ECE461: Digital Communications

Lecture 7: Reliable Communication with Erasures

Introduction

So far, we have seen that arbitrarily reliable communication is possible at non-zero rates provided the receiver is well designed. In this lecture we will take a closer look at simplifying (in a computational sense) the complexity of the receiver design. We break up the receiver into two steps:

1. *Demodulation*: Map the analog received voltage to a finite number of discrete levels. To be concrete, we focus on the following situation: let the transmit voltages be binary (just as in the previous lecture). Then we map the analog voltage into one of *three* possible levels. Two of them correspond to the two levels of the binary modulation at the transmitter. The third, called an *erasure*, models the scenario when the received analog voltage is not enough to make a decision one way or the other.

2. *Decoding*: The second step involves taking the erasures into careful account and recovering the original information bits.

Receiver Design in Two Steps

For concreteness, the discussion in this lecture is limited to *binary* modulation on the AWGN channel:

$$y[m] = x[m] + w[m], \quad m = 1 \dots T; \tag{1}$$

i.e., the transmit voltage is restricted to be $\pm\sqrt{E}$. Further, the transmitter is assumed to be broken up into the two steps described in the previous lecture (sequential modulation and linear coding). For concreteness let us suppose that $-\sqrt{E}$ is transmitted when the corresponding coded bit is 0 and $\sqrt{E}$ is transmitted when the corresponding to when the coded bit is 1.

The ML receiver (from the previous lecture) took the T received voltages and mapped them directly to the information bits. While this is the optimal design, it is also prohibitively expensive from a computational view point. Consider the following simpler two-stage design of the receiver.

1. *Demodulation*: At each time m the received voltage $y[m]$ is mapped into one of three possible choices: Let us fix $c \in (0, 1)$.

   (a) If
   $$y[m] \leq -c\sqrt{E} \tag{2}$$
   then we map into a 0.

   (b) If
   $$y[m] > c\sqrt{E} \tag{3}$$
   then we map into a 1.

Figure 1: Demodulation Operation.

(c) In the intermediate range, i.e.,

$$-c\sqrt{E} \le y[m] \le c\sqrt{E} \tag{4}$$

we map into an "e" (standing for an erasure).

The process is described in Figure 1 and is a very easy step computationally.

2. *Decoding*: We now take the T outputs of the demodulator (one ternary symbol – 0, 1, e – at each time instant) and map them into the information bits.

In the following, we will study each of these two stages carefully and analyze the end-to-end performance.

Demodulation

The idea of making erasures is that when the received voltage is in the intermediate range, we are less sure of whether $+\sqrt{E}$ was transmitted or $-\sqrt{E}$ was transmitted. For instance, if we receive a voltage of 0, then the corresponding transmit voltage could equally likely be $\pm\sqrt{E}$. What is the probability of the demodulation event at any time m resulting in an erasure? It is simply equal to

$$p \overset{\text{def}}{=} \mathbb{P}\left[-c\sqrt{E} \le y[m] \le c\sqrt{E}\right] = Q\left((1-c)\sqrt{\mathsf{SNR}}\right) - Q\left((1+c)\sqrt{\mathsf{SNR}}\right). \tag{5}$$

Here we have denoted SNR to be the ratio of E to the variance of the additive Gaussian noise (σ^2).

Even with the unclear intermediate range marked by erasures, it is possible that a coded bit of 0 (corresponds to a transmit voltage of $-\sqrt{E}$) could result in a demodulated symbol of 1 (corresponds to a received voltage larger than $c\sqrt{E}$). The probability of this event

$$\mathbb{P}\left[w[m] > (1+c)\sqrt{E}\right] = Q\left((1+c)\sqrt{\mathsf{SNR}}\right), \tag{6}$$

gets smaller as c is made larger. We could decide to set c large enough so that this probability is made desirably small. In the following, we will suppose this is small enough that we can ignore it (by setting it to zero).[1]

Let us summarize the demodulation output events:

[1]It is important to keep in mind that no matter how small this probability is, the chance that at least one of such an undesirable event occurs in the transmission of a large packet grows to one as the packet size grows large. This is the same observation we have made earlier in the context of the limitation of sequential communication.

1. When the coded bit is a 0, the output of the demodulated step takes on one of two possible values:

$$\begin{array}{ll} 0 & \text{with probability } 1-p(\text{ cf. Equation (5))} \\ e & \text{with probability } p. \end{array} \tag{7}$$

2. Analogously, when the coded bit is a 1, the output of the demodulated step takes on one of two possible values:

$$\begin{array}{ll} 1 & \text{with probability } 1-p \\ e & \text{with probability } p. \end{array} \tag{8}$$

Decoding

We now have T demodulation outputs (each one of them being either 0, 1 or e). Whenever we get a 0 (1), we are confident that the corresponding code bit could only have been a 0 (1). This is based on our assumption that the chance that a coded bit of 0 will result in a demodulated symbol of 1 is very small; so small, that we have modeled it as zero. Thus the only time instants where the decoder has to do some work is in the erased locations. For T large, what fraction of the locations do we expect to be erasures? Since we are supposing that the coded bits are equally likely to be 0 or 1 and are statistically independent from time to time, the fraction of erasures is p. Indeed, by the *law of large numbers*,

$$\mathbb{P}\left[\left|\frac{\text{Number of erasures over time } T}{T} - p\right| > \delta\right] \to 0, \qquad \text{as } T \to \infty. \tag{9}$$

So, we have approximately $(1-p)T$ of the coded bits recovered correctly at the output of the demodulation step.

The job of the decoder is to use these as inputs and figure out the original information bits. At this point it is useful to take a close look at the coding operation at the transmitter that mapped the information bits into the coded bits (cf. Equation (1) from Lecture 6):

$$\begin{bmatrix} \text{vector of} \\ \text{coded bits} \end{bmatrix} = C \begin{bmatrix} \text{vector of} \\ \text{information} \\ \text{bits} \end{bmatrix}. \tag{10}$$

The size of the coding matrix C is $T \times RT$ where $R < 1$ is the coding rate. Now at the decoder we have available a fraction $(1-p)T$ number of the coded bits with a fair degree of certainty.[2] Thus we can rewrite Equation (10) as

$$\begin{bmatrix} \text{vector of} \\ \text{demodulated} \\ \text{bits} \end{bmatrix} = \tilde{C} \begin{bmatrix} \text{vector of} \\ \text{information} \\ \text{bits} \end{bmatrix}. \tag{11}$$

Here the matrix $\tilde{C}$, of dimension $(1-p)T \times RT$, is a *sub-matrix* of the original linear coding matrix C: it is formed by choosing $(1-p)T$ of the T rows of C (exactly which rows are

[2]Exactly which fraction of the coded bits are available is unclear; all is known is that a total of $(1-p)T$ coded bits are available.

chosen depend on which of the demodulated outputs were *not* erasures). Now, to recover the information bits from the linear set of equations in Equation (11), we need at least as many equations $(1-p)T$ as variables (RT). Further we need at least RT of these equations to be linearly independent. Putting these conditions into mathematical language, we need:

- $R < 1 - p$.
- The matrix $\tilde{C}$ has *rank* at least RT.

The first condition is simply a constraint on the coding rate R. This is readily satisfied by choosing the data rate appropriately at the transmitter. The second condition says something about the linear coding matrix C. We need every subset of RT rows of the matrix C to be linearly independent.

How does one construct such linear codes? This has been the central focus of research for several decades and only recently could we say with some certainty that the final word has been said. The following is a quick summary of this fascinating research story.

1. Consider the random linear code (we studied this in the previous lecture as well): each entry of C is i.i.d. 0 or 1 with probability 0.5 each. It turns out that almost surely the random matrix C has the desired rank property. Thus it is easy to construct linear codes that work for the erasure channel(almost every random linear code works). This is a classical result:

   P. Elias, "Coding for two noisy channels," *Information Theory*, 3rd London Symposium, 1955, pp. 6176.

   The problem with this approach is the decoding complexity – which involves inverting a $(1-p)n \times (1-p)n$ matrix – is $O(n^3)$.

2. **Reed-Solomon Codes**: These are *structured* linear codes that guarantee the decodability condition. They have the additional property that their decoding complexity is smaller: $O(n^2)$. Reed-Solomon codes are used in several data storage applications: for example, hard disks and CDs. You can learn about these codes from *any* textbook on coding theory. Unfortunately one cannot get nonzero data rates from these structured codes.

3. **Digital Fountain Codes**: These are a new class of random linear codes that satisfy the decodability condition with very high probability. The distinguishing feature is that the matrix C is very sparse, i.e., most of its entries are 0. The key feature is a simple decoding algorithm that has complexity $O(n \log(n/\delta))$ with probability larger than $1-\delta$. For a wide class of channels, a sparse linear code admits a simple decoding algorithm, called the *message passing algorithm*. This is a very recent development that has revolutionized coding theory, both theoretically and from a practical point of view. Here are a few references for you to learn more about this exciting area. Anytime there is a such a significant breakthrough, you can expect some entrepreneurial activity surrounding it. The interested student might want to check out `http://www.digitalfountain.com`.

Looking Ahead

We have looked at erasures as an intermediate step to simplify the receiver design. This does not however, by itself, allow arbitrarily reliable communication since the total error probability is dominated by the chance that at least one of the coded bits is demodulated in error (i.e., not as an erasure). To get to arbitrarily reliable communication, we need to model this *cross-over* event as well: coded bits getting demodulated erroneously. A detailed study of such a model and its analysis is a bit beyond the scope of this course. So, while we skip this step, understanding the fundamental limit on the reliable rate of communication after such an analysis is still relevant. This will give us insight into the fundamental tradeoff between the resource of energy and performance (rate and reliability). This is the focus of the next lecture.