CS 2400: Assignment 2

Group 7
Rahul Kejriwal CS14B023
Sai Pavan Dronavalli CS14B041

---

1. Experiment:

The following procedure was followed for communication:

1. We accept input text file (>64 kB). Store the input file as ‘test’ inside input directory.

   Note:
   a. We are using ‘ ` ’ as a special character in our code and we do not allow the use of this character in the text file.

2. The code first append ‘ ` ’ as an EOF delimiting character for convenience.
   It then reads through the file, counting the frequency of different characters in a tree node object attribute.

   We use the frequency information to compute the entropy and output the frequency details along with the entropy to output file ‘symbollist’.

3. We then add all the nodes to a Queue and use the Huffman algorithm to construct a Huffman tree from the tree nodes created earlier.

4. Using the Huffman tree, we then create a codebook and also store it in output file ‘codebook’.

5. Then, we use our codebook to translate our input file into Huffman Code. We output this result to output file ‘srccoded’.
   We generally observe a compression of ~45%.

6. We then do channel coding, using CRC. We add 3 redundancy bits for CRC for 5 data bits. We use the generating polynomial ‘1011’. We then transmit in packets of 8 bits.
   The channel coded input is stored in output file ‘chanelcoded’.

We observe that:

$$channel\ coded\ file\ =\ 8/5 * Huffman\ coded\ file$$

7. We then use modified versions of the provided code for transmitting and receiving the channel coded file.

8. The Client program transmits 8 bit packets to the Server and waits for an acknowledgment message. If the acknowledgment message indicates error in transmission, the Client retransmits the same packet.

   Note:

   a. We are sending packets sequentially. Ideally, we'd like to send packets *asynchronously* for better performance.
   b. Each packet is represented by a single 8-bit character being transmitted.
   c. Transmission of 0 (ASCII value of NULL) leads to errors at the Server side. Thus, we transmitted additional meta-information along with the packet to indicate whether the packet was a 0 and in that case we actually transmit any character (don't care) in the packet.

9. To simulate channel errors, we introduce errors in each bit with probability, P. This P is defined as a constant and can be varied to study the effect on the successful transmission.

   Note:

   a. $Prob.\ of\ successful\ packet\ transmission, P' = (1-P)^n,\ where\ n\ is\ packet\ size$
      $Prob.\ of\ unsuccessful\ packet\ transmission, P'' = 1 - P' = 1 - (1-P)^n$
      In our case, $n = 8$. Hence, the probability of successful packet transmission is significantly smaller than successful transmission of a single bit.

10. The modified server program receives packets and tries to verify the integrity of the packet by computing its CRC.

    If the CRC computed using generating polynomial '1011' turns out to be 0, the data is intact and we extract the 5 message bits and send an acknowledgement indicating success.

    Else we transmit an acknowledgment indicating transmission failure. We wait for successful retransmission and redo the above procedure until we can

successfully extract the 5 message bits.

11. The received message with extracted message bits is then stored in output file recvdWoutCRC. We then use the codebook to decode the Huffman coded message and reobtain the original input message.

12. Lastly, we run a diff on the input and output to see if they match. We observe that they do match.

2. Observations:

2.1 Part 1:

We study the variation of output file sizes and no. of errors wrt input file size first.

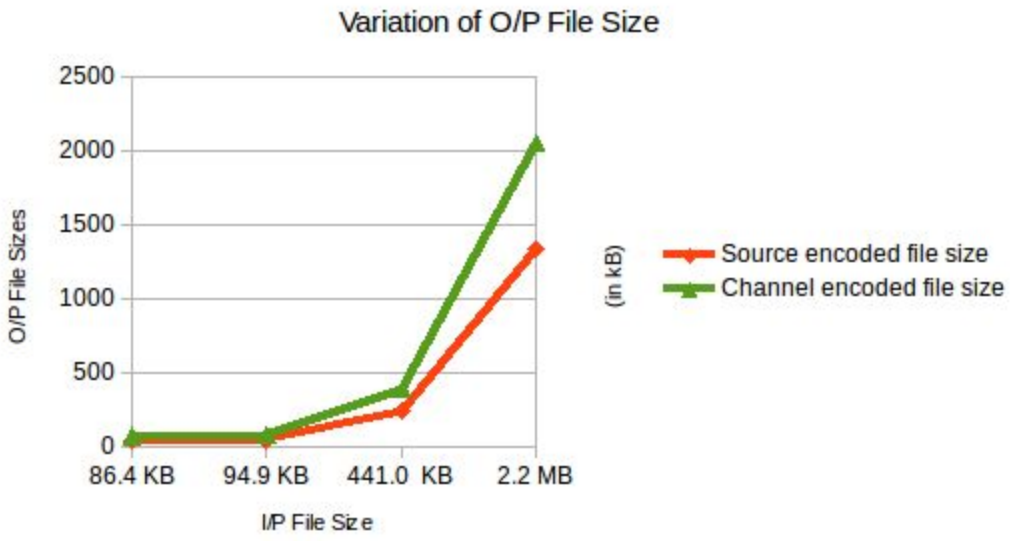
Table 1: We vary input file size keeping $P = 0.008 \ and \ P'' = 0.062$.

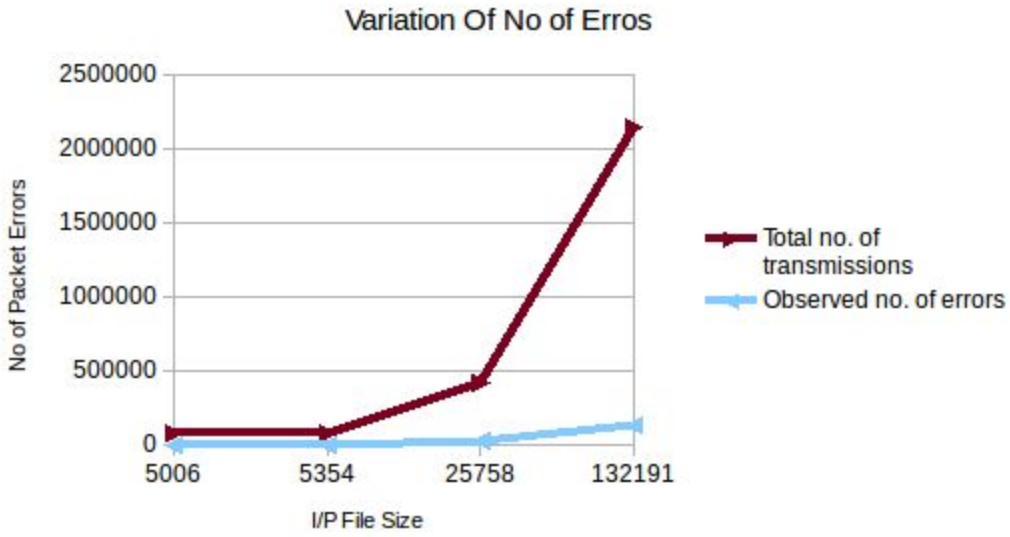
Input file size	Entropy	Source encoded file size	% compression	Channel encoded file size	Total no. of transmissions	Observed no. of errors	Observed prob. of packet wise errors P''
86.4 KB	4.369	47.4 KB	45.14 %	75.9 KB	80870	5006	0.0619
94.9 KB	4.246	50.7 KB	46.58 %	81.2 KB	86528	5354	0.0618
441.0 KB	4.408	245.0 KB	44.44 %	392.1 KB	417812	25758	0.0616
2.2 MB	4.556	1.3 MB	40.90 %	2.0 MB	2138516	132191	0.0618

Note :

1. We are assuming very small bit flip probability (of order of 0.01). Hence, for all practical reasons, the probability of more than 2 bit errors is almost 0 and we can neglect cases where CRC computes to 0 despite the message being incorrectly transmitted.
2. The bit error probability we assume is of order of 10^{-2} which is still much larger than 10^{-5} which is generally observed in real systems. We have used such high P in order to meaningfully study variation of error wrt file size. Else we would have to use much larger files to get any appreciable no. of errors.

Graphical Representation of Observed Data:





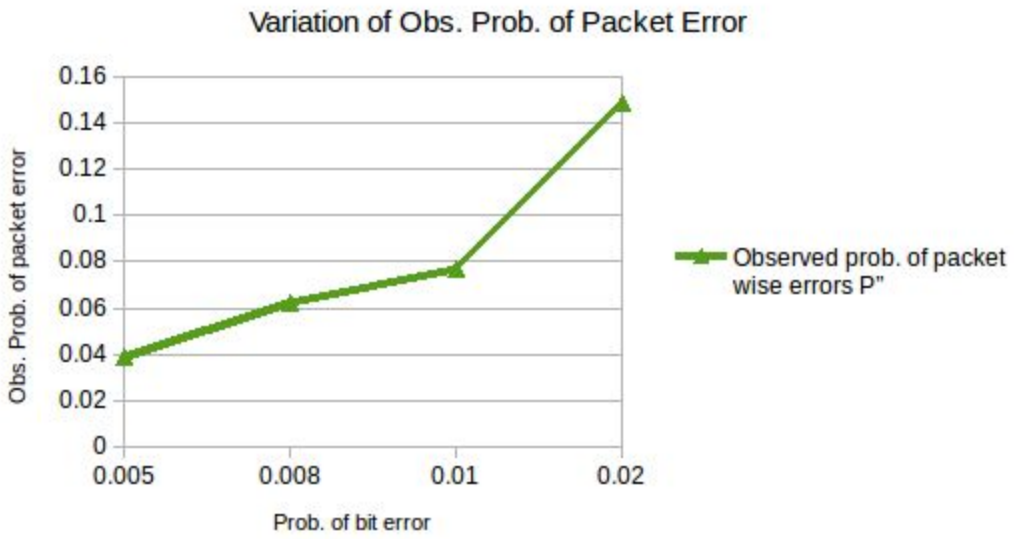
2.2 Part 2:

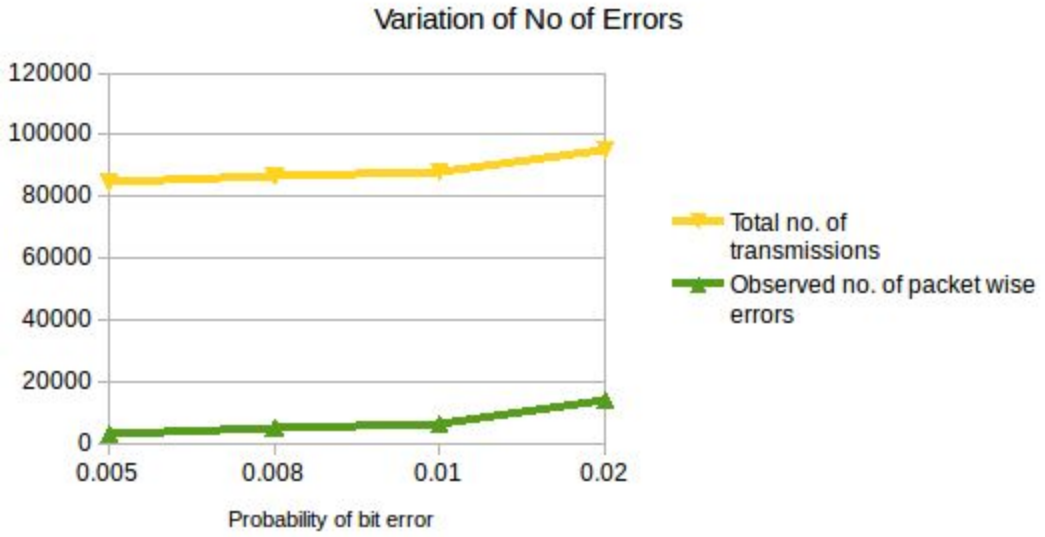
We study the variation of no. of errors wrt probability of error.

Table 2: We vary probability of error keeping input file size constant at $i/p\ file\ size = 94.9\ KB$.

Prob. of bitwise error P	Prob. of packet wise errors P''	Total no. of transmissions	Observed no. of packet wise errors	Observed prob. of packet wise errors P''
0.005	0.039	84478	3304	0.0391
0.008	0.062	86528	5354	0.0619
0.01	0.077	87948	6774	0.0770
0.02	0.149	95292	14118	0.1481

Graphical Representation of Observed Data:





2.3 Part 3:

We study the same by varying both parameters simultaneously.

Table 3:

Input file size	Entropy	Source encoded file size	% compression	Channel encoded file size	Input prob. Of packet wise errors	Total no. of transmissions	Observed no. packet wise errors	Observed prob. of packet wise errors
86.4 KB	4.369	47.4 KB	45.14 %	75.9 KB	0.077	82187	6323	0.0769
94.9 KB	4.246	50.7 KB	46.58 %	81.2 KB	0.149	95292	14118	0.1481
441.0 KB	4.408	245.0 KB	44.44 %	392.1 KB	0.039	407885	15831	0.0388
2.2 MB	4.556	1.3 MB	40.90 %	2.0 MB	0.062	2138516	132191	0.0618