Rateless Codes: *Spinal Codes*

CS M038/GZ06: Mobile and Cloud Computing
Kyle Jamieson

Department of Computer Science
University College London
• Walking speed wireless trace measuring SNR

• Fades last just tens of milliseconds

• Fades are large in magnitude (> 20 dB difference)
Channel *coherence time*

- The sender transmits the wireless signal at carrier frequency $f = c / \lambda$
  - Speed of light: $c$; Wavelength of the signal: $\lambda$

- So, a path length difference of $\lambda/2$ moves the combination of the two signals between *constructive* and *destructive* interference
  - Receiver movement of $\lambda/4$: *coherence distance*
    - Time it takes to move a coherence distance: *coherence time*
      - Walking speed @ 2.4 GHz: $\approx$ 15 milliseconds
Current approach: Modulation adaptation

Metric: Signal-to-noise power ratio (SNR)
Current approach: Coding adaptation

Current approaches adapt a fixed code rate: \( R = \frac{k}{n} \)

Lower code rate more parity bits more redundancy
## 802.11: adapt code rate, modulation

<table>
<thead>
<tr>
<th>Bit-rate</th>
<th>802.11 Standards</th>
<th>DSSS or OFDM</th>
<th>Modulation</th>
<th>Bits per Symbol</th>
<th>Coding Rate</th>
<th>Mega-Symbols per second</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>b</td>
<td>DSSS</td>
<td>BPSK</td>
<td>1</td>
<td>1/11</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td>DSSS</td>
<td>QPSK</td>
<td>2</td>
<td>1/11</td>
<td>11</td>
</tr>
<tr>
<td>5.5</td>
<td>b</td>
<td>DSSS</td>
<td>CCK</td>
<td>1</td>
<td>4/8</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>b</td>
<td>DSSS</td>
<td>CCK</td>
<td>2</td>
<td>4/8</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>a/g</td>
<td>OFDM</td>
<td>BPSK</td>
<td>1</td>
<td>1/2</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>a/g</td>
<td>OFDM</td>
<td>BPSK</td>
<td>1</td>
<td>3/4</td>
<td>12</td>
</tr>
<tr>
<td>12</td>
<td>a/g</td>
<td>OFDM</td>
<td>QPSK</td>
<td>2</td>
<td>1/2</td>
<td>12</td>
</tr>
<tr>
<td>18</td>
<td>a/g</td>
<td>OFDM</td>
<td>QPSK</td>
<td>2</td>
<td>3/4</td>
<td>12</td>
</tr>
<tr>
<td>24</td>
<td>a/g</td>
<td>OFDM</td>
<td>QAM-16</td>
<td>4</td>
<td>1/2</td>
<td>12</td>
</tr>
<tr>
<td>36</td>
<td>a/g</td>
<td>OFDM</td>
<td>QAM-16</td>
<td>4</td>
<td>3/4</td>
<td>12</td>
</tr>
<tr>
<td>48</td>
<td>a/g</td>
<td>OFDM</td>
<td>QAM-64</td>
<td>6</td>
<td>2/3</td>
<td>12</td>
</tr>
<tr>
<td>54</td>
<td>a/g</td>
<td>OFDM</td>
<td>QAM-64</td>
<td>6</td>
<td>3/4</td>
<td>12</td>
</tr>
</tbody>
</table>
Fixed-rate codes require channel adaptation.

Figure 1-4: Theoretical throughput in megabits per second using packets versus signal-to-noise ratio for several modulations, assuming AGWN and a symbol rate of 1 megasymbol per second.

This equation assumes the transmitter sends packets back-to-back, the receiver knows the location of each packet boundary, the receiver can determine the integrity of the data with no overhead, there is no error correction, and the symbol rate is 1 megasymbol per second.

Packetets change the throughput versus S/N graph dramatically; Figure 1-4 shows throughput in megabits per second versus S/N for 1500-byte packets after accounting for packet losses caused by bit-errors. The range where each modulation delivers non-zero throughput but suffers from loss is much smaller in Figure 1-4 than in Figure 1-3.

For most S/N values in the range from 5 to 30 dB, the best bit-rate delivers packets without loss. Bit-rate selection is easier for links that behave as in Figure 1-4 than as in Figure 1-3; the sender can start on the highest bit-rate and switch to another bit-rate whenever the
Current approach: “Adapt and discard”

Throughput = delivery rate \times \text{bitrate} = (1 - \text{BER})^L \times \text{bitrate}

- **Network layer**
- **Frame checksum:** discard errored frames
- **Link layer**
  - Link layer: Request retransmissions
- **Physical layer**
  - Physical layer: Adapt for a low ($\sim 10^{-5}$) bit error rate (BER)
    - Adapt modulation (feedback loop)
    - Adapt fixed coding rate (feedback loop)
Existing rate adaptation algorithms

- **Frame-based**
  - Data
  - ACK
  - Estimate frame loss rate at each bit rate

- **SNR/BER-based**
  - Data
  - SNR using preamble
  - Lookup table: SNR/BER $\rightarrow$ best rate

### Algorithms
- **RRAA**, Wong et al., 2006.
- **SampleRate**, Bicket, 2005.
- **ARF, ONOE**
- **CHARM**, Judd et al., 2008.
- **SoftRate**, Vutukuru et al., 2009
Adaptation strategy depends on coherence time

- **Walking speed (3 km/h):** coherence time ≈ 25 ms
- **Slow driving speed (30 km/h):** coherence time ≈ 2.5 ms
- **Motorway speed (120 km/h):** coherence time ≈ 0.6 ms

Reactive in nature  
Waste capacity  
Choose average: some waste

(Carrier frequency = 1900 MHz)
Today: Rateless codes

- Idea: Sender transmits at a rate higher than the channel can sustain

- Receiver extracts information at the rate the channel can sustain at that instant
  - No adaptation loop is needed!

![Graph showing SINR over time with conservative prediction and lag](image)
Spinal codes: Introduction

• Hash *chunks* of the original message: random mapping between message bits and hashed bits

• *Constellation mapping function* takes hashed bits to constellation points (or can use standard constellations we saw earlier)

• Key design issue: An efficient and practical decoder

• Decoder replays encoder in a clever way
Spinal encoder: Computing the spines

Message $M$

\[ M_1 \text{ (} k \text{ bits)} \rightarrow h \rightarrow s_1 \rightarrow h \rightarrow s_2 \rightarrow h \rightarrow s_3 \rightarrow \ldots \]

- $M_1$, $M_2$, $M_3$, etc. are $k$-bit inputs to the hash function $h$.
- $s_0$, $s_1$, $s_2$, $s_3$, etc. are spine values generated from the hash function.

The key idea is to replace the scale-down property for spinal codes.
The full spinal encoder

Message $M$ 
(n bits)

1
$M_1$ (k bits)

$k + 1$
$M_2$ (k bits)

$2k + 1$
$M_3$ (k bits)

... 

$s_0$ $h$ $s_1$ $h$ $s_2$ $h$ $s_3$

$x_{1,1}$ $x_{1,2}$ $x_{1,3}$

$x_{2,1}$ $x_{2,2}$ $x_{2,3}$

$x_{3,1}$ $x_{3,2}$ $x_{3,3}$

... 

Pass 1

Pass 2

Pass 3
What’s the best shape for a constellation?

- Start with a square constellation (a)
  - Recall, distance of each symbol from origin determines power
  - So, a circle traces constant power points

- Maintaining spacing between constellation points, move points from outside the circle to inside, shaping the constellation (b)
  - This is shaping gain: we maintain error probability, hence throughput, but reduce the average signal power
  - Now we can add more constellation points (c) to restore average power to as it was before. This increases throughput!
An ideal decoder

• Consider all 2^n possible messages that could have been sent. Pick the message that minimizes the distance between:
  1. \( x_{t,l} \): \( t \)th constellation point sent in the \( l \)th pass
  2. \( y_{t,l} \): \( t \)th constellation point received in the \( l \)th pass

• At the receiver, make a vector \( y \) of all received symbols
  — The decoder that minimizes probability of error (maximum likelihood or ML decoder) is then:

\[
\hat{M} = \arg \min_{M' \in \{0,1\}^n} \left\| y - x(M') \right\|^2
\]

\[
= \arg \min_{M' \in \{0,1\}^n} \sum_{all \ t,l} \left| y_{t,l} - x_{t,l}(M') \right|^2
\]

✗ This decoder needs to compute 2^n separate sums (intractable)
The key idea is to replace the scale-down property for spinal codes. Fortunately, it is possible to construct an ML decoder with resources, it is not obvious how to adapt this standard method. In our experiments, we actually obtain rates higher than the maximum rate achievable by the code, while the maximum rate achievable by the code grows linearly with $\frac{1}{k}$, where the transmitter does not have to be a small constant; for example, over the default 802.11b/g bandwidth of 20 MHz, if $n$ bits are transmitted successfully after a single pass, meaning that the estimated message $\hat{y}$ from the ideal ML decoder and reducing the resources constituted by the hash function to replay the encoder at the decoder for spinal codes can take advantage of the structure property, which captures the intuition that we would like the decoder is implementable by starting from the root and associate with each one a distinct spine value from these possibilities. When the decoder has much smaller computational capabilities. When the decoder has much smaller computational capabilities. When the decoder has much smaller computational capabilities.
ML decoder tree (2)

The key idea is to generate the spinal encoder at the decoder using the shared knowledge of the hash function to replace the ideal ML decoder for spinal codes. Fortunately, it is possible to construct an ML decoder with modest computational capabilities. When the decoder has much smaller computational resources, it is not obvious how to adapt this standard method. In this section, we describe a maximum likelihood (ML) decoder for spinal codes. The ML decoder achieves capacity over the BSC channel and nearly achieves capacity over AWGN for spinal codes.

The description of spinal codes has only one parameter, $s_0$, which is the spine value that the encoder would like to compute. Let $h$ be the associated constellation, and $x_i$ be the symbol the encoder would like to compute. Let $S_{i-1}$ be the set of $2^k$ possible values that $x_i$ can take on. The key idea is to replace the ideal ML decoder with a tree that is expanded all the way up to depth $L$. The tree is expanded all the way up to depth $L$ by connecting the new node to its predecessor in the tree.

In our experiments, we actually obtain rates higher than the BSC channel and nearly achieve capacity over AWGN for spinal codes. In this section, we describe a maximum likelihood (ML) decoder for spinal codes. The ML decoder achieves capacity over the BSC channel and nearly achieves capacity over AWGN for spinal codes.

In this section, we describe a maximum likelihood (ML) decoder for spinal codes. The ML decoder achieves capacity over the BSC channel and nearly achieves capacity over AWGN for spinal codes.
Decoding spinal codes: “Scale-down” decoder \((k = 1, B = 4)\)

**1.2 Overview**

**Introduction**

Figure 2: Practical Decoder Tree

Figure 2 shows an example of such a pruned tree. A message is represented by a continuous path from the root node to one of the leaves, and hence there can only be \(B\) candidate messages. The decoder estimates the transmitted message as the one that has minimum total cost among the \(B\) messages corresponding to the leaf nodes at the level \(n = k\) of this pruned tree.

**1.2.3 Puncturing Schedule**

An important part of the actual implementation of this system is the puncturing schedule. This is agreed-upon by the sender and receiver a priori. In the system, as described above, the receiver obtains symbols corresponding to an integral number of passes. The puncturing schedule allows a much more fine-grained control over the achievable rates.

In the first pass, the sender transmits a symbol for every value in the sequence. In any subsequent pass, the sender can choose to transmit only every \(g\)-th value, where \(g\) may change from pass to pass. For example, one option is to use the schedule (8,4,2,1).

That is, the first punctured pass sends every 8th value, the second pass sends every 4th value but only if it is not a multiple of 8, the third pass sends every second value but...
Spinal codes come close to Shannon capacity over an AWGN channel

![Graph showing Spinal 24-bit messages and LDPC 648-bit codewords]

- Spinal 24-bit messages
- LDPC 648-bit codewords

Shannon bound

Fixed-block approx. bound (len=24, err.prob=1e-04)
Spinal Codes: Future work

- Non-uniform constellations (Gaussian)
- Feedback link-layer protocol for rateless spinal codes
- Decoding multiple concurrent transmissions coded with Spinal codes
- Proving that polynomial-time decoder can achieve capacity
- Comparing with other rateless schemes (LT, extensions of LT: Raptor, Irregular Repeat-Accumulate, etc.)
- Implementation in a real-time system (decoder speed)