# Random Notes on CDMA

C. Rose

November 18, 2010

## 1 CDMA

The basic idea behind CDMA is to assign each user a signature (also called a codeword), $s_m(t)$. Users will use antipodal signaling (i.e., multiplying their signatures by $\pm 1$ to send data. Thus, what the receiver sees is (idealistically)

$$r(t) = \sum_m b_m s_m(t) + w(t)$$

where the $b_m$ are the users' bits (again, $\pm$). We also generally assume that the system is *interference limited* so that the noise term is inconsequential. That is, we assume

$$r(t) \approx \sum_m b_m s_m(t)$$

From the perspective of a receiver trying to decode, say, user 1 we have

$$r(t) = b_1 s_1(t) + \sum_{m \neq 1} b_m s_m(t)$$

and we do the usual matched filtering (correlator receiver) where we multiply $r(t)$ by $s_1(t)$ and integrate so that the decision paramater we use to guess what $b_1$ is looks like

$$r_1 = b_1 \mathcal{E}_{s_1} + \sum_{m \neq 1} b_m \int_0^T s_1(t) s_m(t) dt$$

where we've assmed the bit interval is $(0, T)$. What's not necessarily obvious is that we've ALSO assumed that all the users signals arrive at the receiver in lockstep (their individual bit intervals start and stop at the same time). This actually

takes a little bit of doing owing to the finite speed of light and that a distance of $1000$ feet introduces approximately a $1\mu s$ delay. Thus, for a CDMA system to be *synchronous* the system (base) must essentially tell users to advance or delay their transmissions so that all transmissions arrive at the base synchronously. This is not a big deal – TDMA systems do it all the time, but it's worth mentioning (which is obvious because I mentioned it I guess :) ).

*(Also please note that we're purusing a* baseband *development. These signals $s_m(t)$ are actually the product of a baseband signature and a carrier at some high frequency. But it's easier to see what's going on in baseband – and mathematically, since you already understand signal space, the two are completely equivalent approaches.)*

So, now the question is how are those signatures chosen. We could have some "signature authority" that carefully chose the signatures. Ideally, we'd like to have users interfere with each other as little as possible. One approach would be to make the $s_m(t)$ orthogonal (time division, frequency division or a combination of both called frequency hopping are very easy conceptually). One could also simply assign each user a basis function from some arbitrary orthogonal basis.

However, such optimal approaches make the mobile handset a bit more complicated – it has to be able to synthesize the requisite waveforms upon command. This is not a big deal these days, but in the olden times of 1990 when I became a professor, the idea was to make things as simple as possible from a hardware perspective. In addition, there are serpents in the garden of which we have not yet spoken – the radio channel can do some ugly things to you since radio waves bounce all over the place. We can certainly deal with these things via equalization, but it adds another layer of complexity to the hardware.

And one last thing that's not been mentioned yet – there are SYSTEMS issues as well. If you assign waveforms that are orthogonal, nearby bases have to know which ones you're using so that your transmissions don't interfere with theirs. So there's also a coordination problem when you try to get too anally rententive optimal. :)

Now all that said, what we're about to do **is theoretically no better than anything else you could do.** Of course, that did not stop the raging debate about CDMA/TDMA/FDMA at the beginning of the cellular revolution (these debates were actually rather heated!). In fact, there are aspects of CDMA that seem downright boneheaded when you're trying to conserve precious spectrum resources. Nonetheless, there's a certain analytic elegance an simplicity about what follows that's appealing, and it just so happens that in this particular VHS/Beta battle (VCR tapes – which are increasingly a technology that seems modern to me, but

you may not even remember!), it looks like CDMA is winning out by and large.

So, how do you choose your waveforms $s_m(t)$? RANDOMLY! What does that mean? Aren't random waveforms hard to compose? DEFINITELY, so we restrict the waveforms to a particular easy to construct format. We divide the bit interval up into $L$ equal duration "chips" and for a given signature, we randomly choose between $\pm 1$ for each of them! Thus, if $p(t)$ is a unit-energy square pulse on $(0, T/L)$ we have

$$s_m(t) = \sum_{\ell=0}^{L-1} c_{m,\ell} p(t - \ell \frac{T}{L})$$

Where all the $c_{m\ell}$ are iid equiprobably $\pm 1$. What you should notice immediately is that by virtue of chopping up our waveform into little chips, we've increased the bandwidth it occupies by a factor of $L$. For this reason, this approach is also called *spread spectrum*.

This all sounds crazy right? Won't random codewords interfere with each other? Yup, they do. Didn't you just increase the bandwidth used by every user? Yup, we did. However, remember that in a cellular system you have multiple bases. So even if you tried to make your waveforms orthogonal, other bases doing the same thing would mess you up unless they used different frequencies. In a CDMA system, all the bases share the same bandwidth and we simply have to tolerate it. So, oddly enough, at the end of the day it all ends up working out rather well (and with that statement I'm sweeping at least a full decade of fierce debate under the rug).

Now, let's go back to our detection problem. We immediately see that $\mathcal{E}_{s_1} = L$ and that the sum – or *interference* – is a sum of a bunch of random variables, many of them independent. So, to extract user 1's info we do a matched filter on signature $s_1(t)$ as in

$$r_1 = \int_0^T \sum_{m=1}^{M} b_m s_m(t) s_1(t) dt = \int_0^T \sum_{m=1}^{M} b_m \sum_{\ell=0}^{L-1} c_{m,\ell} p(t - \ell \frac{T}{L}) \sum_{n=0}^{L-1} c_{1,n} p(t - n \frac{T}{L}) dt$$

We rearrange as

$$r_1 == \sum_{m=1}^{M} \int_0^T \sum_{\ell=0}^{L-1} \sum_{n=0}^{L-1} p(t - \ell \frac{T}{L}) p(t - n \frac{T}{L}) b_m c_{m,\ell} c_{1,n} dt = \sum_{m=1}^{M} \sum_{\ell=0}^{L-1} b_m c_{m,\ell} c_{1,\ell}$$

because the $p(t - \ell \frac{T}{L})$ are all mutually orthonormal. We can also rewrite this as

$$r_1 = \sum_m b_m \mathbf{c}_1^\top \mathbf{c}_m$$

We then rewrite $r_1$ as

$$r_1 = b_1 L + \sum_{m \neq 1} b_m \sum_{\ell=0}^{L-1} c_{1,\ell} c_{m,\ell} = b_1 L + \sum_{m \neq 1} b_m \mathbf{c}_1^\top \mathbf{c}_m$$

and all these sums make us immediately think AHA! Gaussian! So for a reasonable number of users and chips, we assume that the sum term looks Gaussian enough for government work. In that case we only need it's mean and variance to be able to determine the average error probability for user 1 (and thus for all users since we've set the problem up symmetrically – all users have random waveforms).

*(You'll notice that it is assumed that the base knows the users' waveforms – which seems rather odd since we're assuming random chips. More on that a little later.)*

MEAN:

$$E\left[\sum_{m \neq 1} b_m \sum_{\ell=0}^{L-1} c_{1,\ell} c_{m,\ell}\right] = \sum_{m \neq 1} E[b_m] E\left[\sum_{\ell=0}^{L-1} c_{1,\ell} c_{m,\ell}\right] = 0$$

VARIANCE:

$$\sigma^2 = E\left[\sum_{m \neq 1, k \neq 1} b_m b_k \sum_{\ell=0, r=0}^{L-1} c_{1,\ell} c_{m,\ell} c_{1,r} c_{k,r}\right] = \sum_{m \neq 1, k \neq 1} E[b_m b_k] \sum_{\ell=0, r=0}^{L-1} E\left[c_{1,\ell} c_{1,r}\right] E\left[c_{m,\ell} c_{k,r}\right]$$

Since the $b_m$ and $b_k$ are independent when $k \neq m$ and they can only take on values $\pm 1$ we have

$$\sigma^2 = \sum_{m \neq 1} \sum_{\ell=0, r=0}^{L-1} E\left[c_{1,\ell} c_{1,r}\right] E\left[cm, \ell c_{m,r}\right]$$

The same argument for the chips $c_{i,j}$ leads to

$$\sigma^2 = \sum_{m \neq 1} \sum_{\ell=0}^{L-1} = (M-1)L$$

So the signal to noise ratio is

$$\text{SNR} = \frac{L^2}{(M-1)L} = \frac{L}{M-1}$$

TADA! This is an interesting expression. As the number of chips $L$ goes up, we get better SNR. The number $L$ is called the "processing gain" for this reason. We also notice that for a fixed number of chips, this expression tells us how many users we can tolerate before exceeding some SNR threshold.

For instance, suppose you can tolerate a bit error rate of $10^{-2}$. How big an SNR do you need? Well, we can look up values for the Gaussian CDF (which is often written in terms of the error function which I find confusing) and we see that the signal amplitude needs to be about $2.3$ times less than the noise standard deviation. That means that the SNR needs to be $2.3^2 = 5.29$. So, for a CDMA system with $L = 128$ chips (standard) you can in principle support $M - 1 = 128/5.23 \approx 24$, which means $M = 25$ users at a BER of $0.01$.

Now, there are all sorts of other things I could regale you with. You'll notice that chips are SHARP (unit steps) which means they have ROTTEN spectral characteristics if you're trying to conserve bandwidth. So what's often done is to use things like raised cosine pulses instead of square pulses and all that other usual stuff that's done to decrease bandwidth.

There's also a kinda hidden (but now to be revealed!) cuteness about CDMA (this is called direct sequence CDMA because we assume that each bit stream is modulated with random chips (i.e., user codewords change from bit to bit). Because we use correlator receivers and because radio channels have echoes, you can imagine that time-shifted versions of the signal arrive at the receiver (faint echoes). Well, what if you delayed the signal into your receiver by a few chip-durations. Your receiver (ideally) would ignore the direct path signal and pick up only the echo! If there are multiple echoes, you could pick out each echo path because the codewords when shifted are UNCORRELATED WITH EACH OTHER because the chips themselves are independent!!!

Now, suppose you had a really cool receiver that could do multiple correlations at once. For a given user, you could take the output of these receivers, add them together and thereby increase your SNR even when the channel is trying its best to make your life miserable with echoes. This sort of receiver is called a RAKE receiver (for reasons that are simple to explain with a picture of the actual device – piezo-electric-acoustic properties are employed and the device surface looks like the fingers of a rake).

Now, how to generate the sequence of chips in a simple way. True randomness is really hard to do. But I've already shown you (in class) a simple structure that generates rather random-looking bit sequences (pseudorandom bitstreams) called a feedback shift register. You can look up the exact structure on the web – there are many different configurations and lengths – but the basic ideas is that every

transmitter and every receiver can have one of these simple babies on board. The base station assigns a differnt starting "seed" to each user and since the base knows the seed, it can generate that user's chips sequence at the receiver.

It also turns out that the cross correlation properties of sequences can be made better (truly random sequences have delta function correlation – shift by one chip and the sequences are uncorrelated) by using "Gold Codes" (named after the inventor).

And that's about it for our brief treatment of CDMA.