Chapter 4

An Analysis of Data Link Control Protocols

Pinaki Mitra

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/60962

Abstract

In this chapter we analyze the performance of data link protocols. We consider Stop and Wait ARQ and Sliding Window protocol. In Sliding window protocol, we consider Selective Reject ARQ and Go-Back-N ARQ. There are existing results that analyze the link utilization of these protocols in the presence of error. We have experimented in a somewhat more generalized framework. We have also analyzed and experimented with the number of duplicate error-free packets that will be transmitted.

Keywords: Stop ad Wait ARQ, Sliding Window Protocol, Selective Reject ARQ, Go-Back-N ARQ, Duplicate Error-Free Packet

1. Introduction

There are two types of Data Link Control Protocols:

1. Stop and Wait
2. Sliding Window

1.1. Stop and wait ARQ

In stop and wait flow control the sender sends the data frame whose transmission time is t_f and it reaches the receiver after time t_p, the propagation delay across the communication channel. The receiver sends the acknowledgment frame back to the sender whose transmission time is negligible and the propagation delay is t_p. This is illustrated in Figure 1. The sender sends the next data frame after time ($t_f + 2t_p$) if ACK is positive, otherwise it resends the earlier frame.

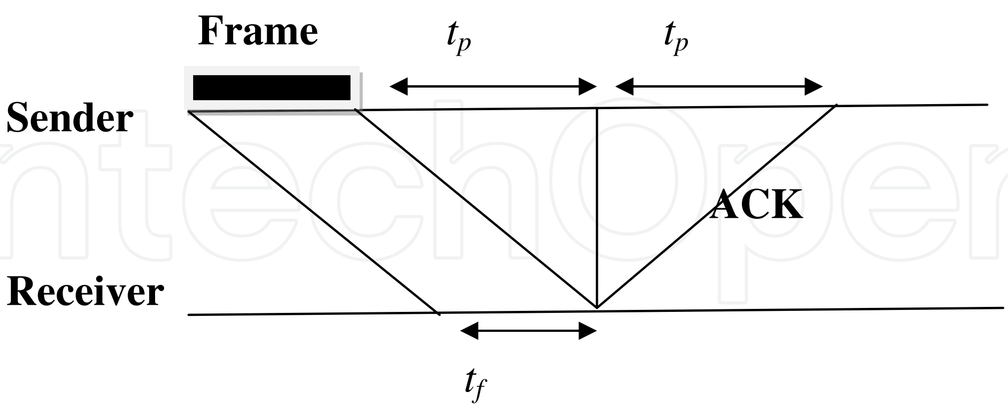

Figure 1. Illustration of Stop and Wait ARQ

The link utilization of this stop and wait protocol is given below:

$$U = \frac{t_f}{t_f + 2t_p}$$

1.2. Sliding window protocol

Here, the sender maintains a window of data frames of size W. In this protocol the sender continues sending packets before the receipt of the ACK (acknowledgment) frame. If the frame transmission time is t_f and the propagation delay is t_p, we can pack A data frames in the time span $t_f + 2t_p$ where

$$A = \frac{t_f + 2t_p}{t_f} = 1 + 2\frac{t_p}{t_f} = 1 + 2a \text{ where } a = \frac{t_p}{t_f}$$

If our window size $W \geq A$ then link utilization $U = 1$ otherwise $U = \frac{W}{A}$. The protocol is illustrated in Figure 2.

This analysis holds for error free transmission. Now if there are transmission errors then in both the protocols the packet has to be resent. For sliding window protocols there are two types of ARQ, namely:

1. Selective Reject ARQ
2. Go-Back-N ARQ

http://dx.doi.org/10.5772/60962

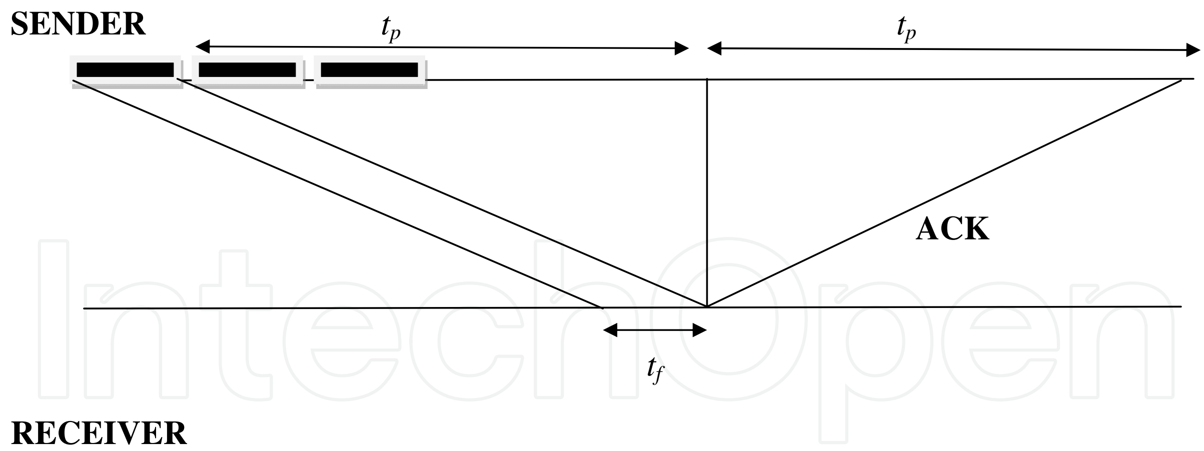

Figure 2. Illustration of Sliding Window Protocol

In Selective Reject ARQ, sliding windows of same size at both sending and receiving ends are maintained. The receiver sends a *RR-N* frame, that is, receive ready frame to the sender for the sending data frame *N*. If any data frame *N* is in error the receiver sends back *SREJ -N*, that is, selective reject signal to the sender. It continues to receive frames with sequence number ≥ *N* from the sender but it does not send any *RR* request for frame ≥ *N*.

In contrast, in Go-Back-N ARQ, sliding windows of unequal size at sending and receiving ends are maintained. In the sending end sliding window size is usually > 1 but in the receiving end sliding window size is usually = 1. Here also the receiver sends a *RR-N* frame, that is, receive ready frame to the sender for the sending data frame *N*. If any data frame *N* is in error, the receiver sends back *Go-Back-N* signal to the sender. It rejects all frames with sequence number ≥ *N* from the sender and it does not send any *RR* request for frame ≥ *N*.

For both these protocols, let N_r be the expected number of retransmissions. Then, the link utilization in the presence of error $U_{error} = \frac{U}{N_r}$. The analysis of N_r is usually carried out by assuming the ACK frame is error free. In this chapter, we try to analyze three situations:

1. The delivered packet is delayed
2. ACK frame is in error
3. ACK frame is delayed

In all the three cases the receiver will receive duplicate error-free packets. We try to analyze the estimate of duplicate packets received by the receiver.

2. Existing analysis of link utilization of data link control protocols

Consider *P* is the probability that the single frame is in error. Then link utilization for **Stop-and-Wait ARQ** is computed as follows:

$$N_r = \sum_{i=1}^{\infty}\left(iP^{i-1}\left(1-P\right)\right) = 1/\left(1-P\right)$$
$$U_{error} = U / N_r = \left(1-P\right)/A$$

Where

$$A = \frac{t_f + 2t_p}{t_f} = 1 + 2\frac{t_p}{t_f} = 1 + 2a \text{ where } a = \frac{t_p}{t_f}$$

For Selective Reject ARQ the link utilization is as follows:

$$U_{error} = U / N_r = \begin{cases} 1-P & W \geq A \\ \frac{W(1-P)}{A} & W < A \end{cases}$$

For Go-Back-N ARQ each error generates a requirement to retransmit *K* frames rather than a single frame.

$$N_r = \sum_{i=1}^{\infty}\left[1+\left(i-1\right)K\,P^{i-1}\left(1-P\right)\right] = 1 - K + K/\left(1-P\right) = \left(1-P+KP\right)/\left(1-P\right)$$

$$U_{error} = U / N_r = \begin{cases} \frac{1-P}{1+2aP} & W \geq A \\ \frac{W(1-P)}{A(1-P+WP)} & W < A \end{cases}$$

Here, we assumed $K = A$ for $W \geq A$ and $K = W$ for $W < A$.

A detailed discussion of the protocol and the above analysis can be found in authored books [1], [2], [3] and [4]. Reader can also refer to the published journal article [5].

3. Modified analysis of link utilization of data link control protocols

We will assume that the probability of a frame not in error is p_1. Also we will assume that the probability the frame is delivered within a time interval T is p_2. The sender will resend the packet if ACK does not arrive after time interval 2T. Let p_3 be the probability that the ACK frame is error free. Let p_4 be the probability that the ACK arrives within time 2T. The probability that the packet is received correctly in one transmission is $p_1p_2p_3p_4$. So the expected number of

http://dx.doi.org/10.5772/60962

retransmissions $N_r = 1/\ p_1p_2p_3p_4$. So in the existing analysis we only need to substitute $P = 1- p_1p_2p_3p_4$ to get the link utilization.

To analyze the case of the receipt of duplicate error free packets we consider the following four events:

i. The delivered packet is free from error. The corresponding probability is p_1. Let us call this as event A_1.

ii. The packet doesn't reach within time T and the receiver asks to resend the packet. The corresponding probability is (1-p_2). Let us call this as event A_2.

iii. The ACK frame is in error and requires retransmission. The corresponding probability is (1-p_3). Let us call this as event A_3.

iv. The ACK frame doesn't arrive within time 2T and the sender resends the packet. The corresponding probability is (1-p_4). Let us call this as event A_4.

The probability that there is duplicate frame transmitted will be:

$$\begin{aligned}
&\mathrm{P}[A_1 \cap (A_2 \cup A_3 \cup A_4)\big] = \mathrm{P}(A_1)\big[\mathrm{P}(A_2) + \mathrm{P}(A_3) + \\
&\mathrm{P}(A_4) - \mathrm{P}(A_2 \cap A_3) - \mathrm{P}(A_3 \cap A_4) - \mathrm{P}(A_4 \cap A_2) - \mathrm{P}(A_2 \cap A_3 \cap A_4)\ ] \\
&= p_1 \begin{bmatrix} (1-p_2) + (1-p_3) + (1-p_4) - (1-p_2)(1-p_3) - \\ (1-p_3)(1-p_4) - (1-p_4)(1-p_2) + (1-p_2)(1-p_3)(1-p_4) \end{bmatrix} = q
\end{aligned}$$

So the expected number of duplicate packets will be $N_r\, q$.

4. Experimental results and discussions

We have experimented with the number of retransmission and the number of duplicate error-free packets transmitted with the following C code:

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
void main()
{
float p[4];
float S0, S1, S2;
int i,j;
float q, term, term_1, term_2;
int n_r, dup;
time_t t;
srand((unsigned) time(&t));
```

```
randomize();
p[0]=(float)random(100)/100.;
p[1]=(float)random(100)/100.;
p[2]=(float)random(100)/100.;
p[3]=(float)random(100)/100.;
printf("p[0]=%f p[1]=%f p[2]=%f p[3]= %f\n\n", p[0], p[1], p[2], p[3]);
S0 = 3-p[1]-p[2]-p[3];
S1 = 0;
for (i= 1; i <4; i++)
for (j=i+1; j < 4; j++)
S1 += (1-p[i])*(1-p[j]);
S2 = (1-p[1])*(1-p[2])*(1-p[3]);
q = p[0]*(S0 - S1 + S2);
term = p[0]*p[1]*p[2]*p[3];
term_1 = 1.0/term;
n_r = (int)term_1; term_2 = q*n_r; dup = (int)term_2;
printf (" n_r = %d dup = %d \n", n_r, dup);
}
```

We have run several times the above mentioned program and obtained the following results:

p[0]	p[1]	p[2]	p[3]	n_r	dup
0.82	0.93	0.64	0.29	7	4
0.85	0.66	0.22	0.33	24	19
0.38	0.71	0.24	0.56	27	9
0.6	0.44	0.14	0.95	28	15
0.13	0.74	0.24	0.54	80	9
0.95	0.47	0.48	0.4	116	109
0.04	0.92	0.64	0.22	192	6
0.21	0.44	0.48	0.07	322	66
0.12	0.12	0.37	0.16	1173	139
0.52	0.24	0.42	0.01	1907	990

Table 1. Number of retransmissions and duplicate error-free packets with different probabilities

In the Table-1, P(A_1) = p[0], P(A_2) = 1 - p[1], P(A_3) = 1 - p[2], P(A_3) = 1 - p[3]. The number of retransmissions N_r = n_r and dup stands for the number of duplicate error-free packets. We have shown the table sorted on n_r. We also obtained the plot illustrated in Figure 3, which shows the interrelationship between n_r and dup.

We can clearly see that as n_r increases monotonically dup doesn't show any such monotonic increasing property. This is apparent from Table 1 where we observe that for n_r = 116 dup = 109 and for n_r = 192 dup = 6.

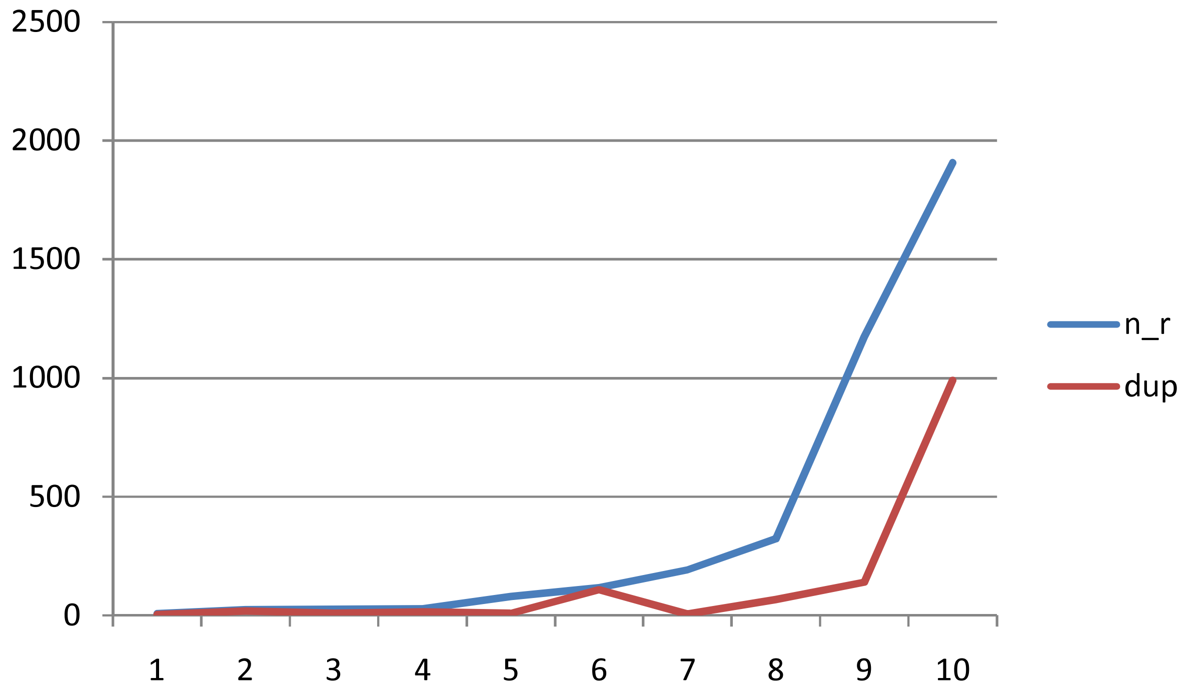

Figure 3. Illustration of the interrelationship between the number of retransmissions and duplicate error-free packets

5. Conclusion and future work

In this chapter we have analyzed and experimented with the number of retransmissions and the number of duplicate error-free packets transmitted in a generalized framework of data transmission model. As a future work we can consider some distribution on the probability of delayed transmission as a function of time.

Author details

Pinaki Mitra

Address all correspondence to: pinaki@iitg.ernet.in

Department of CSE, IIT Guwahati, Guwahati, India

References

[1] William Stallings, *Data and Computer Communications*, 8th Edition, Pearson.

[2] Prakash C. Gupta, *Data Communications and Network,* Prentice Hall India.

[3] Ajit Pal, *Data Communications and Computer Networks,* Prentice Hall India.

[4] Dimitri Bertsekas and Robert Gallager, *Data Networks*, 2nd Edition, Prentice Hall India.

[5] Lin, S., Costello, D., and Miller, M. "Automatic-Repeat-Request Error-Control Schemes." *IEEE Communication Magazine,* December 1984.