Error Control and Information Coding

- No communications service is perfect;
- from time to time data will either be lost completely or will be delivered out of order.
- It is quite common for a protocol to be asked to enhance the error rate inherent in the underlying service. Notice that the word is "enhance"; it is, of course, impossible for a protocol to eliminate errors completely.
- In order that errors can be corrected they must first be detected and this is closely related to the way in which information is encoded for transmission.

- According to the OSI model, the Transport Service is supposed to provide for the error free transfer of information. This being so, error control must take place in the Transport Layer or below.

- Since the Physical Layer has no underlying service, that leaves layers 2-4. Error control may be found in each of these layers. It may also be found in the Application Layer; the "end-to-end argument" leading certain applications not to trust the efforts of the layers below.

<u>Errors and error rate</u>

We can only conclude that a service has been guilty of error if we know Quality of Service it was supposed to provide.
If we are using a service which gives no guarantees about sequenced delivery then we should not complain if it delivers out of sequence. However, there are a couple of situations that may be considered errors no matter what service we ere considering:

i. Delivery of a message which has been corrupted in some way
ii. Non-delivery of a message that was submitted.

- We begin by concentrating on these two error types.
- No service can offer a zero error rate.
- However, it is possible for the error rate to be effectively zero.
- For example, if a 64 Kbps line has a BER of 10-12 there will be one bit in error in every 6 months of continuous operation. In most circumstances this is not worth worrying about; other associated bits of equipment are breaking much more frequently than that.

Two error rates are important

i. The signalled error rate concerns errors that are detected and reported by the underlying service
ii. The residual error rate concerns errors that are missed by the underlying service.

If either of these rates is unacceptable we will have to implement procedures for error correction. if ii) is unacceptable we will also need some method of error detection.

Redundancy

It is obvious that there is a srelling mistake in this sentence. It is obvious because the set of all possible words that can be made from English letters can be partitioned into two sets; English words and non-English words. "Srelling" belongs to the latter set as do dfede, msndasad, and qjykfda. In fact, vastly more "illegal" words can be made than legal ones. Suppose we somehow re-organised spelling so that all combinations of letters spelled legal words. We would not now be able to tell whether a word had been mangled by a transmission service; the result would always be another legal word'

Another way of putting this is to say that our spelling system has "redundancy". All those illegal combinations are wasted as far as the business of encoding information is concerned. Suppose there are around 64000 different words in the English language. 64000 = ~2^{16} so, in principle we only need 16 bits per word. If the words are 7 letters long on average and each letter represents 5 bits (26 =~2^{5}) then each word actually uses about 35 bits. So roughly half the bits are redundant.

What is true for spelling is also true for the way data must be encoded for data transmission. If we are to detect errors then there must be some redundancy. This means that error control has a cost in terms of throughput; the capacity of a channel is effectively reduced since some of the bits it carries are redundant. We need to understand how much redundancy is needed in order to be able to detect errors

Error Detection

To make the problem manageable we will assume that information is to be transferred in fixed-size chunks of m bits (m for "message"). Thus there are 2^m different possible messages that might be sent.
Messages are encoded for transmission as n bits where n = m + r. The additional bits constitute the required redundancy.
Note that it is not necessarily the case that we can classify each bit as "information" or "redundant" any more than we can say which letters in an English word are redundant.
We call each of the n-bit objects a "codeword".
There are 2^n possible codewords but only 2^m of these are legal codewords - the ones corresponding to the messages.
Errors alter bits in codewords. When this happens we would like the result to be an illegal codeword.
If the result is a legal codeword then the error will be missed.
For example, suppose we construct our codewords on the basis of applying even parity to 7-bit ASCII. This gives the following codewords:

A	01000001
B	01000010
C	11000011
D	...

If we alter one bit in any of these codewords the result is an illegal codeword since the parity would be wrong.
However, if we alter two bits then the result will be legal.
We can change "A" to "B" in this way. With this encoding, all legal codewords are a least two "bit changes" apart. The number of bits in which two codewords differ is called the "Hamming distance" (Hamming, 1950).
In the 8-bit ASCII code above, there are no two legal codewords with a Hamming distance of 1. We can say that the minimum Hamming distance is 2.

If a code has a minimum Hamming distance of h then ail (h-1)-bit errors will be detected since they cannot possibly generate another legal code.
Put the other way around:

To detect all d-bit errors => minimum Hamming distance > d+1

ERROR CORRECTION

Notice that not only was it possible to detect the spelling mistake in the first sentence in the above section,it was also possible to correct it. Encodings that allow correction in addition to detection require more redundancy and a greater Hamming distance

Suppose we want to be able to correct all errors of d-bits or fewer. We must ensure that the set of all possible illegal codewords that we can generate from a legal codeword by changing d-bits or fewer does not intersect the similar set generated from any *other legal* codeword. If we receive one of these illegal codewords, (i.e. one generated from a legal code word by altering d-bits or fewer) we will always be able to tell from which legal codeword it was generated. We can think of each legal codeword as being surrounded by a cloud of illegal codewords which can be generated from it by altering d bits or fewer. We can state that

To correct all d-bit errors => minimum Hamming distance > 2d + 1

For example, to correct all 1-bit errors requires a minimum Hamming distance of 3. So 8-bit ASCII has no error correcting properties. Suppose we have a system with just four codewords:

0000000000 0000011111   1111100000   1111111111

This has a minimum Hamming distance of 5, so it should be able to correct all 2-bit errors. If 0000001110 is received then we can assume that we started out with 0000011111. Of course, it may be that we really started out with 0000000000 and 3 bits were altered..

8-bit ASCII has 1 redundant bit in 8 - 12.5% redundancy and yet it has no error correcting properties. How much redundancy would be required to correct all 1 bit errors? If we have m messages and m clouds. Since codewords are n bits long we can generate n illegal codewords from each by changing 1 bit of a legal codeword. So each cloud accounts for n+1 codewords (n illegal plus one legal). So:

Total number of codewords in the clouds is $2^m(n+l)$

This must be less than or equal to the absolute total number of codewords(there may be some illegal codewords not in any clouds) so:

$2^m(n+1) <= 2^n$

$2^m(m+r+l) <= 2^{m+r}$ since $n = m+r$

$m+r+1 <= 2^r$

For ASCII code m = 7 we have

r	**m+r+l**	**2^r**
0	8	1
1	9	2
2	10	4
3	11	8
4	12	16

so we must add at least 4 "check bits" to our 7 message bits - 36.4% redundancy

Hamming not only derived a limit on how many check bits would be needed, he devised a code that operated at that limit - the"Hamming Code".

Hamming Code

We take as an example a Hamming Code for 7-bit ASCII. This must have 4 check bits. The check bits are placed in the power of 2 positions; 1, 2, 4 and 8

11	10	9	8	7	6	5	4	3	2	1
			^				^		^	^

The check bits act as 4 independent parity bits for combinations of the message bits. To see which message bit is checked by which check bit write the number of the message bits as binary numbers.

	Check bit no.			
Message bit no.	8	4	2	1
11	1	0	1	1
10	1	0	1	0
9	1	0	0	1
7	0	1	1	1
6	0	1	1	0
5	0	1	0	1
3	0	0	1	1

This indicates that check bit 8 will check message bits 11, 10 and 9; i.e. bit 8 is chosen to give ever. !odd) parity to bits 11, 10, 9, and 8. The table also shows which check bits will be affected by an error in a given message bit. For example, if bit 9 is altered, check bits 8 and 1 will yield the wrong parity, 4 and 2 will be unaffected. It is only an error in bit 9 that can cause this particular combination. Therefore, if we find the checks for 8 and 1 fail whilst those for 4 and 2 succeed it must be bit 9 that is in error and we can correct it. Note also that if it is one of the check bits that is altered the other checks will be unaffected.

The error algorithm is:

"Write down a 1 under the check bits that fail and a 0 under those that succeed. Interpreting the result as a binary number gives the number of the bit that is in error. "

For example, ASCII "A", even parity, would be encoded as:

11	10	9	8	7	6	5	4	3	2	1
1	0	0	1	0	0	0	0	1	0	0

If this is received as:

11	10	9	8	7	6	5	4	3	2	1
1	0	0	1	0	1	0	0	1	0	0

Then we write

8	4	2	1
0	1	1	0

since the checks for the bits 4 ad 2 fail. This tells us that bit 6 should be corrected. This procedure can be implemented.