CMPUT313 – Assignment 1 Report
Mikus Lorence
ID: 1227388

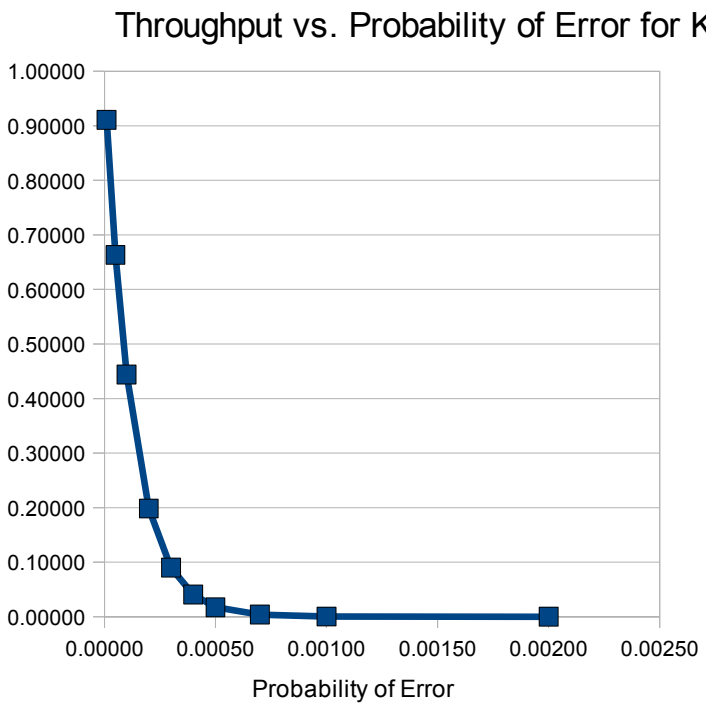
After completing the simulation I began to do some informal tests. I observed that for the special cases where K is 0 and 1, the throughput was exceptionally low, especially when I increased the error probability **e** to values such as 0.0005. With such an error rate, and no HSBC applied to the frame (K = 0), the throughput is a really low value of ~0.07 and the average number of frame transmissions is ~20.8, which means that on average the frame has to be retransmitted around 21 times. By increasing the number of blocks to 1, essentially enabling HSBC, the throughput rises to ~0.085, and the average number of frames transmitted drops to almost half of the previous value, ~11.73. This is expected because with bit error correction enabled, all the cases where there is one error in the frame can be corrected, and with a high error rate such as 0.0005, that eliminates many of the retransmissions.

I then ran more tests with fixed error values while alternating values for K. I chose the following 10 set values for **e**: 0.00001, 0.00005, 0.0001, 0.0002, 0.0003, 0.0004, 0.0005, 0.0007, 0.001, 0.002. I focused more on the value between 0.0001 and 0.001 because I noticed that that was the range of e for which the results went from great to terrible. At an error probability of 0.002 and above, the throughput was 0 for each trial because all blocks were getting more than 1 error. For **F, A,** and **T,** I used the "default" values of 8000, 100, and 5. I used 100,000,000 for the simulation runtime **R** to ensure the simulation reached equilibrium and had accurate results. I ran these sets of tests with fixed **e** values over a set of the following **K** values: 0, 1, 5, 10, 100, 1000, 8000. So the input values were as follows:

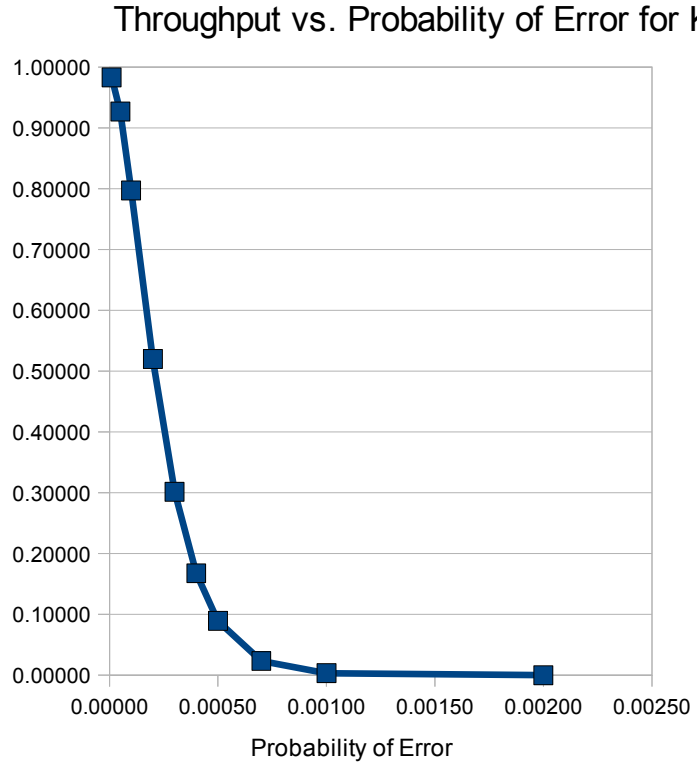
A: 100
K: {0, 1, 5, 10, 100, 1000, 8000}
F: 8000
e: {0.00001, 0.00005, 0.0001, 0.0002, 0.0003, 0.0004, 0.0005, 0.0007, 0.001, 0.002}
R: 100,000,000
T: 5

For the seeds for each trial and each set of tests, I ensured that no trial in the entire tests set had the same seed. So I started with seed 1 and just incremented it for each trial, for each set of tests.

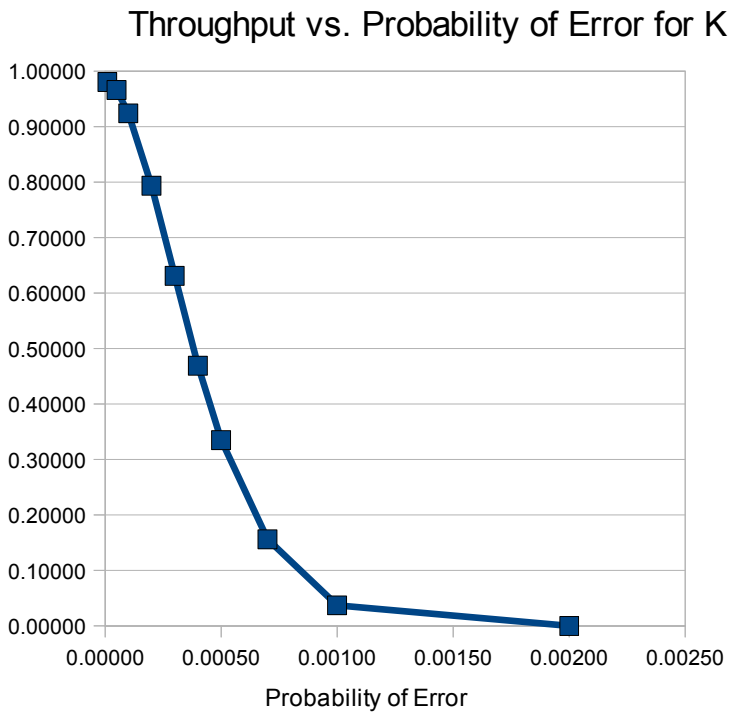
The graphs of throughput vs. e for each value of K, followed by a table with the corresponding Confidence Intervals are on the following pages:



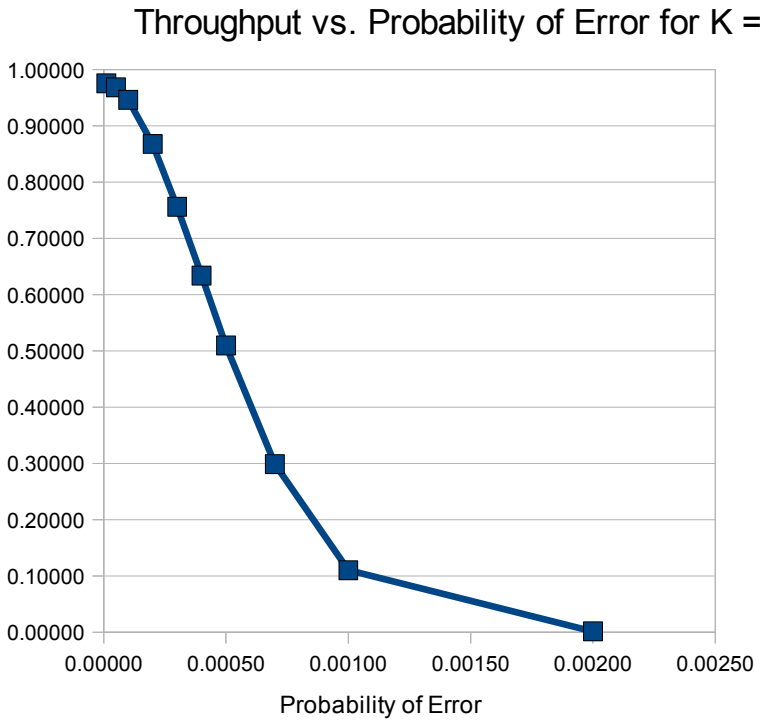
e	Throughput	CI
0.00001	0.90486	0.91716
0.00005	0.64685	0.67965
0.00010	0.43679	0.45177
0.00020	0.19187	0.20495
0.00030	0.08487	0.09517
0.00040	0.03749	0.04351
0.00050	0.01339	0.02140
0.00070	0.00232	0.00549
0.00100	0.00003	0.00071
0.00200	0.00000	0.00000



e	lower bound	upper bound
0.00001	0.98219	0.98384
0.00005	0.92416	0.92954
0.00010	0.79000	0.80442
0.00020	0.50236	0.53727
0.00030	0.29184	0.31163
0.00040	0.15937	0.17592
0.00050	0.07990	0.09845
0.00070	0.01971	0.02695
0.00100	0.00124	0.00535
0.00200	0.00000	0.00000

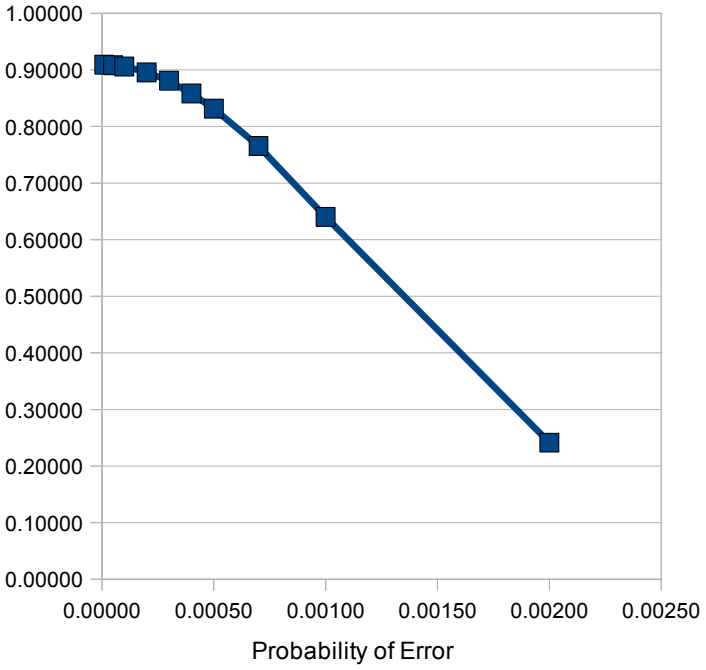


e	lower bound	upper bound
0.00001	0.97971	0.98106
0.00005	0.96451	0.96769
0.00010	0.91851	0.92958
0.00020	0.78296	0.80388
0.00030	0.62213	0.64035
0.00040	0.45318	0.48496
0.00050	0.33076	0.33960
0.00070	0.14795	0.16438
0.00100	0.02908	0.04507
0.00200	-0.00009	0.00031



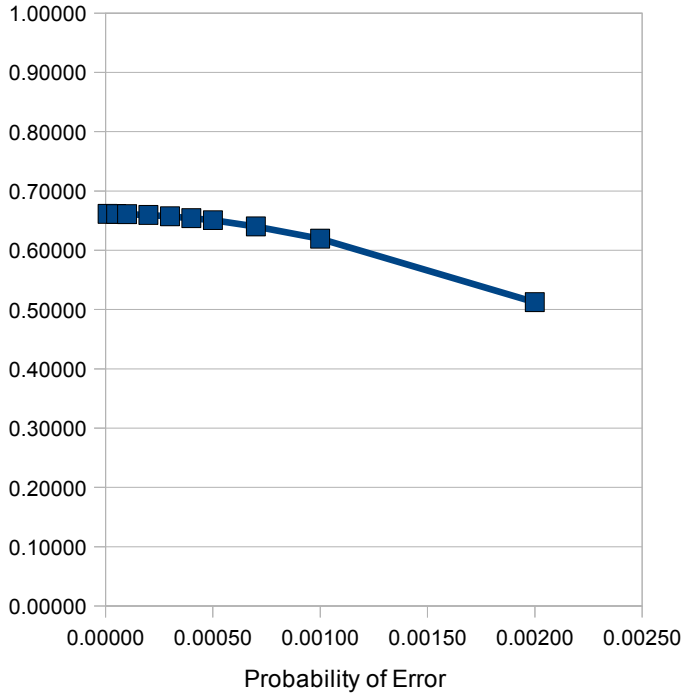
e	lower bound	upper bound
0.00001	0.97505	0.97579
0.00005	0.96653	0.97025
0.00010	0.94122	0.95073
0.00020	0.86365	0.87160
0.00030	0.74347	0.76851
0.00040	0.61782	0.64926
0.00050	0.50309	0.51581
0.00070	0.28447	0.31253
0.00100	0.10563	0.11495
0.00200	0.00082	0.00190

Throughput vs. Probability of Error for K = 100

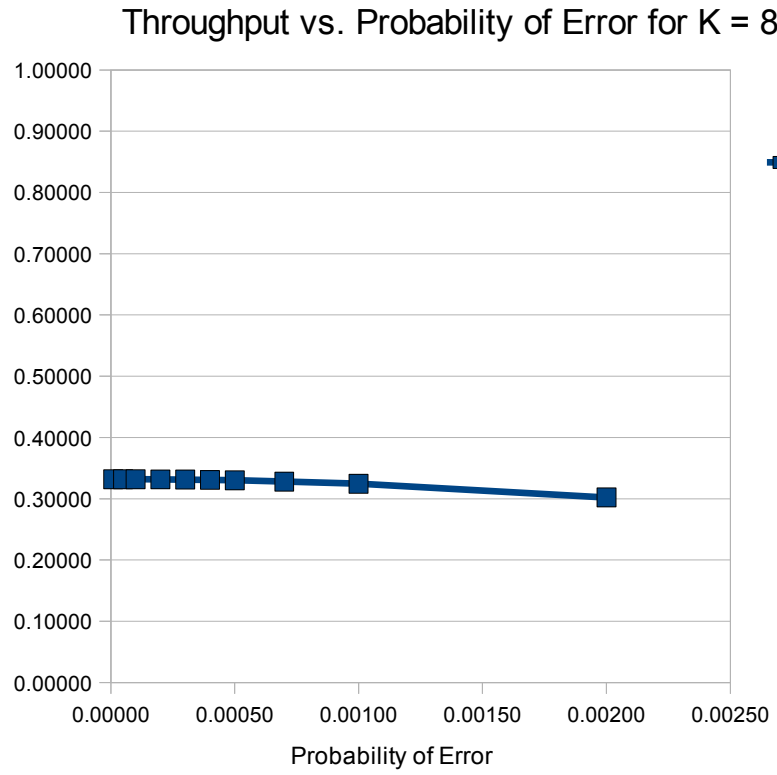


e	lower bound	upper bound
0.00001	0.90894	0.90918
0.00005	0.90756	0.90857
0.00010	0.90533	0.90677
0.00020	0.89294	0.89820
0.00030	0.87514	0.88656
0.00040	0.85100	0.86516
0.00050	0.82717	0.83587
0.00070	0.75940	0.77140
0.00100	0.62772	0.65241
0.00200	0.23498	0.24783

Throughput vs. Probability of Error for K = 1000



e	lower bound	upper bound
0.00001	0.66116	0.66116
0.00005	0.66077	0.66135
0.00010	0.66013	0.66116
0.00020	0.65761	0.66128
0.00030	0.65559	0.65902
0.00040	0.65240	0.65574
0.00050	0.64839	0.65299
0.00070	0.63702	0.64321
0.00100	0.61468	0.62421
0.00200	0.49905	0.52559



e	lower bound	upper bound
0.00001	0.33195	0.33195
0.00005	0.33184	0.33203
0.00010	0.33171	0.33203
0.00020	0.33126	0.33207
0.00030	0.33044	0.33199
0.00040	0.32982	0.33171
0.00050	0.32923	0.33105
0.00070	0.32643	0.32935
0.00100	0.32202	0.32684
0.00200	0.29975	0.30440

For the case where K is 0, even with a very small error rate of 0.00001, the throughput was closer to 0.9 than 1.0 because just one error in any of the blocks caused the whole frame to be resent. As the error rate is increased, the throughput experiences exponential decay.

For a K value of 1, even though there is a little bit more overhead than with K = 0 due to the Hamming's Code being applied to the block, the throughput shows improvement over the K = 0 case because one error is allowed in the block before it has to be retransmitted. At the low error rate of 0.00001, the throughput is almost 1.0 because most of the bit errors in each frame are corrected.

As the K value is increased and the Frame is being broken up into more blocks, the throughput begins to experience less of an exponential decay, and more of a gradual decline. This is because there are more blocks for each frame and therefore each frame is allowed more bit errors before it must be re-transmitted, therefore even with higher probabilities of error, the Hamming's Code is able to recover from those errors.

However, the maximum possible throughput, even with very low probabilities of error such as 0.00001 begins to decline as well. As evident in the cases where K is 1000 and 8000, the throughput with a very low probability of error is only ~0.7 and ~0.3, respectively. This is due to the fact that as the frame is broken up into more blocks, there is more overhead required required for each of those blocks. This is especially related to the relationship between the size of the block and the amount of check bits it requires. If high block sizes such as 2000 or 4000 (when K is 4 or 2), only 11 or 12 check bits are required, a very low fraction of the block size. But when the block size is as small as 8 or 1 (when K is 1000 or 8000), the amount of check bits required is 4 and 1 respectively, which, in the case of K = 8000, doubles the amount of bits that need to be sent. This leads to low throughput even when there are virtually no errors. It is also worth noting that for these high values of K, the throughput at higher error probabilities is much higher than the throughput for lower values of K.

From these observations I can deduce that it is beneficial to break up the frame into various amounts of blocks depending on the situation. If it is known that there is a high probability of error

when transmitting the frames through the network, then it is desirable to use high values of K that would break the frame into a large number of blocks each consisting of a relatively small number of bits. This would ensure that transmission through this high error probability network is more successful than if the frame was broken up into smaller blocks. One must however also taken into account delay. Even though the throughput at high error probabilities is better for higher K values than for lower K values, there is likely a considerable amount of delay that is not conveyed through the graphs presented in this report.

If it is known that the network instead has a very low probability of error, it is more desirable to break up the frame into smaller blocks. This would lead to less higher throughput, less overhead, and therefore less delay in the transmission.

For much larger values of A, I expect the throughput to start lower for the low probabilities of error, and then follow a similar exponential decay as **e** is increased. For example, for value of **A** = 10000, the graph of throughput vs. bit error probability would be compressed in half along the y-axis. This is because for each frame, the receiver must wait a longer period of time before it can transmit another frame. Each individual block isn't affected by A the same way that it is affected by the error probability, and so the throughput decays at a similar rate as with lower values of **A**.

I can conclude that the HSBC can be very useful to the throughput of a communication channel if enough information is available about this channel. One must be able to adjust parameters such as the block size in accordance to variables such as the bit error probability in the given network in order to utilize the HSBC, otherwise it may have a negative impact on the throughput in certain conditions.