ECE 5325/6325: Wireless Communication Systems
Lecture Notes, Spring 2013

Lecture 18

Today: (1) da Silva Discussion, (2) Error Correction Coding, (3) Error Detection (CRC)

- HW 8 due Tue. HW 9 (on Lectures 18, 19) is due Tue April 2. HW 9 is the last material for Exam 3 (April 4).
- Reading — Today: Molisch 14.2. Tue: Molisch 14.1.

1 Forward Error Correction Coding

The material for this section comes largely from Jeff Frolik's MUSE channel coding video, available at:

- `http://www.uvm.edu/~muse/CTA.html`

Def'n: *Forward error correction coding or channel coding*
Adding redundancy to our data at the transmitter with the purpose of detecting and correcting errors at the receiver.

The transmitter takes in data bits and puts out coded bits. Our notation is that for each k data bits input to the FEC operator, the FEC operation will produce $n > k$ coded bits out.

1.1 Block vs. Convolutional Coding

Def'n: (k, n) *Block Code*
A (k, n) block code inputs k-bits which are accumulated (via serial-to-parallel conversion) in a k-length vector $\mathbf{d}$. Block encoding multiplies $\mathbf{d}$ by a $k \times n$ generator matrix, G, to output a n-length bit vector $\mathbf{c}$. Block decoding then multiplies the received vector $\mathbf{r}$ by the syndrome matrix S to determine if any errors occurred and determine which (if any) bits were in error out of the n sent.

The syndrome is just a rearrangement of the transpose of the generator matrix, as shown by example below.

In contrast, a convolutional code is a "running" code. For encoding, bits are input into what is effectively a binary filter, the output bits are dependent on the current and past bits.

Compare the advantages and disadvantages:

- Block code: Advantages: Better for data that is not coming in large streams (bursty data sources, <1000 bits), *e.g.*, wireless sensor networks. Pretty simple computation. Not the best one can do in terms of improving energy efficiency / removing errors. Block codes are used in CDs, DVDs, and disk drives.
- Convolutional codes: Advantages: Best for very large data streams. More energy efficient than block codes when you have large streams of data. Convolutional codes are used in: deep space communication (Voyager program), satellite and terrestrial digital video broadcasting. Disadvantages: Computational complexity increases exponentially in the length of the code. Andrew Viterbi (founder of Qualcomm) is credited with the optimal decoder, called the Viterbi algorithm.

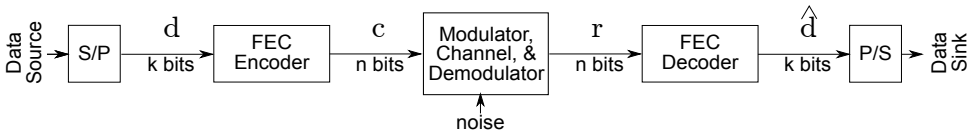

Figure 1: Block diagram of a communication system emphasizing the forward error correction (FEC) "block coding" encoder and decoder.

1.2 Block Code Implementation

Let the input be denoted $\mathbf{d}$, a k-bit vector. Let the output be $\mathbf{c}$, a n-bit vector. Let G be the generator matrix. Then

$$\mathbf{c} = \mathbf{d}G$$

Thus the G matrix has size $k \times n$. This operation is done modulo-2. That is, multiply all of the pairs, sum them, and then take the mod 2 of the result. That is, if the sum of the products is even, the answer is 0, if the sum is odd, the answer is 1.

Def'n: *Systematic*
The first k bits of the n bits output, are the same as the k bits in $\mathbf{d}$.

Example: (6, 3) systematic block code which can correct one bit error
Let G be given by:

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Encode the data bits $\mathbf{d} = [1, 1, 1]$.

Solution: $\mathbf{c} = [1, 1, 1, 0, 0, 0]$

Example: Reception
You receive $\mathbf{r} = [1, 1, 1, 0, 0, 1]$, that is, what you received has an error in the last bit compared to $\mathbf{c}$ (the coded bits that were sent through the channel). What was is the block decoder's estimate of the transmitted data?

Solution: At the receiver, multiply by the syndrome

$$S = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Compute: $\mathbf{r}S = [0, 0, 1]$.

Look at all of the rows of the syndrome. The row number of the syndrome S that matches the output $\mathbf{r}S$, is the same as the number of the bit that was in error. If $\mathbf{r}S$ is all zeros, that indicates that there were no errors. Since the sixth bit was in error, instead of $[1, 1, 1, 0, 0, 1]$, we know the correct coded bits were $[1, 1, 1, 0, 0, 0]$.

Finally, because it is a systematic code, we know the first three bits are the data bits. The receiver will just drop the last three bits.

Example: (7, 4) Block Code

1. Encode $\mathbf{d} = [0, 1, 1, 0]$ with the $(7, 4)$ block code with generator,

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

2. If $\mathbf{r} = [0, 1, 1, 0, 1, 1, 1]$ is received, and S is given as below, what would the receiver determine to be the demodulated bits?

$$S = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3. If $\mathbf{r} = [0, 0, 0, 1, 1, 1, 0]$ is received, what would the receiver determine to be the demodulated bits?

4. If $\mathbf{r} = [1, 0, 0, 1, 1, 1, 0]$ is received, what would the receiver determine to be the demodulated bits?

5. If $\mathbf{r} = [1, 1, 0, 1, 1, 1, 0]$ is received, what would the receiver determine to be the demodulated bits?

Solution: (1) I get $\mathbf{c} = [0, 1, 1, 0, 1, 1, 0]$. (2) Then, multiplying $[0, 1, 1, 0, 1, 1, 1]S$, I get $[0, 0, 1]$, which is the same as the 7th row, which says that the last row was incorrectly received, and so the 7th bit was incorrect. Thus the correct four bits sent were $[0, 1, 1, 0]$. (3) I get $\mathbf{r}S = [0, 0, 0]$ which means no bits were received in error, so the four data bits sent were $[0, 0, 0, 1]$. (4) I get $\mathbf{r}S = [1, 1, 1]$ which means that the first bit was received in error, so the four data bits sent were $[0, 0, 0, 1]$. (5) I get $\mathbf{r}S = [1, 0, 0]$ which means that the receiver thinks the fifth bit was received in error, so the receiver would guess the four data bits were $[1, 1, 0, 1]$.

1.3 Performance and Costs

Using a (7,4) block code, we can correct a single error in the 7 bits. But we need to increase the number of bits to send, which then requires more energy. So when using channel coding, we reduce the transmit power when comparing uncoded and coded transmissions. Still, the probability of bit error goes down for equal $\frac{\mathcal{E}_b}{N_0}$(dB). Equivalently, we can achieve the same bit error rate at 1 dB lower $\frac{\mathcal{E}_b}{N_0}$. This value, 1 dB, is the *coding gain*. In our link budgets, coding goes in the Gains column, added in with the antenna gains.

However, coding requires sending additional bits. So, in a coded system, there is always a ratio of data bits to coded bits, r, called the *code rate*. In the (7,4) block code it is $r = 4$ data bits /7 coded bits. For a fixed bandwidth, this reduces the achievable data rate by r. For a fixed data rate, it increases the bandwidth by a factor of $1/r$.

2 Error Detection via CRC

Besides trying to correct the data, we can also simply try to detect the error. (One or both error correction and error detection can be done.) If the check fails, we might request the data again. For error detection, we have the following packet structure:

1. Header: Information about who the data is meant for, what purpose data is being sent, etc.

2. Payload: Data that we need to communicate. May be of various lengths.

3. Footer: Where the CRC is included. The CRC is always the same length, regardless of the payload length.

The TX calculates the cyclic redundancy check (CRC) from the header and payload and puts it in the footer. The RX then, once it has all of the header and footer, calculates the CRC based on the received header

and footer, and compares it to the one it received. If they match, the RX decides the data was received correctly.

2.1 Generation of the CRC

A y-bit CRC is described by a y-order polynomial. The y refers to the maximum exponent in the CRC polynomial. For example, a 1st order CRC is $c(x) = x+1$. The data we want to send, $\mathbf{d}$, can also be described as a polynomial. For example $\mathbf{d} = [0, 1, 0, 1]$ corresponds to the polynomial $d(x) = 0x^3 + 1x^2 + 0x + 1x^0 = x^2 + 1$. To determine the polynomial, just multiply the (row) bit vector with the column vector $[x^3, x^2, x^1, 1]^T$, or, in general, for n-length data string, multiply by $[x^{n-1}, x^n, \ldots, 1]^T$.

To find the CRC bit, we divide the two polynomials, $d(x) \div c(x)$ (modulo-2) and the CRC bit is the remainder. So in fact the CRC is $d(x) \mod c(x)$. In this example, the CRC is zero.

Then, what we send through the channel is the desired bits appended by the CRC bit, in this example, $[0, 1, 0, 1, 0]$.

Example: CRC for 5 bit payload
Let $\mathbf{d} = [1, 1, 0, 1, 1]$. What is the CRC bit for $c(x) = x + 1$?
Solution: When dividing $d(x) = x^4 + x^3 + x + 1$ by $c(x) = x + 1$, you get a remainder of 0. So 0 should be appended.

For the 1-bit CRC check, there is also a much easier way – to determine the 1-bit CRC, just check if the data (including CRC bit) has an even number of 1s. The one-bit CRC is also called the parity check. Next, we consider a 4th order (four bit) CRC.

Example: CRC calculation with a 4-bit CRC
Let $c(x) = x^4 + x + 1$. Let $d(x) = x^6 + x^3 + x^2 + x + 1$. Because the $c(x)$ polynomial is 4th order, there are four CRC bits. Calculate the CRC bits in this case.
Solution: The solution is $[0, 0, 0, 1]$. See my written solutions on the last page.

2.2 Performance and Costs

Using a CRC, we add a few additional bits to the packet, fewer than a FEC code would add. This allows us to detect an error, but not correct it. However, it is often important to ensure that the data received did not have any errors, beyond any reasonable doubt. For example, for cell phone audio, it may be okay to make a bit error, and have it cause some temporary noise in the audio signal; but when transferring a file, it may be critical not to have the file show up with incorrect data, and not know that it was corrupted. A y-bit CRC will have a missed detection rate around 2^{-y}.

Most systems use both an FEC and CRC to increase the energy efficiency (with the FEC) and to make sure that received data is correct

(with the CRC).