

Exp: 6

8/09/25

Hamming code

Aim: To send blocks without errors &

Write a program to implement error detection

and error correction using Hamming code concept. Make a test run to input data stream and verify error correction feature.

It's a technique used in network to detect and correct errors.

Error correction at Data link layer

Hamming code is a set of error correction that can be used to detect and correct the errors that can occur when data is transmitted from sender to the receiver. It is a technique developed by R.W. Hamming for error correction.

Create sender program with below features.

1. Input to sender file should be a text of any length. Program should convert the text to binary.
2. Apply hamming code concept on binary data and add redundant bits to it.
3. Save this output in a file called channel.

"C:\Users\9195\Downloads\file1.txt" is works

Create a receiver program with below features:

File (WAV, Text, Images) read file

Lab 2: Error Detection

1. Receiver program should read the input from

channel file as input to the

2. Apply hamming code on the binary data & check for errors at the first position.

3. If there is an error, display the position of the error and the addressed word.

4. Else remove the redundant bits and convert the binary data to ascii and display the original message.

Student observation:

Hamming code sender

sent a word via message send class

def char_to_binary(ch):

return format(ord(ch), '08b')

def binary_encode(data):

d1, d2, d3, d4 = [int(bit) for bit in data]

p1 = d1 ^ d2 ^ d3

p2 = d1 ^ d3 ^ d4, 6th bit

Hamming bits P1, P2 = d1 ^ d2 ^ d3, 2nd and 3rd

return f'{p1}{p2}{d1}{d2}{P1}{P2}{d3}{d4}'

text = input("Enter text: ")

with open(channel.txt, "w") as f:

for ch in text:
 : (1. (bin-ch & 0) char-to-binary(ch)
 (2. for i in range(0, 8, 4),
 ((s, code) = binHammingEncoder(bin-ch(i:i+4))
 f.write(code))

point ("Data written to channel.txt with Hamming code".)

Hamming code Receiver

def hamming_decode(code):

b = [0] + [int(bit) for bit in code[1]]

p1 = b[1] ^ b[3] ^ b[5] ^ b[7]

p2 = b[2] ^ b[3] ^ b[6] ^ b[7]

p4 = b[4] ^ b[5] ^ b[6] ^ b[7]

error_pos = p1 * 1 + p2 * 2 + p4 * 4

if error_pos == 0:

point("Error detected at position {error_pos},
correcting...")

b[error_pos] ^ = 1

d1, d2, d3, d4 = b[3], b[5], b[6], b[7]

return f"{{d1}} {{d2}} {{d3}} {{d4}}"

binary_result = ""

with open("channel.txt", "r") as f:

code = f.read():

for i in range(0, len(code), 2):

byte = binary result[i:i+8]

((i+1) * 2, i*2) text, t = chr(int(byte, 2))

point ("Received text after error correction: " text)

Corrected text, formed at position $i + 1$

(...)

Input:

Enter text to send: Hi
([s] d) [s] d [s] d [s] d

[s] d [s] d [s] d [s] d

Output:

([s] d [s] d)

Data writer [s] d channel output & log (0110011...)

Decoded text sent to receiver: Hi Sq

[r] d ^ [d] d ^ [a] d ^ [g] d = Sq

r * Sq + d * Sq + a * Sq + g * Sq = 20q - 60002

∴ 0 = {20q - 60002} d

{20q - 60002} owing to last step error "3" in Sq

(..., 16, 000)

Result:

$\therefore \{20q - 60002\} d$

(r), [d] d, [a] d, [g] d = sb, sb, sb, sb

Hence the hamming code is implemented to
detect " {sb} {sb} {sb} {sb}" & correct
single error.

For ("s", "t") both formed " " \therefore this
 $\therefore (3b) \oplus 3 = 2b$