Power-Aware Rateless Codes in Mobile Wireless Communication

Calum Harrison
University College London
c.harrison@cs.ucl.ac.uk

Kyle Jamieson
University College London
k.jamieson@cs.ucl.ac.uk

ABSTRACT

Rateless error correction codes hold great potential for increasing the capacity of practical wireless networks by obviating the need for transmitters to estimate the highest reliable rate of an unpredictable wireless channel and send information at that rate. But the accumulation and intensive processing of noisy bits works against rateless codes' adoption in mobile devices, where energy is at a premium due to limited battery capacity. In this work, we identify a new tradeoff between energy efficiency and wireless capacity that rateless codes can make in low signal-to-noise ratio or highly variable "grey zone" conditions. We propose *Power-Aware Rateless Codes* (PRC), a design that integrates with the medium access control portion of a rateless wireless system, giving the system a way of selectively sacrificing small amounts of wireless capacity for large savings in decoder computation effort, thus reducing radio power consumption in challenging radio environments.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless*; D.2.8 [**Software Engineering**]: Metrics—*Wireless communication*

General Terms

Algorithms, Design, Performance

Keywords

Wireless, Rateless, IRA, Capacity, Power Aware

1. INTRODUCTION

Rateless codes for noisy channels [8, 9, 18] hold great potential for increasing the capacity of practical wireless networks by obviating the need for transmitters to adapt to the unpredictable wireless channel, estimate the highest reliable rate, and send information at that rate. Instead, the transmitter simply sends at a very high rate, while the receiver accumulates received information in a graph [8], trellis [18], or matrix [9] structure, periodically making decoding attempts and notifying the transmitter when successful. At moments when the channel is poor, the error-correcting capabilities of the rateless code compensate for the high transmission rate, fixing bit errors. At moments when the channel is strong, the high transmission rate implies a high communication rate.

Such rateless codes thus approach the Shannon rate of a point-to-point channel, even when the wireless channel is quickly and randomly varying in quality, is very poor, or both. Furthermore, rateless codes seem a good choice for an energy efficient decoder because compared to fixed-rate codes, they are able to achieve a higher rate at similar SNRs and do not require a full retransmission on decoding failure. This greatly reduces the time that transmitter and receiver radios are active, potentially saving energy.

But the required accumulation and intensive processing of information works against the energy efficiency of rateless codes, especially in mobile devices where energy is at a premium due to limited battery capacity. In fact, in order to approach Shannon capacity, existing approaches require the decoder to work increasingly harder as signal to noise ratio (SNR) decreases, consuming more energy on the margin for the same relative capacity (bits per channel use) gains.

Consider a mains-powered access point transmitting ratelessly-coded information on the downlink to a battery powered mobile. Before the access point's first transmission, it estimates how much information to send to the mobile, and sends slightly more than that estimate. When the channel is stationary, the estimate will be accurate, but in the presence of mobility, bursty interference, or environment-induced channel fading, it may be well off. Once it has received the first transmission, the mobile decodes, expending considerable computational resources. If the decoding attempt fails, the mobile negatively acknowledges the transmission, collects information from a second transmission, and makes another decoding attempt, wasting energy.

We propose *Power-Aware Rateless Codes* (PRC), rateless codes whose decoder integrates with medium access con-

Hotnets '12, October 29–30, 2012, Seattle, WA, USA.

trol, giving the system a way of selectively sacrificing small amounts of wireless capacity for large savings in decoder computation effort. Leveraging the fact that decoders for graph-based codes break processing up into many small iterations, PRC works as follows. After the very first few iterations of decoding (*i.e.*, before expending significant computational resources on decoding the entire packet successfully) the PRC decoder uses internal decoder metrics to estimate the likelihood of the entire decoding attempt being successful, with a modest amount of processing effort. Then, it makes a decision about whether to proceed based on:

- Whether the receiver is battery- or mains-powered.
- If battery-powered, the desired wireless performance level, determined by the scarcity of wireless capacity at that particular moment (or lack thereof).
- If battery-powered, the amount of energy remaining and the user's preference for conserving battery energy.

If the PRC receiver decides not to proceed with the decoding attempt, it aborts decoding for that transmission, thus conserving power. The receiver retains all the received bits, and requests another transmission from the sender.

The contributions of this paper are:

1. The identification of a new tradeoff between energy efficiency and wireless capacity that graph-based codes can make in low or highly variable SNR regimes.
2. A novel graph-based code that offers higher ergodic communication rates over Spinal Codes [18] and Strider [9], while simultaneously reducing power consumption.

With simulation-based experiments, we show that it is possible to construct a high capacity graph-based rateless code that is simultaneously able to quickly abort its decoding effort based on early estimates of the likelihood of decoding success. Evaluated as a whole, PRC (including MAC protocol) slightly increases the fraction of Shannon capacity achievable in the 21–25 dB SNR range while simultaneously requiring between one and two orders of magnitude fewer instructions than Spinal codes, and remaining highly competitive in terms of capacity outside that SNR range.

In the next section, we summarize related work in rateless codes from a power-efficiency standpoint and introduce the reader to rateless graph-based codes, the codes upon which our proposed approach is based. Sections 3 and 4 present the design and a preliminary performance evaluation of PRC, respectively, and Section 5 closes.

2. RELATED WORK AND PRIMER ON GRAPH-BASED CODES

In this section we survey recent work in rateless coding from the perspective of energy efficiency, then present a primer on existing graph-based codes to allow the reader to contrast the novel graph-based code we describe in §3.

We begin with the decoder for a Spinal code, shown in Figure 1. Each level of the decoding tree corresponds to a block of k bits in the n-bit packet, and each of the 2^k edges emanating from a node correspond to the 2^k possible messages in the corresponding k-bit block. As a result, information received at level l tells the decoder about levels l and lower (possibly fixing earlier bit errors), but not about levels higher than l. Consequentially, while Spinal codes achieve a rate close to Shannon capacity, the decoder may not know whether the entire packet will decode or not until it has processed all of the received information.

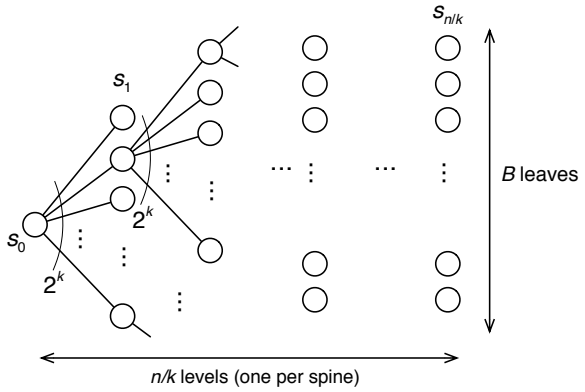

Figure 1: In the Spinal Code decoder, earlier iterations process earlier bits in the packet, while later iterations process successively larger prefixes of packet bits.

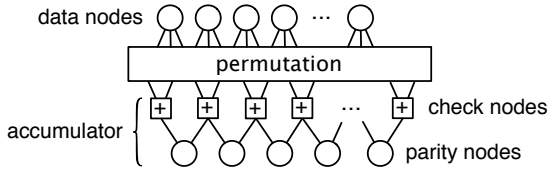

Figure 2: In a graph-based code such as an Irregular Repeat-Accumulate (IRA) code, data nodes connect to an accumulator through a permutation. On each of its iterations, the belief propagation decoder allocates processing power roughly uniformly across all data bits.

Strider [9] works by forming random linear combinations of encoded data blocks as follows:

$$\begin{pmatrix} \text{packet } 1 \\ \text{packet } 2 \\ \vdots \\ \text{packet } M \end{pmatrix} = \mathbf{R} \begin{pmatrix} \text{coded block } 1 \\ \text{coded block } 2 \\ \vdots \\ \text{coded block } K \end{pmatrix} \quad (1)$$

where $\mathbf{R}$ is an $M \times K$ matrix containing random coefficients. Given the M received packets on the left hand side of Equation 1, Strider's decoder first uses all M packets to decode coded block 1, subtracts the effect of coded block 1 from the packets on the left hand side, and reduces the number of columns of $\mathbf{R}$ by one. The decoder continues iteratively until all coded blocks are decoded. Since Strider's decoder uses all received packets to decode each coded block, an approach similar to the one we propose here may be possible; we leave an investigation of such an approach for future work.

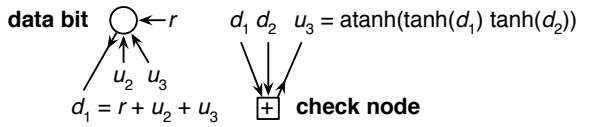

Figure 3: In a binary decoder, the "down" message a data bit produces is the sum of the up messages not incident on the message's edge while the "up" message from a check bit diminishes in certainty if *any* of the constituent down messages is uncertain.

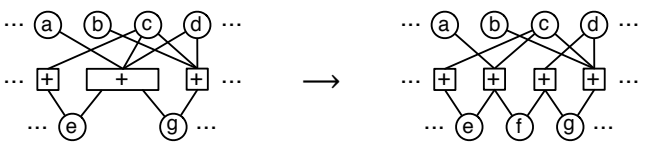

Figure 4: Check splitting reduces the code rate.

2.1 Rateless graph-based codes

Many different types of rateless graph-based codes exist, but they can all be described by a graph such as the one in Figure 2. Here, the information to be communicated is associated with the *data nodes* in the graph, while the *check nodes* represent a constraint on the information, most often that the connected data nodes sum to zero. The data nodes are connected to the accumulator through a random edge permutation, as shown in the figure.

LT [15], Online [17], Raptor [7], Rate-Compatible LDPC [10], and Irregular Repeat-Accumulate (IRA) [13] codes all play on this theme with different graph structures and precoding steps to optimize performance. Such codes have also found use in rateless content distribution on the Internet [4].

The belief propagation decoder of a graph-based code consists of the same graph, where each of the data and parity nodes carries information r about the received bits as distorted by the wireless channel, as shown in Figure 3. The belief propagation decoder operates in *iterations*, where one iteration consists of all data nodes passing messages out of each incident edge, followed by all check nodes passing messages out of each incident edge. In graph based codes, data nodes can represent bits, or elements of a finite field of size q (GF(q)). If data nodes represent bits, then each message is a log-likelihood ratio (LLR) of the two possibilities for the bit, $\log \frac{\Pr(b=1)}{\Pr(b=0)}$. Otherwise, each message is a probability vector of size q where each element of the vector represents the probability that the data is that value in GF(q).

For a binary code, all messages are LLRs, with a large positive LLR indicating strong confidence in a "1", a large negative LLR indicating strong confidence in a "0", and an LLR small in absolute value indicating a weak confidence in that bit. Referring to Figure 3, the *down message* a data node generates on an edge (d_1, *e.g.*) is the sum of all information known about the data with the exception of the information from the edge over which the message departs: $r + u_2 + u_3$. The *up message* a check node generates on an edge (u_3, *e.g.*) is a function of the down messages arriving from connected data nodes, with the exception of the down message arriving over the edge the up message departs. It can be shown that the correct function to use to combine these messages is the hyperbolic tangent [3], but intuitively, the up message from the check node tends to be low confidence (close to zero) if *any* of its constituent down messages are zero. These concepts generalize to higher-order finite fields; for brevity we omit details, but refer the reader to the literature [6], *et al.*

Some strategies for early termination in a belief propagation decoder have been proposed, primarily focusing on binary LDPC decoders [14], as well as Raptor [12] and GF(2^q) [5] codes. These works focus on either the sum of symbol variances or the change in the number of incorrect bits from iteration to iteration. Informal testing suggests that our more sophisticated halting criterion halts decoding in fewer iterations, but we leave a comparative experimental evaluation as future work.

For our purposes in designing an energy-aware decoder, the belief propagation decoder has a very useful property. Referring to Figure 2, since data nodes are connected to parity nodes through a random permutation, within a small number of decoder iterations, it is highly likely that a randomly-chosen pair of data nodes will have been tested against each other by some check node. This observation is the starting point of our energy-aware decoder design (§3.2).

3. DESIGN

In this section, we describe PRC's code design (§3.1), decoder (§3.2), and medium access control protocol (§3.3).

3.1 Graph-based code

Recent work on Irregular Repeat-Accumulate (IRA) codes over high-order finite fields has shown that they come extremely close to capacity on the Additive White Gaussian Noise (AWGN) channel for short packet sizes [6]. Motivated by this positive result on one-dimensional noisy channels, we base PRC's code on an IRA code designed for the two-dimensional Quadrature Amplitude Modulation (QAM) channel that wireless networks use.

We have designed two codes: a GF(256) code that has superior capacity properties (§4.2), and a binary code that has superior power-consumption properties (§4.3). Our codes use check splitting and combining techniques [8], whereby a single high-degree check node is split into two or more check nodes of lower degree, as shown in Figure 4.

3.2 Decoder

As observed in §2, each data node x_i in a graph-based code is associated with a numerical confidence in that node's correctness. This is the LLR in the case of a binary code, and the data node's variance across its probability vector

$$\sigma_i^2 = \sum_{k=0}^{q-1} \left(\Pr\left(x_i = k\right) - \mu_i\right)^2$$

for a GF(q) code, where $\mu_i = \frac{1}{N}\sum_{k=0}^{q-1} \Pr(x_i = k)$.

The distribution of all data nodes' σ_i^2 values s essentially the distribution (across data nodes) of decoder confidences, quantities that change with each iteration of the decoder as it passes messages into and out of check nodes, testing the constraints that the check nodes enforce.

After each transmission from the sender, PRC tracks multiple variance order statistics (VOSs) in its decoder on every iteration, monitoring their trajectories so as to predict whether the packet will eventually decode. Notice that this is a multi-stage decision process: at any iteration the decoder may abort its decoding effort and request more bits.

On receiving each transmission the receiver uses standard communications techniques to estimate the SNR over the packet's preamble. While collisions or fading may impact the body of the packet in different ways than the preamble, we take this simple approach as a starting point, leaving SNR estimation over all received bits to future work.

The PRC decoder proceeds in iterations, estimating the derivative of each VOS by averaging the slope across the three preceding samples. We term a VOS *active* if this estimate predicts that the packet will decode in Ω iterations, where Ω is the average number of iterations required to decode at the estimated SNR.

Halting algorithm.

PRC halts decoding if the derivative of the VOS for percentiles above the 50^{th} percentile decreases to the point that if the derivative were to remain constant it would not achieve 95% of the maximum possible variance. This will result in the early termination of some packets that would otherwise succeed. However, for a small capacity cost the decoder will succeed in much fewer iterations after receiving additional data, and so the total cost of decoding is generally lower. It also halts decoding if the derivative of an active VOS for percentiles below 50% becomes negative. If it runs for Ω iterations without succeeding or halting, PRC halts decoding more readily: if sufficient progress has not been made (*i.e.* any VOSs above the 10^{th} are inactive), or if any VOS's derivative over a single iteration becomes negative.

Let's examine how PRC would behave during decoding of the four packets in Figure 5. Packet (1) would not be interrupted, and successfully decodes. Packet (2) would decode as normal, but at iteration 8, when the derivative of the 20^{th} percentile statistic becomes negative, PRC halts decoding. Early termination of decoding in this instance marginally reduces the rate achieved, but saves energy as the packet decodes in very few iterations with more data. Packet (3) is an undecodable packet that has been slowly improving throughout decoding, however, at Ω it has not shown sufficient improvement and decoding halts. Finally, packet (4) is an undecodable packet that PRC halts early, as the derivatives of higher percentile VOSs have decreased to the point where it is insufficient to reach 95% of the maximum variance by Ω. These examples also serve to illustrate how decoding progress can vary greatly from packet to packet, and the difficulty of accurately determining whether a packet will successfully decode or not.

3.3 Medium access control protocol

The medium access control (MAC) protocol we used for simulations attempts to take advantage of the primary characteristic of rateless codes that allow them to outperform equivalent rated codes: because of variance in the noise, some packets at equivalent SNRs will be better than others. Our MAC therefore optimistically attempts to transmit at slightly above the average rate obtained at a given SNR, then to achieve the target BER.

4. EVALUATION

In this section, we present simulation-based results comparing the rate and power consumption of PRC, Spinal Codes, and Strider. PRC keeps transmitting bits until it achieves a bit error rate (BER) of less than 10^{-6}. Throughout, we use operation counts as a surrogate for power consumption.

4.1 Methodology

We measure the number of operations used by PRC in decoding as the number of operations used to demodulate received symbols, construct probability vectors, run each decoding iteration, calculate variance statistics, and determine whether decoding should continue.

We approximate the number of operations that the Spinal Codes decoder uses as follows. The decoder uses a trellis structure, keeping B leaves and generating $B2^k$ hashes at each stage. Using the one-at-a-time hash function costs 31 operations [18]. Additionally, the decoder computes metrics for each generated spine by computing the Euclidean distance (costing $\phi = 3$ operations) between the spine and of each ζ received symbols for that spine. These metrics are summed over the entire spine. The decoder sorts the resulting metrics, incurring a cost of $B2^k \log(B2^k)$ comparisons and a further three operations when a swap is necessary. Since the hash function maps ζ k-bit inputs randomly within the c-bit symbol space, we can assume that the data being sorted is randomly generated, and so expect half of the comparisons to require a swap. This gives an expected cost of

$$B2^k \cdot 31 + B2^k \cdot \zeta \cdot \phi + B2^k \log(B2^k) \cdot (1 + 0.5 \cdot 3) \quad (2)$$

for each spine.

4.2 Rate comparison

We begin with a performance evaluation of just the codes underlying PRC, Figure 6 shows the comparison of our codes against Spinal codes and the implementations of Strider and Raptor code by Perry *et al.* [18]. Results for Spinal Codes with $B = 16, 64$ are generated using code provided by Perry *et al.*, using the puncturing schedule in [18] and evaluating all B branches at the final spine for the correct message.

The basic codes underlying PRC perform extremely well, with both codes generally outperforming both Raptor and

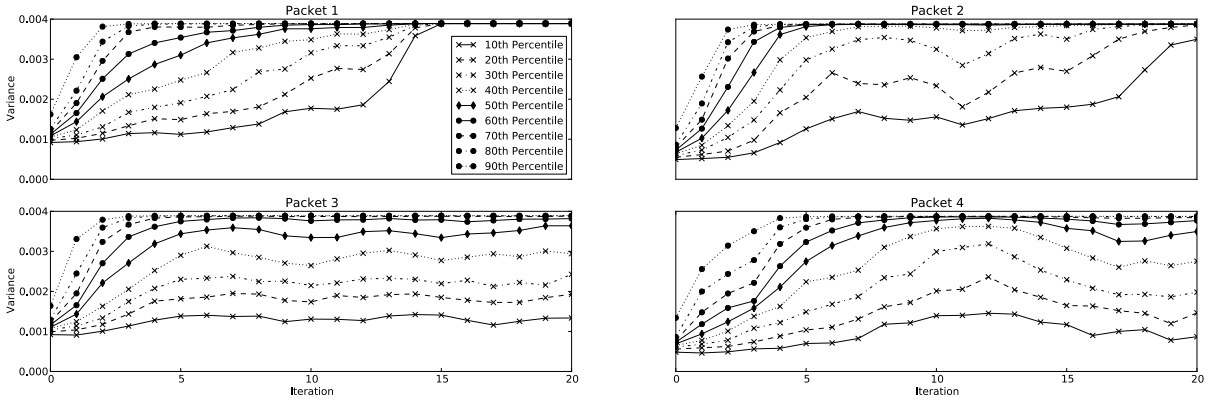

Figure 5: The variance order statistics (VOSs) of several GF(256) packets in the PRC decoder for SNR $= 18.7$ dB, $n = 1024$, rate $= 0.68$, and $\Omega = 20$.

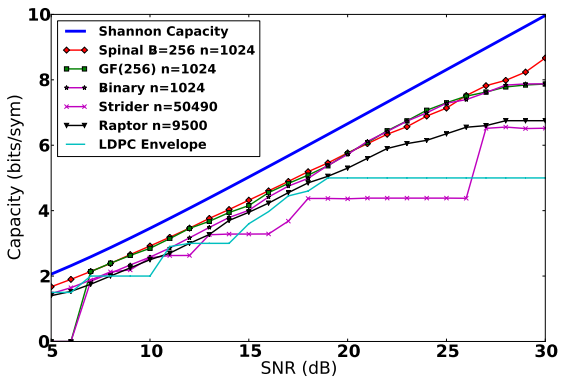

Figure 6: Rate versus signal to noise ratio (SNR) for PRC's graph-based code *without* decoder power awareness, Spinal Codes, Strider, Raptor Codes and the LDPC envelope.

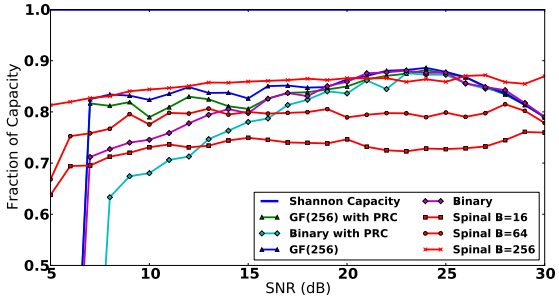

Figure 7: Rate versus signal to noise ratio (SNR) for PRC *with* decoder energy awareness and Spinal codes as a fraction of Shannon capacity.

Strider. Additionally, they are extremely competitive with Spinal Codes, outperforming them between 21–25dB and remaining extremely competitive outside this range. This is particularly true of the GF(256) code as the binary code falls off sharper at lower rates.

Next we examine PRC as a whole; Figure 7 presents the overall rate we achieve with the power-aware decoder and MAC protocol described in §3.2 and §3.3, respectively, as a fraction of Shannon capacity. The rate PRC achieves is extremely competitive with Spinal Codes, marginally outperforming Spinal Codes between 21 dB and 25 dB, while generally staying within 0.25 bits/symbol below this range. In fact, at the lowest SNRs our current implementation is able to function at, PRC is becoming increasingly competitive with Spinal Codes decoder. We also include the results for Spinal Codes with vary values of B.

4.3 Power Efficiency and Savings

We now examine the power efficiency of PRC, using the number of operations required to decode as a surrogate for power efficiency. We begin by evaluating how much power PRC saves over a decoder that uses a fixed number of runs. The maximum number of runs required to successfully decode for our GF(256) and binary codes is generally fifty iterations above 15 dB and scales up to one hundred iterations below that. For PRC decoding, Ω is typically set to 15–40 depending on the SNR estimate.

Figure 8 shows the fraction of operations needed by PRC to successfully decode in place of an equivalent full decoding run. At 28–30 dB the fraction of operations used approaches and surpasses 1.0 due to very few packets requiring additional iterations. As a result, codes with and without PRC decode in roughly the same number of iterations, meaning that the overhead from PRC has a larger effect on the fraction of operations used. Below 28 dB, PRC performs very well comparatively, generally requiring between 20–60% for the GF(256) code and 40–90% operations for the binary code. Also of note is that the demodulation cost for obtaining soft values for the QAM-256 constellation is quite high, approximately 1 million operations at high rates and 3 million at lower rates, which is a not inconsiderable overhead, particularly for the binary code, where the additional overhead required to obtain LLR values for the received bits greatly increases the number of operations required to demodulate compared with obtaining probability vectors for the GF(256) code. This could be improved by intelligently demodulating to avoid this high cost. The high demodulation cost, as well as the higher relative cost of running the halting algorithm on the binary implementation are some reasons

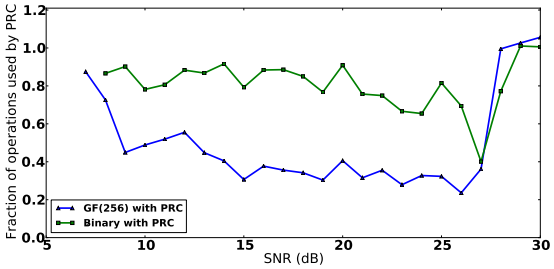

Figure 8: The fraction of operations used by the GF(256) and binary code with PRC versus the same base code.

why the binary code with PRC does not see the same improvements that the GF(256) code sees. This suggests that a less effective but less expensive halting algorithm may be the better choice for a binary implementation. Additionally, as shown in Figure 7, the binary code with PRC exhibits a faster drop in rate than the GF(256) code, indicating that the halting algorithm was poorly tuned for the binary code.

To compare Spinal Codes against our simulation results, we calculate the number of operations Spinal Codes would require for a single decoding attempt. Spinal Codes incurs a relatively small cost for reducing the code rate, slightly less than trebling the number of operations required between 5–30 dB. Changing the value of B is very beneficial to Spinal Codes decoding cost, however Figure 7 shows that it seems to have a large effect on the fraction of capacity obtained.

The operations count for the Raptor Code presented in Figure 9 were obtained from our own implementation of Raptor Codes using a similar construction method to Perry *et al.* [18]. The LT code used the distribution given in the Raptor RFC [16], and the outer LDPC code has code rate 0.95 and has regular left degree 4, the LDPC matrix itself is created using the PEG algorithm [11]. Capacity achieved using this code was disappointing, possibly due to the shorter block length, $n = 1024$, compared with the results produced by Perry *et al.*. Due to the poor rate achieved by this code, it would be unfair to compare at equivalent SNRs to our codes, instead, we present results for a single Raptor decoding attempt using a full 50 iterations at the same rate achieved by Perry *et al.*'s Raptor code. Interestingly, the number of operations for a full run of the belief propagation on the Raptor code is very similar to Spinal Codes, yet the Raptor code exhibits much lower performance.

5. CONCLUSION AND FUTURE WORK

As alluded to in §3, we will investigate more sophisticated irregular rateless IRA code designs to further increase capacity, and shift to lower-order modulations at lower rates on a pre-determined schedule. We will also extend PRC to incorporate ideas from "gear-shift" decoding [2], which proposes switching decoding algorithms in response to predictions about the ease with which received bits will be decoded. Finally, as mentioned in §2, §4, a deeper look into

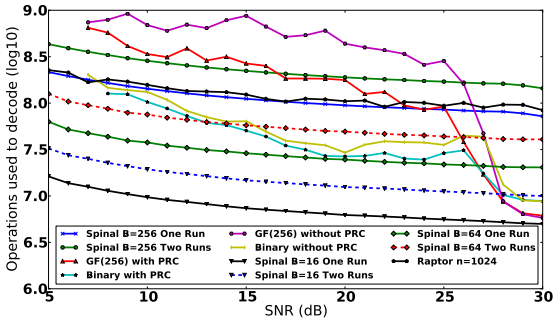

Figure 9: Comparison of operations needed to decode using PRC, number of operations needed without PRC, and Spinal Codes with one and two frame transmissions.

effective halting algorithms, particularly for the binary implementation should further improve energy savings.

Other future work concerns modelling the power savings of our approach. Here an early termination algorithm for a fixed-rate LDPC decoder [1] uses 80 nJ per decoder iteration, on average. A full decoding run, 50 iterations, would use 4 μJ. Our approach reduces the operations between 30–90%, suggesting potential savings of 0.4–3 μJ per packet.

6. REFERENCES

[1] E. Amador, R. Knopp, V. Rezard, and R. Pacalet. Dynamic power management on ldpc decoders. In *Proc. of the* IEEE *Symp. on VLSI and Circuits*, June 2010.

[2] M. Ardakani and F. Kschischang. Gear-shift decoding. IEEE *Trans. on Comms.*, 54(7):1235–1242, July 2006.

[3] J. Barry, E. Lee, and D. Messerschmitt. *Digital communication*. Kluwer Academic Publishers, 2004.

[4] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *Proc. of the* ACM SIGCOMM *Conf.*, 1998.

[5] X. Chen, A. Men, and W. Zhou. A stopping criterion for nonbinary ldpc codes over gf(q). In *Proc. of the* IEEE *Singapore International Conference on Communication Systems* , Nov. 2008.

[6] M. C. Chiu. Bandwidth-efficient modulation codes based on non-binary irregular repeat-accumulate codes. IEEE *Trans. on Info. Theory*, 56(1):152–167, 2010.

[7] O. Etesami and A. Shokrollahi. Raptor codes on binary memoryless symmetric channels. IEEE *Trans. I. Theory*, 52(5):2033–2051, 2006.

[8] M. Good and F. Kschischang. Incremental redundancy via check splitting. In *Proc. Biennial Sym. on Comm.*, pages 55–58, 2006.

[9] A. Gudipati and S. Katti. Strider: Automatic rate adaptation and collision handling. In *Proc. of the* ACM SIGCOMM *Conf.*, 2011.

[10] J. Ha, J. Kim, and S. W. McLaughlin. Rate-compatible puncturing of low-density parity-check codes. IEEE *Trans. on Info. Theory*, 50(11):2824–2836, 2004.

[11] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold. Progressive edge-growth tanner graphs. In *Proc. of* IEEE *Global Telecommunications Conf.*, volume 2, pages 995 –1001, 2001.

[12] A. A. Hussein, A. Oka, and L. Lampe. Decoding with early termination for raptor. IEEE *Comm. Letters*, 12(6), June 2008.

[13] H. Jin, A. Khandekar, and R. McEliece. Irregular repeat-accumulate codes. In *Proc. Symp. Turbo Codes Related Topics*, 2000.

[14] F. Kienle and N. Wehn. Low complexity stopping criterion for ldpc code decoders. In *Proc. of* IEEE *VTC*, pages 606–609, June 2005.

[15] M. Luby. LT codes. In *Proc. of the* IEEE FOCS *Symp.*, 2002.

[16] M. Luby, A. Shokrollahi, M. Watson, and T. Stockhammer. Raptor forward error correction scheme for object delivery, Oct. 2007.

[17] P. Maymounkov. Online codes. Technical Report 833, NYU, 2002.

[18] J. Perry, P. Iannucci, K. Fleming, H. Balakrishnan, and D. Shah. Spinal codes. In *Proc. of the* ACM SIGCOMM *Conf.*, 2012.