In [1]:
```
######################
# Project 3
# ESE 572
# Layer 2 Channel Coding
######################

# Choose a probability of bit error to use when simulating the receiving end.

# prob = 0.1
prob = 0.01
```

In [2]:
```
######################
# Step 1
######################

# Function to convert string to binary using ASCII encoding
def string_to_binary(string):
    binary = ''
    binaryS = ''
    binaryC = ''
    counter = 0
    for char in string:
        ascii_code = ord(char) # Get ASCII code of character
        if counter == 24:
            binary += format(ascii_code, '08b')
            binaryS += format(ascii_code, '08b') + ' '
            binaryC += format(ascii_code, '08b') + '\n'
            counter = 0
        else:
            binary += format(ascii_code, '08b')
            binaryS += format(ascii_code, '08b') + ' '
            binaryC += format(ascii_code, '08b')
            counter += 1
    return binary, binaryS, binaryC

# Load the document
filename = 'input.txt'
text = ''
with open(filename,'r') as i:
    text += i.readline()

binary_data, binaryS_data, binaryC_data = string_to_binary(text)

# binary_data is single line
# binaryS_data contain spaces to delimitate each character
# binaryC_data is split as chunks of 200 bits
```

In [3]:
```
######################
# Step 2
######################

# Create CRC for g(D) = [D16 + D12 + D5 + 1]
crc_str = '10001000000100001' # divisor
int_crc_str = int(crc_str,2)
frames = []

binary_chunks = binaryC_data.split('\n')
for chunk in binary_chunks[:-1]:
    temp = chunk + "0000000000000000"
    temp = int(temp,2)
    crc = temp % int_crc_str
    crc = format(crc, '016b')
    if len(crc) > 16:
        crc = crc[-16:]
    frames.append(chunk + crc)

print((frames[0]))
```

```
0100100101101110001000000011000100111000001101010011001100101100001000000111000001110010011011110110110101101001011011001100101011011100111010000100000010100110111010000101110001000000100110001101111111111000101110100
```

In [4]:
```
######################
# Step 3
######################
## Channel Coding **NEW**

import numpy as np

def encode_message(msg, matrix):
    code = np.mod(np.dot(msg, matrix), 2)
    return code
```

```
def decode_message(code, valid_codewords):
    incorrect_vec = []
    incorrect = 0
    for codeword in valid_codewords:
        for ind in range(len(codeword)):
            if codeword[ind] != code[ind]:
                incorrect += 1
        incorrect_vec.append(incorrect)
        incorrect = 0
    closest = np.min(incorrect_vec)
    closest = np.where(incorrect_vec == closest)[0][0]
    decoded_msg = valid_data[closest]
    return decoded_msg

G = np.array([[1,1,1,1,1,1,1,1],
              [0,0,0,0,1,1,1,1],
              [0,0,1,1,0,0,1,1],
              [0,1,0,1,0,1,0,1]])

valid_data = [[0,0,0,0], [0,0,0,1], [0,0,1,0], [0,0,1,1],
              [0,1,0,0], [0,1,0,1], [0,1,1,0], [0,1,1,1],
              [1,0,0,0], [1,0,0,1], [1,0,1,0], [1,0,1,1],
              [1,1,0,0], [1,1,0,1], [1,1,1,0], [1,1,1,1]]

valid_codewords = []
for data in valid_data:
    valid_codewords.append(encode_message(data,G))
print(valid_codewords)

############ Example ############
msg = np.array([0, 1, 1, 1])
print("Sent Message:", msg)
print()

code = encode_message(msg, G)
print("Encoded code:", code)
print()

decoded_msg = decode_message(code, valid_codewords)
print("Decoded message:", decoded_msg)
############ End Example ############

def array_to_string(array):
    output = ""
    for each in array:
        output += str(each)
    return output

def string_to_array(string):
    output = []
    for each in string:
        output.append(int(each))
    output = np.array(output)
    return output

def RM_string(input_string):
    output_string = ""
    ind = 0
    while(ind <= len(input_string)):
        dat = input_string[ind:ind+4]
        data_string = string_to_array(dat)
        if data_string.size != 0:
            data_string = data_string.reshape((1,4))
            encoded = encode_message(data_string,G)[0]
            output_string += array_to_string(encoded)
        else:
            output_string += str(dat[-1:])
        ind += 4
    return output_string

binaryRM = []
for frame in frames:
    binaryRM.append(RM_string(frame))

print(len(binaryRM[0]))
```

```
[array([0, 0, 0, 0, 0, 0, 0, 0]), array([0, 1, 0, 1, 0, 1, 0, 1]), array([0, 0, 1, 1, 0, 0, 1, 1]), array([0, 1, 1, 0, 0,
1, 1, 0]), array([0, 0, 0, 0, 1, 1, 1, 1]), array([0, 1, 0, 1, 1, 0, 1, 0]), array([0, 0, 1, 1, 1, 1, 0, 0]), array([0,
1, 1, 0, 1, 0, 0, 1]), array([1, 1, 1, 1, 1, 1, 1, 1]), array([1, 0, 1, 0, 1, 0, 1, 0]), array([1, 1, 0, 0, 1, 1, 0, 0]),
array([1, 0, 0, 1, 1, 0, 0, 1]), array([1, 1, 1, 1, 0, 0, 0, 0]), array([1, 0, 1, 0, 0, 1, 0, 1]), array([1, 1, 0, 0, 0,
0, 1, 1]), array([1, 0, 0, 1, 0, 1, 1, 0])]
Sent Message: [0 1 1 1]

Encoded code: [0 1 1 0 1 0 0 1]
```

```
Decoded message: [0, 1, 1, 1]
432
```

In [5]:

```
#######################
# Step 4
#######################

# flag = '01111110'
flag = '000111111111111111111111000' # tripled
framed_frames = []
bitstuff=0
for binRM in binaryRM:
    i = 0

    while(i <= (len(binRM) - 18)):
        if binRM[i:i+18] == '111111111111111111':
            binRM = binRM[:i+17] + '0' + binRM[i+17:] # bit stuff after any string of five 1s
            bitstuff+=1
            i += 18
        else:
            i += 1
    framed_frames.append(flag + binRM)

framed_frames.append(flag) # after last FEC frame, insert another flag
print('bits stuffed: '+str(bitstuff))
print(framed_frames[0])
```

```
bits stuffed: 0
000111111111111111111111000000011111010101000111100110000110011001100000000001100110010101010110011011111111101100110010110100
110011001100110001100111111100000011001100000000001101001000000000110100100110011001111001001011000111100101001010011110010
10101000111100110000110011110001011010001111001100001101101001000011110011001100000000001011010011001100110100100001111001
100111100001100110011000000000000001111111100000011110010010110100101101111111111001100111001100
```

In [6]:

```
#######################
# Sequence to Transmit
#######################

sequence = ''
for fram in framed_frames:
    sequence += fram
    # print(len(fram))

print(len(sequence))
```

```
20544
```

In [7]:

```
#######################
# Step 5: Error
#######################

import random

lim = {0.1: 9,
       0.01: 99}

rxbits = ''
errors = 0

for bit in sequence:
    rnum = random.randint(0,lim[prob])
    # rnum = 1 # no errors
    if rnum == 0: # error bit
        errors += 1
        if bit == '1':
            bit = '0'
        else:
            bit = '1'
    rxbits += bit
print("Number added random errors:", errors)
print(len(rxbits))
```

```
Number added random errors: 231
20544
```

In [8]:

```
#######################
# Step 5: Unstuffing
#######################

def unstuff(bits):
    i = 0
    while i < len(bits)-3:
        #print(bits[i:i+19])
        if bits[i:i+19] == '1111111111111111101':
            print(bits[i:i+19])
```

```
            bits = bits[:i+17] + bits[i+18:] # remove bit stuffing
            print(bits[i:i+19])
            i += 18
        else:
            i += 1
    return bits

rxbits = unstuff(rxbits)
print(len(rxbits))
```

```
20544
```

In [9]:

```
######################
# Step 5: Fixing
######################

corrected_bits = rxbits
# finding flags to reseparate frames
i = 0
start = -1
end = -1
new_frames = []
flag_flag = 0

while (i < len(corrected_bits)):
    if corrected_bits[i:i+24] == flag:
        flag_flag = 1
        if start != -1:
            end = i
            temp = corrected_bits[start:end]
            new_frames.append(temp)
            start = i
        else:
            start = i
    if flag_flag:
        i += 24
        flag_flag = 0
    else:
        i += 1

new_frames.append(corrected_bits[start:])

new_frames = new_frames[:-1]
print(new_frames[0])
print(len(new_frames[-2]))

# frames separated, unstuffed but need :: untripled, CRC checked, and then converted to text
```

```
000111111111111111111100000001111111010100011110011000011001100110000000001101110010101010110011011111111011001100101101001100110011000110001111100000011001100000000011010010000000001101001001100110011110010010110001111001010010100111100101010001111001100001100111110001011010001111001100001101101000000011010011001100001000010110100110011001101001000011110011001111000000000000011111111100000011110010010110100101101111111110011001110011001100
456
```

In [10]:

```
######################
# Step 5: Remove flag
######################

rxseqs = []
for nf in new_frames:
    rxseqs.append(nf[24:])

print(len(rxseqs[0]))
# need :: CRC checked, and then converted to text
```

```
432
```

In [11]:

```
######################
# Step 5: Decode RM
######################

temp_rxseqs = []
for frame in rxseqs:
    # frame = frame[:-8]
    temp_frame = ""
    count = 0
    while(count < len(frame)):
        message = string_to_array(frame[count:count+8])
        decoded_msg = decode_message(message, valid_codewords)
        temp_frame += array_to_string(decoded_msg)
        count += 8
    temp_frame += frame[count:]
    temp_rxseqs.append(temp_frame)
    print(temp_frame)
```

```
rxseqs = temp_rxseqs
# print(rxseqs)
print(len(rxseqs[0]))
```

```
01001001011011100010000000110001001110000001101010011001100101100001000000111000000111001001101111011011010110100101101110
01100101011011100111010000100000001010011011101000010111000100000001001100011011111111110001011101
0
011101010110100101110011001000000110110101100101011100100110001101101000011000010110111001110100001000000101011101100001
01111100101101101011000010110111000100000001000011011100100110111101110111001000000001011011100010
1
011000010110111001100100001000000110100001101001011100110010000000111000000110000101110011011101000110111101110010001011
000
0100000001010111011010010110110001101100011010010110000101101101001000000100011101011110111000
00
011100100110010101100101011011100110110001100101011000010110011000100000001000101011011000110100101101111011101000001000
00
1001010011100100010111000101100000100000011000110110111101101110011000110110010110010011011010
01
011100100110111001100101011001000010000000110000101100010011011110111010101110100001000000011101000110100001100101001000
000
1101100011000010110001101101011001000000110111101100110001000000110100101101110111100101010001
0
011100110111010001101001011101000111010101110100011010010110111101101110011100110010000000110111101100110001000000110100
00
1101001011001110110100001100101011100100010000000110110001100101011000010111001000011111000111
10
011011100110100101101110011001110010000000110100101101110001000000111010001101000011001010010000000110011101110010011011
110
11101110110100101101110011001110010000000110110101101001011001000111011101100101101110011010000
1
011100110111010000101100001000000110110001100101011001000010000000111010001101000011001010010000000110011001101111011101
010
1101110011001000110100101101110011001110010000000110111101100110001000000101011101100111101001
01
011000010111001101101000011010010110111001100111011101000110111101101110001000000101010101101110011010010111011001100101
010
11100100111001101101001011101000111100100100000011010010110111000100000001010011000000000111010
11
011101000010111000100000001001100011011110111010101101001011100110010111000100000001000100011101010111001001101001011011
100
1100111001000000111010001101000011001010010000000011000100111000000110100001100000111010111001
001
011100110010000000110000101101110011001000010000000011010100110000001110011001011000010000000111011101100001011101100110
01010
111001100100000001101111011001100010000000110100101101011011010110100101100111000011100000001110100100011000111001001100
0
010110111001110100011100110010000000110011001101100011011110110111101100100011001010110010000100000001101001011011100111
010
001101111100100000001010011011101000010111000100000001001100011011110001101010100110010010001100011101010110100101110011
0010
110000010000000110001001101111011011110111001101110100011010010110111001100111001000000011101000110100001100101001000000
01110
00001101111011100000011101010110110001100001011101000000111100110100
1
011010010110111101100000000100000001101111011001100010000000111010001101000011001010010000000111100101101111011101010110
11100
1100111001000000110001101101001011101000111100100101110001000000010101110110100100000001111001011
011101000110100000010000000111010001101000011001010111001101100101001000000110111001100101011101110110001101101111011011
010
1100101011100100111001100100000001100011011000010110110101100101001000000011000011001100001101
110
001000000011100000011100100110010101110011011100110110100101101110011001110010000000110111001100101011001010110010000100
0000
1100110011011110111001000100000001100101011001000111010101100011011000010111010011011100010101
11
011010010110111101101110001000000001011010010000000110001001101111011101000110100000010000000110100101101110011001000111
01010
1110011011101000111001001101001011000010110110000100000001110100011100100110000111111100000101
00
011010010110111001101001011011100110011100100000001100001011011100110010000100000001100010011000010111001101101001011000
110
010000000110011101100101011011100110010101110010011000010110110000100000001100011000010101011010
0
011011110111010101110010011100110110010101110011001000000001011010010000000110001101101111011011100110010001110101011000
110
1110100011001010110010000100000001101111011101010111010001110011011010010110010000000001101000
100
011001010010000000110111101100110001000000110111001101111011100100110101100001011011000010000000111011101101111011100
100
1101011011010010110111001100111001000000011010000110111101110101011100100111001111010110001011
11
001011100010000000101001101101111001000000011101000110100001100101001000000110011001101001011100100111001101110100001000
000
1100101011001000111010101100011011000010111010001101001011011110110111001100001011010111000100101001000110001101100001000
0001110011011101000110010101110000000100000011011110110011000100000001110100011010000110010100100000011110010110111101110
10
101101110011001110010000000101011101100001011100110110100001101001011100010001101001
011011100110011101110100001101111011011100010000000101010101101110011010010111011001100101011100100111001101101001011101
000
11110010010000000111011101100001011100110010000000111010001101111001000000011001011110011100111000
0111001101110100011000010110001001101100011010010111001101101000001000000110000101101110001000000110010101111011001100
1010
1101110011010010110111001100111001000000011100000011100100110111101100111011100100010010100011111
0110000101101101010000000110111101101110001000000100111101100011011101000110111101100010011001010111001000100000001100
100
011001000101100000100000001100010011100000011010100110100001011100010000000100111111100111000100110
0111011001100101011100100010000000111010001101000011001010010000000111001101110101011000110110001101100101011001010110010
00
11010010110111001100111001000000110010001100101011000110110000101100100011001010011011100111010
011100110010110000100000001110100011010000110010100100000011000110110111101101110011101000110100101101110011101010110100
10
110111001100111001000000110010101100100011101010110001101100001011101000110100110001100011100000
011011110110111000100000001110000011100100110111101100111011100100110000101101101001000000111010101101110011001000110010
10
111001001110111011001010110111001110100000001101101011000010110111001111001011101010100011001001000011000100000011000
1101101000011000010110111001100111011001010111001100101110001000000101010001101000011001010010000000111010101101110011010
0
1011101100110010101110010011100110110100101110100001111001001000000110000101010111
011001100110110001101111011101010111001001101001011100110110100001100101011001000010000000110000101110100001000000011010010
11101000111001100100000001101100011011110110110000101110100011010010110111101110111011100111000110001100110111000100
0
00011010010110111000100000001100100011011110111011101101110011101000110111101110111011011100010000000101001101110100000101
11
000100000001001100011011110111010101101001011100110010000000110011000111111101110010
01101111011100100000001101001011101000111001100100000001100110011010010111001001110011011101000010000000110101001100000
0
010000000111100101100101011000010111001001110011001011000010000000110011101110010011001010101100
01101111011101110110100101101110011001110010000000110011001110010011011110110100100000001100001011011100010000000110010
10
111011001100101011011100110100101101110011001110010000000111000000111001001101111100000010101001010010001100011001110111
00
100110000010110110100100000001110100011011110010000000110000101101110001000000110100101101110011100110111010001101001011101
0
0011101010111010001101001011011110110111000100000001101111011001100001011100000001
011001100110010101110010011010010110111001100111001000000110000100100000001100110011101010110110001101100001000000111001
10
110110001100001011101000110010100100000001101111011001100010000000111001101100011111011100000011
0110100101100101011011100111010001101001011001100110100101100011001000000110110001101001011000100110010101110010011100
100
1100000101101100000100000001100001011100100111010001110011001000000011000010110111000000111001001010
01100100001000000001100011011011000110000101110011011100110110100101100011011000010110110000100000001100011011011110111010
10
111001001110011011001010100000001101111011001100110011001100101011100100110100100010111111110011
0110111001100111011100110010111000100000001001001011011100010000000111010001101001011101101011001010010110000010000000111001
10
11000110110100001101111011011110110110001110011001000000110111101100110001000000111101110110100010010001100011011000110
00
010111011100100000001100001011011100110010000100000001100110011010010111001001101001011011100110010100100000001100001011100
100111010001000111001
100100000001110111011001010111001001100101001000000110000101100100010001111111101101
0110010001100101011001000010111000100000001001001011011100010000000110001001110000001110010011000100101100000100000001110
1000
1101000011001010010000000111001101100011011010000110111101101111011011000111001100100000000001101100100
01
01100001011000110111000101110101011010010111001001100101001000000111010001101000011001010010000000101001101110
1000
010111000100000001001100011011110111010101101001011100110010000000100110101100101100000010110111100100100011000110010001
1010
010110001101100001011011000010000000100001101101111011011000110110001100101011001110110010100100000011101000110111100100
00
0011001100110111101110010011010010100000001100001001000000110110101101100010000000100100011000110010101100100011010010110
0011011000010110110000100000011001000110010101110000011000010111001001110100011011010110010101101110011101000010110000
100
00001110111011010000110100101100011011010000001000000101011010001101
216
```

In [12]:
```
######################
# Step 5: Check CRC
######################

# Create CRC for g(D) = [D16 + D12 + D5 + 1]
crc_str = '10001000000100001' # divisor
int_crc_str = int(crc_str,2)
failed = 0
total = 0

for rxs in rxseqs:
    crc_rx = rxs[-16:]
    beginning = rxs[:-16] + "0000000000000000"
    beginning = int(beginning,2)
    crc = beginning % int_crc_str
    crc = format(crc,'016b')
    if len(crc) > 16:
        crc = crc[-16:]
    if crc != crc_rx:
        print("crc: "+str(crc)+" crc_rx: "+str(crc_rx))
        failed += 1
    total += 1

print('Number of Failed Frames with Error Probability ' + str(prob) + ': --- ' + str(failed) + ' --- \n')
print('Failed Frames: ' + str((failed / total) * 100) + '%' + ' of ' + str(total) + ' total frames recieved.')
```

```
crc: 1010010100010101 crc_rx: 1001001101101001
crc: 1110001111100011 crc_rx: 0001111001101001
crc: 1101111111010110 crc_rx: 0000001111001011
crc: 0100001011110111 crc_rx: 1100010001101001
crc: 1100011000001101 crc_rx: 0110000101010111
crc: 0111110101100000 crc_rx: 0011111101110010
crc: 0011000101101011 crc_rx: 0001011100000001
crc: 1100000100110111 crc_rx: 0100011111101101
crc: 0001100111010101 crc_rx: 0101011010001101
Number of Failed Frames with Error Probability 0.01: --- 9 ---

Failed Frames: 26.47058823529412% of 34 total frames recieved.
```

In [13]:
```
######################
# Convert Back to Text
######################

all_rx_bits = ''
char_arr = []
paragraph = ''
i = 0

for rxs in rxseqs:
    if len(rxs[:-16]) < 200:
        continue
    else:
        all_rx_bits += rxs[:-16]

while i < len(all_rx_bits)-7:
    char_arr.append(all_rx_bits[i:i+8])
    i += 8

for cb in char_arr:
    character = chr(int(cb,2))
    if (character!='~'):
        paragraph += character

print(paragraph)
```

```
In 1853, prominent St. Louis merchant Wayman Crow and his pastor, William Greenleaf Eliot Jr., concerned about the lack of institutions of higher learning in the growing midwest, led the founding of Washington University in St. Louis. During th e 1840s and 50s, waves of immig HÇ& çG2 fÆööFVB çFò 7Bâ Æñªd uis, boosting the populatio` of the young city. With these n ewcomers came a pressing need for education - both industrial training and basic general courses - conducted outside of n ormal working hours. So the first educationak HÆÂ 7FW öb F R ÷Vær v 6 æwFöâ Væ fW'6 G v 2 Fò W7F &Æ 6 â WfVæ ær &öw & Ò öâ ö7Fö&W" #"Â SBâ ÷fW" F R 7V66VVF ær FV6 FW2Â F R 6öçF çV ær VGV6 F öâ &öw& Ò VæFW'vVçB Ö ç Td changes. The univers ity flourished at its locatiovç Æâ â F÷vçF÷vâ 7Bâ Æ÷V 2 f÷" G2 f '7B S V '2Â w&÷v ær g&öÒ â WfVæ ær &ø* gram to an in stitution offering a full slate of scientific, liberal arts and classical course offerings. In time, schools of {´HÆÆ r æ B f ær 'G2 vW&R FFVBâ â Â F R 66 ööÂ 7 V &VB F R 7Bâ Æ÷V 2 ÖX-ä dical College to form a ml@HÆVF 6 Â FW 'FÖVçB Â v 6
```

In [14]:
```
# In 1853, prominent St. Louis merchant Wayman Crow and his pastor, William Greenleaf Eliot Jr., concerned about
# the lack of institutions of higher learning in the growing midwest, led the founding of Washington University
# in St. Louis. During the 1840s and 50s, waves of immigrants flooded into St. Louis, boosting the population of
# the young city. With these newcomers came a pressing need for education - both industrial training and basic
# general courses - conducted outside of normal working hours. So the first educational step of the young
# Washington University was to establish an evening program on October 22, 1854. Over the succeeding decades,
# the continuing education program underwent many changes. The university flourished at its location in downtown
```

```
# St. Louis for its first 50 years, growing from an evening program to an institution offering a full slate of
# scientific, liberal arts and classical course offerings. In time, schools of law and fine arts were added. In
# 1891, the school acquired the St. Louis Medical College to form a medical department, which merged with the
# Missouri Medical College in 1899.
```