error occurs in a manuscript, or a crackle obliterates words in a telephone conversation, it is often possible to recreate the words by considering the context of the sentence. Sometimes we build in extra redundancy for use in 'noisy' environments: the use of 'alpha', 'bravo', and 'charlie' in place of 'a', 'b', and 'c' in radiotelephony, for example.

Suppose that on our computer we send a word of x bits in length, consisting of y bits of real data and z redundant bits (i.e. $x = y + z$). In our explanation of parity we had a value of seven for y and one for z. For Hamming codes, z will need to be proportionately larger. Now let's assume that a single-bit error can occur in any of the x bits (our z redundant bits are of course just as prone to error as the y data bits). If the chance of a bit error in a word is, say, one in a million, then the chance of two errors in a word is one in a million million, so we'll ignore this possibility.

When the data is received at the other end, there will be x+1 eventualities. Either there will be no errors, or the first data bit will be in error, and so on up to the xth bit. Now, with z redundant bits we can represent 2^z situations, so that for the word to be proof against one bit error:

$$2^z \geqslant y+z+1$$

If y is seven (for ASCII codes), then z will need to be four. If y is four (as in our example in the panel) then z will need to be three. However, if y is 16 then z need be increased only to 5. It follows that Hamming codes are far more efficient for longer word-lengths than for short ones.

In a Hamming code, each of the redundant bits acts as an even-parity check on a different combination of bits in the word. If any bit is flipped in transmission then one or more of the check bits will be wrong and the combination of these bits will point to the erroneous bit in the word (see example). The receiving computer's software can then simply flip that bit back again.

The key to the way that Hamming codes work is the different combinations of bits upon which each Hamming bit acts as a parity check. The total number of bits is effectively divided into several different but overlapping sets — devised so that no two bits appear in the same combination of sets. The receiving computer performs parity checks on the same sets as the sending device did to create the Hamming code. If any one of the bits, including the Hamming bits, has been flipped in transmission, then one or more of these sets will not pass the parity test. The combination of tests failed points to a unique bit.

Some computers employ Hamming codes even for their internal memory operations. When this is the case, it is possible to remove one whole RAM chip and watch the computer continue to function! Some military computers take the principle of redundancy to the extreme of duplicating every single component in the computer, and comparing the results from the two halves after each operation.

How A Hamming Code Works