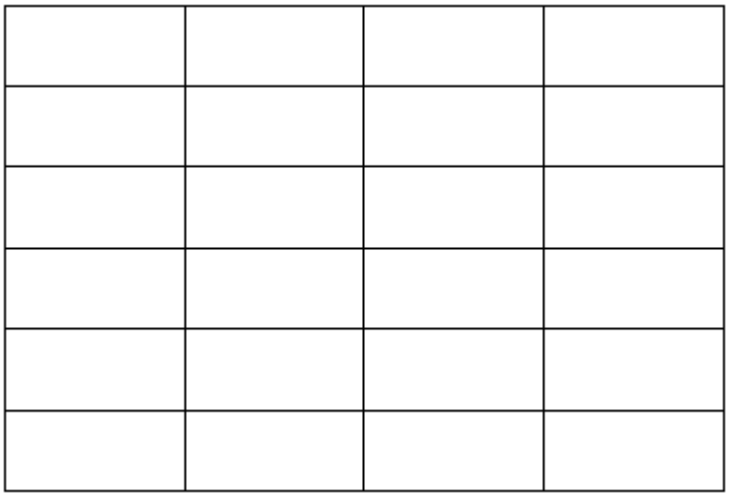
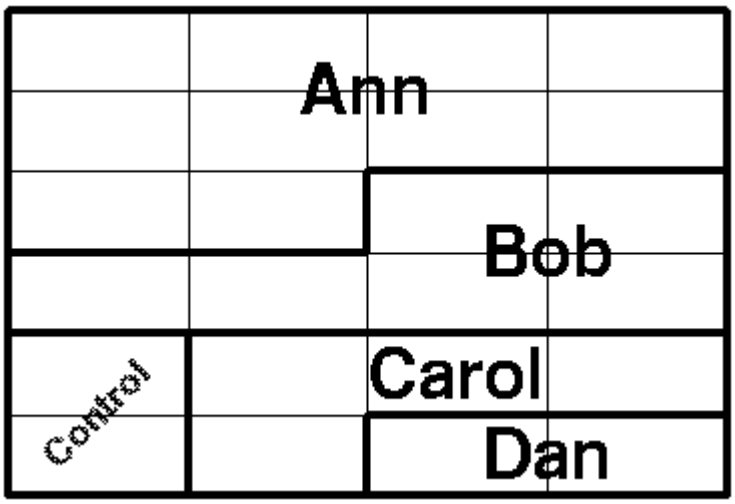
UNDERSTANDING THE LTE DOWNLINK : PART 1

The downlink is the part of a cellular network that transmits data from a cell tower to a cell phone. The 4G LTE downlink is sophisticated and complicated – very complicated. A useful technique for learning any complex subject is to do it in stages, so we'll start with a very over-simplified example of how the downlink works.

Data transmission over the downlink always starts with an empty table, like so:



The first step is to divide up that table amongst the various users that will be getting data. Once that is done we have something like the following:



In our example the table will be holding data for four users – Ann, Bob, Carol & Dan. In the lower left hand corner you'll notice a region labeled "control". That area describes how the table is divided. It says that the first 2 ½ rows belong to Ann, the next 1 ½ rows to Bob, etc. The control information is always in the same place (the lower left in this example). The rest of the table is dynamic and will change over time. In this table Ann got the biggest share. In another table it might be Carol. Do not assume that there are only four users of this cell tower at this time. Zeke is also using his smart phone – but he will not be getting any data from this particular table.

Every user will have a stream of data (ones & zeroes) waiting to be transmitted to him or her. The next step is to load the table with some of the data that is targeted for each user, and we end up with something like this:

00	11	11	01
10	00	00	01
01	01	101	111
111	110	001	010
1100	00	00	00
1010	00	1101	0011

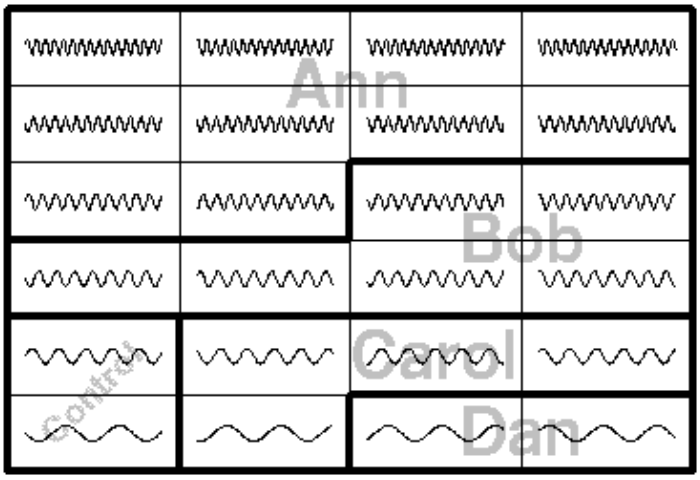
You'll notice that the number of bits in each table entry varies. It will always be either 2 bits, 4 bits, or 6 bits. The decision about how many bits to use is a trade-off between throughput and robustness. More bits means more data going through, but also a higher risk of data corruption. If you go with 2 bits when in fact 6 would have worked then your data rate is only one third of what it could be. But if you try to use 6 bits when the radio conditions only support 2 bits then the data will be lost and your data rate is zero.

You might wonder how different manufacturers can distinguish their LTE products – given that they are all building equipment to the same set of specifications. Well, part of the answer is in the previous two steps. The LTE specifications say nothing about how to allocate table entries amongst users. Do a good job and all your customers will be reasonably happy, even though they're sharing a very limited resource. Do a bad job, and people will be complaining about slow throughput, unacceptable delays, and so on. The specs also say nothing about choosing the number of bits per table entry. You want to be as aggressive as possible without crossing the line where data starts to be lost.

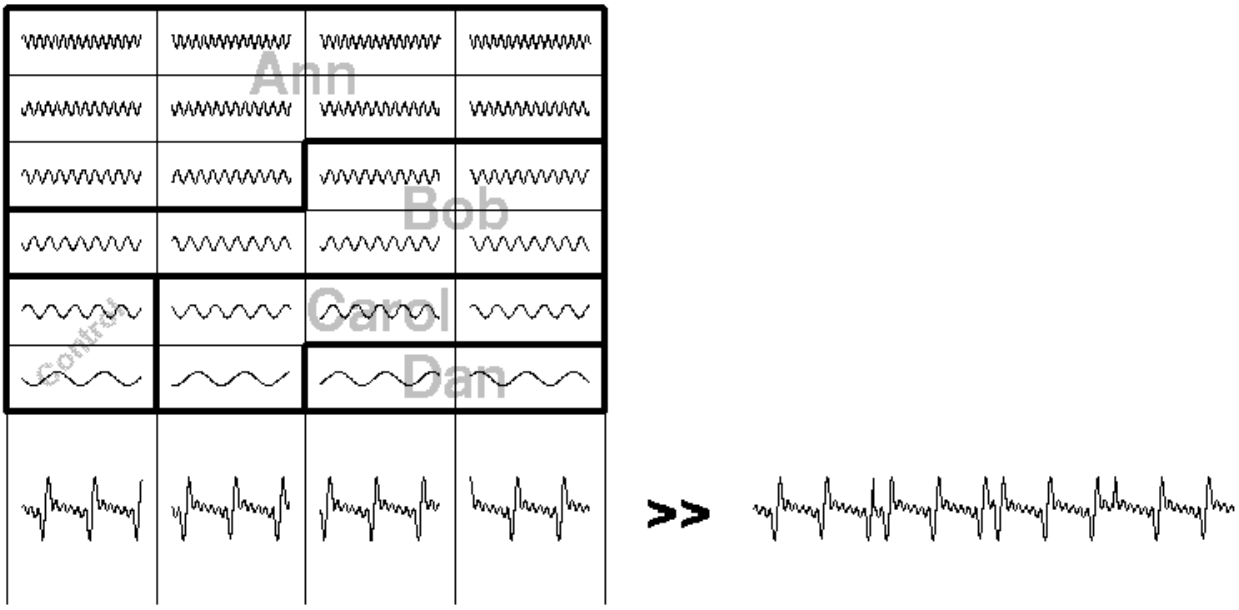
How well you walk that knife edge will be part of what determines how well your equipment performs.

There is a lot of leeway in how a manufacturer implements the previous two steps. From here on out there's no leeway whatsoever. Everything is pinned down with absolute precision.

The next step is to use a carefully specified algorithm to convert the data bits in each table entry to a squiggle. Our table now resembles this:



We then add together the squiggles in each column to produce a composite squiggle, and after that, you concatenate the column composite squiggles into one long continuous squiggle. These two steps are illustrated in the following diagram:



The last step? Take that final squiggle and pass it off to the radio for transmission. The job of a radio transmitter is to take squiggles and convert them into radio waves. In the case of an AM radio station the squiggle represents an audio signal, perhaps taken from a microphone. In the case of LTE the squiggle was artificially generated. It doesn't matter to the radio where the squiggle comes from.

The radio will take 0.5 milliseconds to transmit the squiggle. What do we do in the mean time? Repeat the whole process!! Start with another empty table and go through the same sequence. We have 0.5 milliseconds to generate another squiggle, because once that first 0.5 milliseconds are up the radio is going to be "hungry" for more squiggle, and we'd darn well better be ready to feed it!

The whole process continues ad infinitum. A continuous stream of information being broadcast, generated in 0.5 millisecond chunks.

What about on the receiving side, at your cell phone? It's just the whole process in reverse. The cell phone receiver takes a 0.5 millisecond chunk of signal, then breaks it up into 4 pieces, each of which represents that composite squiggle that we created by summing a column. It then de-composites the signal into 6 individual pieces – those are our 6 rows – thus creating a table just like we had in our diagrams. It then converts the squiggles back into bits. The cell phone looks at the control section, and then grabs any data intended for that cell phone.

NOTE 1: You might be wondering about this business about "de-compositing" the signal. Is that really possible? Well, in general, no. If you add up a bunch of arbitrary squiggles there is in general no way to reverse that action. But these are not arbitrary squiggles! They are very, very carefully chosen squiggles – picked in fact just so that this de-compositing is possible. Mathematically inclined readers might have guessed that this involves the Fourier transform and inverse Fourier transform, and they'd be right.

NOTE 2: You might be wondering how the control region, with a maximum capacity of 12 bits, could actually contain a description of the table allocation. Well, in this toy example it can't. A real LTE table is much larger, the control region is much larger, and it can in fact describe what parts of the table belong to what users.

NOTE 3: Each cell phone is only supposed to look at the data intended for that cell phone – but of course it receives the entire table. What's to stop Ann from looking at Carol's data? Well, nothing really. But it won't do her much good. The data sent over LTE is encrypted. So even though it's not hard to look at someone else's encrypted data stream, without the correct decryption key all you have is a random sequence of ones and zeroes.

NOTE 4: An actual implementation would probably not operate exactly as I've described here – which is very "brute force". A real implementation would effectively do the same thing, but in a more clever and efficient manner.