CS 2400 : Assignment 2

Rella Hemanth Reddy CS14B024

Madisetty Uday Theja CS14B044

Problem Description :

In this assignment , we study the probability of occurrence of unique symbols in arbitrarily large text data and compute the entropy. Later we source code the data using Huffman coding algorithm and generate a code book for the symbols in the text data .

Later we divide the data into packets of some fixed length(7 in our case) and channel encode each packet . Then , we assume some arbitrary bit rate error and generate bit errors in each packet . Then , we utilise the client server programs provided to send the signal across the channel. We then recover the text from the received data and check whether there is an error or not by using the Cyclic Redundancy Check(CRC) and request for a re-transmission if it has an error . We do so until we generate the same text data as the input file .

Part I : Calculating Probabilities and Entropy

For doing this , we consider an array of size 256 . We chose this number because this is the number of unique symbols implemented in C without counting the special symbols . We scan through the file and store the frequency of the symbol in the index of the array equal to its ASCII value. This process is done until we reach the end of the file .

Now , the summation of each index of the array gives us the Sample Space (say $Count_{Total}$) . Let $Count_{Index}$ give the total number of occurrences of a symbol with ASCII value as Index , then Probability of Occurrence of character can be given by the formula ,

$$P_{Index} = (Count_{Index})/(Count_{Total})$$

After calculating probability , the entropy can be calculated by using the relation ,

$$\mathrm{H}(X) = \sum_{i=1}^{n} \mathrm{P}(x_i)\,\mathrm{I}(x_i) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i),$$

Where H(X) is the entropy of X which is the text data and $I(x_i)$ is the information content of x_i .

Part II : Huffman Coding Algorithm and generating the Code Book

Using Huffman coding we generate a codebook corresponding to the characters in the text data using the probabilities . Huffman code uses the fact that the symbol occurring the most number of times must be represented by least number of bits possible in order to transmit the information using the least possible bits which also takes care of the fact that channel noise can be minimised using less number of bits for a symbol that occurs many times .

Part III : Generating Bit errors for each packet

We chose some small arbitrary bit error which in real life signifies the error due to the channel in reading a particular data . Later we generate a random number between 0 and 1 and compare it with the arbitrary bit error initially fixed . If the random error is less than the fixed error , we revert the bit . Otherwise , we transmit the same bit . The same thing we do for all the bits in the packet and transmit the packet across the channel using the UDP Client Server programs .

Part IV : Error Checking

From the data transmitted across the channel , we recover the text and check whether it has an error or not . We do this by implementing the Cyclic Redundancy Check and ask for a re-transmission if we come across any error . We do this until we reach a state of no error is reached . That is , a state where the CRC gives a remainder 0 .

Observations :

Entropy :

Entropy is the minimum average length of codeword . We check this by calculating the average code length of all symbols after codebook is generated . This is compared with the theoretical value of Entropy . We observe that the Huffman coding algorithm provides more than 95 % efficiency in this case .

We also make use of the fact that Huffman coding algorithm is uniquely decodable . We find the use of this fact in the writing the decode function.

Note that the two codebooks and efficiencies generated below are for two different files with different file sizes .

```
Huffman Codebook:
e: 000
s: 0010
y: 0011000
-: 00110010
x: 0011001100
1: 0011001101
(: 0011001110
F: 0011001111
w: 001101
m: 00111
n: 0100
r: 0101
h: 0110
v: 0111000
6: 0111001000
S: 0111001001
): 0111001010
5: 0111001011
K: 01110011
u: 011101

: 0111100
,: 0111101
P: 0111110000
R: 0111110001
G: 011111001
T: 01111101
b: 01111110
E: 011111110
I: 011111111
a: 1000
l: 10010
d: 10011
t: 1010
i: 1011
f: 110000
p: 110001
g: 11001
.: 1101000
k: 11010010
D: 1101001100
9: 11010011010
:: 11010011011
': 110100111
c: 110101
o: 11011
 : 111

Entropy: 4.335738
Avg word length: 4.369603
Efficiency: 99.224983 %
```

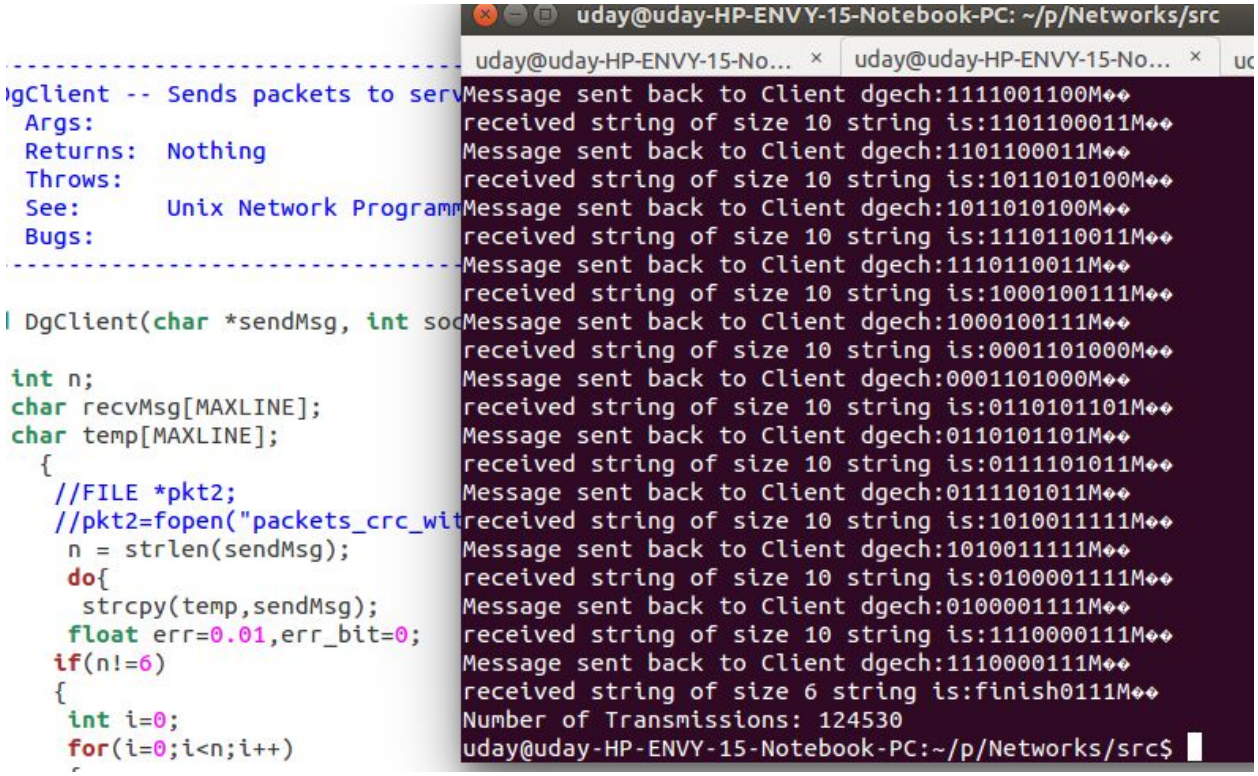
```
c: 10010
f: 100110
b: 100111
d: 10100
y: 101010
C: 1010110
D: 1010111000
E: 1010111001
U: 1010111010
7: 10101110110
z: 10101110111
P: 101011110
:: 101011111
t: 1011
m: 110000
.: 1100010
k: 11000110
4: 11000111
h: 11001
x: 11010000
v: 11010001
p: 1101001
u: 110101
s: 11011
o: 11100

: 1110100
H: 1110101000
L: 1110101001
R: 111010101
T: 111010110
M: 11101011100
6: 11101011101
-: 11101011110
5: 11101011111
l: 111011
i: 11110
I: 111110000
(: 1111100010
j: 11111000110
F: 11111000111
,: 111110010
2: 111110011
S: 111110100
0: 111110101
): 1111101100
1: 1111101101
A: 1111101110
B: 1111101111
g: 1111110
W: 111111100
q: 111111101
w: 11111111
```

Number of transmissions and Error Probability :

We see that upon increasing the error probability , the number of re-transmissions increase as each packet now has a greater chance to have an error .

Note the error probability implemented in the background code.



```
--------------------------------
gClient -- Sends packets to serv
 Args:
 Returns:  Nothing
 Throws:
 See:      Unix Network Programm
 Bugs:
--------------------------------

 DgClient(char *sendMsg, int soc

int n;
char recvMsg[MAXLINE];
char temp[MAXLINE];
  {
   //FILE *pkt2;
   //pkt2=fopen("packets_crc_wit
    n = strlen(sendMsg);
    do{
     strcpy(temp,sendMsg);
    float err=0.15,err_bit=0;
   if(n!=6)
   {
    int i=0;
    for(i=0;i<n;i++)
```

```
uday@uday-HP-ENVY-15-Notebook-PC: ~/p/Networks/src
uday@uday-HP-ENVY-15-No... × uday@uday-HP-ENVY-15-No... ×
Message sent back to Client dgech:1100111110;U◆
received string of size 10 string is:1100001111;U◆
Message sent back to Client dgech:1100001111;U◆
received string of size 10 string is:0100001111;U◆
Message sent back to Client dgech:0100001111;U◆
received string of size 10 string is:1110000101;U◆
Message sent back to Client dgech:1110000101;U◆
received string of size 10 string is:1110001001;U◆
Message sent back to Client dgech:1110001001;U◆
received string of size 10 string is:1110000101;U◆
Message sent back to Client dgech:1110000101;U◆
received string of size 10 string is:0110000111;U◆
Message sent back to Client dgech:0110000111;U◆
received string of size 10 string is:0110000111;U◆
Message sent back to Client dgech:0110000111;U◆
received string of size 10 string is:0110000111;U◆
Message sent back to Client dgech:0110000111;U◆
received string of size 10 string is:1110000110;U◆
Message sent back to Client dgech:1110000110;U◆
received string of size 10 string is:1110000111;U◆
Message sent back to Client dgech:1110000111;U◆
received string of size 6 string is:finish0111;U◆
Number of Transmissions: 466823
uday@uday-HP-ENVY-15-Notebook-PC:~/p/Networks/src$
```

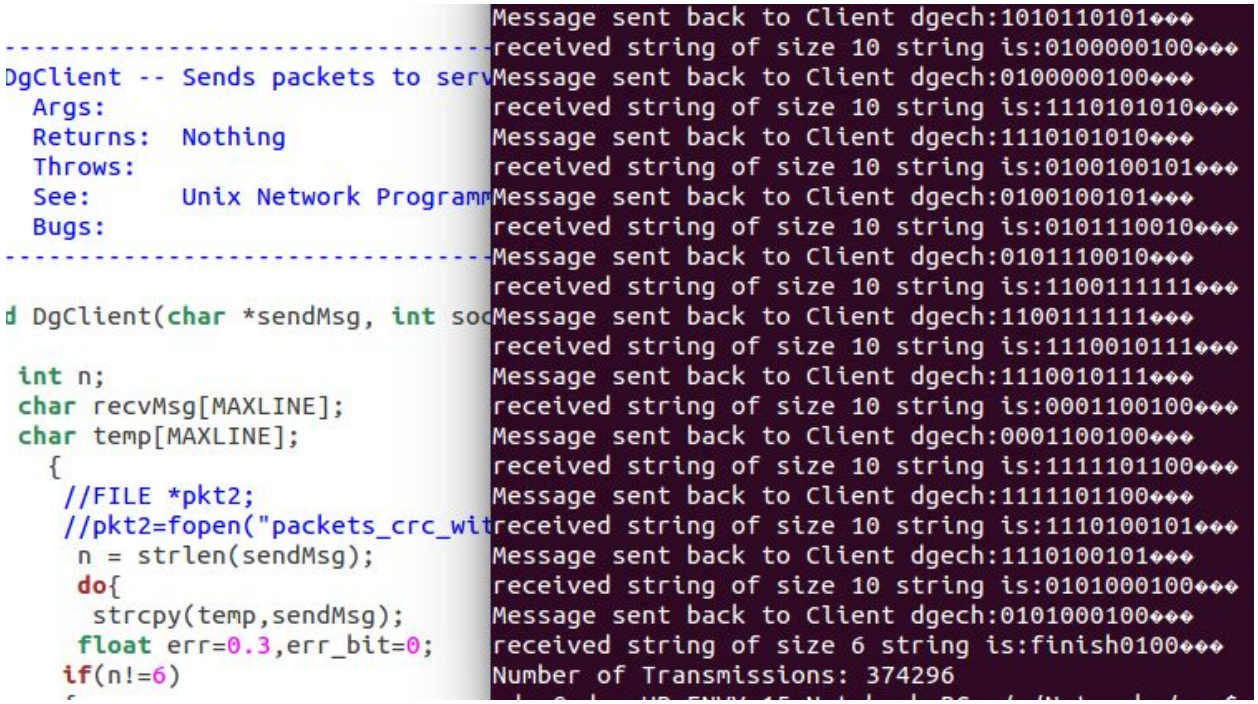
```
--------------------------------
Client -- Sends packets to serv
 Args:
 Returns:  Nothing
 Throws:
 See:      Unix Network Programm
 Bugs:
--------------------------------

 DgClient(char *sendMsg, int soc

nt n;
har recvMsg[MAXLINE];
har temp[MAXLINE];
  {
   //FILE *pkt2;
   //pkt2=fopen("packets_crc_wit
    n = strlen(sendMsg);
    do{
     strcpy(temp,sendMsg);
    float err=0.3,err_bit=0;
   if(n!=6)
   {
    int i=0;
    for(i=0;i<n;i++)
```

```
uday@uday-HP-ENVY-15-Notebook-PC: ~/p/Networks/src
uday@uday-HP-ENVY-15-No... × uday@uday-HP-ENVY-15-No... ×
Message sent back to Client dgech:1111111111e◆◆
received string of size 10 string is:0110000110e◆◆
Message sent back to Client dgech:0110000110e◆◆
received string of size 10 string is:1100000100e◆◆
Message sent back to Client dgech:1100000100e◆◆
received string of size 10 string is:1101110011e◆◆
Message sent back to Client dgech:1101110011e◆◆
received string of size 10 string is:0010000100e◆◆
Message sent back to Client dgech:0010000100e◆◆
received string of size 10 string is:1100100111e◆◆
Message sent back to Client dgech:1100100111e◆◆
received string of size 10 string is:1010010101e◆◆
Message sent back to Client dgech:1010010101e◆◆
received string of size 10 string is:1110110101e◆◆
Message sent back to Client dgech:1110110101e◆◆
received string of size 10 string is:0101000111e◆◆
Message sent back to Client dgech:0101000111e◆◆
received string of size 10 string is:1110000011e◆◆
Message sent back to Client dgech:1110000011e◆◆
received string of size 10 string is:1100100101e◆◆
Message sent back to Client dgech:1100100101e◆◆
received string of size 6 string is:finish0101e◆◆
Number of Transmissions: 864516
uday@uday-HP-ENVY-15-Notebook-PC:~/p/Networks/src$
```

The screenshot below is the number of transmissions for different input file .



```
--------------------------------
DgClient -- Sends packets to serv
  Args:
  Returns:  Nothing
  Throws:
  See:      Unix Network Programm
  Bugs:
--------------------------------

d DgClient(char *sendMsg, int soc

 int n;
 char recvMsg[MAXLINE];
 char temp[MAXLINE];
   {
    //FILE *pkt2;
    //pkt2=fopen("packets_crc_wit
     n = strlen(sendMsg);
     do{
      strcpy(temp,sendMsg);
     float err=0.3,err_bit=0;
    if(n!=6)
```

```
Message sent back to Client dgech:1010110101◆◆◆
received string of size 10 string is:0100000100◆◆◆
Message sent back to Client dgech:0100000100◆◆◆
received string of size 10 string is:1110101010◆◆◆
Message sent back to Client dgech:1110101010◆◆◆
received string of size 10 string is:0100100101◆◆◆
Message sent back to Client dgech:0100100101◆◆◆
received string of size 10 string is:0101110010◆◆◆
Message sent back to Client dgech:0101110010◆◆◆
received string of size 10 string is:1100111111◆◆◆
Message sent back to Client dgech:1100111111◆◆◆
received string of size 10 string is:1110010111◆◆◆
Message sent back to Client dgech:1110010111◆◆◆
received string of size 10 string is:0001100100◆◆◆
Message sent back to Client dgech:0001100100◆◆◆
received string of size 10 string is:1111101100◆◆◆
Message sent back to Client dgech:1111101100◆◆◆
received string of size 10 string is:1110100101◆◆◆
Message sent back to Client dgech:1110100101◆◆◆
received string of size 10 string is:0101000100◆◆◆
Message sent back to Client dgech:0101000100◆◆◆
received string of size 6 string is:finish0100◆◆◆
Number of Transmissions: 374296
```

File compression :

Any ASCII character can be represented by a set of 8 bits as there are a total of 256 ASCII characters . Hence after decoding is done , every set of 8 bits is taken and stored as a character which results in significant decrease in file size .

The screenshots generated below are for two different files.

```
uday@uday-HP-ENVY-15-Notebook-PC:~/p/Networks/src$ gcc channelcoding.c -lm
uday@uday-HP-ENVY-15-Notebook-PC:~/p/Networks/src$ ./a.out
extra_bits: 2
number_of_packets: 112766
input_size: 180.647995 KB
huffman_encoded_size: 98.669998 KB
compression: 45.379967 %
uday@uday-HP-ENVY-15-Notebook-PC:~/p/Networks/src$
```

```
uday@uday-HP-ENVY-15-Notebook-PC:~/p/Networks/
uday@uday-HP-ENVY-15-Notebook-PC:~/p/Networks/
extra_bits: 0
number_of_packets: 48601
input_size: 77.672997 KB
huffman_encoded_size: 42.525875 KB
compression: 45.250118 %
```