EE3204: Computer Networks Lab Assignment
Venkatesan Harish, A0121828H

In this assignment, we explore the impact of error probability and data unit size of each packet on the throughput/data rate (KB/s) and message transfer time (ms).

The code implements a stop-and-wait ARQ which essentially waits for a successful acknowledgement (ACK) to be received for each packet it transmits before moving on to the next packet.

Assumptions

We assume that the only out-of-normal flow will be when the packet received is damaged. Hence, we assume packet loss, acknowledgement loss/damage do not occur.

We also assume the error probability to be just a simulation. Hence, the code implemented simulates this error probability by making a decision if the packet is damaged when received by the server through utilizing a random number generator.

We assume that timeout is not used in our scenarios.
We assume a simulation under localhost on a computer.
We assume propagation time to be constant.

Simulating Error Probabilities

The user enters the error probability (as a percentage) to be simulated as a command line argument when running tcp_server (e.g.: "`./tcp_server 10.5`" will simulate an error probability of 10.5%). Using this probability, we can classify some of the received packets as damaged. To do this, we utilize a random number generator initialized with a time seed.

"`((float) rand()/(float) (RAND_MAX)) * 100;`" yields a random float from 0.00 to 100.00. If this number is less than or equal to the error probability specified, we consider the currently received packet as damaged. Hence, we send a negative acknowledgement (NACK) to the sender. The sender, upon receiving the NACK, re-sends the previous packet again. Else, if the number is more the error probability, the server sends an ACK to indicate that the packet was received successfully, thereby allowing the sender to proceed to the transmission of the next packet.

NACK vs Timeout

Utilising a NACK is better than timeout in this case since we are only considering the case of packet damage. If the NACK is able to reach the sender faster than the timeout specified, then the sender can reduce the time spent waiting for the ACK/NACK from the receiver. This will improve the link utilisation.

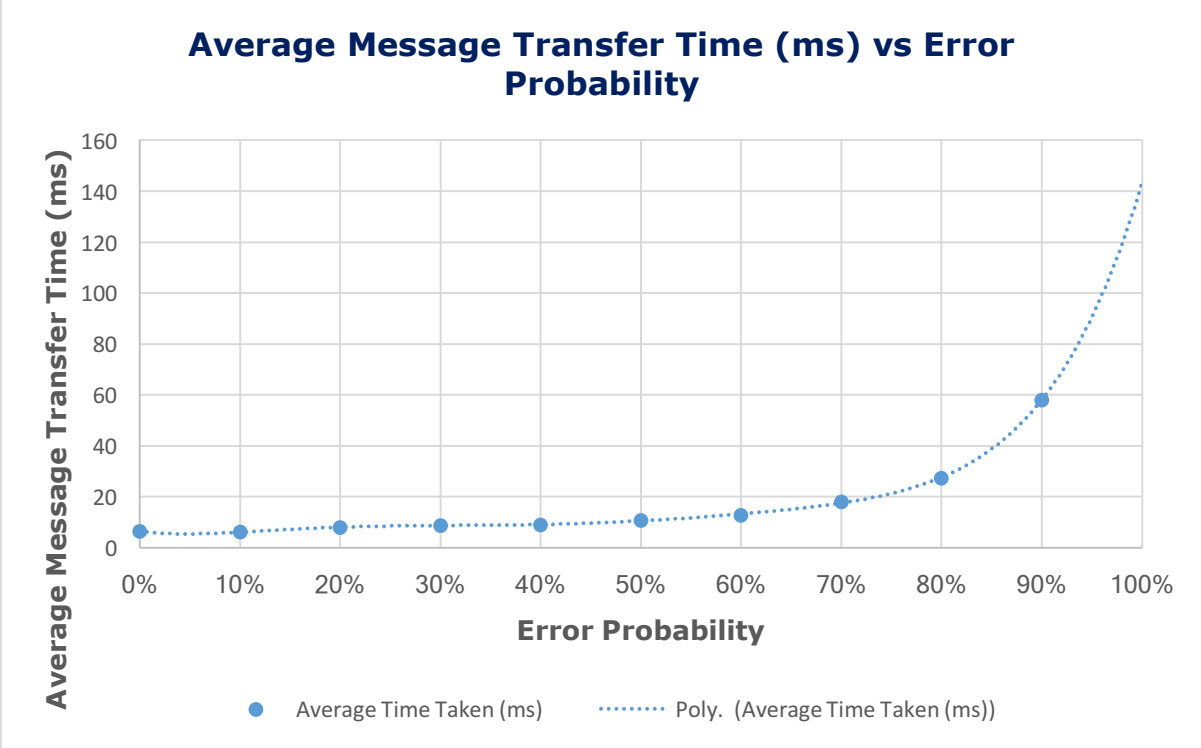
Error Probabilities vs Message Transfer Time (s) and Average Data Rate (KBps)

The following results on the message transfer time and average data rate (KBps) were observed when the various error probabilities were simulated.

Error Rate	Average Data Rate (KB/s)	Average Time Taken (ms)
0%	9462.26813	6.5598
10%	9718.52871	6.1962
20%	7773.70435	7.9882
30%	6894.49930	8.8014
40%	6698.89774	9.0872
50%	4728.33804	10.9028
60%	4728.33804	12.706
70%	3324.72484	18.1154
80%	2196.89612	27.4788
90%	1031.02764	58.163

The error probabilities were chosen in an interval of 10% to achieve maximum spread over the data.

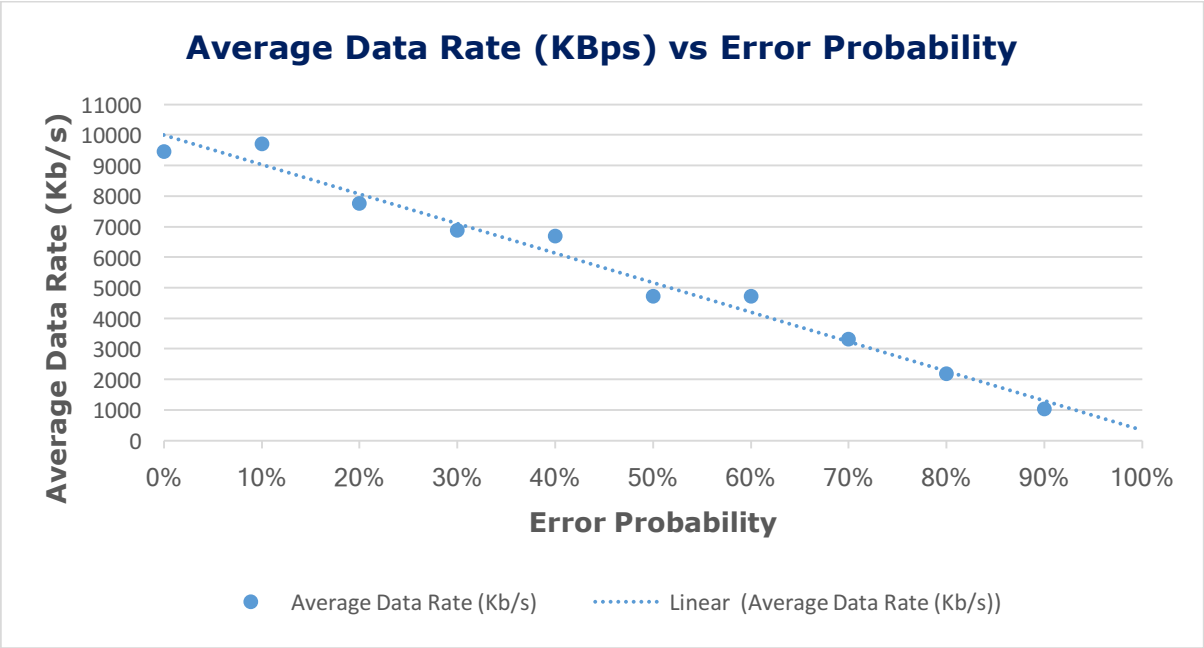
Plotting the above data, we obtain:



Analysis of trend

$U = T_f / (N_r * T_t) = (1 – P_f) / (1 + 2a)$, $a = (Tp/Tf)$

- Throughput rate = Throughput / Bandwidth (which is numerically equivalent to U)
- Throughput = (Data size) / (Total Transfer Time)
- Transfer time = Data size / Throughput
- Throughput = U * Bandwidth = $((1 – P_f) * B) / (1 + 2a)$
- Transfer time = Data size / $((1 – P_f) * B) / (1 + 2a)$ = Data size * $(1 + 2a) / (1 – Pf)$
- With data size being constant, and a being constant, we obtain a relationship of $y = 1 / (1 – x)$, where $x = (1 – Pf)$, for the message transfer time with regards to the error probability (Pf) as reflected in the above graph.



Analysis of Trend

As we can see, the average data rate decreases when the error probability increases. As error probability increases, the number of data packets being damaged when transmitted increases. Such, more retransmissions of the same packets are required which increases the overall transfer time for the message. Data rate is given by (Data size / Message Transfer Time). As message transfer time increases, data rate decreases.

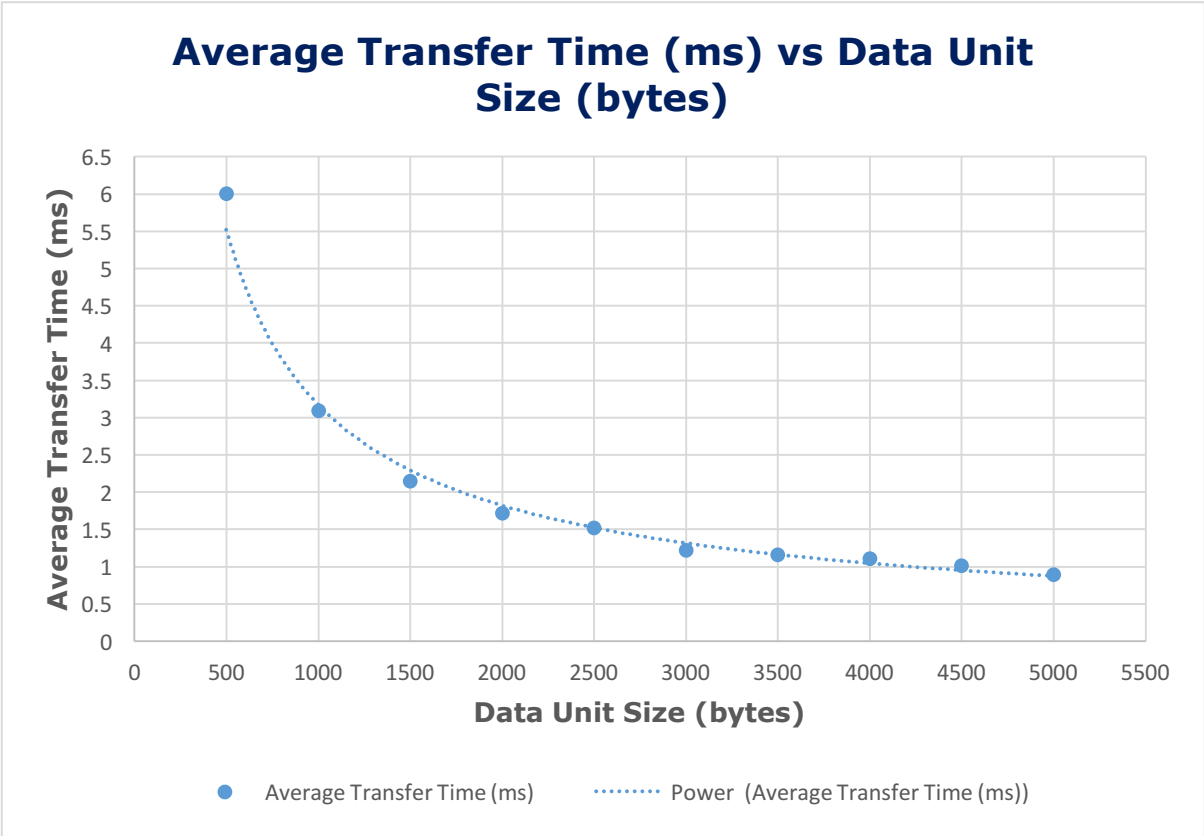
The average data rate, theoretically, is the inverse of the above expression for the message transfer time, $y = 1 / (1 – x)$. This yields, $y = 1-x$, which is represented by the linear relationship shown above.

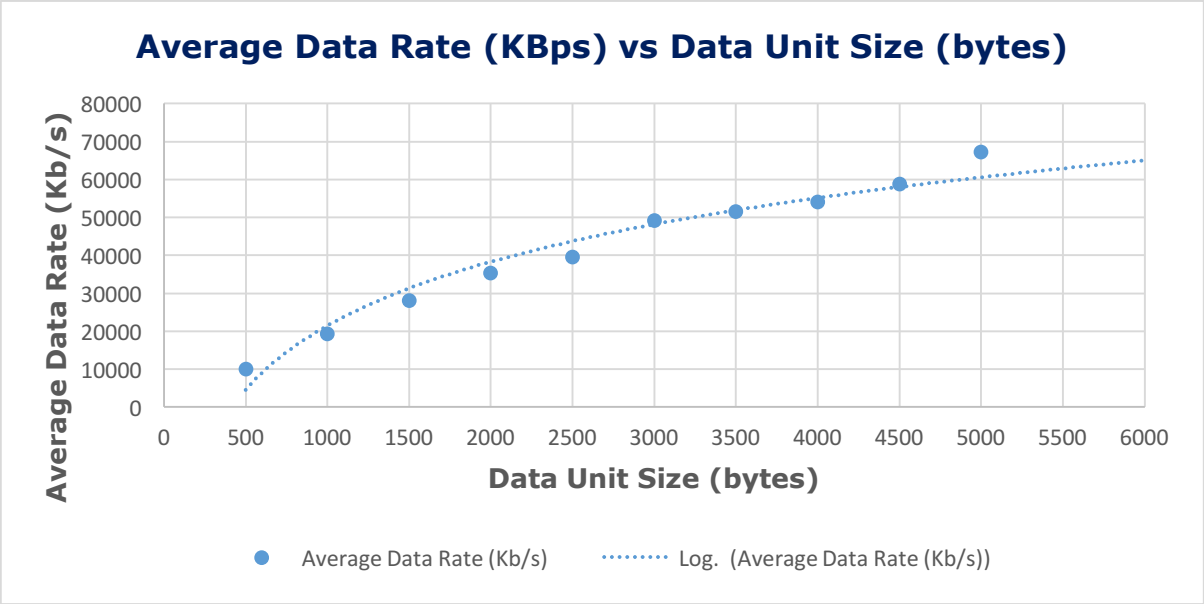
Data Unit Sizes vs Message Transfer Time (s) and Average Data Rate (KBps)

The following results on the message transfer time and average data rate (KBps) were observed when the various error probabilities were simulated.

Data Unit Size (bytes)	Average Data Rate (KB/s)	Average Transfer Time (ms)
500	10076.24456	6.004
1000	19360.7671	3.0966
1500	28091.39671	2.1486
2000	35366.83926	1.715
2500	39607.94109	1.5182
3000	49207.77359	1.2228
3500	51482.8684	1.1634
4000	54103.88177	1.1078
4500	58755.00113	1.0178
5000	67173.24615	0.8908

The data unit sizes were chosen in intervals of 500 bytes to achieve a reasonable spread of the packet sizes. The data was then plotted, as shown in the following graphs.





Hence, it is observed that the average transfer time decreases as the data unit size increases. The average data rate which is inversely proportional to the message transfer time hence increases. With the data unit being larger, we need to transmit a lower number of packets as compared to a scenario with smaller data unit. Hence, the proportion of overhead in terms of the header length decreases per packet. This increases the overall throughput.

Overall, it seems to taper off as we increase the packet length beyond a point as the proportion of the header in comparison with the data decreases.

However, in reality this may be constrained by the factors of packet loss/damaged etc. This will require the packet to be retransmitted. As the packet size increases, the number of bytes being retransmitted also increases. This causes the throughput to reach a plateau at some point, beyond which it may decrease.

References

- EE3204 Lab Handouts & Lecture Notes