ITS 323 – DATA LINK PROTOCOLS EXAMPLES

1 Efficiency, Utilisation and Throughput

Efficiency (or utilisation) is a measure of the ratio of time spent sending *useful data* (that is, the actual message) versus the total time for the data to be successfully sent (which also includes propagation times, time for transmitting headers, acknowledgements, retransmissions).

Efficiency is given as a fraction (less than 1.0) or percentage. For example, the efficiency is 0.65 or 65%.

Throughput is the rate at *useful data* is received by the receiver. So it is given as a rate, e.g. bits per second. Typically, if we know the efficiency and original data rate, we can calculate the throughput:

Throughput = Efficiency x DataRate

For example, with a 1Mb/s data rate and efficiency of 0.65, the throughput is 650kb/s.

2 Stop and Wait Flow Control

Lets look at the Stop and Wait Flow control protocol with some example times. Figure 1 gives an example of Stop and Wait where 3 data frames are sent.

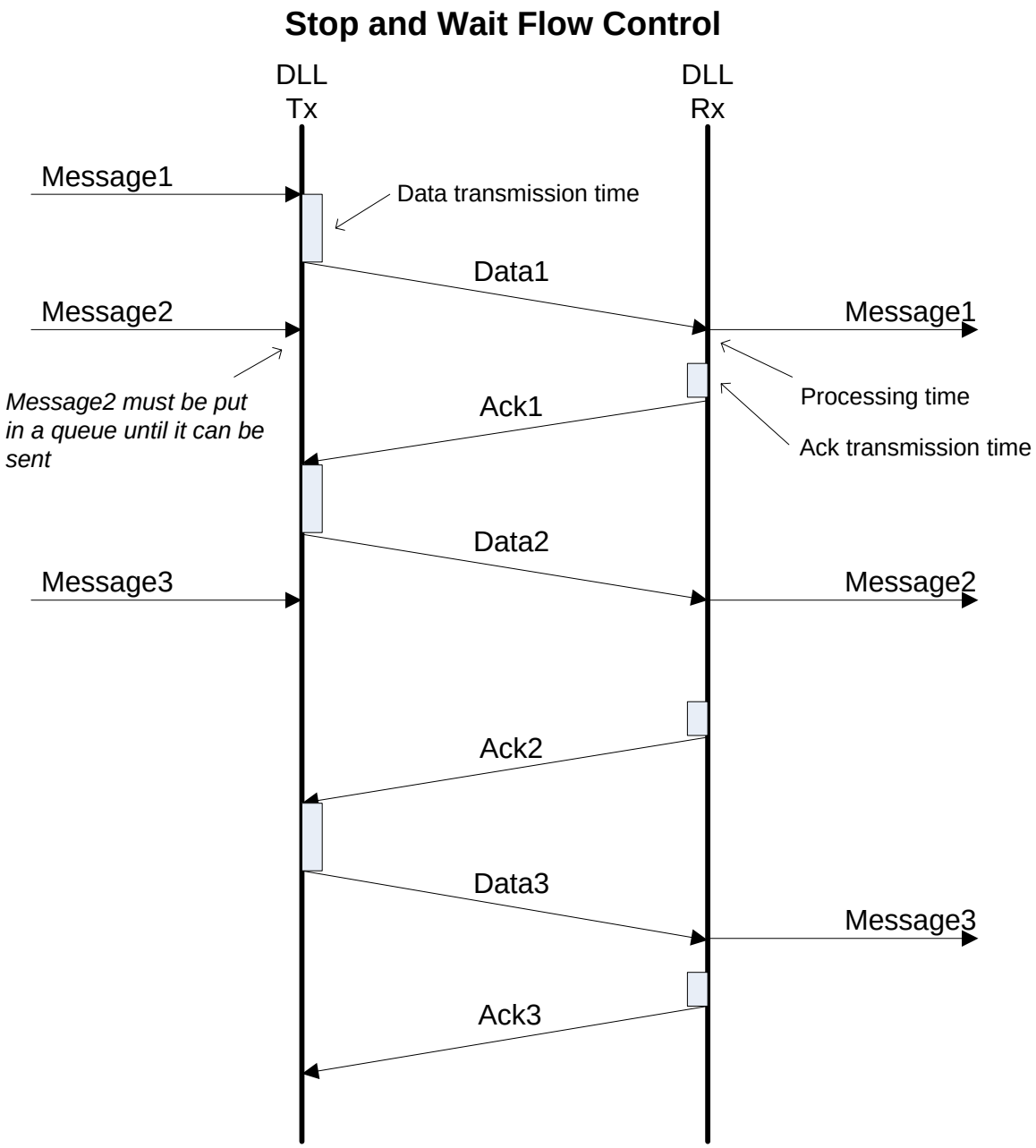


Figure 1: Stop and Wait Flow control

2.1 Time for Stop and Wait Transfer

A general equation that defines the time of sending a message is:

T = DataTransmissionTime + DataPropagationTime + DataProcessingTime + AckTransmissionTime + AckPropagationTime + AckProcessingTime

Lets simplify this, and assume the AckProcessingTime is 0, and we know the propagation time is independent of the frame.

T = DataTransmissionTime + DataProcessingTime + AckTransmissionTime + 2 x PropagationTime

2.2 Stop and Wait Example

So now lets consider an example:

- Each message is 1000 bytes, and the Data Link Layer adds a 20 byte header
- The Ack frame is 20 bytes (that is, entire header)
- Propagation velocity is 2 x 10^8 m/s
- Link distance is 2km
- Data rate is 1Mb/s
- A Data frame takes 1usec to process.

So, we know that:

$$TransmissionTime = \frac{FrameSize}{DataRate}$$

And

$$\Pr opagationTime = \frac{LinkDis\tan ce}{\Pr opagationVelocity}$$

So the total time to send a message is:

$$\begin{aligned} T &= \frac{8\times(1000+20)}{1\times10^6} + 1\times10^{-6} + \frac{8\times20}{1\times10^6} + 2\frac{2000}{2\times10^8} \\ &= 8160\times10^{-6} + 1\times10^{-6} + 160\times10^{-6} + 20\times10^{-6} \\ &= 8.341ms \end{aligned}$$

Lets look at the efficiency of this transmission. It takes a total of 8.341ms to complete the entire transmission. But the actual useful data we want to send is 1000 bytes, which is (1000x8)/1000000 seconds, or 8ms, worth of data. So the efficiency is:

$$\begin{aligned} Eff &= \frac{8}{8.341} \\ &= 0.959 \\ Throughput &= 959kb/s \end{aligned}$$

2.3 Equation for Stop and Wait Efficiency

From this, we can derive a simple equation of the efficiency of Stop and Wait. Lets now assume the DataProcessingTime is also 0. Also, if we assume the AckSize is very small compared to the DataSize (e.g. 20 bytes compared to 1000 bytes), then the AckTransmissionTime can be considered 0.

$$Eff = \frac{DataTransmissionTime}{DataTransmissionTime + AckTransmissionTime + 2 \times PropagationTime}$$
$$= \frac{DataTransmissionTime}{DataTransmissionTime + 2 \times PropagationTime}$$
$$= \frac{1}{1 + 2 \times \frac{p}{d}}$$

where p = PropagationTime and d = DataTransmissionTime

In some examples (such as the lecture notes and Stallings textbook) you may see a = (p/d) so that the efficiency is:

$$Eff = \frac{1}{1 + 2a}$$

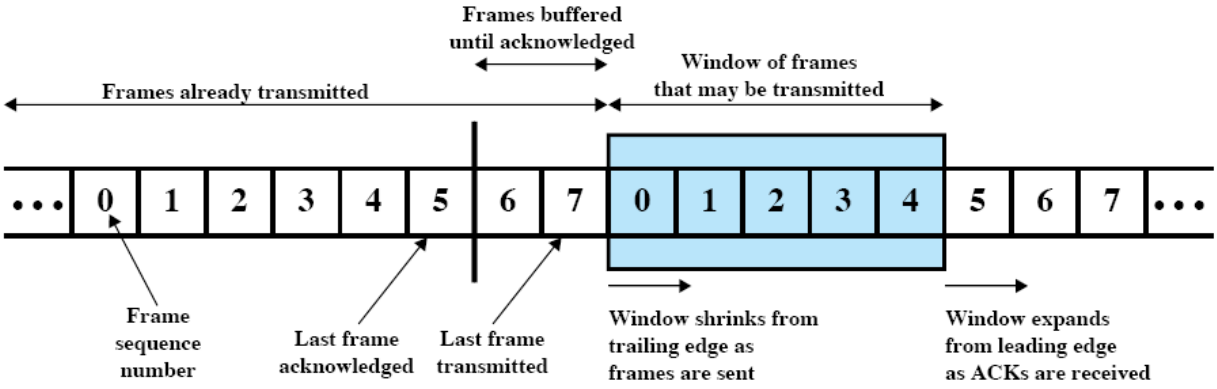
Note that Efficiency is the same as Utilization.

3 Sliding Window Flow Control

3.1 Windowing Concepts

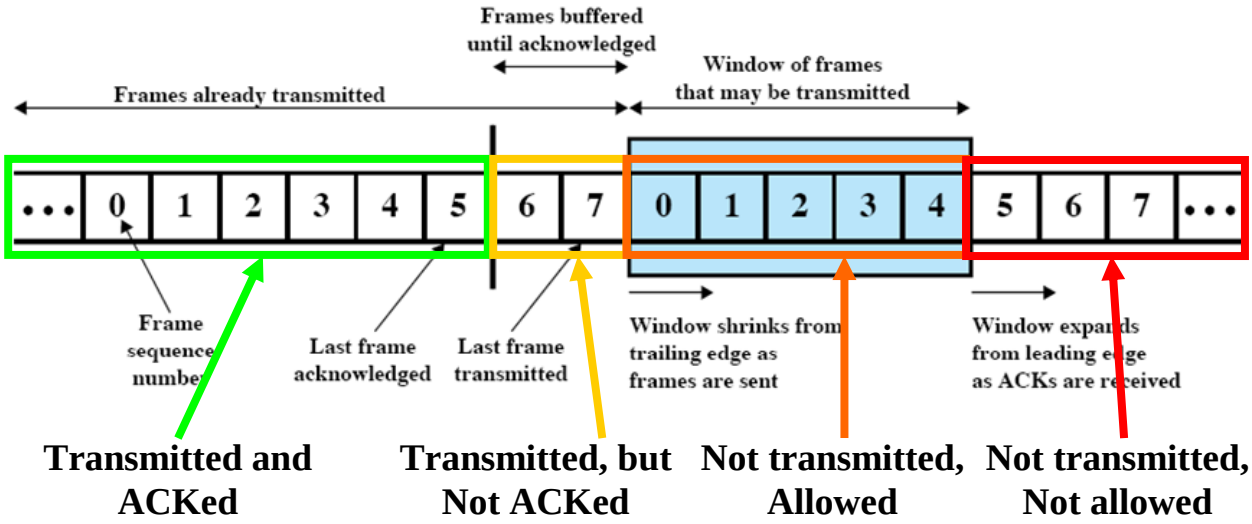
Both the sender (source or transmitter) and receiver (destination) maintain variables to manage the frames they can send and receive. These variables can be visualised as follows.

At the source, there are frames to transmit. Frames are sent in order based on a sequence number. In the figure below, the boxes with numbers indicate the frames. This shows a snapshot in time of the source.



To simplify the above, the frames can be considered in four groups:

1. Frames that have been transmitted and acknowledged.
2. Frames that have been transmitted, but not acknowledged
3. Frames that have not been transmitted, but we are allowed to send
4. Frames that have not been transmitted, and we are not allowed to send



The maximum number of frames that can be transmitted without being ACKed is 2^k-1, the maximum Window size. This is the sum of:

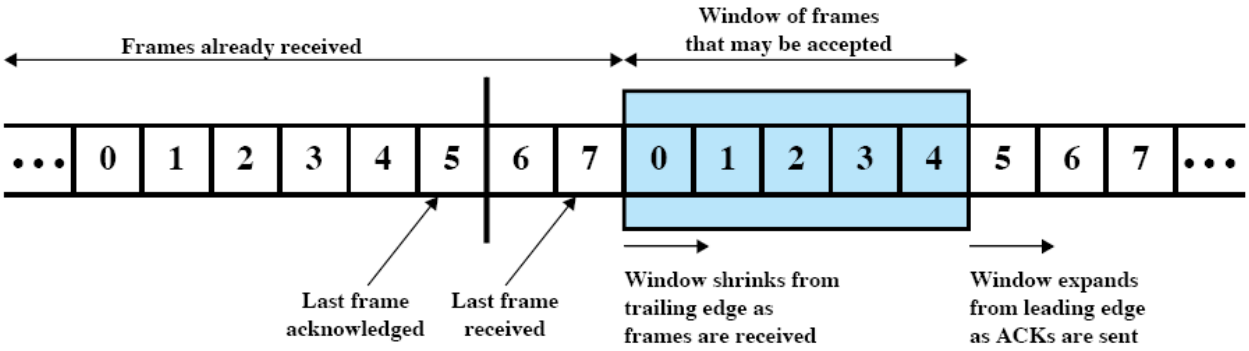
Transmitted, but not ACKed + No transmitted, allowed

In the example, we are allowed to transmit up to 7, but we have only transmitted 2 (frames 6 and 7). Therefore the current window size is 5 frames (frames 0, 1, 2, 3, 4).

If a frame is Acknowledged, then it will move from group 2 (transmitted but not ACKed) to group 1 (Transmitted and ACKed). Also the number of frames we are allowed to transmit will increase. For example, if frame 6 is ACKed, then it will become "Transmitted and ACKed" and the "Not transmitted, allowed" will grow to include frames 0, 1, 2, 3, 4 and 5.

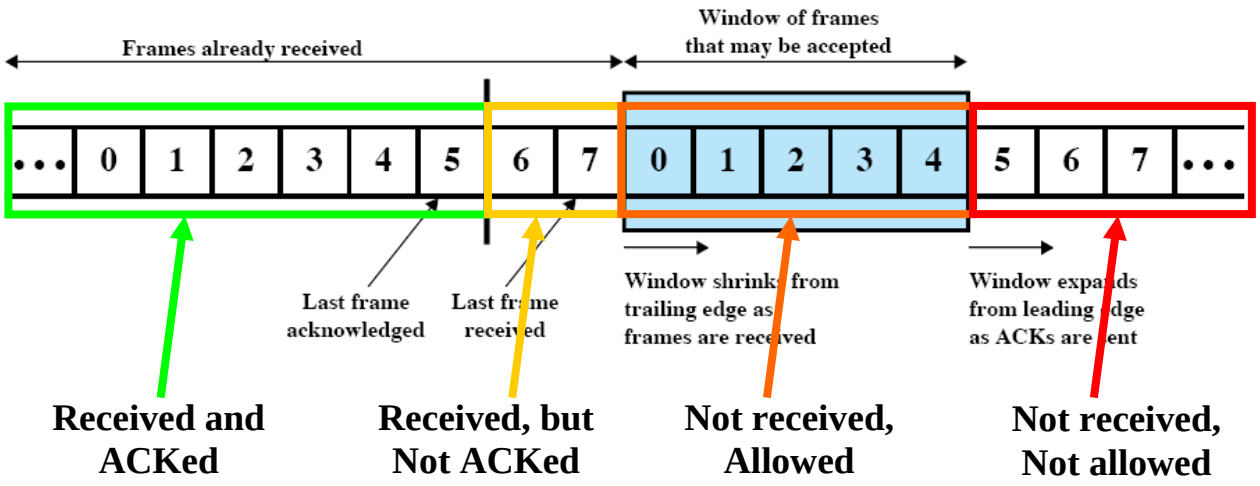
If a frame is transmitted, then it will move from group 3 (not transmitted, but allowed) to group 2 (transmitted but not ACKed). For example, if frame 0 is transmitted, it will become "Transmitted but not ACKed".

The destination or receiver also keeps track of the window in a similar way.



To simplify the above, the frames can be considered in four groups:

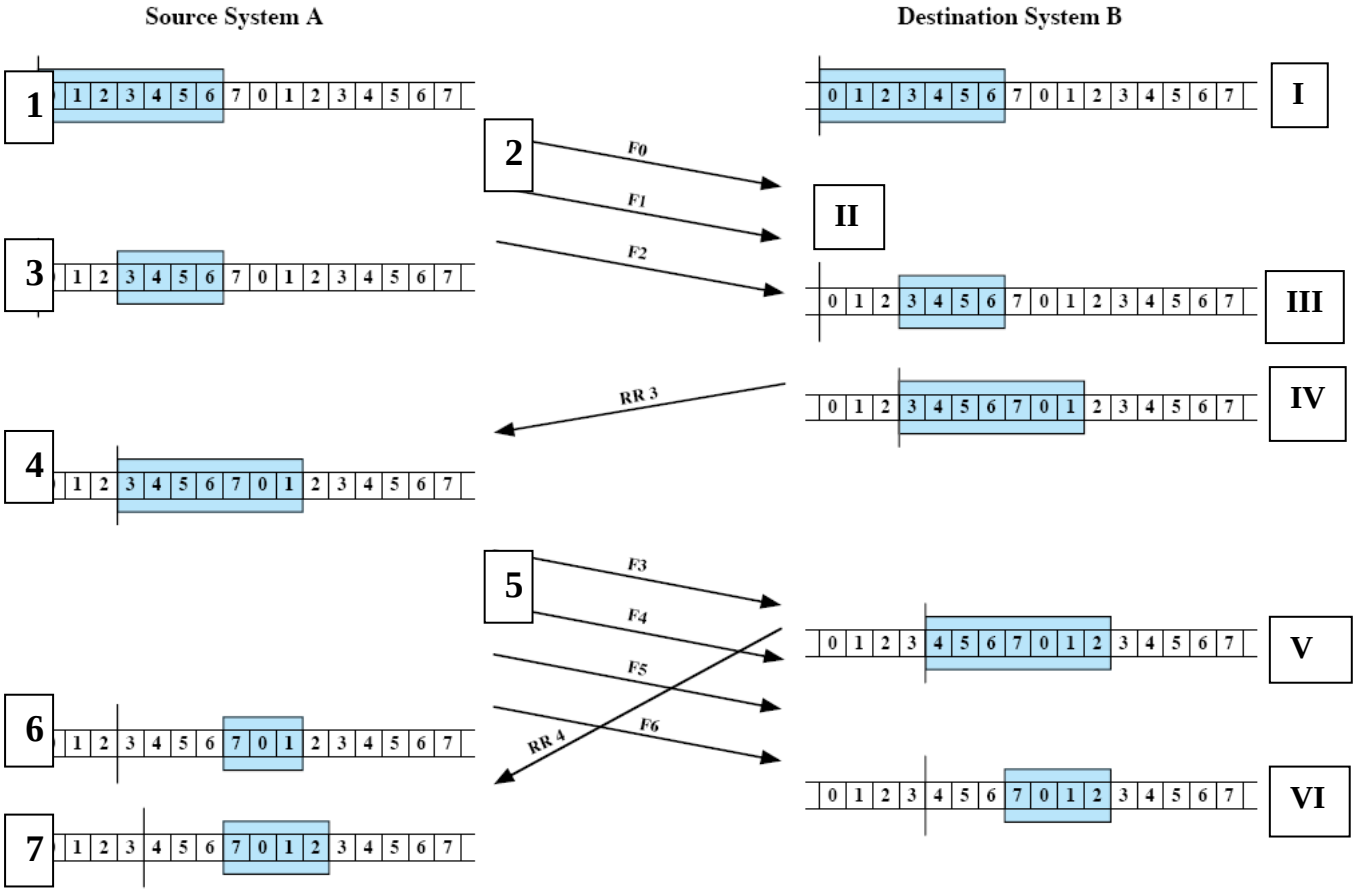
1. Frames that have been received and acknowledged.
2. Frames that have been received, but not acknowledged
3. Frames that have not been received, but we are allowed (or expect) to receive
4. Frames that have not been received, and we are not allowed (or expect) to receive



Similar actions to the source are taken if the destination receives a frame or sends and ACK.

3.2 Windowing Example

Lets look at how the windowing mechanism works in sliding window, using the example from the textbook.



3.2.1 The Source A (or Transmitter)

The maximum window size is 7 (since with a 3-bit sequence number, the maximum window size is 2^3-1).

1. At the start, the Source has no frames transmitted, and therefore can send up to 7 frames.

2. The Source sends 3 frames: F0, F1 and F2
3. Now there are 3 frames transmitted but not ACKed (0, 1, 2) and 4 frames not transmitted, but allowed (3, 4, 5, 6).
4. After receiving the RR3 (Receive Ready 3, which means: "I have received all the frames you transmitted up until frame 2"), frames 0, 1 and 2 are ACKed. So now there are 0 frames transmitted with no ACK. Hence the number of frames that are allowed to be transmitted is 7: 3, 4, 5, 6, 7, 0, 1.
5. The Source sends 4 frames: F3, F4, F5, F6.
6. Now there are 4 frames transmitted but not ACKed (3, 4, 5, 6) and 3 frames not transmitted but allowed (7, 0, 1).
7. After receiving RR4 (which means: "I have received all frames you transmitted up until frame 3"), frame 3 is ACKed. So now there are 3 frames transmitted with no ACK and 4 frames allowed to be transmitted.

3.2.2 The Destination B (Receiver)

I. At the start, the Destination is ready to receive up to 7 frames.
II. The Destination receives 3 frames: F0, F1 and F2.
III. Now there are 3 frames received but not ACKed (0, 1, 2) and 4 frames ready to be received (3, 4, 5, 6).
IV. After sending RR3, frames 0, 1 and 2 have been ACKed, and now the Destination is ready to receive 7 frames.
V. After receiving frame F3, the Destination sends an ACK (RR4) saying it is ready to receive frame 4. The receipt of F3 followed by sending of RR4 effectively shifts the window along 1 frame.
VI. After receiving frames F4, F5 and F6 they become frames that are received but not yet ACKed, and there are 4 frames in the window that we are ready to receive.

3.3 Efficiency

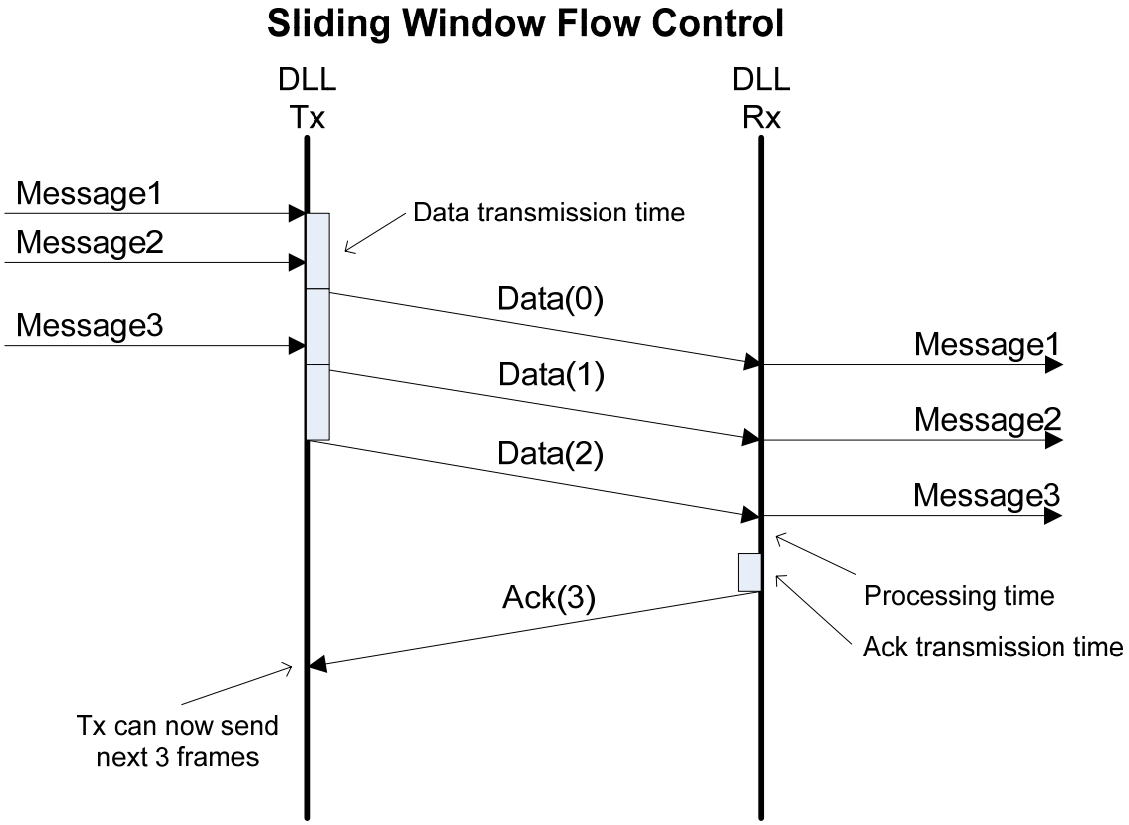


Figure 2: Sliding Window Flow control

Lets look at the best case efficiency of sliding window. By best case we mean that there are some cases when this can't be achieved, especially of the receiver acknowledges frames on a more regular basis.

The sender can transmit W frames. Lets assume the receiver sends one ACK frame after it has received W DATA frames. The ACK indicates that the receiver is ready to receive the next W frames. [Note: it is not necessary for the receiver to send only one ACK for each set of W frames – instead the receiver may send an ACK for each frame. Depending on the transmission and propagation times, this in fact may be more efficient.]

So the time it takes to receive the ACK is:

T = W (DataTransmission) + Propagation + Processing + AckTransmission + Propagation

And the time to send useful data is:

UsefulTime = W x DataTransmission

Hence, we can calculate an efficiency (or throughput). Lets assume AckTransmission and Processing times are 0.

$$Eff = \frac{W \times DataTransmissionTime}{W \times (DataTranssmisionTime) + 2 \times PropagationTime}$$

$$= \frac{Wd}{Wd + 2p}$$

$$= \frac{W}{W + 2\,{}^{p}\!/_{d}}$$

3.4 Comparison of Stop and Wait versus Sliding Window

The plot in Figure 3 shows a comparison of the efficiency of Stop and Wait versus several Sliding Window protocols (with different values of W). We can see that Sliding Window is more efficient than Stop and Wait, and also more efficient with a bigger window.

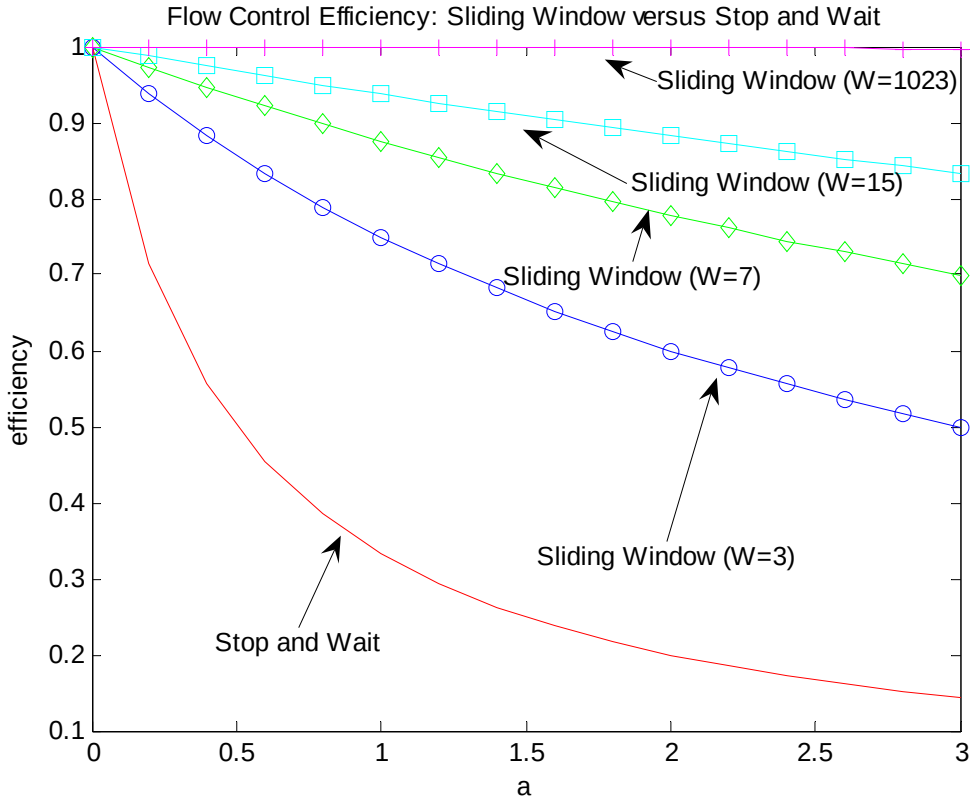


Figure 3: Flow control efficiency

But the disadvantage of Sliding Window is that it is more complex. First, the sender and receiver need to store variables to keep track of the frames sent and received. Also, the receiver must maintain a buffer for W frames. These add to the complexity.

4 Stop and Wait ARQ

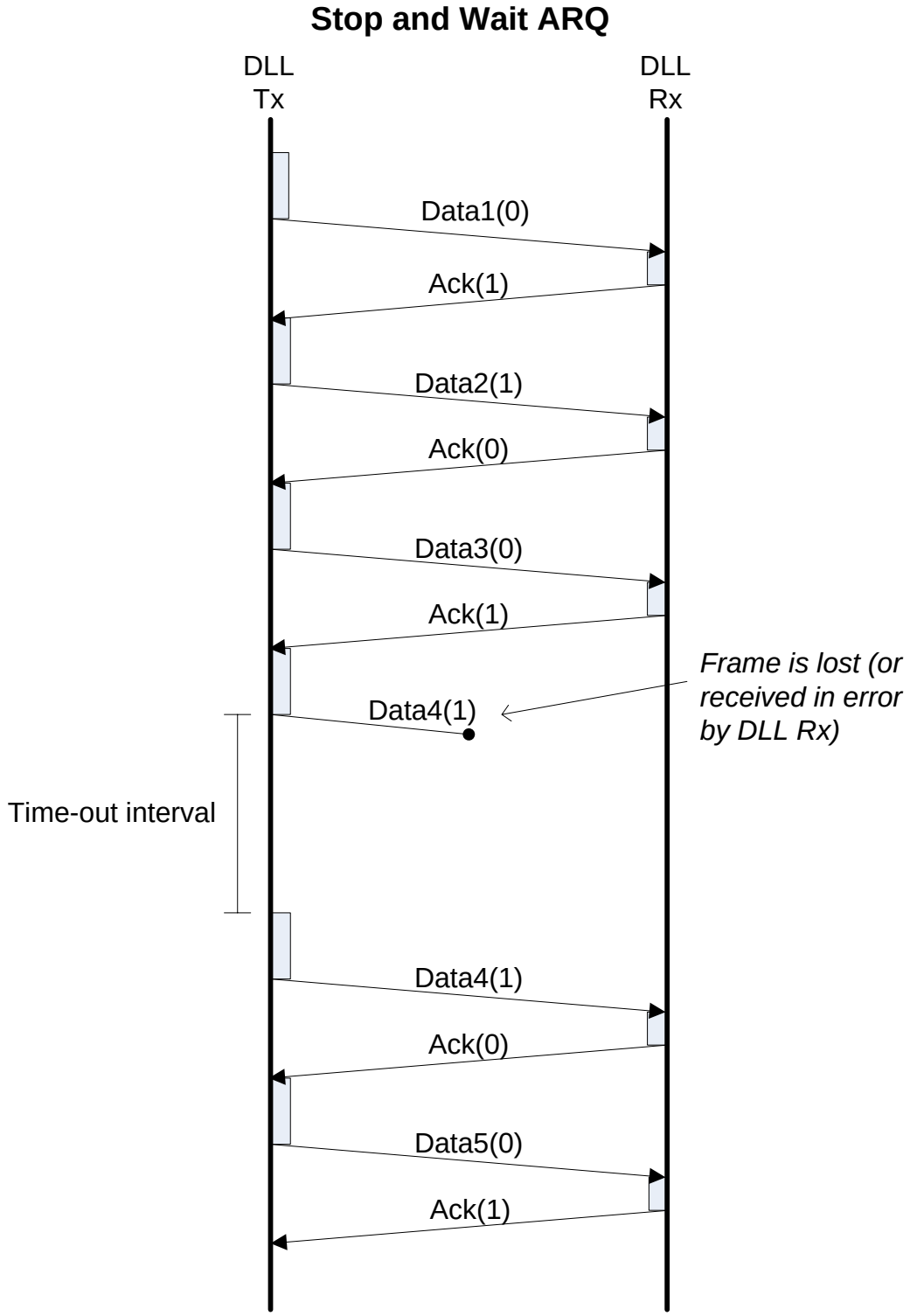


Figure 4: Stop and Wait ARQ

4.1 Efficiency

Lets analyse the efficiency of the Stop and Wait ARQ. We have to make several assumptions:

- The probability a Data frame is lost (or received in error) is P
- The probability an ACK frame is lost is 0 – this simplifies our analysis, and is ok since the ACK frame is very small, there is less chance it is in error.
- A retransmitted frame is never in error – this is not true in practice (and violates our assumption than probability of error is P), but again simplifies our analysis

- Ack transmission time and processing time are 0 (same as what we assumed in Stop and Wait Flow Control)

So for a successful transmission we have (same as Stop and Wait Flow Control):

$$Eff_{succ} = \frac{d}{d + 2p}$$

But what if we get an error. In this case, we transmitted the Data frame, and after a Timeout realise we need to retransmit. The retransmitted frame is successful.

$$Eff_{error} = \frac{d}{d + Timeout + d + 2p}$$

$$= \frac{d}{2d + 2p + Timeout}$$

So:

$$Eff = (1 - P).Eff_{succ} + P.Eff_{error}$$

$$= \frac{(1 - P)d}{d + 2p} + \frac{P.d}{2d + 2p + Timeout}$$

$$= \frac{(1 - P)}{1 + 2a} + \frac{P}{2 + 2a + Timeout}$$

So now, the efficiency depends on:

1. Ratio of propagation time to transmission time (a or p/d)
2. Probability of error, P
3. Timeout

The larger any of the above values, the less efficient Stop and Wait ARQ is. We could do more detailed analysis that consider the effects of lost or errored ACKs. See the textbook if you are interested.

4.2 Timeout Interval

What value should our timeout interval be?

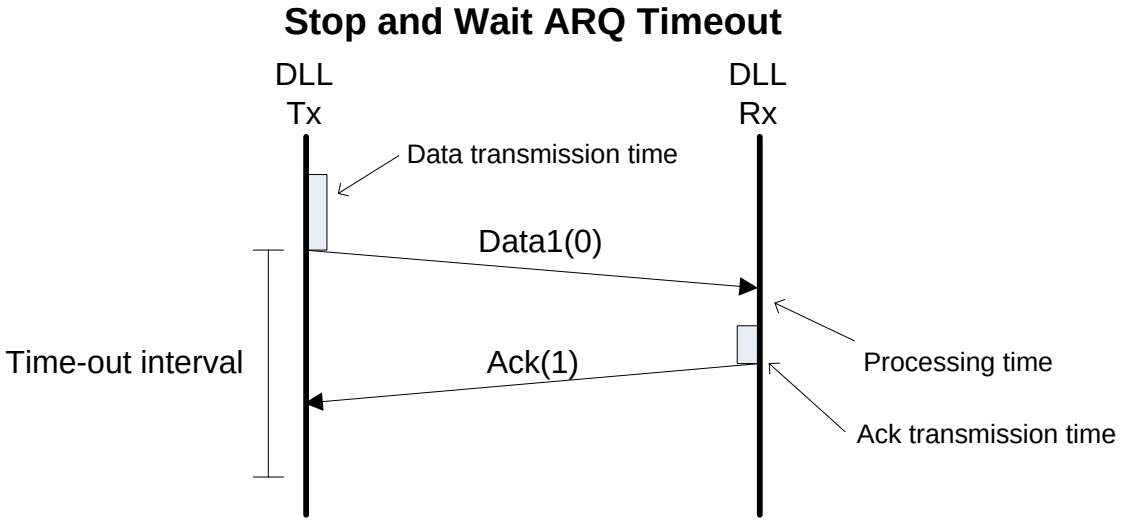


Figure 5: Timeout interval for Stop and Wait ARQ

Once the Tx has sent the DATA frame, the time it takes to receive the ACK is:

PropagationTime + ProcessingTime + AckTransmissionTime + PropagationTime

If the Timeout interval is less than this time, then the Tx may wrongly assume a lost frame, and retransmit. This is a waste (we saw in the Stop and Wait ARQ efficiency that the more retransmissions, the lower the efficiency).

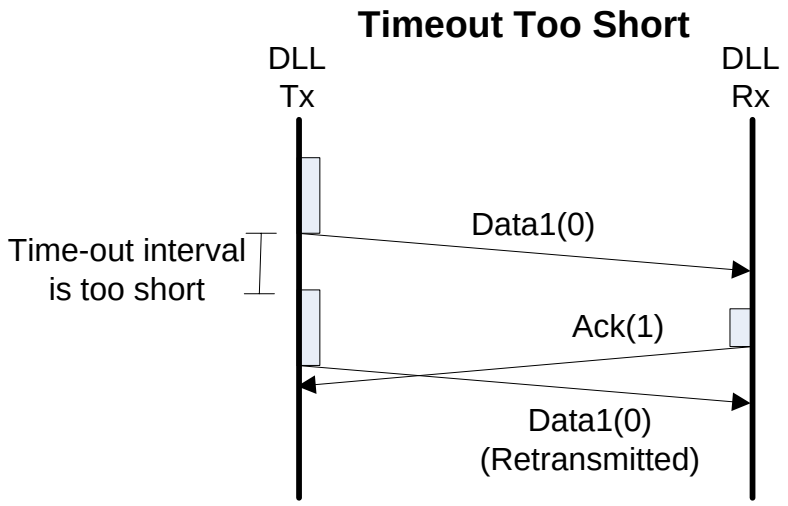


Figure 6: Stop and Wait ARQ with timeout interval too short

If the Timeout interval is *much* greater than this time, then the Tx will wait for a long time before retransmitting in event of an error (or lost frame). This will again decrease our efficiency. (Larger Timeout, smaller efficiency).

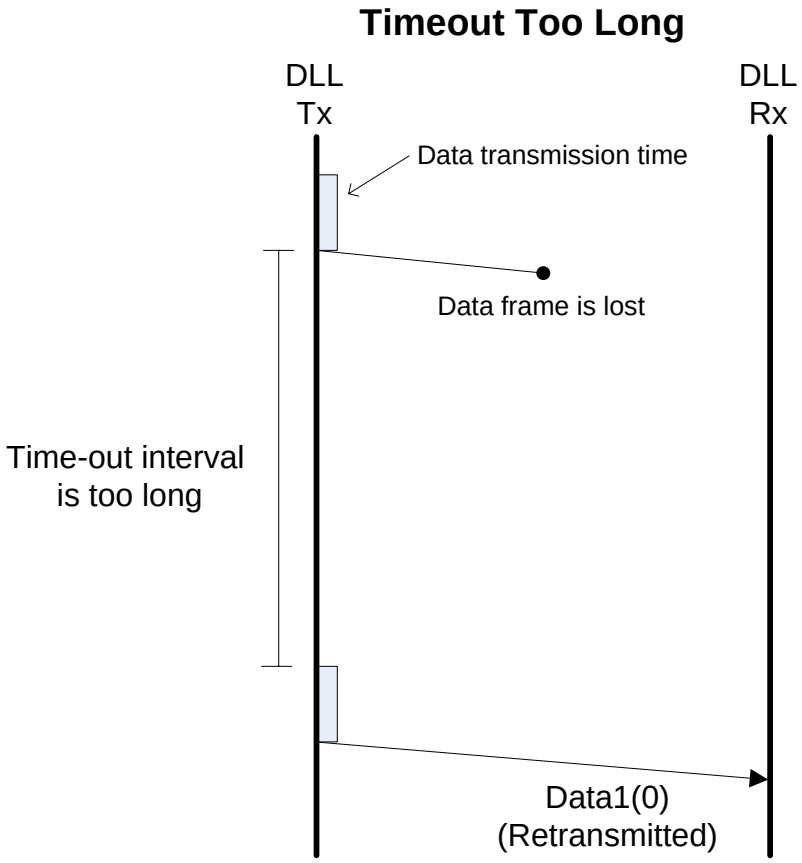


Figure 7: Stop and Wait ARQ with timeout interval too long

So the best value for the Timeout interval is:

Timeout = PropagationTime + ProcessingTime + AckTransmissionTime + PropagationTime

But in practice, it is difficult to accurately know the:

- ProcessingTime: it depends on the computer, and may vary from nanoseconds (very small) to milliseconds
- PropagationTime: although over wired links, this is quite constant, what about a wireless link where the distance changes (because you are mobile)? The propagation time will therefore change, and hence difficult to know. Even worse, in a network with multiple links, the transmitter will not know the propagation time of every link. In practice, the propagation time in a network may vary from microseconds up to 100's of milliseconds (even possibly seconds).

In summary, choosing a good value for the Timeout interval is hard! Choosing a bad value for the Timeout interval can reduce the efficiency significantly!

5 Go-Back-N ARQ

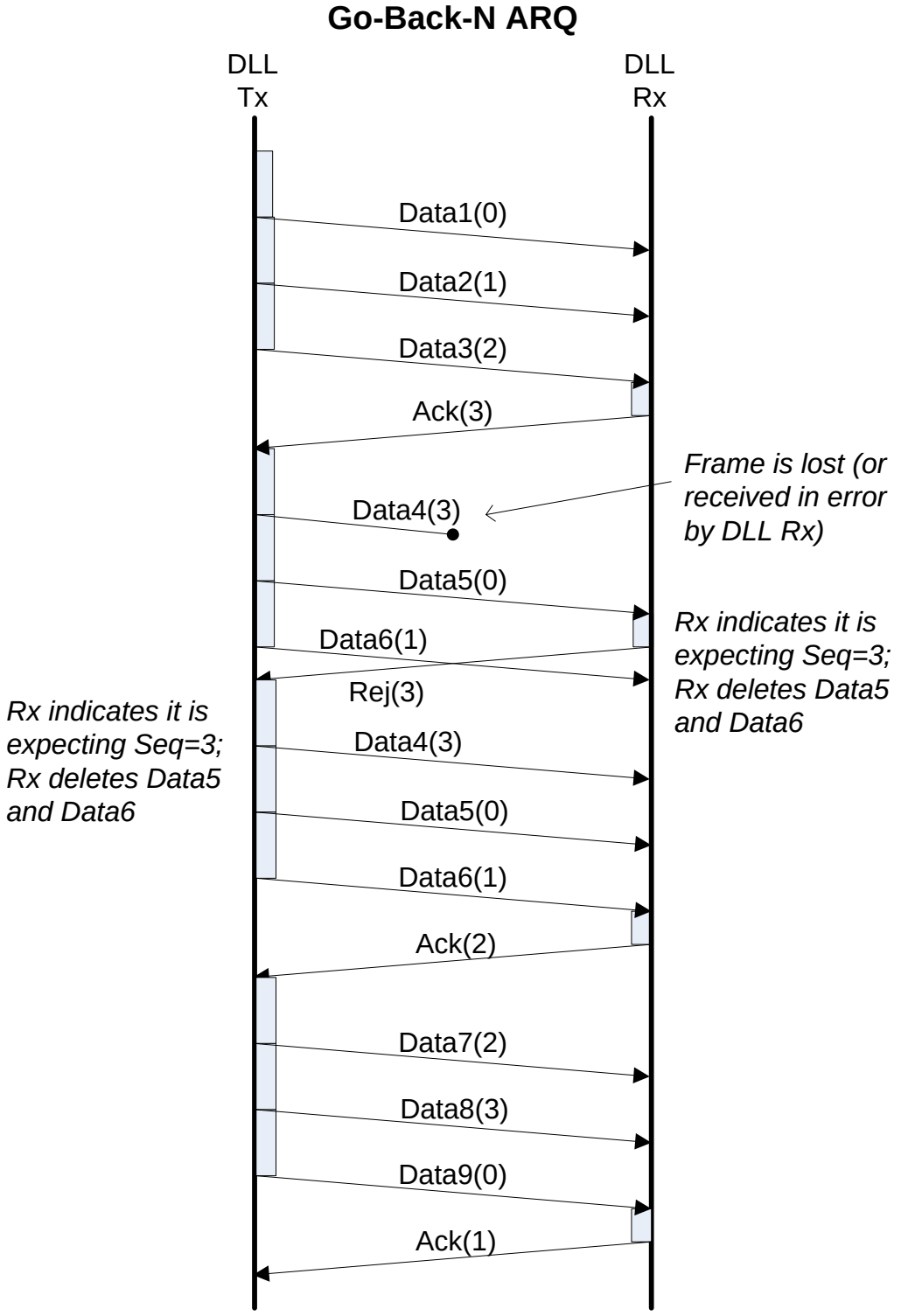


Figure 8: Go-Back-N ARQ with window size 3

In the above Go-Back-N example the Window Size is 3, and the Receiver sends an ACK (or ReceiveReady) after receiving the 3 frames. If the receiver receivers a frame it is not expecting (e.g. because a previous frame was lost), then the receiver immediately sends a Reject message.

Note that the sender needs to retransmit all frames previously sent (and unacknowledged) since the rejected frame with sequence number 3.

6 Selective Repeat ARQ

The concepts are the same as Go-Back-N except the sender only needs to retransmit the frame indicated in the Selective Reject message.