Faculdade de Engenharia da Universidade do Porto

Lab 1 – Protocol Efficiency Report

Computer Networks
Class 2

Emilie Kernivinen - up202306154
Luís Gonçalves - up202305120

Summary

This report presents a performance analysis of an implementation of the Stop-and-Wait Automatic Repeat Request data link protocol. The protocol was tested under different conditions with varying parameters and its throughput and efficiency compared with the theoretical results. The measured values showed good correlation to theoretical efficiencies but were in general around 20% slower due to implementation overhead.

Introduction

Reliable data transfer is a key function of the data link layer. The Stop-and-Wait ARQ protocol provides this reliability by ensuring that each frame is acknowledged before the next one is sent, retransmitting any that are lost or arrive corrupted.

In this project, a Stop-and-Wait ARQ protocol was implemented to enable communication between two Linux-based computers using an RS-232 serial link. It implements framing of packets from a layer above it, error detection, acknowledgements, and retransmissions to guarantee correct delivery while maintaining layer independence.

This report focuses on evaluating its performance by measuring its throughput and efficiency under different conditions.

Methodology

To study the impact of different transmission conditions, parameters such as baudrate (C), frame error rate (FER), propagation delay (T_{prop}), and frame size (L) were varied and the results compared with theoretical expectations for efficiency (S) given by the formula:

$$S = \frac{1-FER}{1+2a} \text{ where } a = \frac{Tprop}{Tframe},\ Tframe = \frac{L}{C},\ FER = 1 - (1 - BER)^{L}$$

Data collection was mostly automated using two bash scripts generated by AI. These scripts launched a virtual cable and ran transmitter and receiver 10 times for each variation, storing the time for each run and averaging them in the end. When not specified, parameter values are the default when running the virtual cable, i.e. $C = 9600\,bps$, $FER = 0$, $Tprop = 0\,s$, $L = 8048\,b$ and the file transmitted is the default *penguin.gif*. The data was then plotted with the following results.

Results

C (bps)	Time (s)	Data R (bps)	Data S (%)	Theoretical S (%)
2400	47,324	1854,112	0,772	0,994
4800	23,862	3677,143	0,766	0,988
9600	12,098	7252,769	0,755	0,976
19200	6,199	14154,541	0,737	0,954
57600	2,283	38433,639	0,667	0,874

Table 1 - Baudrate impact T_{prop} = 0.01s

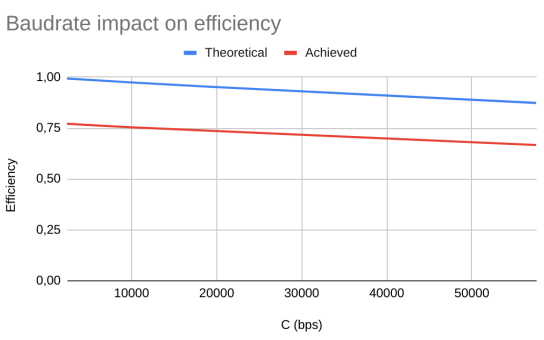

Image 1 - Baudrate impact T_{prop} = 0.01s

Tprop (µs)	Time (s)	Data R (bps)	Data S (%)	Theoretical S (%)
0	11,764	7458,687	0,776	1
10	11,764	7458,433	0,776	0,999
10000	12,097	7252,888	0,755	0,976
100000	14,964	5863,594	0,61	0,807
1000000	43,764	2004,921	0,208	0,295

Table 2 - Propagation Delay impact

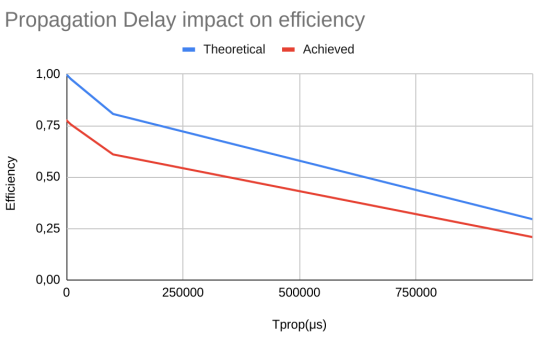

Image 2 - Propagation Delay impact

L (b)	Time (s)	Data R (bps)	Data S (%)	Theoretical S (%)
2048	12,246	7164,881	0,746	1
4048	11,924	7358,481	0,766	1
8048	11,764	7458,497	0,776	1
16048	11,676	7514,58	0,782	1
32048	11,633	7542,68	0,785	1
64048	11,618	7552,158	0,786	1

Table 3 - Frame Size impact T_{prop} = 0s

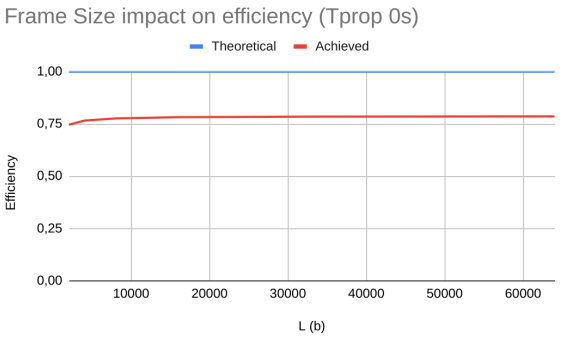

Image 3 - Frame Size impact T_{prop} = 0s

L (b)	Time (s)	Data R (bps)	Data S (%)	Theoretical S (%)
2048	22,046	3979,915	0,414	0,516
4048	17,325	5064,588	0,527	0,678
8048	14,964	5863,476	0,61	0,807
16048	13,677	6415,441	0,668	0,893
32048	13,033	6732,448	0,701	0,943
64048	12,818	6845,373	0,713	0,97

Table 4 - Frame Size impact T_{prop} = 0,1s

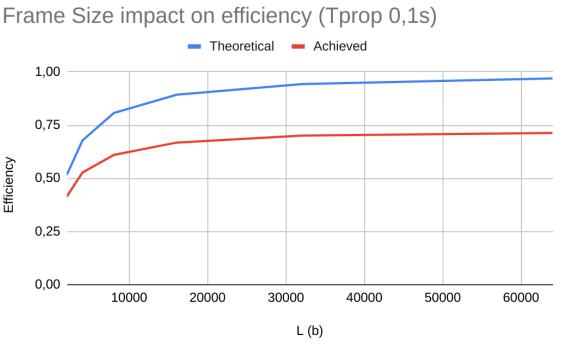

Image 4 - Frame Size impact T_{prop} = 0,1s

BER (%)	FER (%)	Time (s)	Data R (bps)	Data S (%)	Theoretical S (%)
0	0	11,76	7458,56	0,776	1
0,000001	0,008	11,87	7391,956	0,769	0,992
0,000025	0,182	14,20	6178,893	0,643	0,818
0,00005	0,331	22,64	3875,122	0,403	0,669
0,0001	0,552	23,31	3763,946	0,392	0,448

Table 5 - Frame Error Rate impact

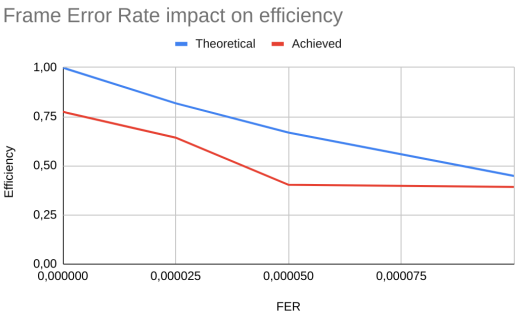

Image 5 - Frame Error Rate impact

Conclusion

Baudrate: Increasing baudrate generates a directly proportional increase in throughput. However, efficiency decreases with higher baudrates because of the increase in the normalized propagation delay parameter a, as predicted by theory. Baudrate increases from 2400 to 57600bps, showing throughput gains from 1854 to 38434bps, while efficiency drops from 0.994 to 0.874.

Propagation Delay: Propagation delay significantly reduces both efficiency and throughput. As T_{prop} increases from 0 to 1s, efficiency S drops from 1 to 0.295, matching the theoretical model's predictions of a sharp reduction. At the same time, throughput R is also greatly reduced from 7458 to 2004 bps. However, for values of $T_{prop} \leq 0.01$s efficiency and throughput are roughly stable with $S \geq 0.976$ and R in the 7250-7500bps range.

Frame Size: At lower propagation delays ($T_{prop} << 0.1$s), frame size has minimal impact on efficiency and throughput although smaller frame sizes have a lower efficiency due to protocol overhead. However, at $T_{prop} = 0.1$s, increasing frame size significantly improves both S and R demonstrating that frame size becomes a critical optimization factor in higher delay conditions. We have seen that throughput goes from 3979 to 6845bps and efficiency from 0.414 to 0.713 when the frame size varies from 2048 to 64048b.

Frame Error Rate: As FER increases, efficiency decreases as predicted by the theoretical model. While FER variations produced noisier data due to the inherent randomness of error generation, the measured efficiency trends closely followed theoretical predictions with FER varying from 0 to 0.552, S fell from 0.776 to 0.392 and R from 7459 to 3764bps. Higher FER values could be tested but it would be too time consuming.

Experimental tests saw efficiency fall, on average, approximately 20% below theoretical predictions likely due to implementation overhead and simplifying assumptions in these formulas (such as instantaneous ACK transmission). Despite the gap, measured results closely followed theoretical trends, validating the model's applicability to real-world protocol design.