FACTFILE:
GCE DIGITAL TECHNOLOGY

UNIT A2 1: INFORMATION SYSTEMS

Rewarding Learning

Error Detection and Correction

Learning Outcomes

Students should be able to:

- Describe and evaluate methods of detecting and correcting data transmission errors: parity bits, checksums, echo checking and cyclic redundancy check (CRC);

Content in Error Detection and Correction

- Describe and evaluate methods of detecting and correcting data transmission errors: parity bits, checksums, echo checking and cyclic redundancy check (CRC).

Describe and evaluate methods of detecting and correcting data transmission errors:

Parity bits

If we consider ASCII codes for each character they do not make full use of the byte allocated as only 7 bits are required to represent each character in ASCII uniquely. The leftmost bit is never used or in some cases defaulted to a zero. This bit is then used as a Parity Bit when transmitting data.

Consider sending the word 'FOG' from one device to another device on a network.

Character	ASCII	EVEN parity	ODD parity
F	100 0110	1100 0110	0100 0110
O	100 1111	1100 1111	0100 1111
G	100 0111	0100 0111	1100 0111

During transmission the parity bit would accompany (be part of) the ASCII code. EVEN parity would ensure that the total number of "1 BIT's" is EVEN in each ASCII code. Whereas ODD parity would ensure that the total number of "1 BIT's" is ODD in each ASCII code.

Let's assume in this case we use EVEN parity to transmit FOG and the result is corrupted or the data has been interfered with.

Character Sent	***ASCII code received***	***Character received***
F	1100 0100	F
O	1100 1111	O
G	0100 0111	G

Odd number of 1's

This simple example shows that an error is detected, by adding up the 1's and getting an odd

number. However, it can't, at this stage, work out which bit is in error. 'Even Parity' checking can only detect an odd number of bits being corrupted (one, three etc.) If the errors have caused an even number of bits to change, then parity checking cannot detect the problem.

This is why more complicated error checking schemes have been developed. But parity checks are still very common as it is such a simple way of detecting errors.

To include error detection and error correction we need to include vertical and horizontal parity checks as shown below:

1	1	0	0	0	1	0	0
1	1	0	0	1	1	1	1
0	1	0	0	0	1	1	1
	1	0	0	1	1	1	0

This shows an example were parity can both detect and correct an error and is a method known as 'Hamming code'. Using this method with large amounts of data would be considerably slow coupled with the possibility of the parity bits becoming corrupted and leading to error within parity.

Checksums

Checksums are associated with packets of data and are an integral part of the packet.

Packet Header	Destination Address	Source Address	Packet Address	DATA	Checksum (CRC)

Checksums involve counting the number of 1 BIT's in a packet and sending this as a mathematical total at the end of the packet.

10101100	1100110	0010111	1010001	0001100	0010100

Data contains a total of 20 1's

Checksum

The process involves the data packet 'checking itself' at the destination looking for errors; if errors are found the data will be transmitted again.

A more complex form of a checksum is sometimes referred to as the Cyclic Redundancy Check (CRC) and involves using a complex mathematical formula (see below).

Echo Checking

This method of error detection involves sending data to a particular destination (terminal). When the data arrives at the desired destination an exact copy is retransmitted back to the sending terminal and this copy is compared with the original copy. If the data differs in any way the data will automatically be re-transmitted. Echo checking is also known as a "loopback" check.

Cyclic Redundancy Check (CRC)

Whereas Parity checking will check if all the bits add up to an even (or odd) number, CRC goes further and is able to detect more than one error within a packet.

Cyclic Redundancy Check (CRC) is an error detection mechanism in which a special number is appended to a block of data in order to detect any changes introduced during storage (or transmission).

The CRC is recalculated on retrieval (or reception) and compared to the value originally transmitted, which can reveal certain types of error. For example, a single corrupted bit in the data results in a one-bit change in the calculated CRC, but multiple corrupt bits may cancel each other out.