Chapter 4. Reliable Communication

Table of Contents

Introduction to Network Applications and Network Usage

Context

This unit follows on directly from unit 3.

Introduction

Data communication as described in the pervious unit would, by and large, be reliable if the presence and effects of noise could be ignored. Unfortunately, they cannot. This unit begins by showing how noise may be described and its effect on data signals assessed. Essentially, noise acts to distort data signals so that, on occasion, one signal is mistaken for the other which results in an error in the data received over a data transmission link. Depending on what the data represents, the effects of such an error may be anything from negligible to catastrophic. This means not only that some way of dealing with errors is needed, but also that various ways are needed so that a method appropriate to each specific transmission may be chosen.

The broad principles of error handling methods are explained with reference to simple methods for detecting errors and for correcting errors. Error detection requires that the occurrence of an error be detected. Error correction is more complex in that it involves not only the detection of the occurrence of an error but also the determination of the position of the error.

The content of the unit is summarised by introducing the Data Link layer as a software layer responsible for error handling (its other responsibilities are not stressed) which allows errors to be dealt with automatically thereby providing the basis for reliable point-to-point communication.

Objectives

At the end of this module, you should be able to:

- explain how noise causes errors in data transmission;
- assess the effects of errors;
- understand how errors may be dealt with;

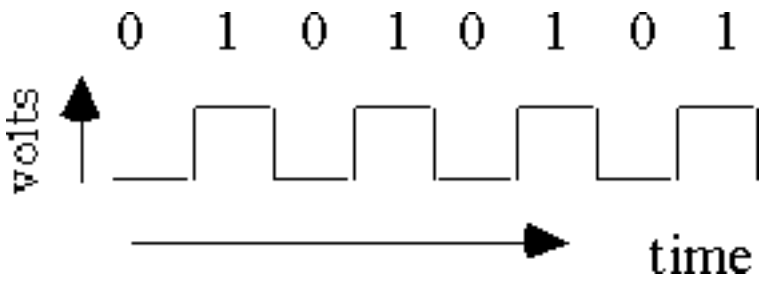
Content

In parallel with this unit, you should read the relevant parts from your textbooks.

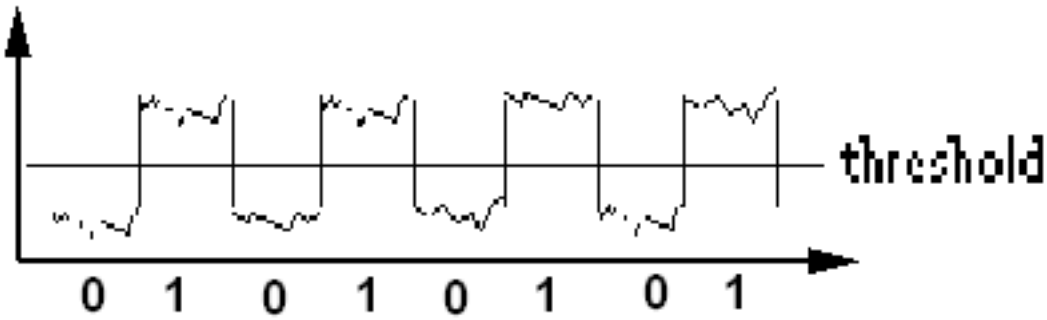
The occurance of errors

The basic task at the sending end of a data communication link is, repeatedly, to accept an item of data and to send the corresponding signal. The task at the receiving end is, repeatedly, to decide which signal was received and to generate the corresponding item of data. The decision is simple, since it involves no more than a choice between two possibilities. In ideal conditions, nothing should go wrong, but conditions for communication are not usually ideal. As we saw in the last unit, there is usually some kind of 'noise' present in the communication channel. This noise acts to distort and degrade the communication signals. It sometimes happens that the distorted version of the signal for (say) a 0, can be mistaken for the signal for a 1. In this case, there will be an error in the received data.

The way that an error occurs can be illustrated as follows. Suppose that two computers can send data directly between their interfaces. The sending computer will send:



In ideal conditions, the receiving computer will receive the same signal and, by interpreting a high voltage as a 1 and a low voltage as a 0, will ensure that the data is transferred correctly. If there is a small amount of noise in the channel, the received signal will be:



It is no longer clear which is the 'high' voltage and which the 'low'. But by setting a threshold midway between high and low, anything above the threshold may be treated as a 1, and anything below as a 0, and no errors have occurred.

However, if there is a large amount of noise, the received signal could be:

This would cause an error in the fifth digit as indicated.

To Do

Now do Review Question 1.

The effects of errors

Suppose that, as part of an e-mail transmission, an 'A' is being sent using the ASCII code. These eight digits are transmitted:

01000001

Now, if an error occurs in the second digit (that is, in the second from the right or second least significant digit), what effect will it have? The received digits will be:

01000011

and this decodes to give a 'C'. So the effect of the error is to cause the receipt of an erroneous character. The error can be said to be a small one in the sense that in the alphabet the erroneous character is only a small distance away from the correct one.

If an error occurs in the fourth digit, the received sequence will be:

01001001

which decodes to 'I', and represents a larger error in the sense defined above. This shows that the magnitude of the effect of an error can depend on the position of the error.

However, if an error occurs in the eighth digit, so that the received sequence is:

11000001

we know that an error has occurred, because all ASCII codes begin with a 0. This error has in fact been detected because it produces a violation of a fixed requirement. It can, in fact, be corrected, since it is known that only a 0 is permitted in this position.

To Do

Now do Review Question 2.

Dealing with Errors

One way to deal with errors is to detect their occurrence. This is done with an error detecting code. The idea is that the sending computer adds extra binary digits to the data in such a way as to impose a constraint that will be violated should an error occur. The receiving computer can detect the occurrence of errors by testing for violation of the constraint. If the constraint holds, it indicates that no errors have occurred, and the receiving computer removes the extra bits and accepts the data. Violation of the constraint indicates that errors have occurred. In this way, their occurrence is detected, and the receiving computer rejects the transmission and requests a retransmission.

To Do

Now do Review Question 3.

Errors can also be corrected. This involves not only detecting that an error has occurred but also locating the position of the error. If a binary digit in a specific location is known to be incorrect, it can be corrected simply by changing it from a 1 to a 0 or vice versa, as necessary.

To Do

Now do Review Question 4.

Error Detection

Detecting a single error

A simple scheme to detect the occurrence of a single error makes use of the idea of parity. The parity of a block can be even or odd. It is determined by counting the number of 1s in the block. Thus, the block 1111 has even parity because it contains four 1s and four is even, while the block 0100 has odd parity because it contains one 1 and one is odd.

The scheme works like this:

Transmitter
: Divides the sequence of bits to be sent into fixed length blocks. In this example, we take the block length as four:

  01000011011100000111100001111
  0100 0011 0111 0000 1111 0000 1111

  Adds a parity bit to each block.

	0100*1* 0011*0* 0111*1* 0000*0* 1111*0*
	Sends the bits of the block sequentially.
Receiver	If the parity is correct (in this example, if it is even), the parity bit is deleted,
	the remaining data is accepted and correct reception is acknowledged.
	If the parity of the block is not correct, the block is rejected and
	a retransmission is requested.

Consider what happens on receipt of:

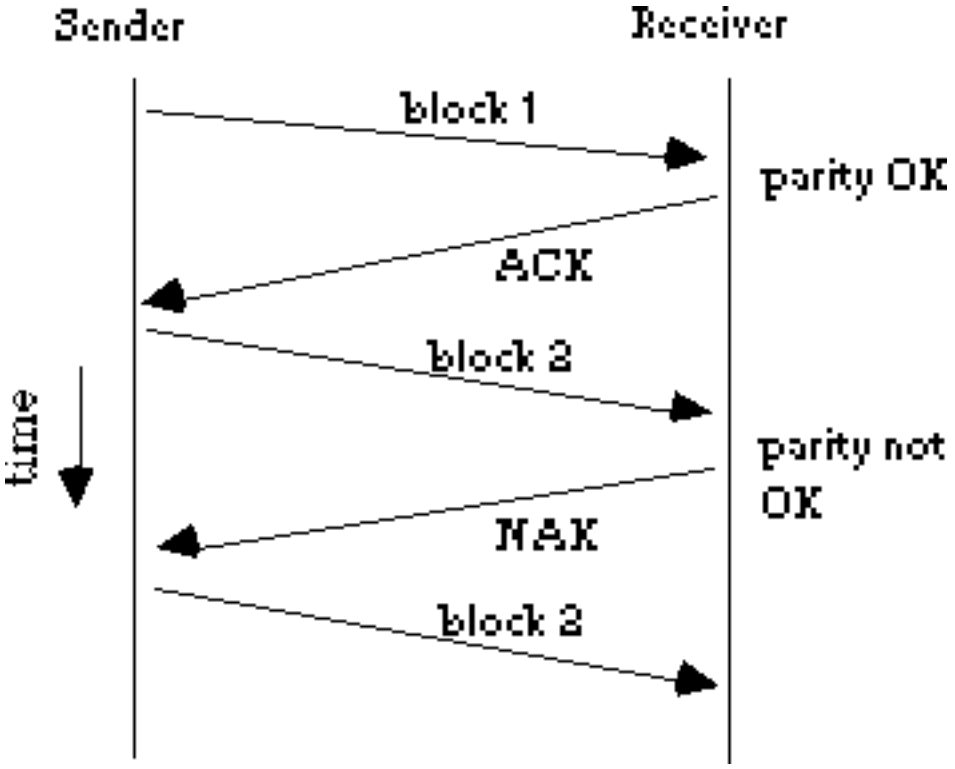
01001 01110 01111 00001 11110

The first block has even parity. The parity bit is deleted and the data is accepted.

The second block has odd parity. It is rejected, and a retransmission is requested.

When the second block has been correctly received, the third block can be dealt with. It has even parity, and its data content is accepted. And so on.

The error detection process is:



This is an example of a protocol.

To Do

Now do Review Questions 5 and 6.

Detecting Multiple Errors

The scheme presented above for detecting a single error by adding a single parity bit can also detect some patterns of multiple errors, although it cannot consistently detect all patterns of multiple errors.

(To remind you, it can detect any pattern containing an odd number of errors but no pattern containing an even number of errors.) However, by carefully adding more than one parity bit, it ought to be possible to devise ways of correcting multiple errors.

Q: Will the addition of one parity check for the first half of a block and another for the second half of a block detect all pairs of errors that may occur?

A: No. A pair of errors occurring in the same half of the block won't be detected.

Q: Will the addition of one parity check for the bits in the odd-numbered positions in a block and another for the even-numbered positions in a block detect all pairs of errors that may occur?

A: No. A pair of errors both occurring in odd-numbered (or even-numbered) positions in the block won't be detected.

Q: Will the combination of checking the two halves of a block and the even- and odd-numbered positions in a block detect all pairs of errors?

A: No. A pair of errors both occurring in odd-numbered (or even-numbered) positions in the same half of a block won't be detected.

It seems that detecting multiple errors isn't so easy. In practice, it may be preferable to transmit blocks that are sufficiently short that more than one error is unlikely to affect them. There are ways of detecting multiple errors, though. They depend on the computation of a digest of the contents of a block that is then appended to the block. The parity bit used to detect a single error can be seen as a very simple digest.

To Do

Now carry out Activity 1.

Error Correction

Another way to deal with errors is to be able to correct them. To do this, it is necessary to be able not only to detect an error but also to determine its position. Once the digit in a certain position is known to be wrong, it can be corrected by inverting it.

The following procedure will correct a single error in a block by arranging it in an array and adding a pattern of parity bits. First, as with error detection, divide the data into blocks:

01000011011100001111100001111

0100 0011 0111 0000 1111 0000 1111

Arrange each block into an array:

01 00 01 00 11

00 11 11 00 11

Then add parity digits to the rows and columns of each array:

01*1* 00*0* 01*1*

00*0* 11*0* 11*0*

011 110 101

Send each array row by row:

01100001100011011001111010 1

At the receiving end, the process is repeated. The parity checks not only detect the error, but also locate it. We can test this by trying it on the received block that would result if an error had occurred in the fifth digit to be sent:

011010011

These digits are put into their array:

011

010

011

Checking the parity of each row and column gives:

0 1 1 yes

0 1 0 no

0 1 1 yes

y n y

e o e

s _ s

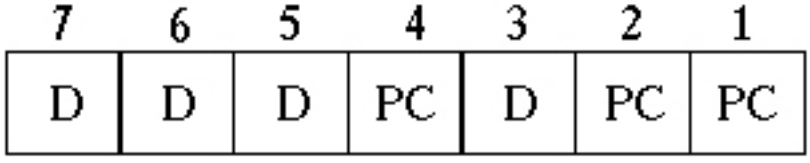
The row and the column containing the error are located, and this gives its position so that it can be corrected.

To Do

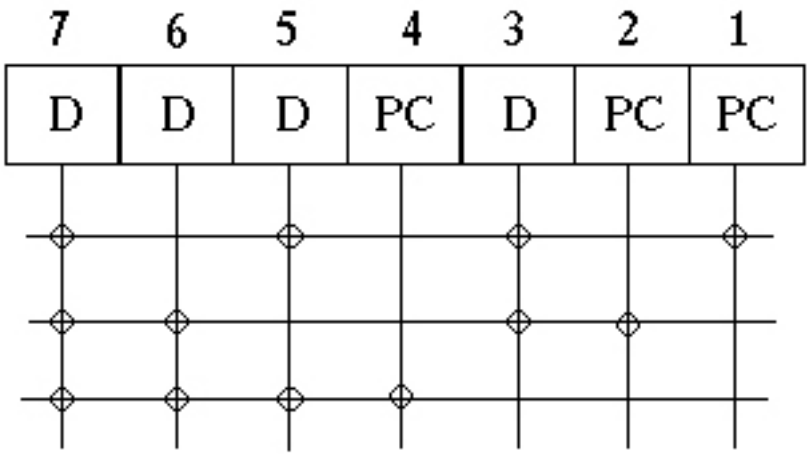
Now do Review Questions 7, 8 and 9.

Error Correction 2

The error correction scheme described above catches the principle of error correction but is, by the standards of commonly used schemes, rather primitive. An example of a more sophisticated error correcting code is the Hamming code. It also divides data into blocks and adds a pattern of parity check digits. A simple Hamming code intended to correct one error operates by dividing data into blocks of four and adding three even parity checks in the pattern shown, where D denotes a data digit and PC a parity check:

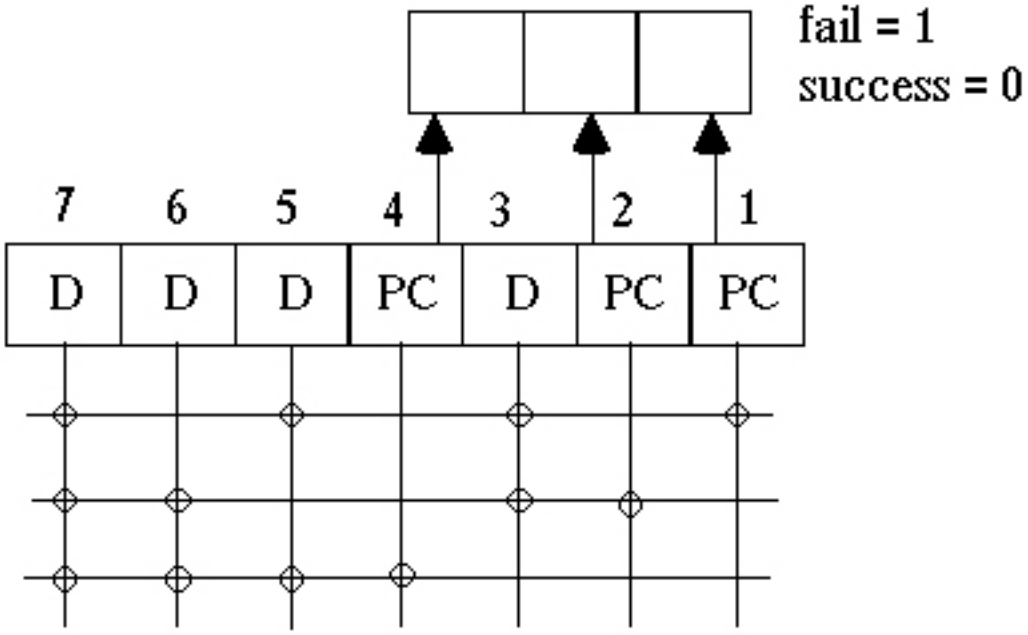


Note that the parity checks occupy positions 1, 2 and 4. The parity checks each check the data digits in different positions, as shown. The parity check in position 1, for example, checks the data digits in positions 3, 5 and 7.



If the data block is 1010, for example, then after computing the parity check, the entire block is 1010010, and this block is signalled over the channel.

When the block is received, the parity check digits are examined and used to determine the position of any error. The procedure is to examine the parity checks, and to record a success by 0 and a failure by 1. The resulting number is zero if there is no error: otherwise it gives the position of the error.



If an error occurs during transmission at position 6, the received block will be 1110010. The parity checks in positions 4, 2 and 1 will, respectively, fail, fail and succeed, causing 110 to be recorded. This is the binary version of 6, which locates the error at position 6.

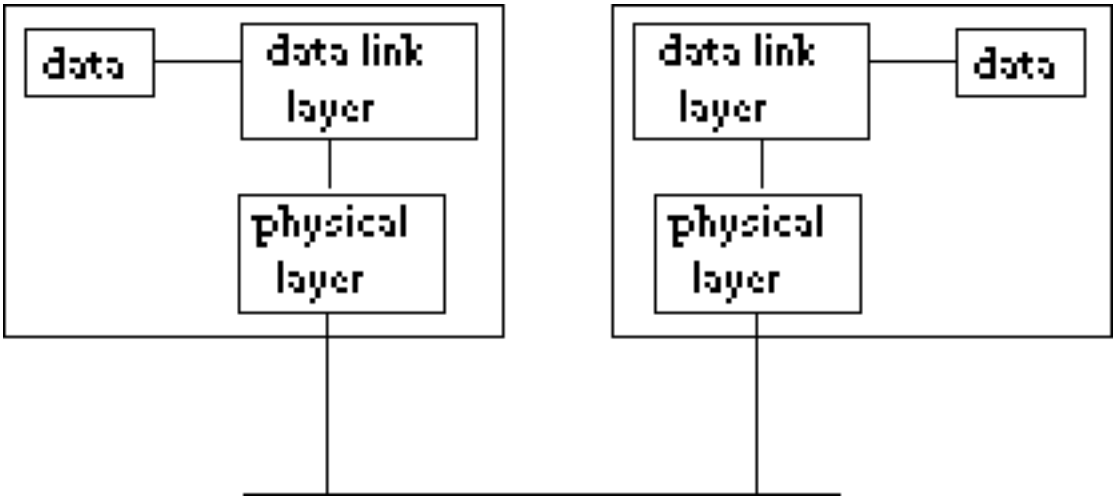
To Do

Now carry out Activity 2.

The Data Link Layer

In dealing with errors, the determination and addition of parity bits to each block is a computation that can be done by the sending computer. Similarly, checking parity bits and taking subsequent action, whether requesting a retransmission or computing the position of an error and then correcting it, is a computation that can be done by the receiving computer. The procedures for this are part of what is

known as the 'data link' layer. Adding a data link layer above the physical layer is sufficient to convert an unreliable 'raw' link into a reliable link.



It is possible for a data link layer to offer error detection as its means of dealing with errors. Equally, error correction could be offered. A flexible system would be able to offer both, making it possible to select the type of service required from those on offer.

To Do

Now do Review Question 10.

Flow Control

The flow of messages between computers must be controlled to ensure, for example, that a message is not sent to a computer that is not ready to accept one, or that a high-speed sending computer does not overload a slower receiving computer. The requirement is called flow control, and it is also a responsibility of the data link layer.

To Do

Now do Review Question 11 and Activity 3.

Activities>

Activity 1 - Error Detection

Find out about the Cyclic Redundancy Check (CRC), which is one of the more commonly used ways of detecting multiple errors. It uses the idea of a digest.

Activity 2 - Error Correction

Test the Hamming code by giving it different single errors to correct. Then see if it is capable of dealing with pairs of errors either by correcting or detecting them. Can you see why the Hamming code works to correct a single error?

Activity 3 - Flow Control

Devise a basic protocol for flow control by simplifying the protocol given above for error detection with retransmissions.

You can find a discussion of this activity at the end of the chapter.

Activity 4 - Hamming Distance

When communication is achieved by transmitting blocks of binary digits of a fixed length, determine a condition on the blocks in terms of the Hamming distance that must be met if a single error in any block is to be detected.

You can find a discussion of this activity at the end of the chapter.

Activity 5 - Hamming Distance 2

When communication is achieved by transmitting blocks of binary digits of a fixed length, determine a condition on the blocks in terms of the Hamming distance that must be met if a single error in any block is to be corrected.

You can find a discussion of this activity at the end of the chapter.

Review Questions

Review Question 1

Suppose that a data communication system is sending one tone for a 1 and another for a 0. Draw the two basic signals generated for transmission when they have respectively two and four complete cycles in a signalling interval. (The ratio of the frequencies of the two tones is 2:4.) With reference to your diagram, devise a simple decision process to determine which signal has been received. Show how the presence of noise could cause an erroneous decision to be made. Show that as the amount of noise increases relative to the signal, errors become more likely.

You can find an answer/comment for this review question at the end of the chapter.

Review Question 2

Describe in rather more detail than you did for the similar question in the previous unit the effect, and the importance, of a single error during the transmission of a text file, a bit-mapped black-and-white image file, a computer program and an ATM transaction?

You can find an answer/comment for this review question at the end of the chapter.

Review Question 3

Suppose that, in sending text and with the intention of detecting errors, the ASCII code for each character is sent twice in succession. At the bit level, how could the constraint be expressed? Could the occurrence of a single bit error be detected? Could other patterns of bit errors be detected? Are there patterns of errors that would not be detected?

You can find an answer/comment for this review question at the end of the chapter.

Review Question 4

The pattern 01000010 is received, and it is known that there are errors in the first and second digits (from the right). Which character was sent?

You can find an answer/comment for this review question at the end of the chapter.

Review Question 5

Why does the described scheme work? (Hint: what constraint does the scheme impose, and how does the occurrence of an error affect the constraint?)

You can find an answer/comment for this review question at the end of the chapter.

Review Question 6

Can this scheme detect other patterns of errors? Are there patterns it cannot correct?

You can find an answer/comment for this review question at the end of the chapter.

Review Question 7

Will this error correction scheme correct multiple errors?

You can find an answer/comment for this review question at the end of the chapter.

Review Question 8

Single letters represented by their 7-bit ASCII code preceded by a 0 have been coded for error correction as in the unit but using an array with two rows and four columns, and the following sequences have been received. Which letters were sent?

- 010011111100110
- 010001101110010

You can find an answer/comment for this review question at the end of the chapter.

Review Question 9

What sort of numbers make the best block sizes in the sense that they require the smallest proportion of parity bits per correctable block?

[Hint: try the different matrix arrangements for error correction when the block length is a composite number like 12, 16 or 60.]

You can find an answer/comment for this review question at the end of the chapter.

Review Question 10

In what circumstances would you choose to have errors handled by error detection rather than error correction, and vice versa?

You can find an answer/comment for this review question at the end of the chapter.

Review Question 11

Show that the procedure given above for error detection with retransmissions is, in a sense, exerting flow control.

You can find an answer/comment for this review question at the end of the chapter.

Discussion Topics

1. What will happen if the address of a message is corrupted during transmission? How can this be dealt with?

2. What will happen if an ACK is corrupted during the error detection process?

To deal with this eventuality, a timer can be set at the sending computer, and if an ACK is not received before the timer expires, the block is sent again. What can go wrong with this protocol? [Hint: Think of what might happen when the retransmitted block carries a coded command.]

3. The idea of a digest introduced in this unit has other applications. Discuss ways in which it can be used as a means of providing authentication and security.

Answers and Comments

Activity 3

Devise a basic protocol for flow control by simplifying the protocol given above for error detection with retransmissions.

Activity 4

If one error occurs in a block, the Hamming distance between the resulting erroneous block and the original correct block is one. This erroneous block must not coincide with one of the blocks that it is possible to send. This means that the distance between every pair of blocks that can be sent must be at least two.

Activity 5

The argument begins as before. If one error occurs in a block, the Hamming distance between the resulting erroneous block and the original correct block is one. For error correction to be possible, it is not enough that the erroneous block does not coincide with one of the blocks it is possible to send. It must also be nearer to the original than it is to any of the others so that it can be corrected by changing to the one it is nearest to. This means that the distance between every pair of blocks that can be sent must be at least three.

Review Quetion 1

A decision could be made on the basis of the number of zero-crossings in an interval. Since, in the absence of noise, these numbers are 4 and 8, a threshold could be set at 6 so that any number below that would be taken to indicate a 1 and any number above to indicate a 0. Noise fluctuations can upset the number of zero crossings, and a large noise fluctuation has more scope for doing so than a small one.

Review Quetion 2

A text file: An erroneous character, which may not be noticed but, even if it is, will not affect the comprehension of the text.

A bitmap: an erroneous pixel, usually not noticeable.

A program: an erroneous instruction, usually disastrous when the program is run.

An ATM Transaction: At worst, an erroneous amount, which would be disastrous.

Review Quetion 3

The constraint is that sixteen bits are sent for a character, the first eight of which are the same as the second eight. A single error anywhere in the sixteen would be detected because it would cause the first eight to differ from the second eight. Other patterns of errors that would be detected include any pattern that is restricted to the either first eight bits or the second eight bits. Any pattern of errors that affected bits in the same positions in the first and second eight would not be detected.

Review Quetion 4

Correcting the erroneous digits gives the correct message as 01000001 and this is the ASCII code for 'A'.

Review Quetion 5

The constraint is that the number of 1s in a transmitted block must be even. An error either changes a 1 to a 0, reducing the number of 1s by one and making it odd, or changes a 0 to a 1, increasing the number of 1s by one and making it odd. Thus, the occurrence of a single error violates the imposed constraint allowing the error to be detected.

Review Quetion 6

It can detect any odd number of errors in a block, but cannot detect an even number of errors.

Review Quetion 7

Yes. If two errors are located so that they occupy both different rows and columns on the grid, they can be corrected. (If they occupy the same row, or the same column, they cannot.)

Review Quetion 8

GM

Review Quetion 9

Square numbers, such as 9, 16 and 25, because they generate a lower ratio of check digits to information digits.

Review Quetion 10

If speed of delivery is not an issue, error detection could be used because the time taken for any retransmissions that might be required would not matter. If time is important, error correction would be preferable, since correct data can be delivered with no need for any retransmissions.

Review Quetion 11

The receiving computer is making the sending computer wait until it receives an ACK or a NAK before proceeding. It is, incidentally, also controlling what the sending computer does when it proceeds.