Rateless spinal codes

Rateless Spinal Codes

Jonathan Perry, Hari Balakrishnan, and Devavrat Shah
Massachusetts Institute of Technology
Cambridge, MA, USA
{yonch,hari,devavrat}@mit.edu

ABSTRACT

A fundamental problem in wireless networks is to develop communication protocols that achieve high throughput in the face of noise, interference, and fading, all of which vary with time. An ideal solution is a *rateless* wireless system, in which the sender encodes data without any explicit estimation or adaptation, implicitly adapting to the level of noise or interference. In this paper, we present a novel rateless code, the *spinal code*, which uses a hash function over the message bits to produce pseudo-random bits that in turn can be mapped directly to a dense constellation for transmission. Results from theoretical analysis and simulations show that spinal codes essentially achieve Shannon capacity, and out-perform best-known fixed rate block codes.

CATEGORIES AND SUBJECT DESCRIPTORS

C.2.1 [**Network Architecture and Design**]: Wireless communication

GENERAL TERMS

Algorithms, Design, Performance

KEYWORDS

Wireless, rateless, channel code, capacity, practical decoder

1. INTRODUCTION

Achieving high communication rates over wireless networks is difficult because wireless channel conditions vary with time, even at time-scales shorter than a single packet transmission time. To achieve high throughput, a communication protocol must not only operate at a high rate given constant channel conditions, but must also adapt well to variations in noise, attenuation, interference, and multipath fading.

Current wireless networks, including 802.11 (Wi-Fi) and various wide-area cellular wireless standards approach this problem by providing a large number of physical layer (PHY) configurations, including a variety of channel codes, various parameters for these codes, several choices of symbol sets (i.e., constellation) over which to modulate bits, and a way to map groups of bits to symbols. The link and sub-network layers implement bit rate adaptation policies to dynamically select and configure the discrete choices and parameters provided by the PHY. The selection of a suitable bit rate is made by observing channel conditions, such as the signal-to-noise ratio (SNR) from a preamble [7, 8] or pilot tones, interference-free bit error rate [19], dispersion in the constellation space [13], frame loss rate [20], or the time taken to successfully transmit a frame [2].

The status quo has two shortcomings. First, it greatly increases the complexity of wireless systems because they have to implement a variety of mechanisms and policies across different layers of the traditional protocol stack. Second, because this approach is inherently reactive in nature, the resulting performance may be sub-optimal, especially in situations where fading or interference from other transmitters causes channel conditions to change quickly or in unpredictable ways.

An alternative approach to these problems is to use a *rateless code* between the sender and receiver, as previous work has noted [3, 5]. Ideally, with such a code, the sender encodes and transmits the coded data at a rate higher than what the channel can currently sustain, and keeps going until the receiver determines that it has correctly decoded all the data and informs the sender to move to the next packet. For this approach to work and achieve a high rate, the sender must encode and modulate the data in a way that allows the receiver to efficiently decode the data in the presence of channel impediments. An ideal rateless code has the property that communication can occur at a rate close to the capacity of the channel, but *without* the sender having to estimate the channel quality and adapt the bit rate explicitly.

To enable the design of such a rateless wireless system, this paper introduces *spinal codes*, a new family of rateless codes. These codes are designed for both analog channels and digital channels: i.e., they can code message bits in a packet directly to symbols for transmission, or to coded bits, which can in turn be sent using any symbol set. If one has full control over all the layers of the protocol stack, the first approach is beneficial because the decoder extracts information from the

Hotnets '11, November 14–15, 2011, Cambridge, MA, USA.

raw received symbol, avoiding loss of soft information caused by manipulation by a demapper. If not, one can still use spinal codes over commodity PHY hardware implementations and improve the rates and error resilience over the status quo.

At the heart of spinal codes lies the use of a *hash function*, which is applied repeatedly in a sequential manner to successive segments of the original message bits to produce a random, nonlinear mapping between message bits and coded bits. The sequential nature of the hashed map makes the encoding linear in the message size. If the system has control over the PHY, the coded bits are then mapped using an *integrated constellation mapping function* to an extremely dense symbol set. Because of the high density, a lot of information can be conveyed when noise is low. When noise is high, robustness comes from the code, which is capable of operating at SNR values as low as -10 dB. The advantage of this method is that the sender no longer has to make any decision of what modulation scheme to use in light of (estimated) channel conditions, or indeed even bother estimating what these conditions might be. An alternative, when modifications to the PHY are infeasible or undesirable, is to transmit the coded bits directly over a traditional modulation method designed to transmit bits, in which case spinal codes operate over a binary channel.

Despite the nonlinearity of spinal codes, we show how the sequential nature of the encoder enables an *efficient practical decoder*, constructed by "replaying" a version of the encoder over the received symbols (or bits) at the receiver [1]. Our practical decoder has a natural "scale down" property: it can operate with any amount of computation resource and attempts to achieve the best performance using the given resources. We present simulation results demonstrating that the scaled-down decoder, which approximates an ideal decoder, achieves nearly optimal code rates with small resource requirements. We also show that spinal codes significantly outperform state-of-the-art low-density parity check (LDPC) codes, especially for small block sizes.

This paper also gives two theorems (without proof, for want of space), which state that spinal codes are essentially capacity-achieving codes for the additive white Gaussian noise (AWGN) channel as well as the binary symmetric channel (BSC). To the best of our knowledge, spinal codes are the first rateless codes that essentially achieve capacity over both AWGN and BSC channels, for which there is an efficient encoder and a practical decoder, and the first codes constructed using hash functions.

2. Related Work

Rateless codes have a long history starting with classical ARQ schemes, but ARQ generally does not come close to capacity. From an information-theoretic perspective, rateless codes for AWGN or BSC, which are characterized by a single noise or error parameter, are well-understood. Shannon's random codebook approach achieves capacity for these channels, and is inherently rateless, but the approach is computationally intractable.

The desire for computationally efficient, capacity-achieving rateless codes led to discovery of Raptor codes by Shokrollahi [15]. Built upon the LT codes of Luby [10], they achieve capacity for Binary Erasure Channel (BEC) where packets are erased (lost) with some probability. There have been interesting attempts made to extend the Raptor code or punctured LDPC/Turbo codes for the AWGN channel [6, 14, 16, 11, 9, 1], but little is known in terms of the closeness to capacity of these approaches.

A very different and promising approach to design of capacity achieving rateless codes for AWGN channel was put forth by Erez, Trott, and Wornell [3]. Their "layered approach" uses existing base codes and combines them to produce transmission symbols in a rateless way. They show that by producing rateless symbols using the appropriate selection of linear combinations of symbols generated by the base codes, capacity can be achieved by the resulting rateless code as the number of layers becomes large enough, provided the fixed-rate base code achieves capacity at some fixed SNR. This elegant construction, in a sense, takes capacity-achieving codes at a given SNR to produce capacity achieving rateless codes for *any* SNR by varying the number of layers. This work, though primarily theoretical in nature, offers the promise of a practical design. Indeed, recent work on Strider by Gudipati and Katti [4, 5] presents the design and implementation of a code along the principles described by Erez et al. Strider exhibits good empirical performance and suggests that the layered approach is practical.

In contrast, spinal codes are not layered codes and do not rely on existing base codes (such as LDPC). Unlike Strider, which takes an existing fixed-rate code and symbol set system and makes modifications to the lowest physical layer procedures to achieve linear combinations of symbols, the construction of spinal codes provides a single mechanism to overcome channel impediments. As such, we think it might be a simpler approach because it does not require any other codes or choice of symbol sets in the system; the code can directly convert message bits to symbols. That said, our goal here is not to claim that spinal codes exhibit superior performance—we don't yet know—but to show a new and different way of constructing practical capacity-achieving rateless codes.

We note a superficial similarity between spinal code and Trellis Coded Modulation (TCM) [18, 17] because TCM codes bits to symbols. TCM was crafted specifically to achieve high *minimum* distance between codewords under a sparse constellation for convolutional codes, whereas spinal codes aim to attain higher *average* distance, obviating the need for sparse constellations. TCM is not rateless, does not achieve capacity for AWGN, is not workable (in any obvious way) for BSC, and is generally specific to convolutional codes.

3. Spinal Codes

[1] Replaying the encoder allows inference of the hash input bits using the hash function; an inverse of the hash function is not required.

This section describes the encoder and decoder for spinal codes. We describe them in the context of a system that has full control of the physical layer, so the encoder produces a sequence of symbols for transmission and the decoder operates on the received symbol sequence to produce an estimate of the original message bits. By slightly modifying the encoder and decoder, it is straightforward to apply the code to a system that has an existing mapping from (coded) bits to symbols, and analyze the performance under the binary symmetric channel model.

The encoding procedure takes the input message bits, $M = m_1 m_2 \ldots m_n$, and produces a potentially infinite sequence of symbols on the *I-Q* (quadrature) plane. At the receiver, the PHY receives a stream of symbols on the *I-Q* plane. The decoder processes this stream sequentially, continuing until the message is successfully decoded, or until it (or the sender) gives up, causing the sender to proceed to the next message.

Spinal codes are rateless: the encoder can produce as many symbols as necessary from a given sequence of message bits. It is straightforward to adapt the code to run at various fixed rates, though we expect the rateless instantiations to be more useful in practical wireless communication systems.

3.1 Encoding

At the core of the spinal code is a *random hash function*, h, which takes two inputs—(i) a real number in $[0,1)$ and (ii) a bit-string (from the message) of length k bits—and returns a real number in $[0,1)$. That is,

$$h : [0,1) \times \{0,1\}^k \to [0,1).$$

Conceptually the input real number and output have infinite precision, though in practice, they will have some finite precision. This conceptual abstraction is convenient; because there are many ways to produce as many output bits as needed, it is not a problematic assumption (e.g., using repeated hashing with different known salts, as needed).

We choose h uniformly at random, based on a random seed, from $\mathscr{H}$, a family of hash functions. The encoder and decoder both know h and agree on the initial value for the first argument to h, denoted s_0.

Let $\bar{m} = (m_1, \ldots, m_k)$ be a k-bit string ($k \geq 1$). We make the following standard assumptions about the random hash function:

(i) *Uniformity.* For any $(s;\bar{m})$,

$$\mathbb{P}\big(h(s;\bar{m}) \leq x\big) = x, \quad \text{for any } x \in (0,1). \tag{1}$$

(ii) *Independence.* For any $\ell \geq 2$ with $(s_i;\bar{m}_i)$ for $1 \leq i \leq \ell$,

$$\mathbb{P}\big(\cap_{i=1}^{\ell} h(s_i;\bar{m}_i) \leq x_i\big) = \prod_{i=1}^{\ell} x_i, \tag{2}$$

when $(s_i;\bar{m}_i) \neq (s_j;\bar{m}_j)$ for $1 \leq i \neq j \leq \ell$ with $x_i \in (0,1)$ for $1 \leq i \leq \ell$. Here by $(s;\bar{m}) \neq (s';\bar{m}')$ we mean either $s \neq s'$ or $\bar{m}$ and $\bar{m}'$ differ in at least one bit.

Spine generation and encoding passes. The encoder takes a message block $M = m_1, \ldots, m_n$ as input and generates a possibly infinite stream of constellation points. To do so, the encoder first produces the **spine** of the code, and then makes one or more *passes* over the spine to transmit symbols.

The encoder divides the message M into non-overlapping segments of size k bits each: $M = M_1, M_2, \ldots M_{n/k}$ with $M_t = m_{(t-1)k+1} m_{(t-1)k+2} \ldots m_{tk} \in \{0,1\}^k$ for $1 \leq t \leq n/k$. Then, it computes a series of n/k *spine values*: $s_t = h(s_{t-1}, M_t)$. These values $s_1, s_2, \ldots, s_{n/k}$ are the *spine* corresponding to M.

The encoder then generates symbols by taking $2c$ different bits at a time from each spine value, in order, and mapping them to a single symbol using a deterministic *constellation mapping function*. This mapping is done in passes. In each pass, the encoder generates n/k new constellation points, taking the next $2c$ bits from each successive spine value.

Figure 1 illustrates the encoding process. It shows the sequential structure of the encoder in which the previous spine value is an input to the hash function to generate the next one. When the pass ends, the next pass begins, unless the receiver informs the sender that the message has been decoded successfully or the sender decides that too much time has been spent sending the current message and that it must be abandoned.

More precisely, in the ℓ^{th} pass ($\ell \geq 1$), the encoder generates the t^{th} constellation point or coded bit as follows ($1 \leq t \leq n/k$):

1. Let the infinite precision bit representation of s_t be $b_1 b_2 b_3 \ldots$.
2. Let $b'_1, \ldots, b'_{2c}$ denote the $2c$ bits in the representation of s_t starting from position $2c(\ell-1)$: for $\ell = 1$, it is $b_1, \ldots, b_{2c}$; for $\ell = 2$, it is $b_{2c+1}, \ldots, b_{4c}$ and in general it is $b_{2c(\ell-1)+1}, \ldots b_{2c\ell}$.
3. Use the constellation mapping function, f, to map $b'_1, \ldots, b'_{2c}$ to a constellation point. This coded bit or constellation point gets transmitted. (For a binary channel, use b'_1 as the coded bit.)

   There are many possible constellation mappings; in this paper, we use a simple linear one. Take the $2c$ bits and consider the first c bits as the I part and the last c bits as the Q part. The I (resp. Q) coordinate mapped to the range $[-P^*, P^*]$ is:

$$(b'_1 \ldots b'_c) \to (-1)^{b'_1} \frac{(b'_2 \ldots b'_c)}{2^{c-1}-1} \cdot P^* \tag{3}$$

One can use other constellation mappings in a spinal code. A promising one that we are currently analyzing and experimenting with is a truncated Gaussian function. The value of c should be large enough so the constellation mapping can sustain high rates when SNR is high. When the SNR is low, the large c is not needed, although there is no loss incurred by the extra precision: the decoding process is built to correctly take into account that MSBs are less likely to be erroneous than LSBs. In fact, this compensation is inherent to the decoder, since inference is made using symbols rather than individual bits.

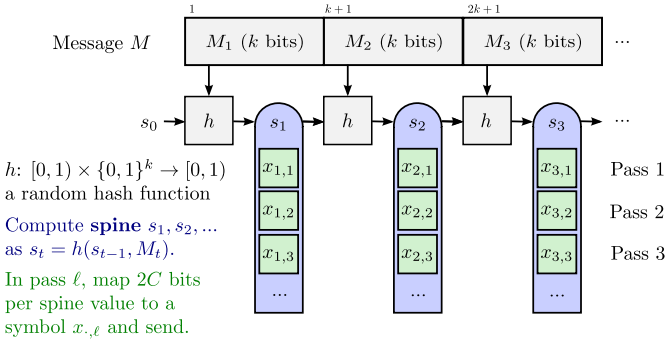

Figure 1: Encoding process.

The description of spinal codes has only one parameter, k, that we have not yet discussed. As we will see next, the computational complexity of the decoder grows exponentially with k, while the maximum rate achievable by the code grows linearly with k. In the description thus far, the maximum rate is k bits/symbol, achieved when the receiver decodes successfully after a single pass, meaning n bits are transmitted in n/k symbols. We expect k to be a small constant; for example, over the default 802.11b/g bandwidth of 20 MHz, if we pick k to be 8 bits/symbol, the maximum link bit rate will be 160 Megabits/s; if the bandwidth doubles to 40 MHz, the same k would give us a maximum bit rate of 320 Megabits/s. In our experiments, we actually obtain rates higher than k bits/symbol using *puncturing*, where the transmitter does not send each successive spine value in every pass.

3.2 Decoding Spinal Codes

In this section, we describe a maximum likelihood (ML) decoder and a practical "scale-down" version of the decoder for spinal codes. The ML decoder achieves capacity over the BSC channel and nearly achieves capacity over AWGN channels; we state the theorems in the next section. The ML decoder has exponential decoding complexity, a property shared by the decoders of some other codes (e.g., classical sphere decoding). An important consideration in practice is whether a *practical* decoder is implementable by starting from the ideal ML decoder and reducing the resources consumed; we use the term *graceful scale-down* to refer to this property, which captures the intuition that we would like the performance to reduce in a gradual way with the reduction in the computational resources at the decoder. We show how a decoder for spinal codes can take advantage of the structure provided by the hash function to replay the encoder at the decoder and achieve this property. In section 5, we show simulation results that demonstrate that modest computational resources provide performance close to capacity, and outperform state-of-the-art fixed-rate block codes across a range of SNR values.

Suppose the transmitter has made L passes over message $M = m_1 \ldots m_n$ resulting in transmission of $N = Ln/k$ symbols in total: Let $x_{t,\ell}(M)$ be the t^{th} symbol sent in the ℓ^{th} pass, $1 \le t \le n/k,\ \ 1 \le \ell \le L$. The receiver sees $y_{t,\ell} = x_{t,\ell} + w_{t,\ell}$, where $w_{t,\ell}$ represents additive noise. For the AWGN channel, w is an independent and identically distributed (iid) complex symmetric Gaussian of mean 0 and variance σ^2.

At the end of the L^{th} pass, the receiver, given observations $\bar{y} = \left(y_{t,\ell},\ 1 \le t \le n/k,\ 1 \le \ell \le L\right)$, wishes to estimate $M \in \{0,1\}^n$ by estimating the corresponding transmitted symbols $\bar{x} = \left(x_{t,\ell}(M),\ 1 \le t \le n/k,\ 1 \le \ell \le L\right)$. It is well-known that the ML rule, which minimizes probability of error with respect to a uniform prior, is

$$\hat{M} \in \arg\min_{M' \in \{0,1\}^n} (\|\bar{y} - \bar{x}(M')\|^2). \tag{4}$$

That is, the estimated message $\hat{M} \in \{0,1\}^n$ is the one such that the encoded vector, $\bar{x}(\hat{M})$, is closest (in ℓ_2 distance) to $\bar{y}$.

For the BSC channel, the ML decoder turns out to be essentially the same as that for AWGN channel: the estimated message $\hat{M} \in \{0,1\}^n$, given the received bit sequence $\bar{y} \in \{0,1\}^N$, is the one such that the transmitted bit sequence $\bar{x}(\hat{M})$ is closest to $\bar{y}$ in terms of the Hamming distance; i.e., replace the ℓ_2 distance in (4) by the Hamming distance.

The standard implementation of an ML decoder would require an exhaustive search over the space of all 2^n possibilities. When the decoder has much smaller computational resources, it is not obvious how to adapt this standard method. Fortunately, it is possible to construct an ML decoder with the scale-down property for spinal codes.

Ideal ML decoder for spinal codes. The key idea is to use the shared knowledge of the hash function to *replay the encoder at the decoder* over the set of received symbols and all possible combinations of k-bit inputs to the hash function at each stage. Using the same notation as before, let the t^{th} received symbol in the first pass be $y_{t,1}$ for $1 \le t \le n/k$. The decoder knows the initial spine state $s_0 = 0$. It starts generating a *decoding tree* starting from s_0 as the root of the tree. Now, the spine value s_1 could be one of 2^k possibilities: $h(s_0, M_1)$ for each $M_1 \in \{0,1\}^k$. Generate 2^k children of the root and associate with each one a distinct spine value from these 2^k values. Let $x_{1,1}(s_1)$ be the symbol the encoder would have generated from s_1. To each such node of the tree with spine value s_1, upon receiving the first symbol $y_{1,1}$, associate a *cost* equal to $\|y_{1,1} - x_{1,1}(s_1)\|^2$.

Now, recursively grow this tree, with each node having 2^k children and associate a cost to each edge in the tree upon receiving new symbols, as follows. Given a tree up to level $t-1$, for a node at level $t-1$ with associated spine value s_{t-1}, the 2^k children have associated spine values $h(s_{t-1}, M_t)$ for $M_t \in \{0,1\}^k$. Let $x_{t,1}(s_t)$ be the associated constellation points (symbols). Upon receiving the t^{th} symbol in the first pass, $y_{t,1}$, associate a cost of $\|y_{t,1} - x_{t,1}(s_t)\|^2$ to the edge connecting the new node to its predecessor in the tree. In this way, the tree is expanded all the way up to depth n/k with each node having degree 2^k. Thus the entire tree has $2^{(n/k)\times k} = 2^n$ distinct leaf nodes.

The path from the root to the leaf node with the smallest cost gives the most likely message. For the BSC channel, each

edge cost should be replaced with the Hamming distance of the received bits and the bits obtained from the spine value produced by hashing the previous spine value (node) and the k bits corresponding to the edge in the tree.

If the receiver has not successfully decoded the message after the first pass, the sender moves on to sending the next pass. The sender continues to send successive passes until the receiver determines that the message has been decoded correctly, using a CRC at the end of each pass, for example. To decode in pass L, the decoder stores all previously received symbols and computes the cost on each edge of the tree as $\sum_{i=1}^{L} \|y_{t,i} - x_{t,i}(s_t)\|^2$ (or, for BSC, the sum of the corresponding Hamming distances).

Practical graceful scale-down decoder. Probabilistically we expect that most messages will have a much higher cost compared to the message with maximum likelihood (at least when we have enough information). This suggests that it is reasonable to maintain information about a small number of messages that are likely to have smaller overall cost at each level of the tree. A simple greedy heuristic, which at each stage of the tree maintains only a small number of nodes, enables a practical decoder to be built for spinal codes. For each level of the decoding tree, the practical decoder maintains no more than B nodes. When it receives the next symbol, it temporarily expands each node to $B \cdot 2^k$ possible nodes, calculates the cumulative path cost to each of these temporary nodes, and then maintains only the B lowest-cost ones (breaking ties arbitrarily). As B grows, the rate achieved by the decoder gets closer to capacity. Interestingly, we show in the next section that even small values of B achieve high rates close to capacity. The complexity of this practical decoder is linear in the message length, exhibiting graceful scale-down (the complexity is exponential in k, but k is a small constant, ≤ 8 in practice).

4. Some Properties of Spinal Codes

Nonlinearity and large state space. Unlike most existing practical codes such as convolutional, turbo, Raptor, etc., spinal codes are nonlinear. That is, the output symbols (coded bits) are not obtained as linear combinations of input message bits, and the sum of any two codewords is not necessarily a valid codeword. There are two important benefits of using nonlinear hash functions. First, the moment two messages differ in 1 bit, their output coded sequences have a large difference, making it easier for the decoder to distinguish between them. Second, they allow good coded sequences to be generated without requiring complicated encoding operations such as the multiplication of message bits by a random matrix or requiring the use of complex graph structures; hash functions achieve exactly this goal, and we benefit from the wealth of research and practice in developing good hash functions (which are often developed for more stringent adversarial settings).

THEOREM 1 (AWGN CHANNEL PERFORMANCE). *Let* $\mathsf{C}_{\mathsf{awgn}}(\mathsf{SNR})$ *be the AWGN channel capacity per channel use, then for any number of passes* L *such that*

$$L \cdot \left[\mathsf{C}_{\mathsf{awgn}}(\mathsf{SNR}) - \frac{1}{2}\log(\frac{\pi e}{6})\right] > k$$

the BER goes to 0 *as* n *goes to* ∞.

This theorem states that in our rateless setting, the decoder can start decoding correctly (BER ≈ 0), as long as the number of passes made by the encoder is just sufficient enough. This is smaller by a constant $\Delta \equiv \frac{1}{2}\log\left(\frac{\pi e}{6}\right) \approx 0.25$ for all SNR values compared to the capacity established by Shannon. Δ is a small fraction of the capacity for large SNR. For example, for $\mathsf{SNR} = 30$ dB, the capacity in two dimensions is roughly 10 bits/s/Hz and hence our code achieves approximately 97.5% of the Shannon capacity. This guarantee may not be tight; we believe that this loss is in part inherent to the linear constellation mapping (unlike a Gaussian mapping) and in part a limitation of our proof. For example, at low SNR, we achieve much higher rates in practice than predicted by the above result, as shown in the next section. Further, our result is stronger than what is stated above because for any value of n when BER is not strictly 0, the erroneous bits are always in the last few bits, a property that we can use in practice by adding some known trailing bits to each coded message.

THEOREM 2 (BSC PERFORMANCE). *Let* $\mathsf{C}_{\mathsf{bsc}}(p)$ *be the BSC channel capacity, then for any number of passes* L *such that* $L \cdot \mathsf{C}_{\mathsf{bsc}}(p) > k$, *the BER goes to* 0 *as* n *goes to* ∞.

That is, the spinal code achieves capacity for BSC with an ML decoder.

5. Simulation Results

We now present some simulation results using the practical spinal code decoder over an AWGN channel and compare the performance of the code with the Shannon capacity over a range of SNR values. We also compare its performance to various fixed-rate LDPC codes, which are known to have good performance at certain SNR values. As we shall see, the spinal code performs near optimally even for small block lengths with constrained computational power (small B) and outperforms LDPC codes even under fixed-SNR conditions.

Figure 2 shows the rate achieved by the practical decoder with $B = 16$, message length $m = 24$, $k = 8$, and $c = 10$ over an AWGN channel with SNR values between -10 dB to 40 dB. To simulate quantization of an ADC, the receiver quantizes each dimension to 14 bits. In these experiments we assume that the receiver informs the sender as soon as it is able to fully decode the data; this allows us to isolate the evaluation of the performance of spinal codes. Indeed, an eventual system using spinal codes (or for that matter any rateless code) ought to use a feedback protocol to achieve the best possible trade-off between throughput and latency.

In the figure, the x-axis shows the SNR (dB), and the y-axis shows the rate achieved in bits/symbol. The top most curve shows the Shannon capacity bound. Somewhat surprisingly, spinal codes achieve rates very close to the Shannon capacity

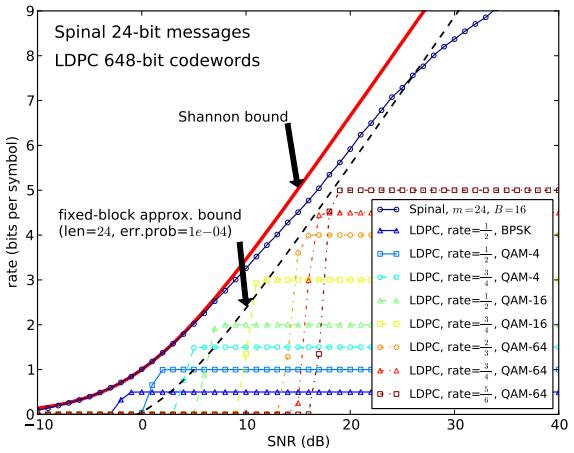

Figure 2: Rates achieved by the spinal code for different SNR with message length $m = 24$, $B = 16$, $k = 8$. "Shannon bound" is Shannon capacity; approximate bound is from [12].

for the entire range of the SNR, getting extremely close to Shannon's limitation for small SNR values. At high SNR, rates higher than $k = 8$ are achieved using puncturing.

The figure compares the spinal code with LDPC codes from the high-throughput mode of 802.11n with 648-bit codewords, decoded with a powerful decoder (40-iteration belief propagation decoder using soft information). As can be seen from the figure, the spinal code outperforms LDPC codes, which are good fixed-rate block codes, across all SNRs. At low SNR values, the benefits are especially large.

Finally, the spinal code yields, for a wide SNR range, higher rates than the fundamental upper bound on the performance of rated codes with block length 24, and error probability below 10^{-4} [12] (the dashed line). That is, the rateless nature of spinal code allows it to outperform *any* rated code of block length 24 for all SNR ≤ 25 dB (we have similar results for other block lengths, but the SNR thresholds differ with length). Note that in this picture, the LDPC codes have a block size > 24 and are therefore not subject to the bound, but an LDPC code over a block length of 24 would be lower than the bound shown by the dashed line.

6. Conclusion and Future Work

This paper introduced *spinal codes*, a new family of rateless codes that achieves near-optimal performance across a wide range of SNR values (-10 dB through 40 dB), as exhibited through simulations. They are the first class of rateless codes that have an efficient encoder, essentially achieve capacity for both AWGN and BSC channel (with ML decoding), and exhibit the graceful scale-down property enabling practical decoding at high rates. Spinal codes are practical because they take advantage of both the structural properties and the low cost provided by hash functions.

We conclude by mentioning several next steps that need to be taken to further demonstrate the utility of spinal codes. First, investigating different constellation mappings other than the linear one; for example, a Gaussian mapping is likely to improve performance. Second, developing a feedback link-layer protocol for rateless spinal codes. Third, developing the ability to decode multiple concurrent transmissions coded with spinal codes, and a suitable MAC protocol. Fourth, the simulation results strongly suggest that one can prove that a polynomial-time decoder can essentially achieve capacity; proving that property would be a significant additional contribution, and will likely entail a slightly different decoding algorithm. Fifth, a detailed comparison with other rated and rateless schemes that have been developed. And last but not least, implementing the encoder and decoder in a system to demonstrate reliable, high-speed operation.

Acknowledgments

We thank Peter Iannucci for several insightful remarks, and Tom Richardson and Lizhong Zheng for helpful comments. Jonathan Perry's work was supported by the Irwin and Joan Jacobs Presidential Fellowship and by the Claude E. Shannon Research Assistantship.

References

[1] R. Barron, C. Lo, and J. Shapiro. Global design methods for raptor codes using binary and higher-order modulations. In *IEEE MILCOM*, 2009.

[2] J. Bicket. Bit-Rate Selection in Wireless Networks. Master's thesis, Massachusetts Institute of Technology, Feb. 2005.

[3] U. Erez, M. Trott, and G. Wornell. Coding for Gaussian Channels. In *ISIT 05-06, journal version on Arxiv*, 2007.

[4] A. Gudipati and S. Katti. Automatic rate adaptation. In *Hotnets*, 2010.

[5] A. Gudipati and S. Katti. Strider: Automatic rate adaptation and collision handling. In *SIGCOMM*, 2011.

[6] J. Ha, J. Kim, and S. McLaughlin. Rate-compatible puncturing of low-density parity-check codes. *IEEE Trans. on Info. Theory*, 2004.

[7] G. Holland, N. Vaidya, and P. Bahl. A Rate-Adaptive MAC Protocol for Multihop Wireless Networks. In *MobiCom*, 2001.

[8] G. Judd, X. Wang, and P. Steenkiste. Efficient Channel-aware Rate Adaptation in Dynamic Environments. In *MobiSys*, June 2008.

[9] J. Li and K. Narayanan. Rate-compatible low density parity check codes for capacity-approaching ARQ scheme in packet data communications. In *Int. Conf. on Comm., Internet, and Info. Tech.*, 2002.

[10] M. Luby. LT codes. In *FOCS*, 2003.

[11] R. Mantha and F. Kschischang. A capacity-approaching hybrid ARQ scheme using turbo codes. In *GLOBECOM*, 1999.

[12] Y. Polyanskiy, H. Poor, and S. Verdú. Channel coding rate in the finite blocklength regime. *IEEE Trans. on Info. Theory*, 56(5), 2010.

[13] S. Sen, N. Santhapuri, R. Choudhury, and S. Nelakuditi. AccuRate: Constellation-based rate estimation in wireless networks. *NSDI*, 2010.

[14] S. Sesia, G. Caire, and G. Vivier. Incremental redundancy hybrid ARQ schemes based on low-density parity-check codes. *IEEE Trans. on Comm.*, 52(8):1311–1321, 2004.

[15] A. Shokrollahi. Raptor codes. *IEEE Trans. Info. Theory*, 52(6), 2006.

[16] E. Soljanin, N. Varnica, and P. Whiting. Incremental redundancy hybrid ARQ with LDPC and Raptor codes. *IEEE Trans. on Info. Theory*, 2005.

[17] G. Ungerboeck. Channel coding with multilevel/phase signals. *IEEE Trans. on Info. Theory*, IT-28(1):55–67, Jan. 1982.

[18] G. Ungerboeck and I. Csajka. On improving data-link performance by increasing the channel alphabet and introducing sequence coding. In *ISIT*, 1976.

[19] M. Vutukuru, H. Balakrishnan, and K. Jamieson. Cross-Layer Wireless Bit Rate Adaptation. In *SIGCOMM*, 2009.

[20] S. Wong, H. Yang, S. Lu, and V. Bharghavan. Robust Rate Adaptation for 802.11 Wireless Networks. In *MobiCom*, 2006.