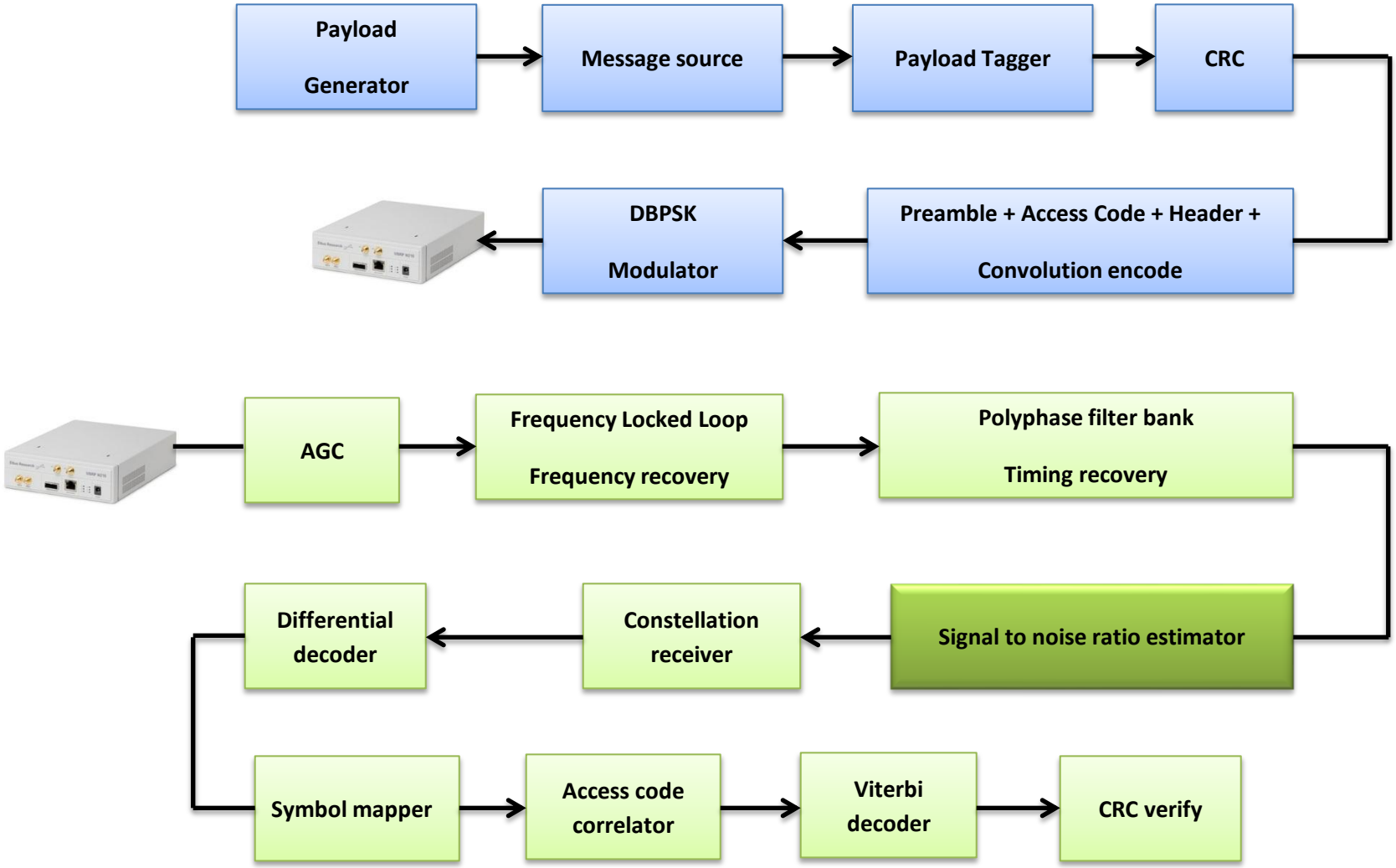
M2 M4 based Signal to Noise Ratio Estimator from “A Comparison of SNR Estimation Techniques for the AWGN Channel” -- David R. Pauluzzi and Norman C. Beaulieu


Bit Error Rate vs Signal to Noise Ratio (Eb/N0) in dB
Bit Error Rate conditioned on Packet being acquired successfully
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
4.389 dB
Uncoded System
Coded System
5
6
7
8
9
10
11
12
Signal to Noise Ratio Eb/N0 in dB, (Eb = Energy per information Bit)


Packet Failure Rate vs Signal to Noise Ratio (Eb/N0) in dB
Packet Failure Rate conditioned on successful packet acquisition
10^0
10^{-1}
10^{-2}
0.9928
Uncoded System
Coded system
4
5
6
7
8
9
10
11
12
13
Signal to Noise Ratio (Eb/N0) in dB, (Eb = Energy per Information Bit)

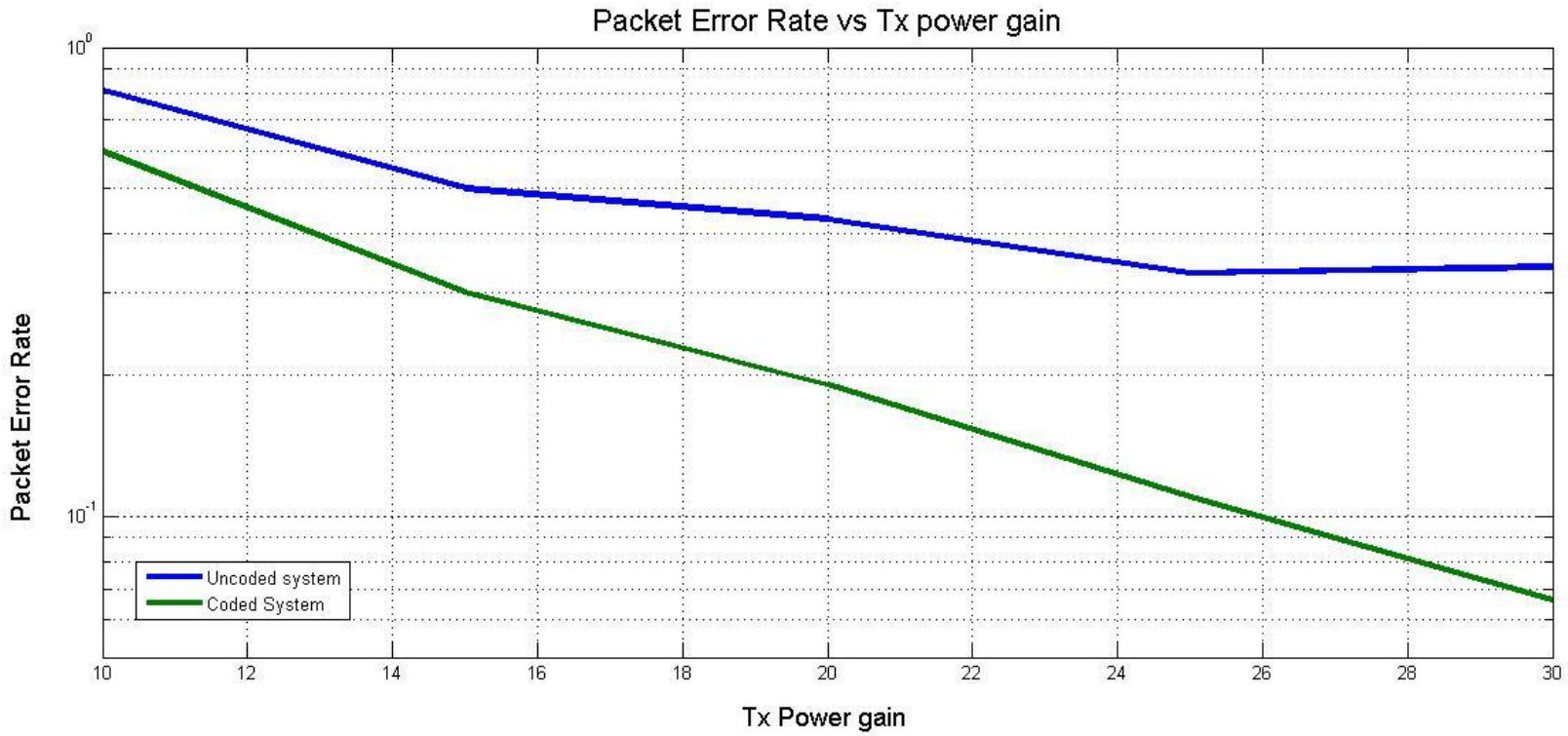

Packet Acquisition Failure Rate vs Signal to Noise Ratio (Eb/N0) in dB
Packet Acquisition Failure Rate
10^0
10^{-1}
5
6
7
8
9
10
11
12
Signal to Noise Ratio Eb/ N0 in dB (where Eb = Energy per information bit)

Hardware test



M2 M4 based Signal to Noise Ratio Estimator from “A Comparison of SNR Estimation Techniques for the AWGN Channel” -- David R. Pauluzzi and Norman C. Beaulieu

PER


Packet Error Rate vs Tx power gain
Packet Error Rate
Tx Power gain
10^0
10^{-1}
10
12
14
16
18
20
22
24
26
28
30
Uncoded system
Coded System

Conclusion

- Good exposure towards programming using GNU Radio
- Understanding some of the physical layer blocks and implementing them using C++
- Improvement in the performance of benchmark script
- A step towards developing a testbed for MAC layer testing using GNU Radio

Thank you !!