Telecommunication Networks
15B11EC611

DATA LINK LAYER: FLOW AND ERROR CONTROL

- **This PPT is containing the discussion of Flow and Error Control of Data Link Layer.**

- **Kindly refer page numbers: 209 to 221 of the Book_2_Data-and-Computer-Communications-by-WilliamStallings for detailed discussion.**

- **Kindly refer page numbers: 311, 312, 318 to 335 of the Book_1_Data-Communications-and-Networking (4th Edition) - By Forouzan for detailed discussion.**

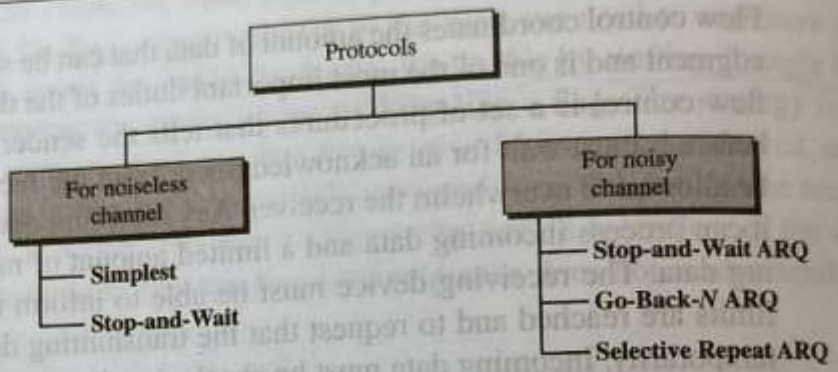
- **Kindly note: For topics, follow this PPT, and for detailed discussion of the topics, follow the book.**

FLOW CONTROL

- **Flow control is a set of procedures that tells the sender how much data it can transmit before it must wait for an acknowledgment from the receiver.**
- **Any receiving device has a limited speed at which it can process incoming data and a limited amount of memory in which to store incoming data.**
- **Each receiving device has a block of memory, called a *buffer, reserved for storing incoming* data until they are processed.**
- **If the buffer begins to fill up, the receiver must be able to tell the sender to halt transmission until it is once again able to receive.**

- **There are two methods of flow control:**
  - **Stop-and-wait**
  - **Sliding window**

Figure 11.5 *Taxonomy of protocols discussed in this chapter*



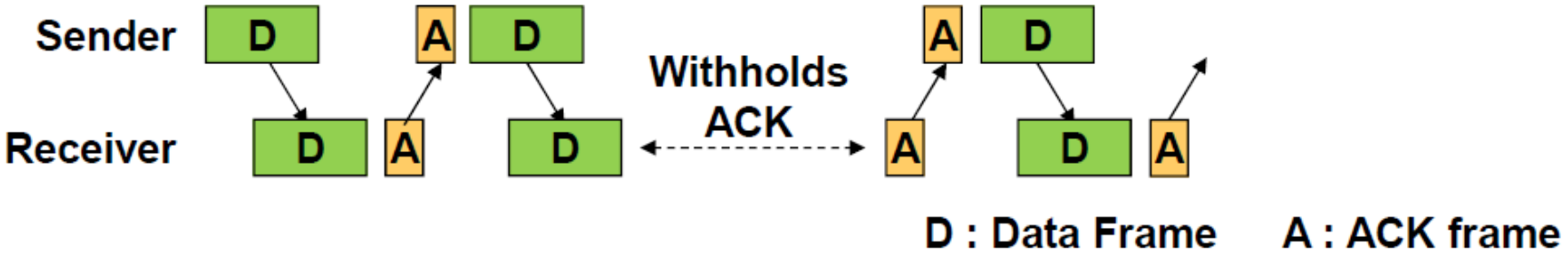
There is a difference between the protocols we discuss here and those used in real networks. All the protocols we discuss are unidirectional in the sense that the data frames travel from one node, called the sender, to another node, called the receiver. Although special frames, called **acknowledgment (ACK)** and **negative acknowledgment** (NAK) can flow in the opposite direction for flow and error control purposes, data flow in only one direction.

In a real-life network, the data link protocols are implemented as bidirectional; data flow in both directions. In these protocols the flow and error control information such as ACKs and NAKs is included in the data frames in a technique called **piggybacking**. Because bidirectional protocols are more complex than unidirectional ones, we chose the latter for our discussion. If they are understood, they can be extended to bidirectional protocols. We leave this extension as an exercise.

Data Link Flow Control Mechanisms

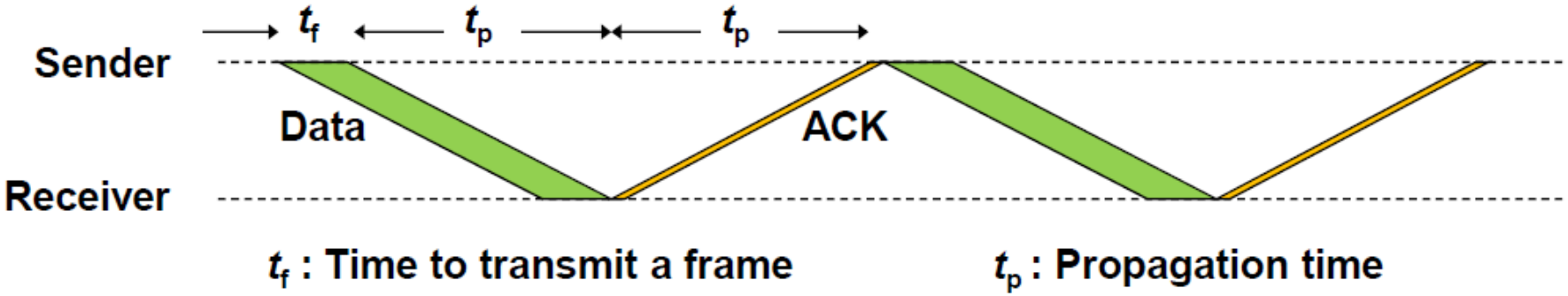
Stop-and-Wait Flow Control

- The sender sends a frame and waits for ACK before sending the next frame.
- The receiver exercises flow control by withholding_ACK.



Link utilization efficiency (U) – No errors case.

- **Link utilization efficiency is the fraction of time the link is effectively utilized for sending frames.**



- **Time required to transmit a frame (t_f)**

  t_f = L/R where L = frame length, R = Bit rate

- **If t_p is propagation time from sender to receiver, the frame is fully received after t_f + t_p sec.**
- **Assuming that ACK transmission time is negligible, ACK is received after total interval of (t_f + 2t_p).**

Link utilization efficiency (U) – No errors case.

- Link utilization efficiency U is given by

  $U = t_f / (t_f + 2t_p) = 1/(1+2A)$ where $A = t_p / t_f$

- Link utilization efficiency is poor when propagation delay is high and frame size is small.

- Approximate analysis based on bandwidth-delay product (N)
- If A >> 1, we can write

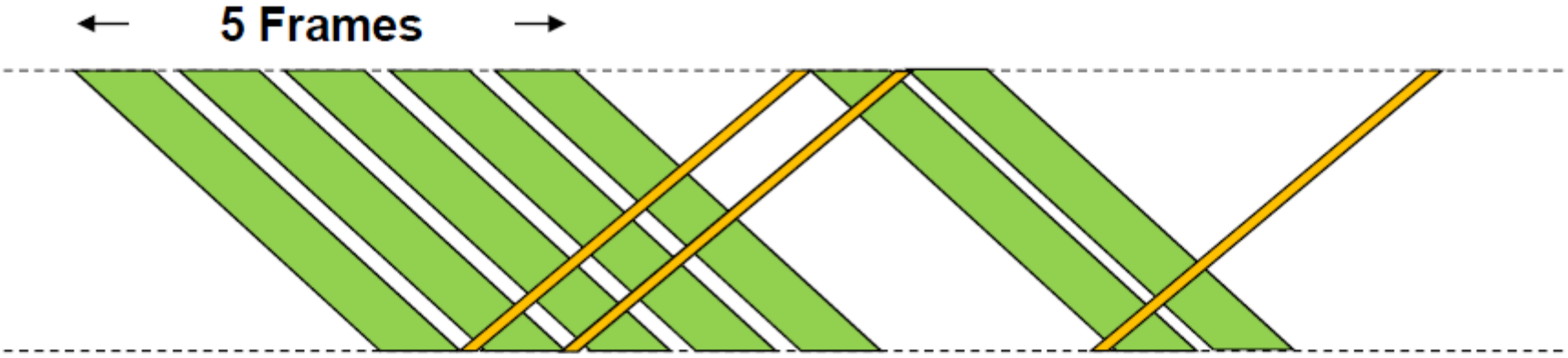
  $U = 1/(1+2A) = 1/(2A) = t_f/(2t_p) = L/(2t_pR) = L/N$ [$t_f = L/R$]

- Bandwidth-delay product $N = 2t_pR$ is number of bits that can be transmitted in one round trip delay.

- If the sender sends one L-bit frame per one round trip delay,
  - link utilization efficiency (U) = L / N.

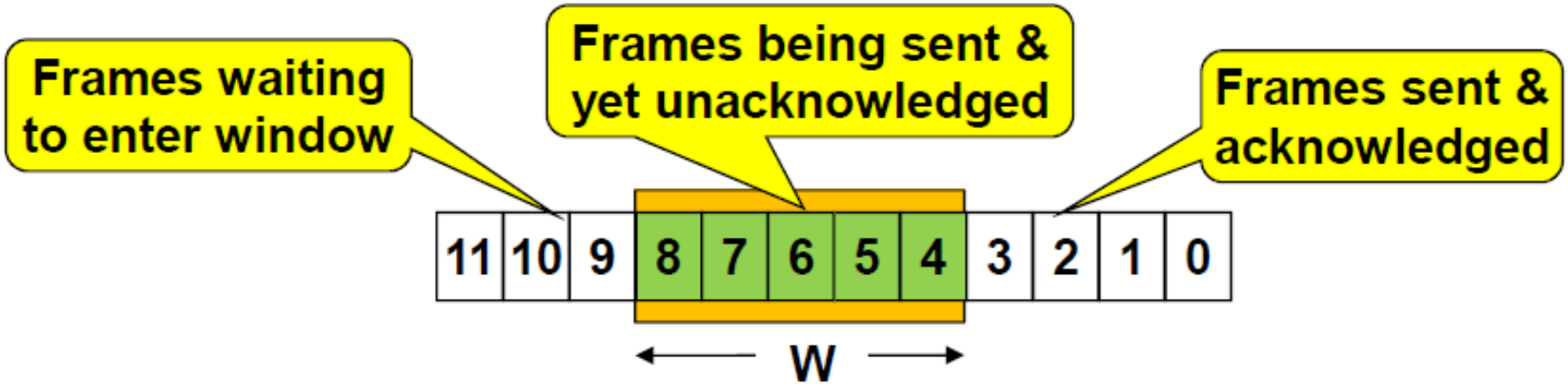
Sliding Window Flow Control

- **Limitation of Stop-and wait → Link utilization efficiency is poor as the sender remains idle until acknowledgement is received for a transmitted frame.**
- **Sliding window flow control allows the sender to send several frames without waiting for acknowledgement.**
- **Thus link utilization is improved.**



Sliding Window Flow Control

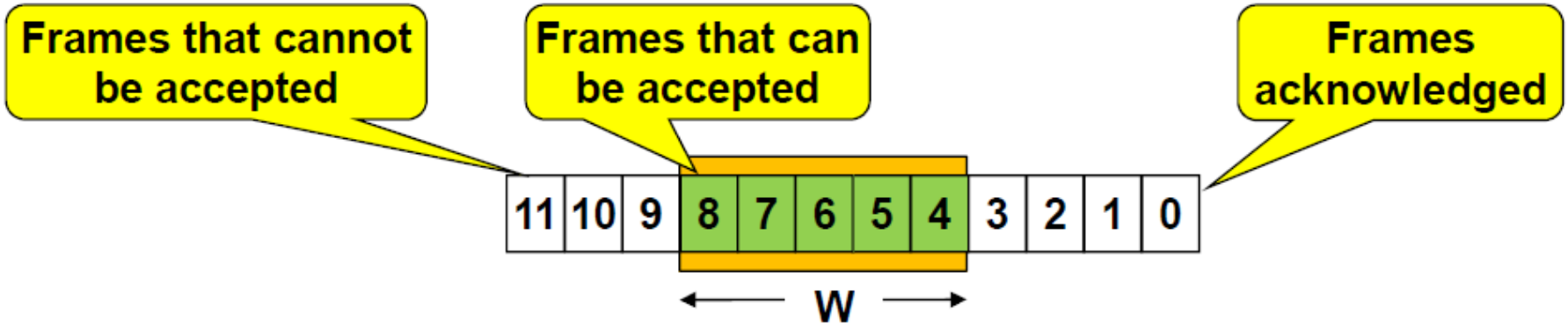
Sliding Window at Sending End



- Sender maintains a window of fixed size (W).
- All the frames carry a sequence number.
- Window contains sequence numbers of the frames that can be sent without waiting for acknowledgement.
- A sequence number of sent frame remains in the window till it is acknowledged.
- An acknowledgement carries a sequence number (N).
  - N is the sequence number of next frame expected. All previous frames (up to N-1) stand acknowledged.
- When acknowledgement is received,
  - Sequence numbers of acknowledged frames move out of window.
  - Window slides to include frames ready for transmission.

Sliding Window Flow Control

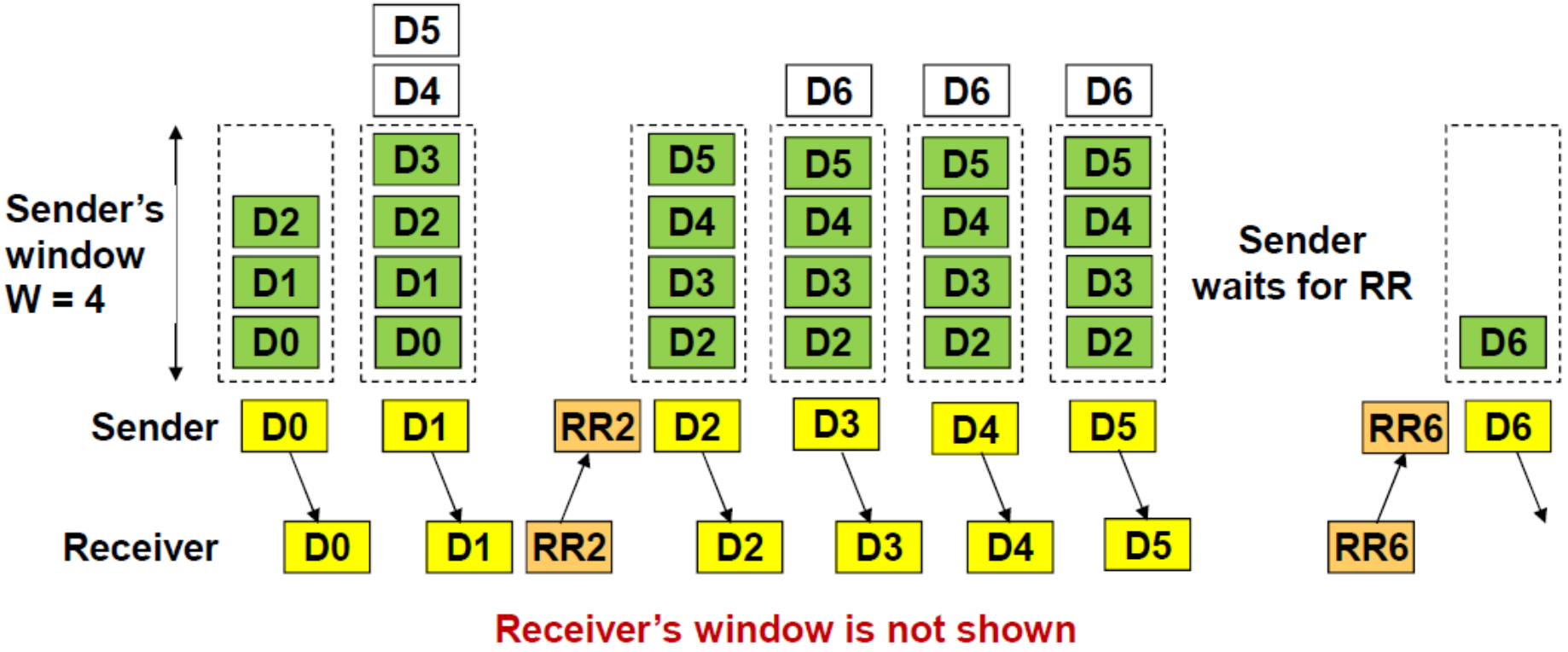
Sliding Window at Receiving End



- Receiver 's window is of same size as sender's window.
- Receiver maintains a window that contains sequence numbers of data frames that receiver is ready to accept.
  - Any received data frame bearing sequence number that is not in the window is discarded.
- As an acknowledgement (RR) is sent, window slides up to the sequence number of next data frame expected.
- Data frames received out of sequence are kept in buffer till missing data frames are received.

Sliding Window Flow Control

- Data frames and ACKs (ACK is called Receive Ready – RR) carry a sequence number.
- RR-N acknowledges all frames up to N-1. It is also indication that the receiver wants data frame N.
- All frames are acknowledged but may not be individually.



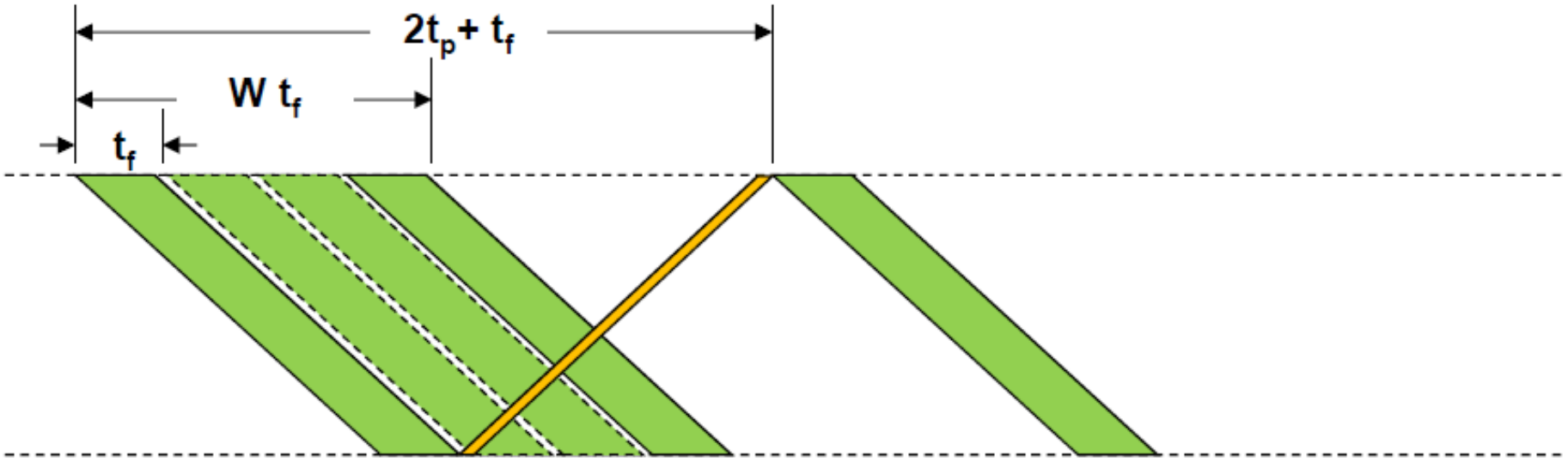
Link utilization

Case 1: Sender exhausts its window before it receives RR.

- Sender sends W frames in time Wt_f and waits for RR.
- RR arrives after time $t_f + 2t_p$
- This Case occurs when
  - $Wt_f < t_f + 2t_p$ or $W < 1 + 2A$ where $A = t_p / t_f$

Link utilization

- $U = Wt_f/(t_f + 2t_p)$ or $U = W/(1 + 2A)$



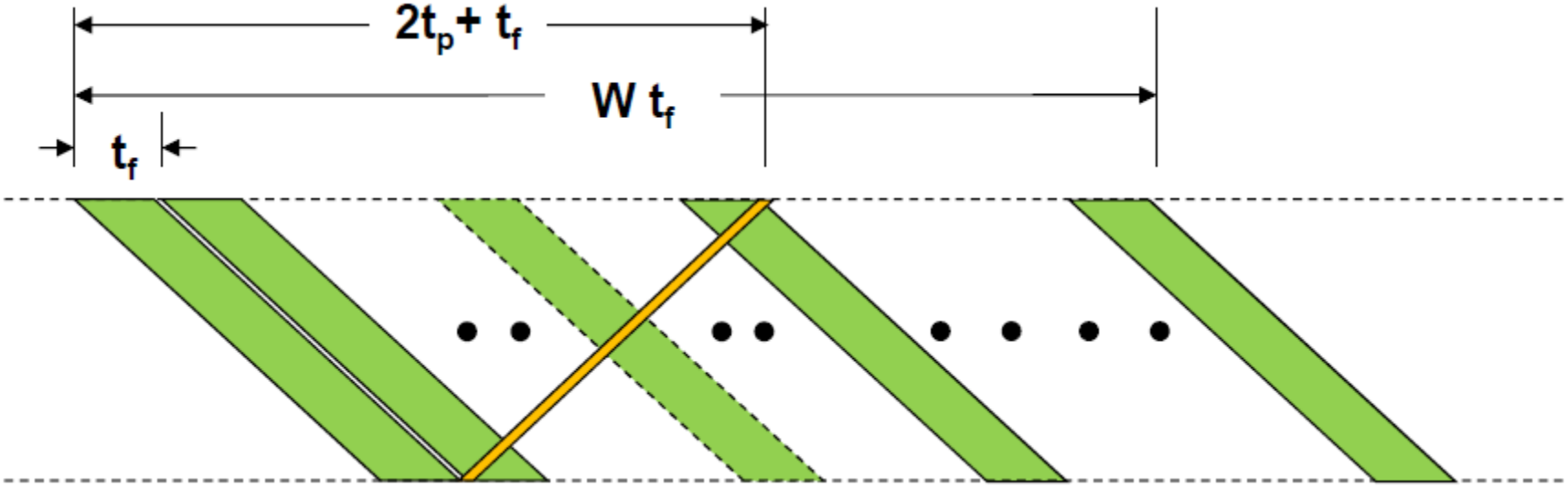
Link utilization

Case 2: Sender keeps receiving RRs before it exhausts.

- Sender keeps sending the frames without any interruption
- This situation occurs when time to transmit W frames is more than $(t_f + 2t_p)$
  - $Wt_f \geq t_f + 2t_p$ or $W \geq 1 + 2A$ where $A = t_p / t_f$

Link utilization efficiency (U)

- $U = 1$



ERROR CONTROL

- **Types of errors**
  - **Content errors : Error in the frame.**
  - **Flow integrity errors: Lost, duplicate frames/acknowledgements.**

- ***Error control* refers to methods of error detection and retransmission.**
- **Error control in data link layer is often implemented simply:**
  - **Any time an error is detected in an exchange, specified frames are retransmitted. This process is called automatic repeat request (ARQ).**

- **Error control in the data link layer is based on automatic repeat request (ARQ), which is the retransmission of data.**

ERROR CONTROL

- **The most common techniques for error control are based on some or all of the following ingredients:**
  - **Error detection:**
  - **Positive acknowledgment: The destination returns a positive acknowledgment to successfully received, error-free frames.**
  - **Retransmission after timeout: The source retransmits a frame that has not been acknowledged after a predetermined amount of time.**
  - **Negative acknowledgment and retransmission: The destination returns a negative acknowledgment to frames in which an error is detected. The source retransmits such frames.**
- **All these mechanisms are referred to as automatic repeat request (ARQ).**
- **Three versions of ARQ have been standardized:**
  1. **Stop-and-wait ARQ**
  2. **Go-back-N ARQ**
  3. **Selective-reject ARQ**

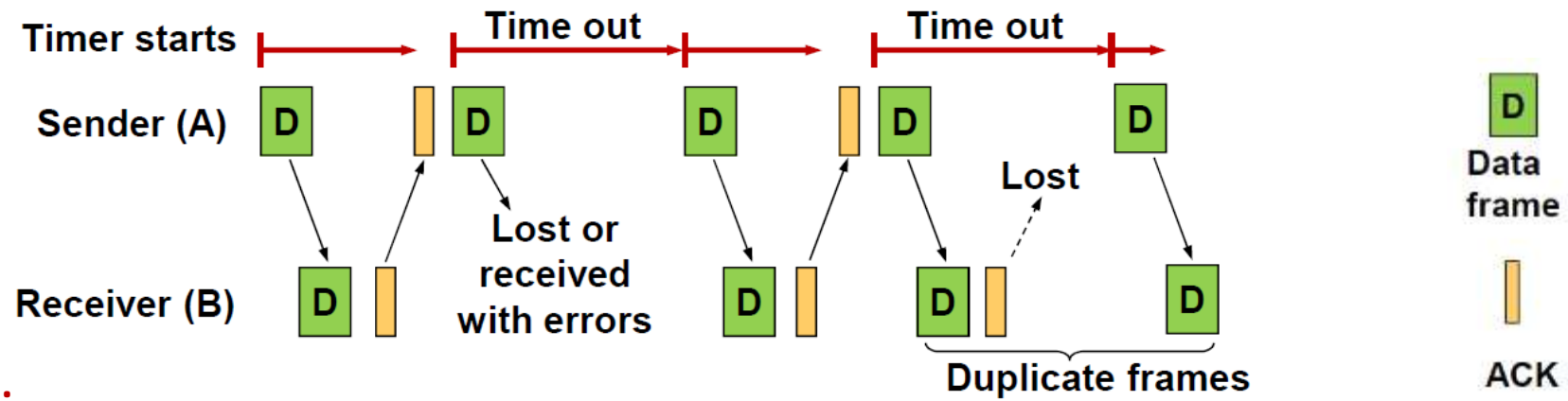
Stop-and-Wait ARQ

- The source station transmits a single frame and then must await an acknowledgment (ACK). No other data frames can be sent until the destination station's reply arrives at the source station.
- Two sorts of errors could occur.
  1. First, the frame that arrives at the destination could be damaged. To account for this possibility, the source station is equipped with a timer.
  2. The second sort of error is a damaged acknowledgment. Consider the following situation. Station A sends a frame. The frame is received correctly by station B, which responds with an acknowledgment (ACK). The ACK is damaged in transit and is not recognizable by A, which will therefore time out and resend the same frame. This duplicate frame arrives and is accepted by B. B has therefore accepted two copies of the same frame as if they were separate. To avoid this duplicacy problem, frames are alternately labeled with 0 or 1, and positive acknowledgments are of the form ACK0 and ACK1. In keeping with the sliding-window convention, an ACK0 acknowledges receipt of a frame numbered 1 and indicates that the receiver is ready for a frame numbered 0.

Stop-and-Wait ARQ

Error control using time outs (ACK not received case)

- Sender sends a frame (keeps a copy of frame) & waits for acknowledgement. After time out retransmits the frame.



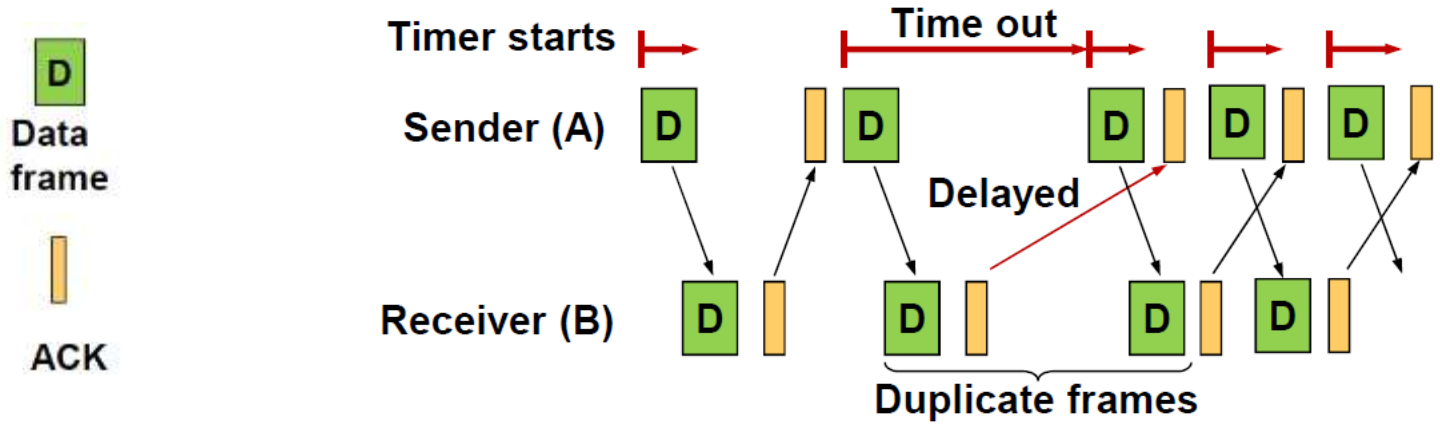
Steps:

1. A sends data frame & gets ACK before time out.
2. A sends next data frame which is lost in transit or received with errors by B.
3. After time out A retransmits the data frame.
4. B receives data frame & returns ACK.
5. A sends next data frame after receipt of ACK.
6. B receives data frame & returns ACK. ACK is lost in transit.
7. A retransmits data frame after time out.

Undetected error situation → B gets duplicate frame but is unaware of this fact. Session continues without detection of this error.

Stop-and-Wait ARQ

Error control using time outs (ACK delayed case)



Steps:

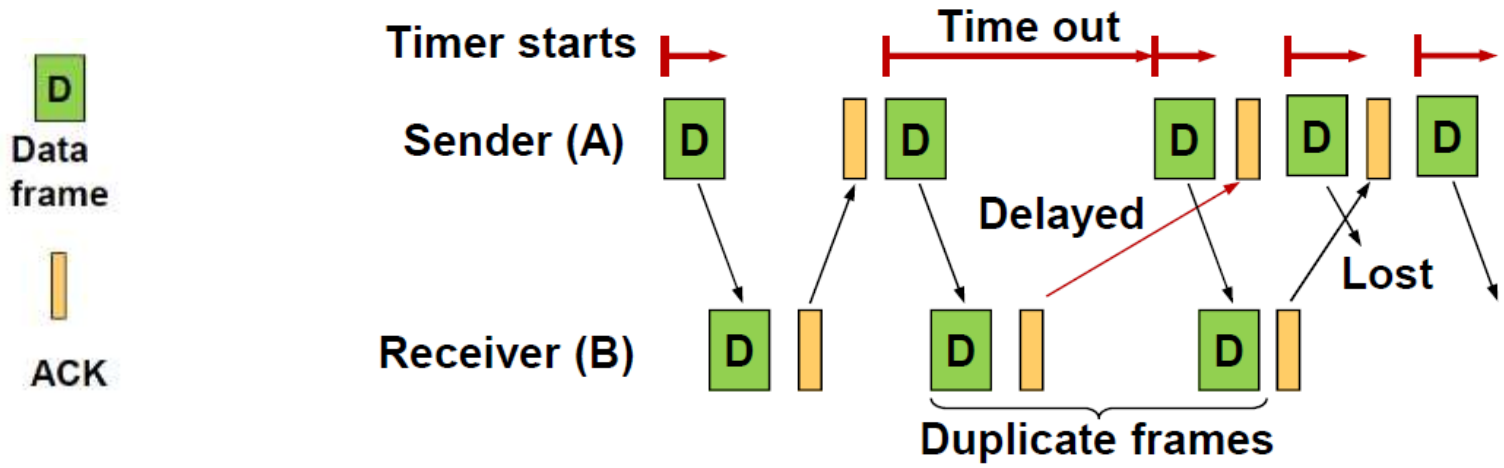
1. B receives data frame & returns ACK.
2. ACK gets delayed. A retransmits data frame after time out. B gets duplicate frame. B releases ACK.
3. A gets delayed ACK. It transmits next data frame. Immediately thereafter it receives ACK of duplicate frame. But A assumes it is ACK of frame just released.
4. Thereafter the communication loses synchronism.

Undetected error situation

- Duplicate frame.
- Communication loses synchronism. ACKs pertain to previous frame.

Stop-and-Wait ARQ

Error control using time outs (ACK delayed and Data frame lost case)



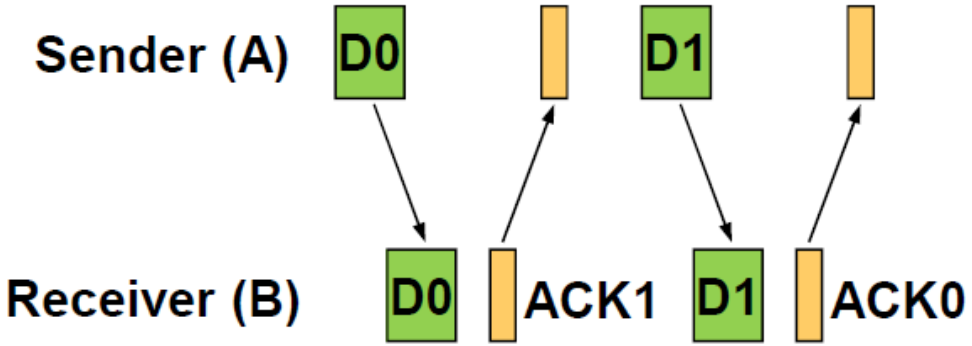
Undetected error situation

- **A & B both are unaware of the loss of frame and their session continues.**

❖ **To avoid this problem, frames are alternately labelled with 0 or 1, and positive acknowledgments are of the form ACK0 and ACK1.**

Stop-and-Wait ARQ

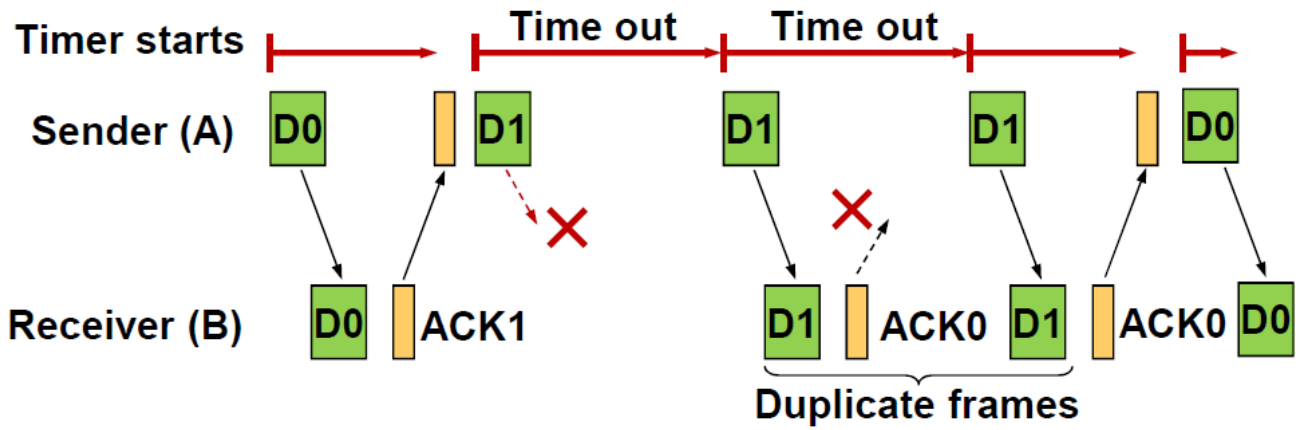
Error control using numbered frames



- Data frames are given numbers 0 or 1 alternatively.
- ACK0 indicates that the required data frame is D0.
- ACK1 indicates that the required data frame is D1.
- All data frames are always acknowledged, even duplicate data frames.

Stop-and-Wait ARQ

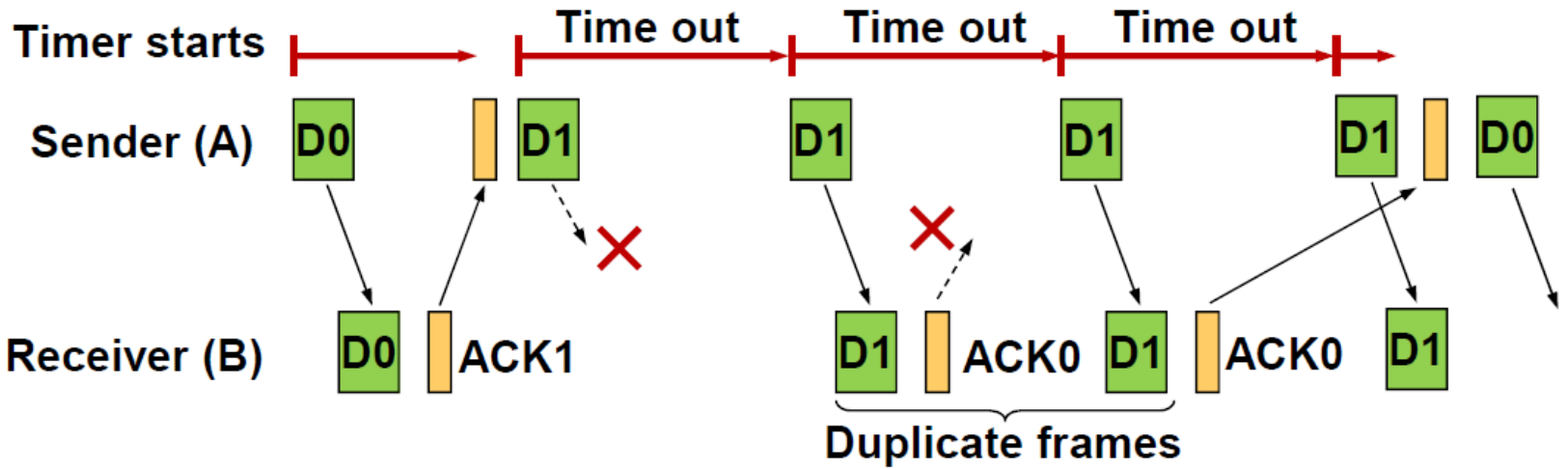
Error control using numbered frames (Lost data or ACK frames)



- **Duplicate D1 is readily detected by B. B discards the frame and again sends ACK0.**
- **A sends next frame D0.**

Stop-and-Wait ARQ

Error control using numbered frames (Delayed ACK)



- Duplicate D1 is detected by B. B discards the frame and again sends ACK0.
- ACK0 is delayed. After time out A retransmits D1.
- B discards D1 as it wants D0. A receives delayed ACK0. It sends next frame D0.

Note:- Error situation is handled effectively.

Stop-and-Wait ARQ

- **Limitations of error control using time outs:**
  - **Inefficient link utilization because the sender must wait for time out for retransmission.**
  - **Generation of duplicate frames.**
  - **Loss of a frame in transit can go undetected.**
  - **Communication can get desynchronized.**

- **Limitation of error control using numbered frames:**
  - **Inefficient link utilization because the sender must wait for time out for retransmission.**

Error Control in Sliding Window

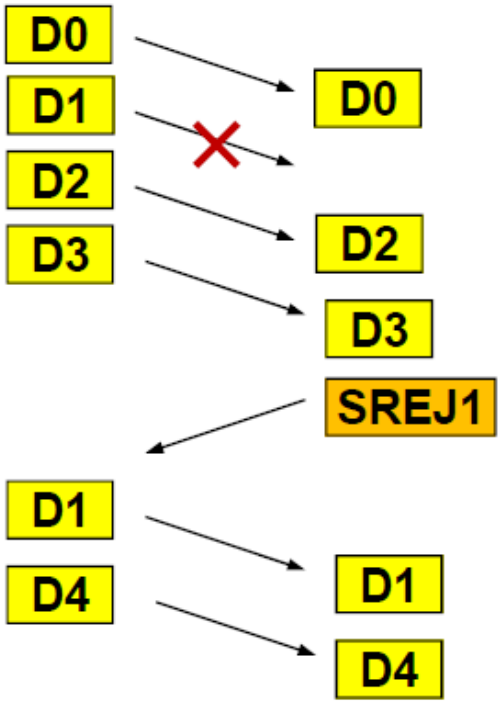
- All data frames & acknowledgements (ACK is called Receive Ready - RR) are numbered.
- A data frame received with content errors is only discarded without initiating any action.
  - Because receiver doesn't know which frame is in error. Error may be in the frame number.
- Receiver detects a missing data frame when it receives an out of sequence frame.

- There are two modes of Sliding Window Error Control
  1. Go-back-N ARQ
  2. Selective-reject ARQ

Error Control in Sliding Window

Selective-reject ARQ:

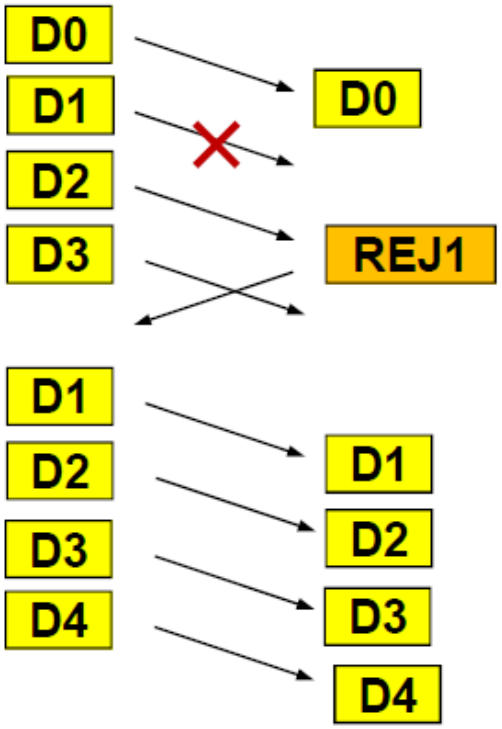
When receiver detects a missing data frame, it requests for retransmission of the missing data frame (N) only by sending SREJ-N.



Selective-reject ARQ

Go-back-N ARQ :

When receiver detects a missing data frame, it requests for retransmission of all the data frames from the missing data frame (N) onwards by sending REJ-N.



Go-back-N ARQ

Go-back-*N* ARQ

- In the Go-Back-N Protocol, the sequence numbers are modulo 2^m , where m is the size of the sequence number field in bits.

Resending a Frame

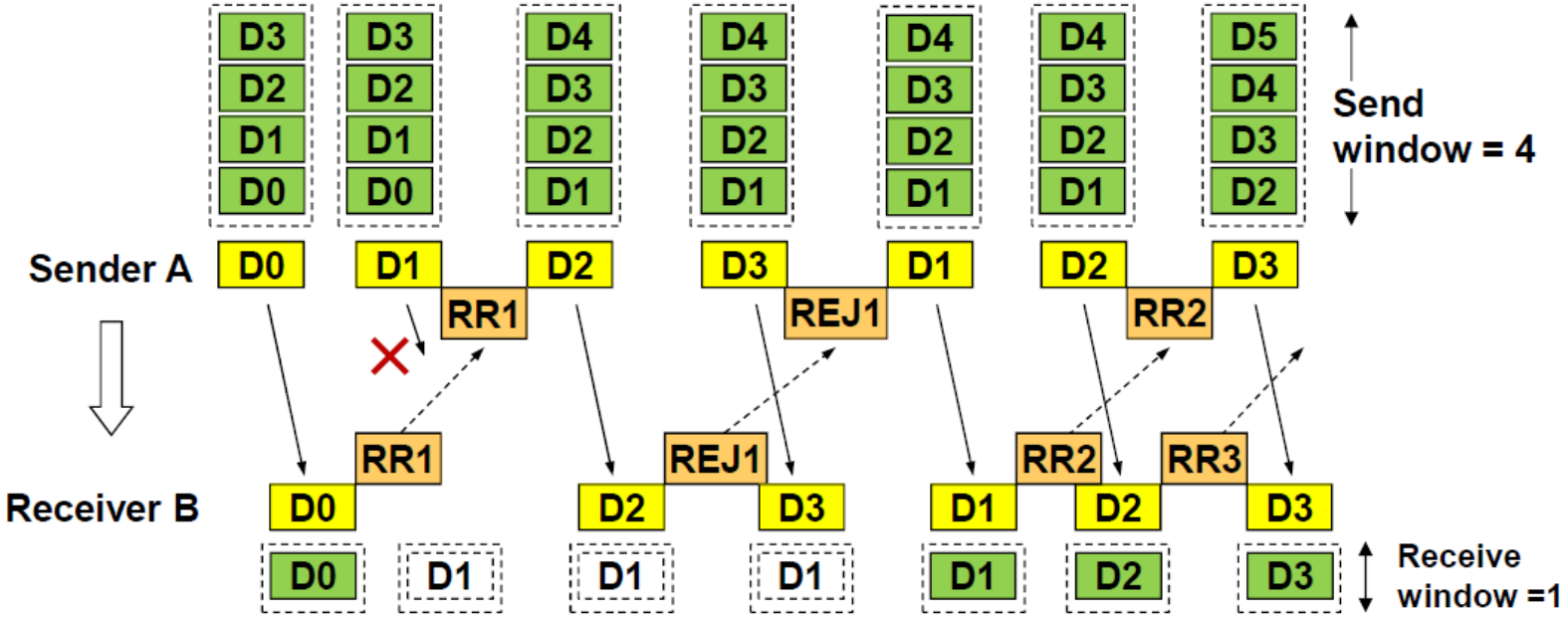
- When the timer expires, the sender resends all outstanding frames. For example, suppose the sender has already sent frame 6, but the timer for frame 3 expires. This means that frame 3 has not been acknowledged; the sender goes back and sends frames 3, 4,5, and 6 again. That is why the protocol is called Go-Back-N ARQ.

- In Go-Back-N ARQ, the size of the send window must be less than 2^m, the size of the receiver window is always 1.

- Stop-and-Wait ARQ is a special case of Go-Back-N ARQ in which the size of the send window is 1.

Go-back-*N* ARQ

- Receive window = 1 irrespective of send window size.



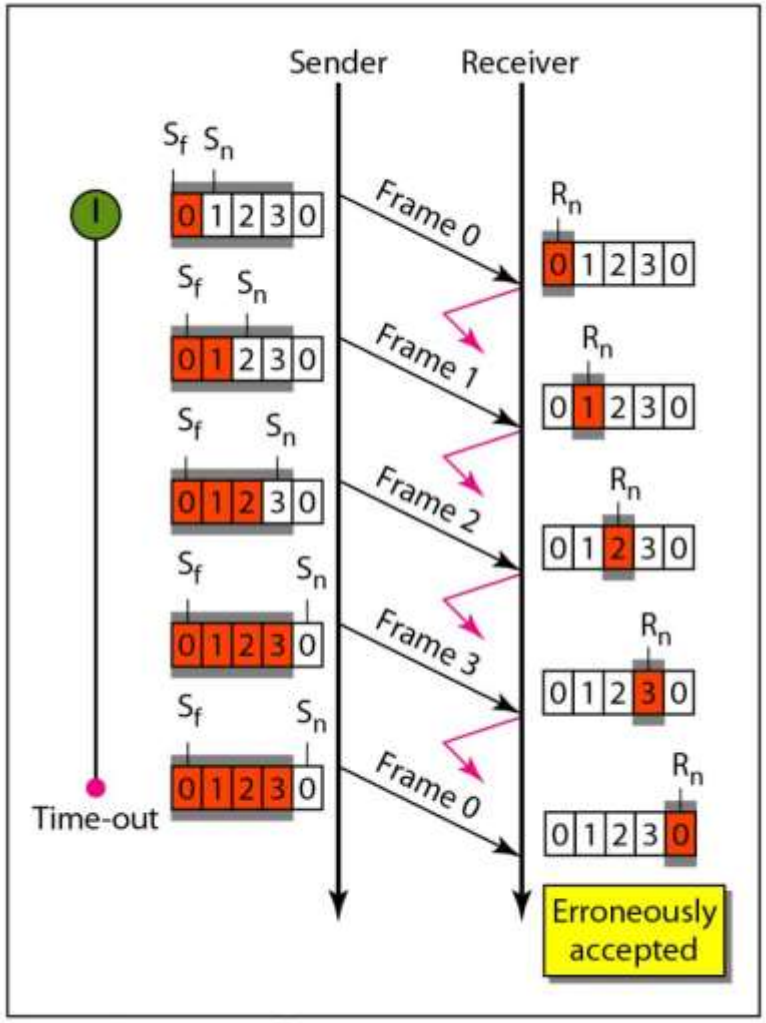
- A has its send window at D0-D3. It sends D0.
- B's receive window is at D0. It receives D0, replies with RR1 and slides receive window to D1.
- A sends D1 which is lost in transit/received with errors by B.
- A receives RR1. It slides window to D1-D4. It sends D2 & D3.
- B does not accepts D2 as its window does not contain this sequence number. B sends REJ1. B receives D3. D3 is also not accepted.
- A receives REJ1. It sends D1.
- B receives D1. It sends RR2 and slides window to D2.
- A sends D2 & D3. It receives RR2. It slides its window to D2-D5.
- B receives D2 and releases RR3. It slides its window to D3.

Go-back-*N* ARQ

In Go-Back-N ARQ, the size of the send window must be less than 2^m.

In order to understand, first consider the window size = 2^m .

As an example, choose m = 2, i.e. window size = 4



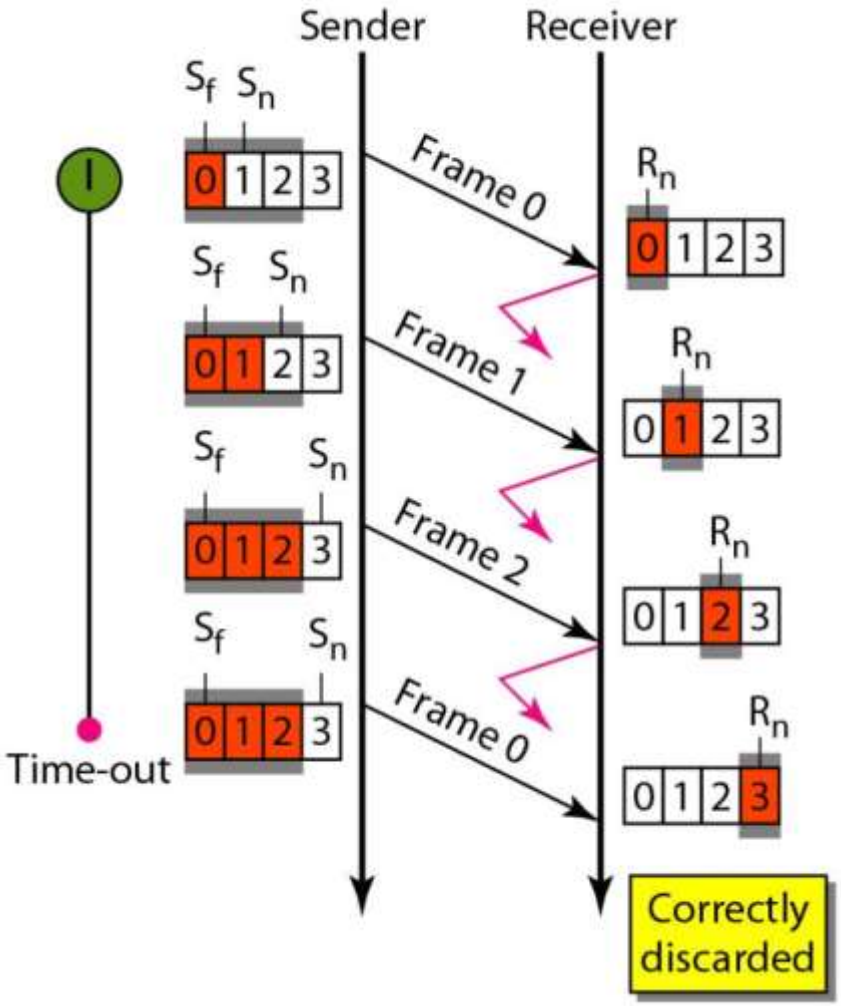
b. Window size = 2^m

1. The timer starts as soon as the first frame is sent. As the first frame receives, Receiver sends ACK. but assume that it is lost due to some reasons.
2. But as, window size is 4 here, the sender will continue sending next frames, till it reaches the max window size. (4 in this case).
3. As there is no loss in the frames, the frames will successfully reach the receiver.
4. Assume all the ACK is lost. and after a while timer times out. So the sender will assume the frames are lost or the ACK are lost. So it will resent the data from beginning.
5. But at the receiver end, the frames are successfully arrived, it will be waiting for next frame (in this case 0). And sender will also send the frame 0. Which is the previous frame. But It will be erroneously accepted by the receiver.
6. So it is better to have window size(N) has to be smaller than the sequence number space(S).

Go-back-*N* ARQ

In Go-Back-N ARQ, the size of the send window must be less than 2^m.

Now consider the window size = $2^m - 1 = 3$ (here)



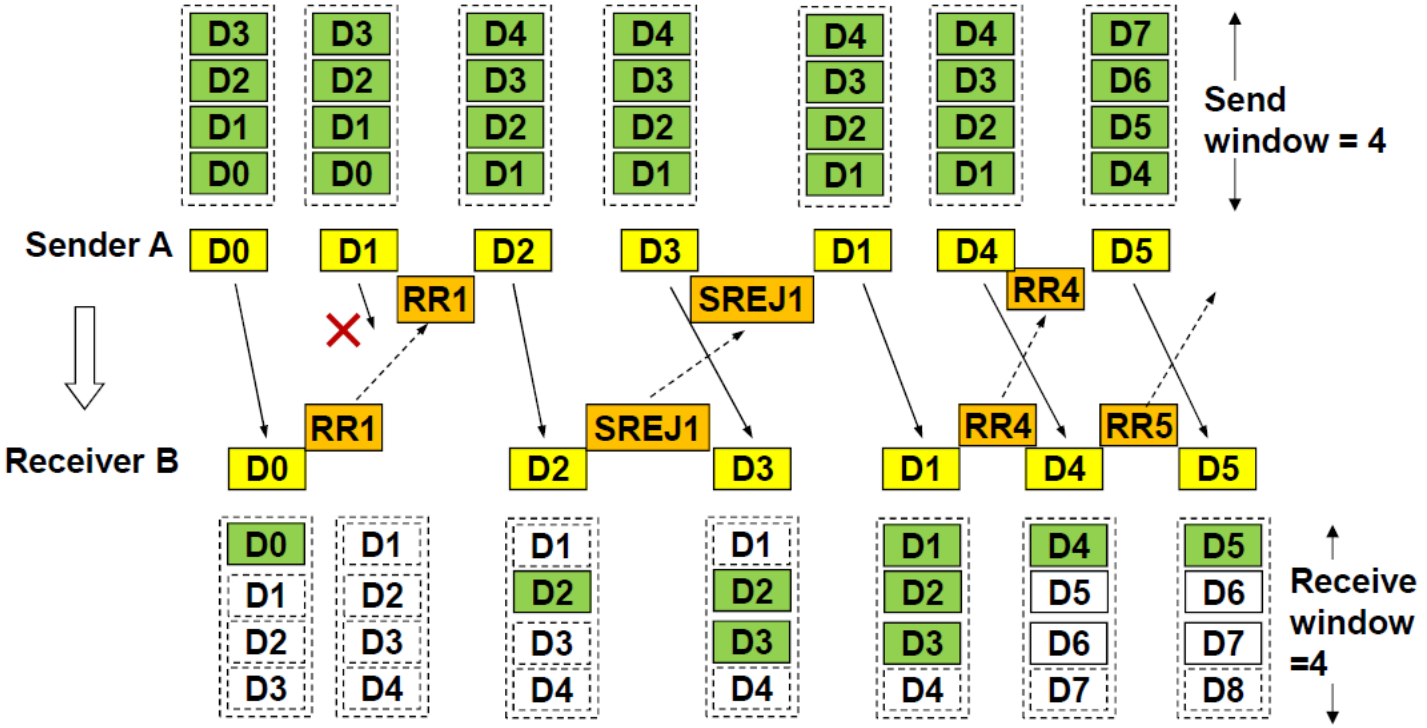
1. **In this diagram the window size is 2^m-1 or 3. In this case also assume the same as we've assumed in the previous case. But the difference is, At last the receiver will be waiting for frame 3 and the sender will send frame 0. So the duplicate frame is correctly discarded. And hence the sender will send the whole set of frame again.**

2. **Therefore in 2nd case there is more possibility of successful communication.**

Go-back-*N* ARQ → Selective-reject ARQ

- Go-Back-N ARQ protocol is very inefficient for a noisy link.
- In a noisy link a frame has a higher probability of damage, which means the resending of multiple frames.
- This resending uses up the bandwidth and slows down the transmission.
- For noisy links, there is another mechanism that does not resend N frames when just one frame is damaged; only the damaged frame is resent. This mechanism is called Selective-reject ARQ.
- It is more efficient for noisy links, but the processing at the receiver is more complex.
  - The transmitter requires more complex logic to be able to send a frame out of sequence. Because of such complications, select-reject ARQ is much less widely used than go-back-N ARQ.
  - Selective reject is a useful choice for a satellite link because of the long propagation delay involved.

Selective-reject ARQ

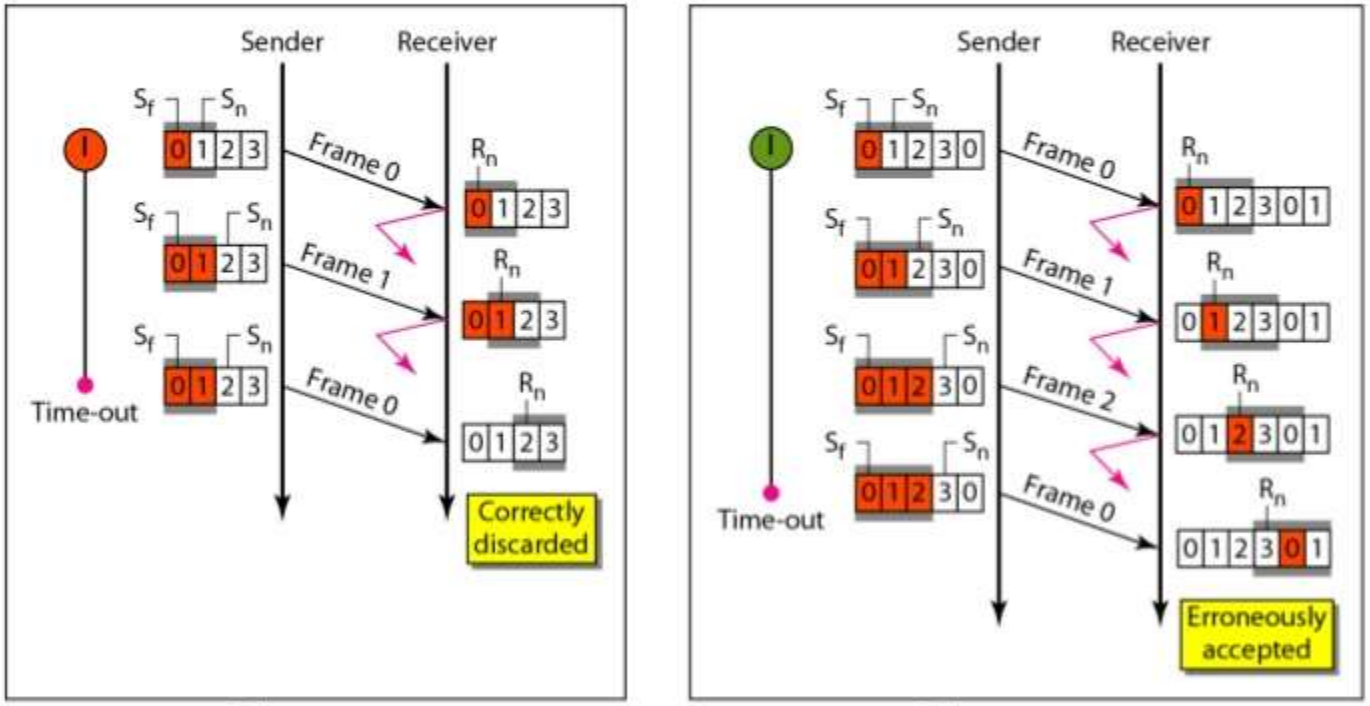


- A has its send window at D0-D3. It sends D0.
- B's receive window is at D0-D3. It receives D0, replies with RR1 and slides receive window to D1-D4.
- A sends D1. D1 is lost in transit/received with errors by B.
- A receives RR1, slides window to D1-D4, sends D2 & D3.
- B accepts D2 and keeps it in its buffer as D1 is yet to be received. B sends SREJ1.
- B receives D3. D3 is treated in the same manner as D2.
- A receives SREJ1. It sends D1.
- B receives D1. As D1, D2 and D3 have been received, it sends RR4. Meanwhile A sends D4.
- A receives RR4, slides its window to D4-D7, sends D5. Meanwhile B receives D4 and releases RR5.

Window Sizes

- The size of the sender and receiver windows must be at most one half of 2^m.
- For an example, we choose m = 2, which means the size of the window is $2^m/2$, or 2.
- Figure 11.21 compares a window size of 2 with a window size of 3.

Figure 11.21 *Selective Repeat ARQ, window size*



Window Sizes

- If the size of the window is 2 and all acknowledgments are lost, the timer for frame 0 expires and frame 0 is resent.
- However, the window of the receiver is now expecting frame 2, not frame 0, so this duplicate frame is correctly discarded.

- When the size of the window is 3 and all acknowledgments are lost, the sender sends a duplicate of frame 0.
- However, this time, the window of the receiver expects to receive frame 0 (0 is part of the window), so it accepts frame 0, not as a duplicate, but as the first frame in the next cycle. This is clearly an error.

THANK YOU