FLOW CONTROL

By:
Dr. Navneet Kaur

Flow Control

- Flow control coordinates the amount of data that can be sent before receiving an acknowledgment and is one of the most important duties of the data link layer.
- In most protocols, 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.
- The receiving device must be able to inform the sending device before those limits are reached and to request that the transmitting device send fewer frames or stop temporarily.
- **Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment.**

Error Control

- Error control is both error detection and error correction.
- It allows the receiver to inform the sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender.
- Any time an error is detected in an exchange, specified frames are retransmitted. This process is called automatic repeat request (ARQ).

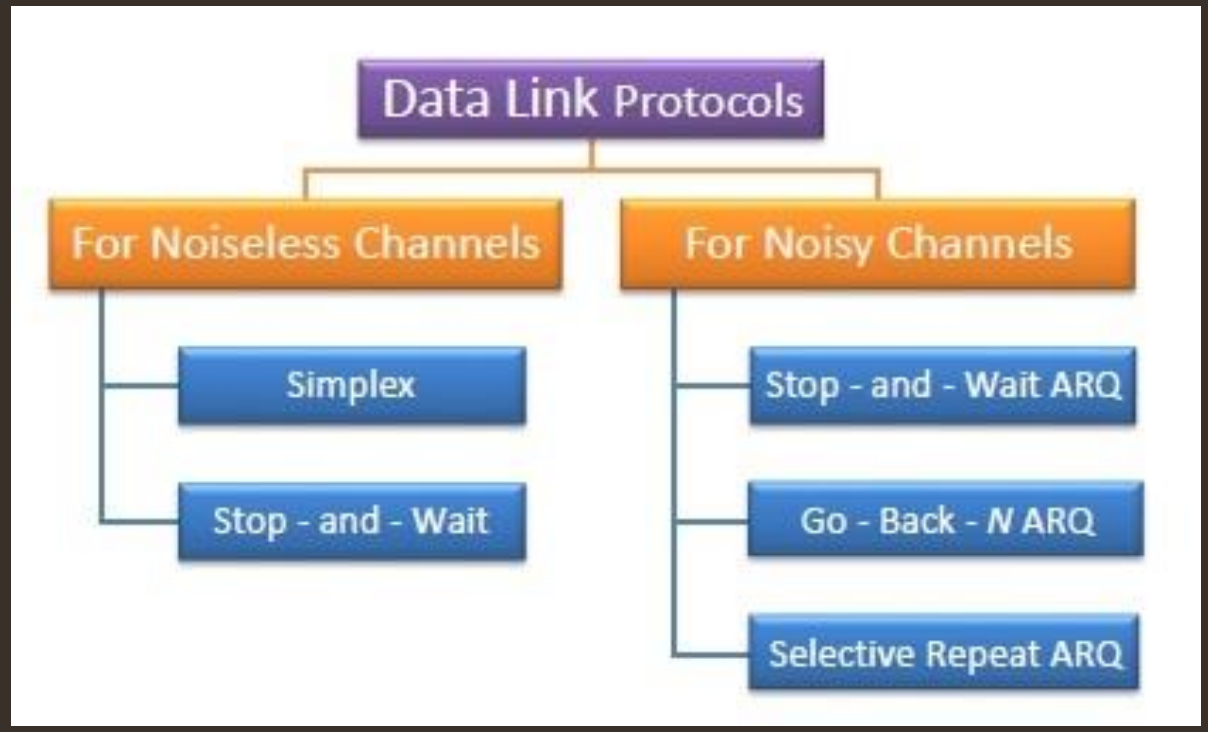
- **Error control in the data link layer is based on automatic repeat request, which is the retransmission of data.**

Elementary Data Link Protocols

- Protocols in the **data link layer** are designed so that this layer can perform its basic functions: framing, error control and flow control.
- Framing is the process of dividing bit - streams from physical layer into data frames whose size ranges from a few hundred to a few thousand bytes.
- Error control mechanisms deals with transmission errors and retransmission of corrupted and lost frames.
- Flow control regulates speed of delivery and so that a fast sender does not drown a slow receiver.

Types of Data Link Protocols

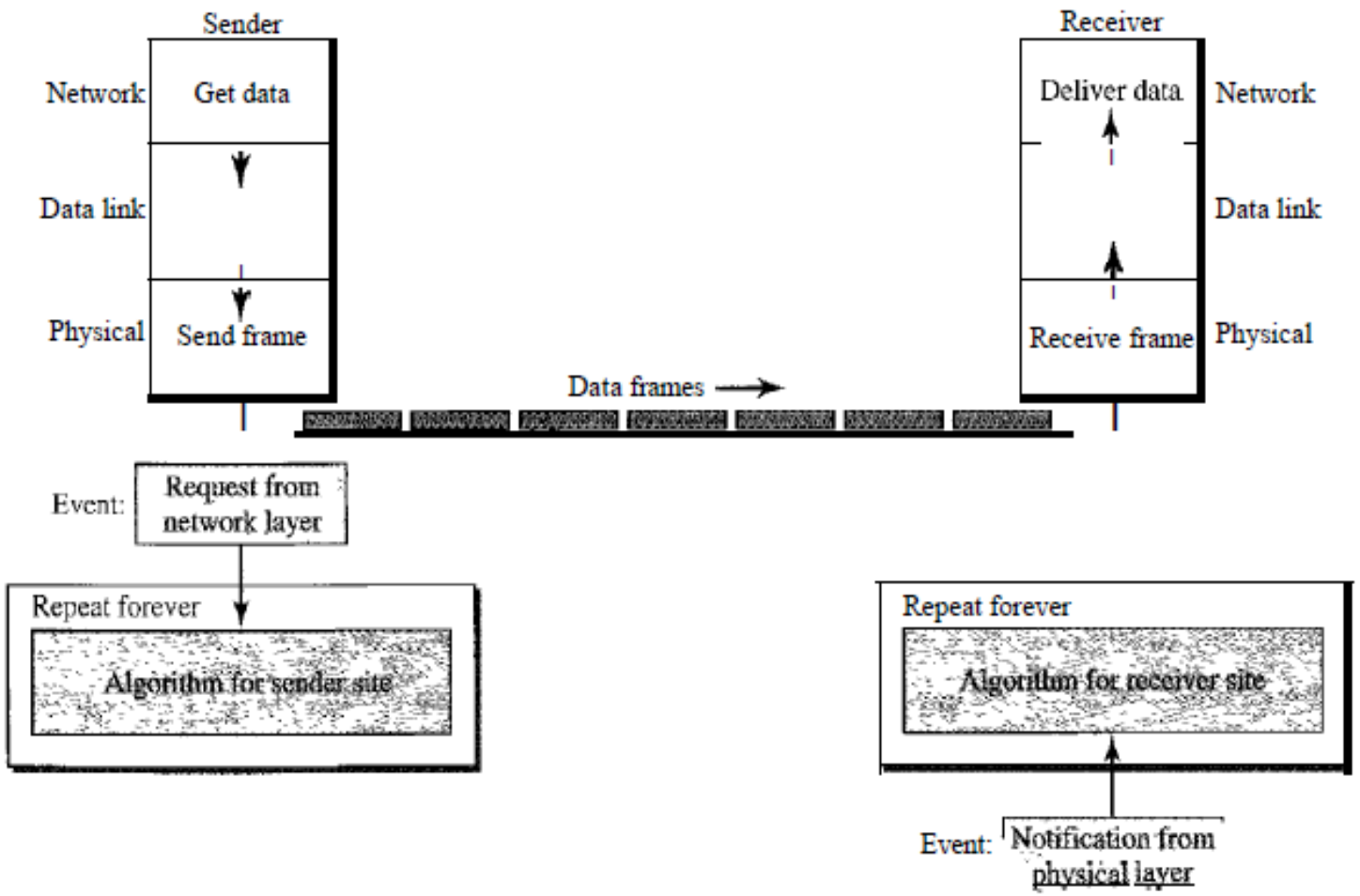
- Data link protocols can be broadly divided into two categories, depending on whether the transmission channel is noiseless or noisy.



Simplex Protocol

- The Simplex protocol is hypothetical protocol designed for unidirectional data transmission over an ideal channel, i.e. a channel through which transmission can never go wrong.
- It has distinct procedures for sender and receiver.
- The sender simply sends all its data available onto the channel as soon as they are available its buffer.
- The receiver is assumed to process all incoming data instantly.
- It is hypothetical since it does not handle flow control or error control.

Figure 11.6 *The design of the simplest protocol with no flow or error control*



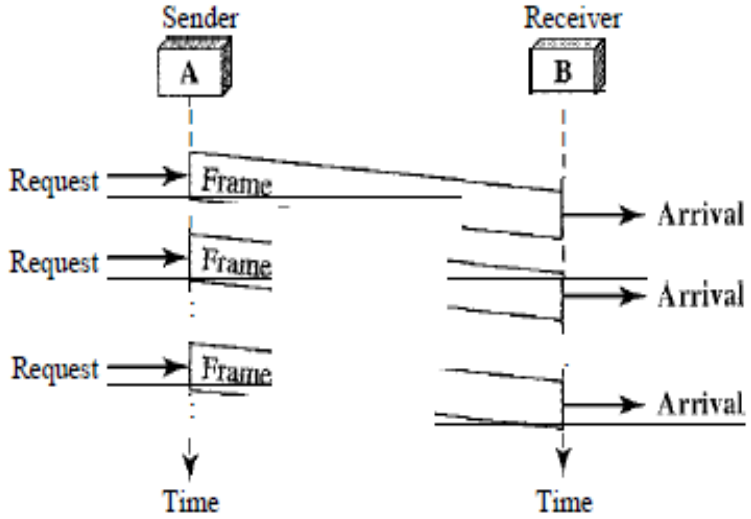
- There is no need for flow control in this scheme.
- The data link layer at the sender site gets data from its network layer, makes a frame out of the data, and sends it.
- The data link layer at the receiver site receives a frame from its physical layer, extracts data from the frame, and delivers the data to its network layer.
- The data link layers of the sender and receiver provide transmission services for their network layers.
- The data link layers use the services provided by their physical layers (such as signaling, multiplexing, and so on) for the physical transmission of bits. Figure 11.6 shows a design.

Example

Figure 11.7 shows an example of communication using this protocol. It is very simple. The sender sends a sequence of frames without even thinking about the receiver. To send three frames, three events occur at the sender site and three events at the receiver site.

Note that the data frames are shown by tilted boxes; the height of the box defines the transmission time difference between the first bit and the last bit in the frame.

Figure 11.7 *Flow diagram for Example 11.1*



Stop-and-Wait Protocol

- If data frames arrive at the receiver site faster than they can be processed, the frames must be stored until their use. Normally, the receiver does not have enough storage space, especially if it is receiving data from many sources.
- This may result in either the discarding of frames or denial of service.
- To prevent the receiver from becoming overwhelmed with frames, we somehow need to tell the sender to slow down.
- There must be feedback from the receiver to the sender.
- The protocol we discuss now is called the Stop-and-Wait Protocol because the sender sends one frame, stops until it receives confirmation from the receiver (okay to go ahead), and then sends the next frame. We still have unidirectional communication for data frames, but auxiliary ACK frames (simple tokens of acknowledgment) travel from the other direction.
- We add flow control to our previous protocol.

Stop-and-Wait Protocol

Working :

- The sender sends data to the receiver.
- The sender stops and waits for the acknowledgment.
- The receiver receives the data and processes it.
- The receiver sends an acknowledgment for the above data to the sender.
- The sender sends data to the receiver after receiving the acknowledgment of previously sent data.
- The process is unidirectional and continues until the sender sends the **End of Transmission (EoT)** frame.

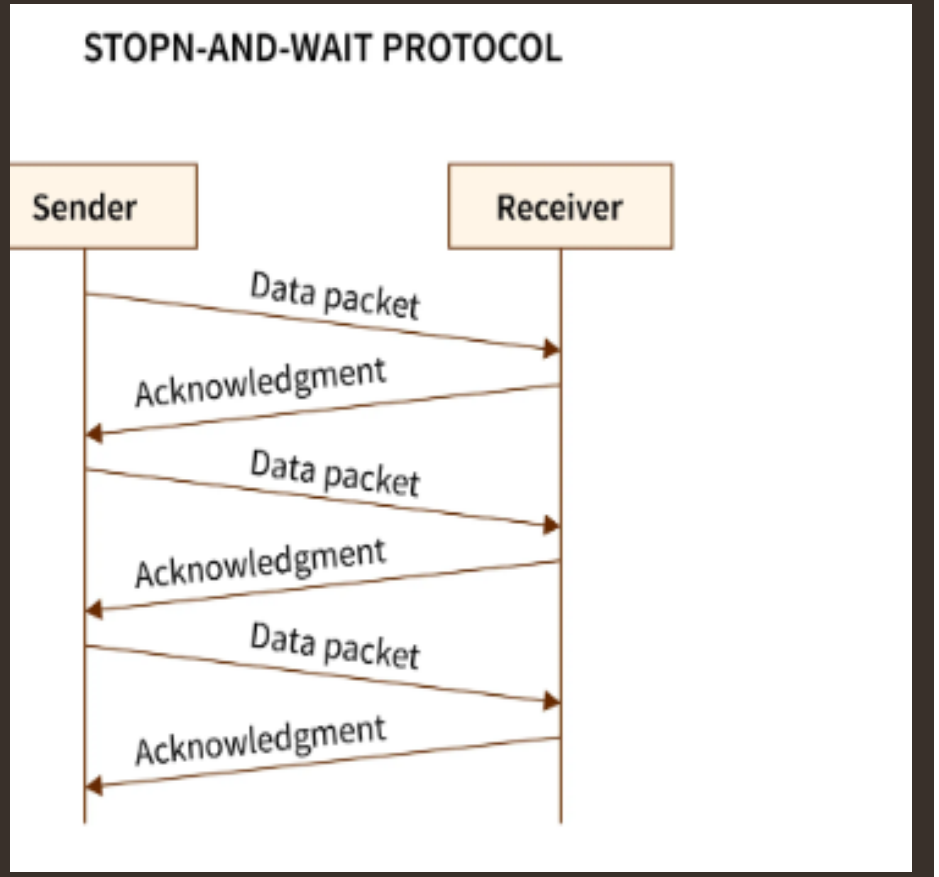
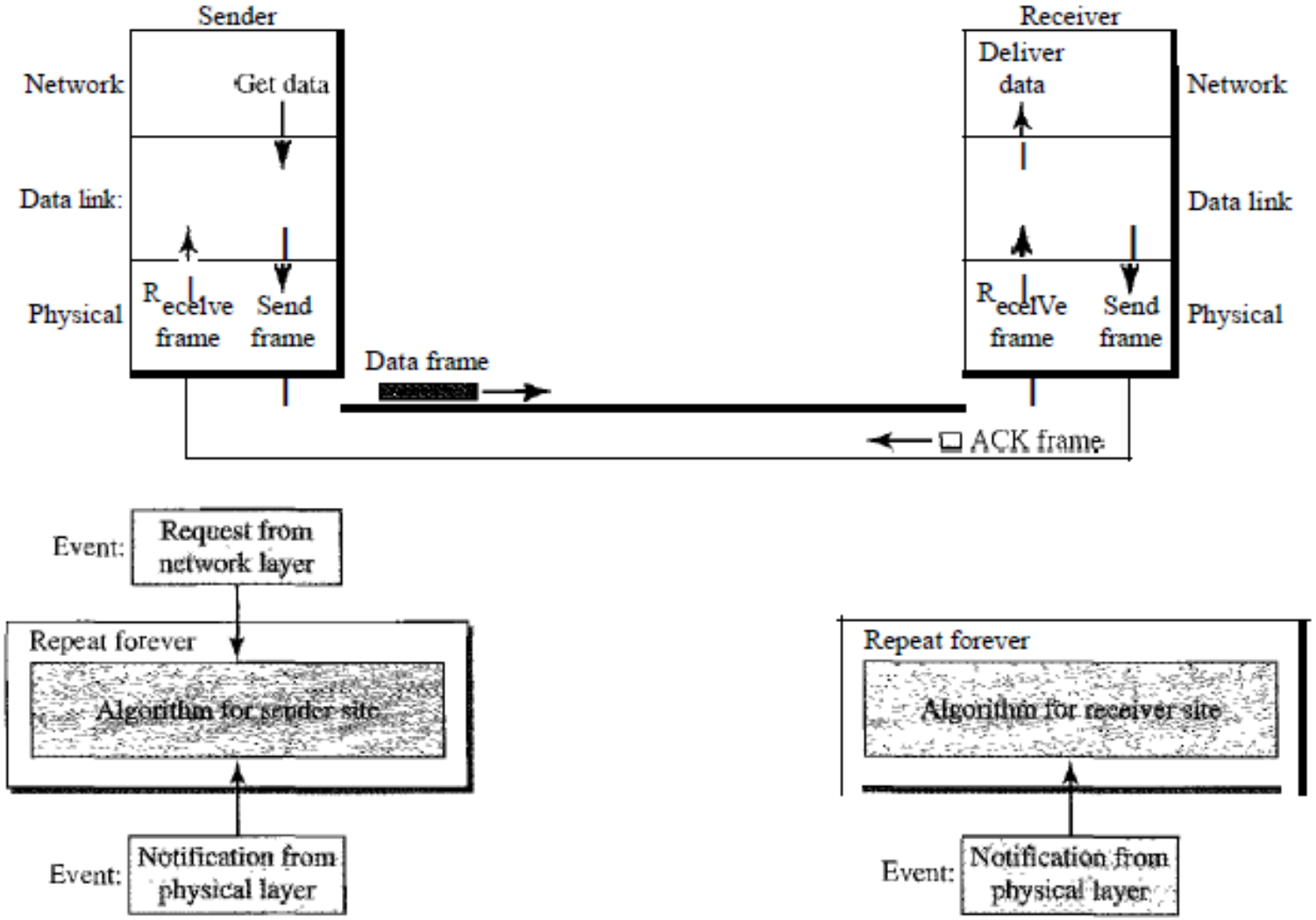


Figure 11.8 *Design of Stop-and-Wait Protocol*



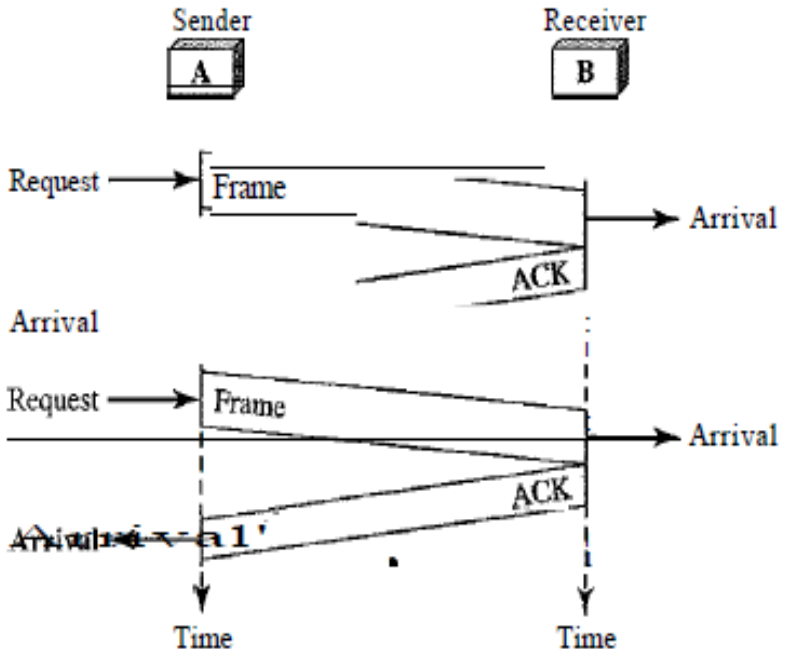
Example

Figure 11.9 shows an example of communication using this protocol. It is still very simple. The sender sends one frame and waits for feedback from the receiver.

When the ACK arrives, the sender sends the next frame.

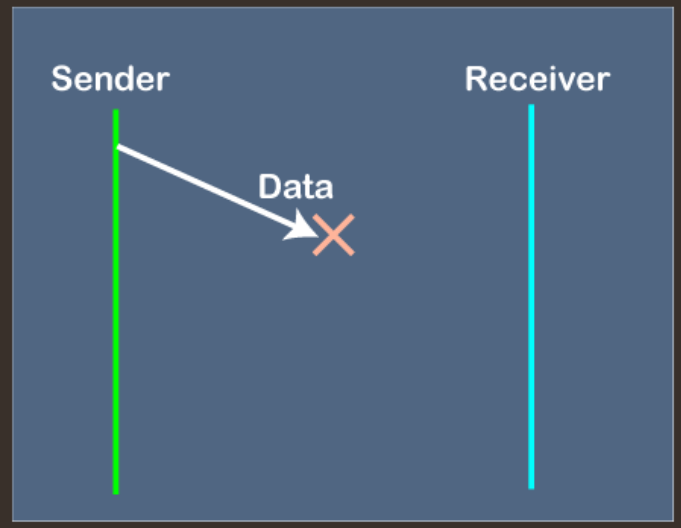
Note that sending two frames in the protocol involves the sender in four events and the receiver in two events.

Figure 11.9 *Flow diagramfor Example 11.2*



Disadvantages of Stop and Wait protocol

The following are the problems associated with a stop and wait protocol:

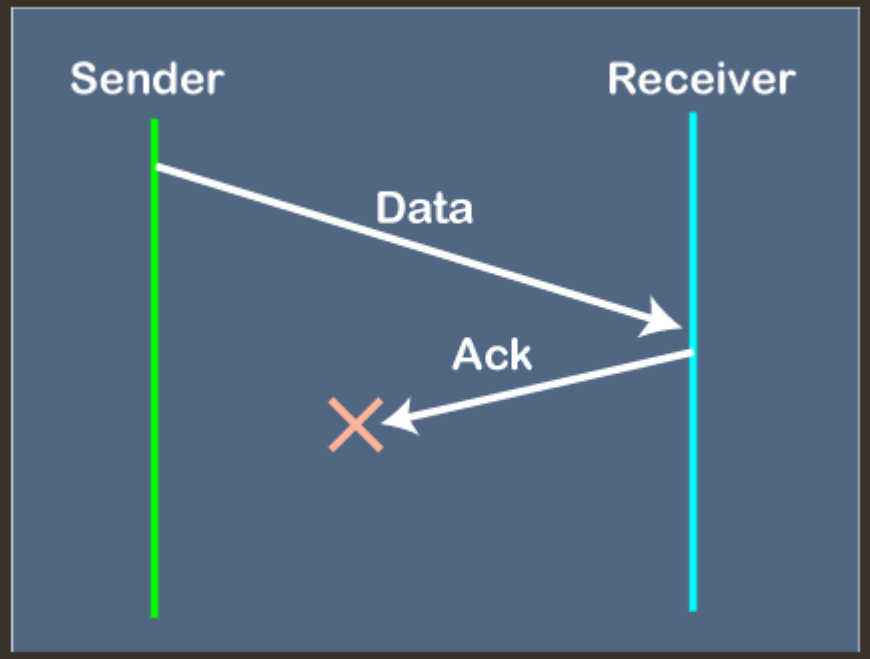


1. Problems occur due to lost data

- Suppose the sender sends the data and the data is lost. The receiver is waiting for the data for a long time. Since the data is not received by the receiver, so it does not send any acknowledgment. Since the sender does not receive any acknowledgment so it will not send the next packet. This problem occurs due to the lost data.
- **In this case, two problems occur:**
- Sender waits for an infinite amount of time for an acknowledgment.
- Receiver waits for an infinite amount of time for a data.

2. Problems occur due to lost acknowledgment

Suppose the sender sends the data and it has also been received by the receiver. On receiving the packet, the receiver sends the acknowledgment. In this case, the acknowledgment is lost in a network, so there is no chance for the sender to receive the acknowledgment. There is also no chance for the sender to send the next packet as in stop and wait protocol, the next packet cannot be sent until the acknowledgment of the previous packet is received.

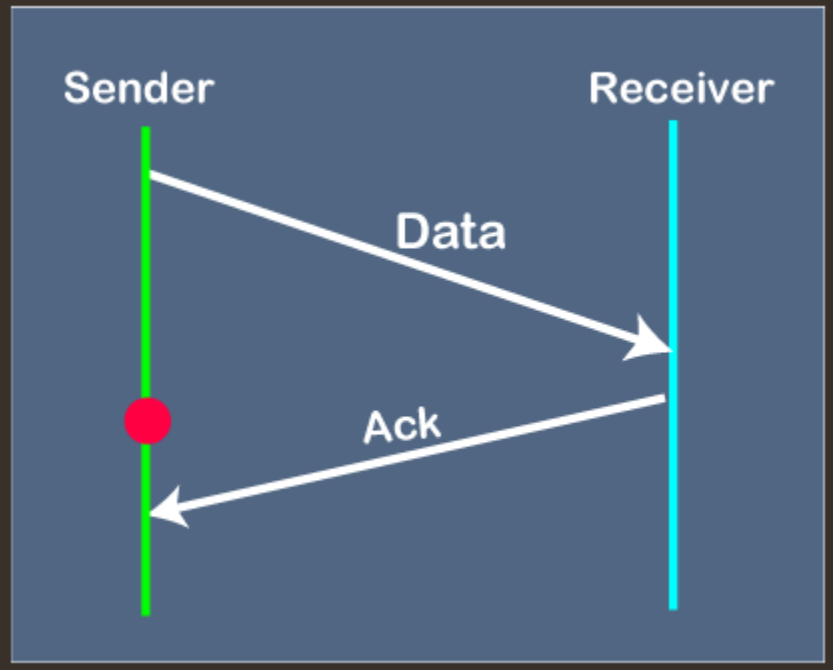


In this case, one problem occurs:

•Sender waits for an infinite amount of time for an acknowledgment.

3. Problem due to the delayed data or acknowledgment

Suppose the sender sends the data and it has also been received by the receiver. The receiver then sends the acknowledgment but the acknowledgment is received after the timeout period on the sender's side. As the acknowledgment is received late, so acknowledgment can be wrongly considered as the acknowledgment of some other data packet.



NOISY CHANNELS

Although the Stop-and-Wait Protocol gives us an idea of how to add flow control to its predecessor, noiseless channels are nonexistent. We can ignore the error (as we sometimes do), or we need to add error control to our protocols. We discuss three protocols in this section that use error control.

Stop-and-Wait Automatic Repeat Request

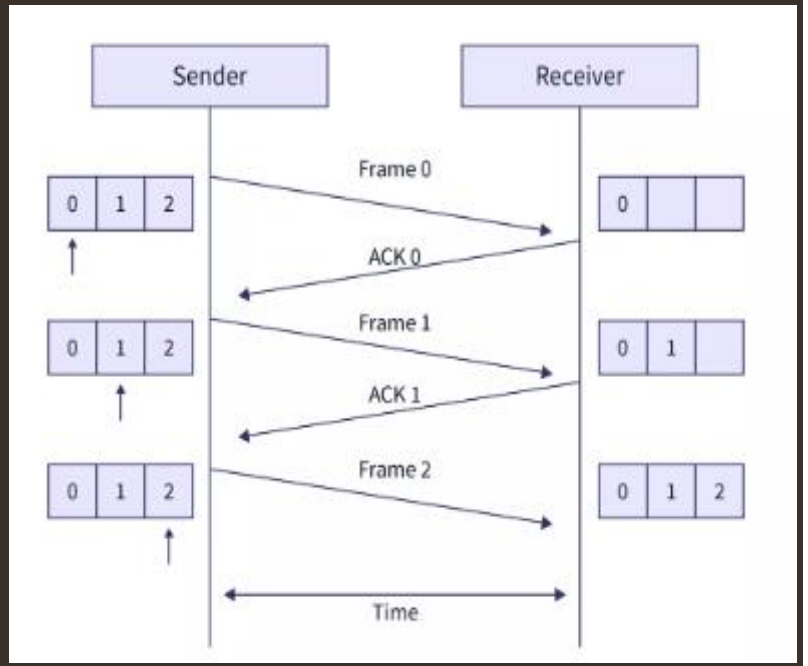
- Our first protocol, called the Stop-and-Wait Automatic Repeat Request (Stop-and Wait ARQ), adds a simple error control mechanism to the Stop-and-Wait Protocol.
- To detect and correct corrupted frames, we need to add redundancy bits to our data frame. When the frame arrives at the receiver site, it is checked and if it is corrupted, it is silently discarded. The detection of errors in this protocol is manifested by the silence of the receiver.
- Lost frames are more difficult to handle than corrupted ones. In our previous protocols, there was no way to identify a frame. The received frame could be the correct one, or a duplicate, or a frame out of order. The solution is to number the frames. When the receiver receives a data frame that is out of order, this means that frames were either lost or duplicated.

Stop-and-Wait Automatic Repeat Request

- In the stop and wait ARQ protocol, ARQ stands for Automatic Repeat Request. ARQ is an error-control strategy that ensures that a sequence of information is delivered in order and without any error or duplications despite transmission errors and losses.
- In the stop and wait ARQ, the sender also keeps a copy of the currently sending frame so that if the receiver does not receive the frame then it can retransmit it. In stop and wait ARQ, the sender sets a timer for each frame so whenever the timer is over and the sender has not received any acknowledgment for the frame, then the sender knows that the particular frame is either lost or damaged. So, the sender sends back the lost or damaged frame once the timer is out. So, we can see that the sender needs to wait for the timer to expire before retransmission
- There is no negative acknowledgment of the lost or damaged frames. So, there is no NACK (negative acknowledgment) in case of stop and wait ARQ.

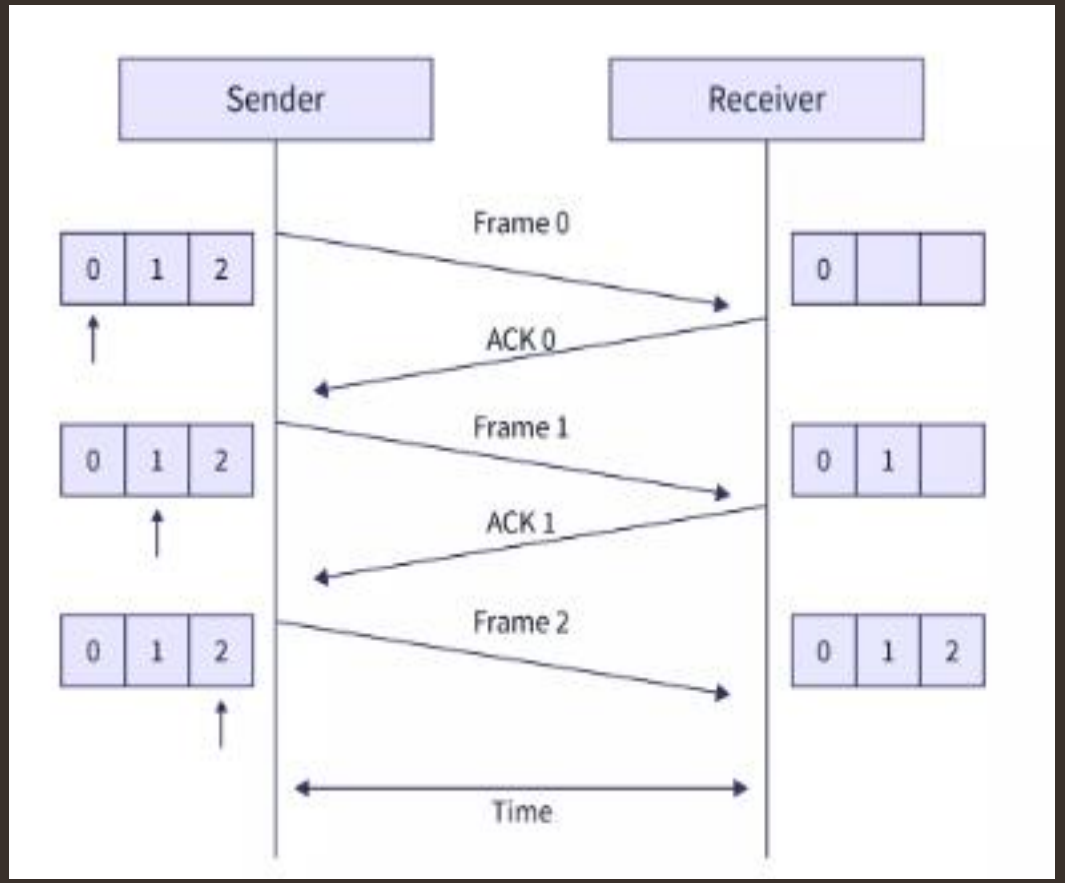
Working Principle of Stop and Wait ARQ

- In the stop and wait ARQ, both the sender and the receiver have windows of the same size. The window on the sender's side covers the sequence of data packets that are sent (or to be sent). On the other hand, the window on the receiver's side covers the sequence of data packets that are received (or to be received).
- The size of the sender's window is 1. The window size of the receiver is the same as that of the sender i.e. 1. The sender's window size is represented using **Ws** and the receiver's window size is represented using **Wr**.
- The overall working of the stop and wait ARQ is simple. Initially, the sender sends one frame as the window size is 1. The receiver on the other end receives the frame and sends the ACK for the correctly received frame. The sender waits for the ACK until the timer expires. If the sender does not receive the ACK within the timer limit, it re-transmits the frame for which the ACK has not been received.
- Now, let us take an example to visualize the working of stop and wait ARQ or how the data frame is transmitted using the stop and wait ARQ protocol. The image below shows the transmission of frames.



Working Principle of Stop and Wait ARQ

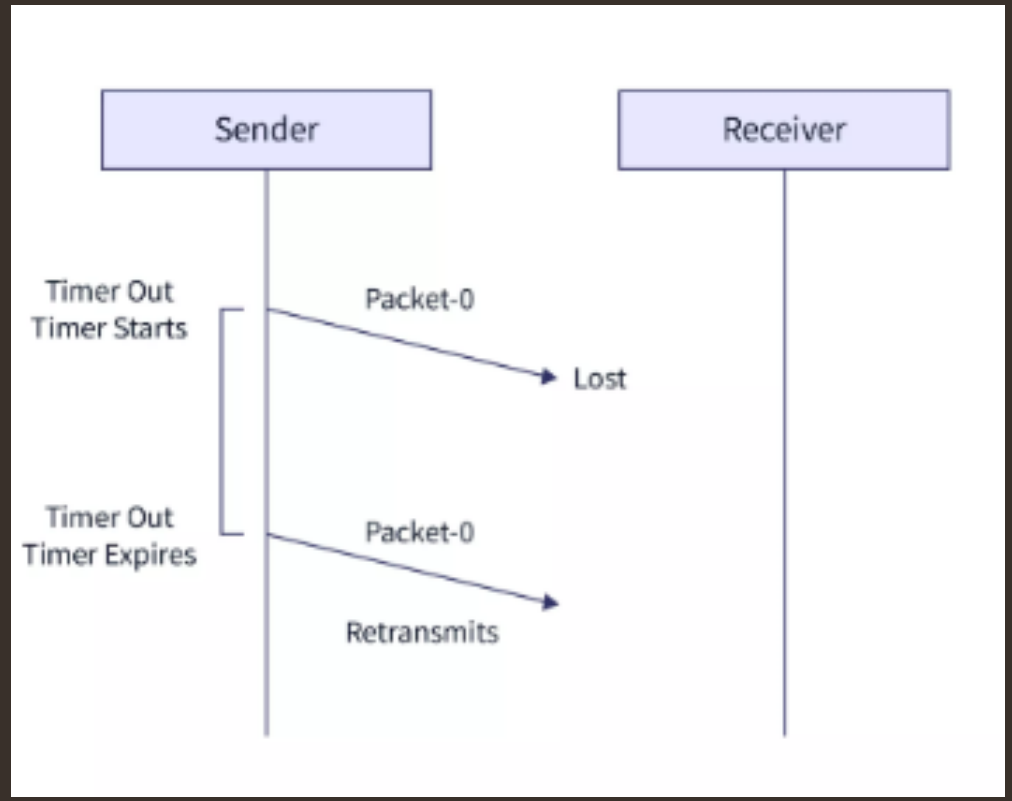
- The steps of data transmission can be:
- The sender sends frame 0.
- The sender waits for ACK from the receiver.
- The receiver receives the frame and sends back ACK 0.
- Again the sender sends the frame 1 and this process is continued till all the frames have been received by the receiver.



Let us discuss the various problems of the Stop and Wait ARQ -

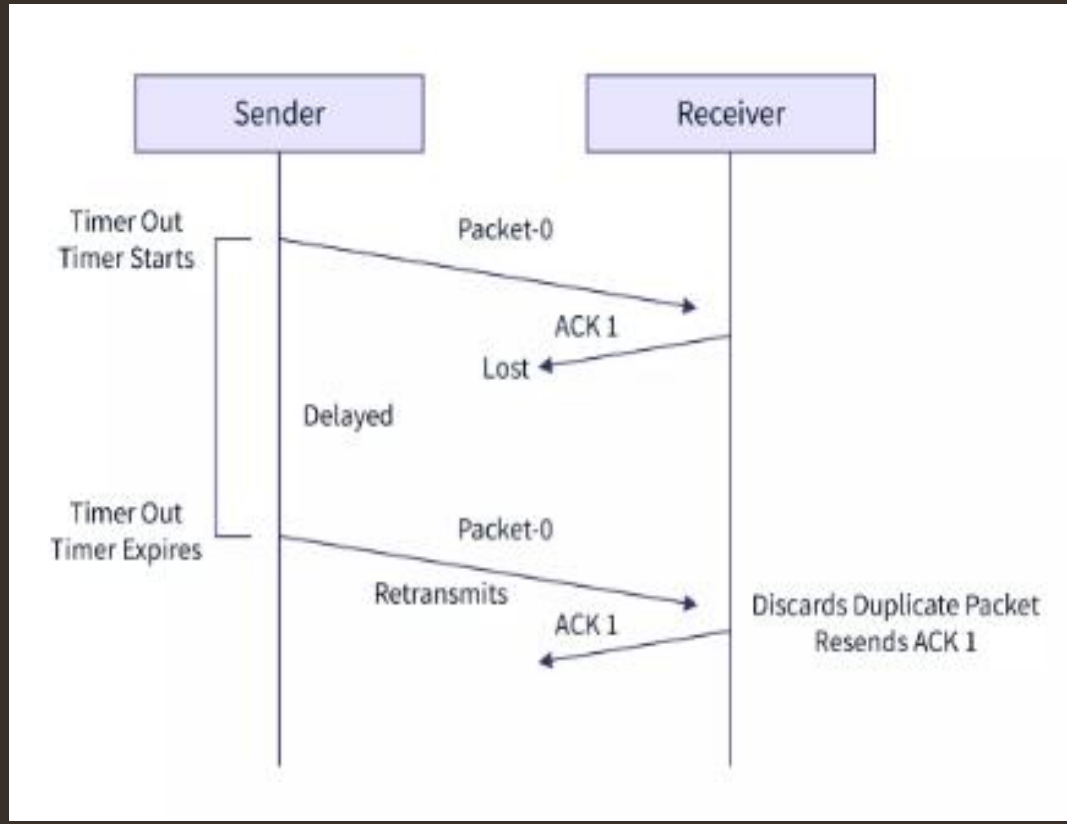
1. Problem of Lost Data Packet

When the sender sends the data packet and the receiver does not receive the data packet, it means that the data is lost in between the transmission. So, to overcome this type of problem, the sender uses a timer. When the sender sends the data packet, it starts a timer. If the timer goes off before receiving the acknowledgment from the receiver, the sender retransmits the same data packet. Refer to the image below for better visualization.



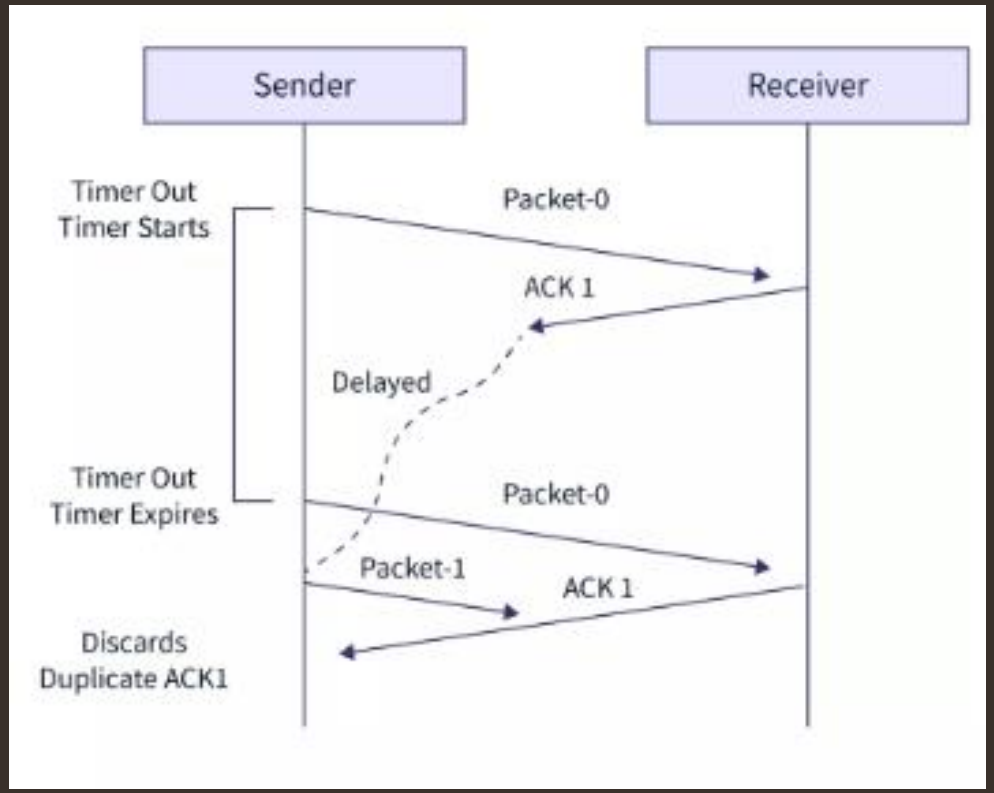
2. Problem of Lost Acknowledgement

When the sender sends the data packet and the receiver receives the data packet but the acknowledgment from the receiver is not received. It means that the acknowledgment is lost in between the transmission. So, to overcome this type of problem, the sender uses sequence numbering. When the sender sends the data packet, it attaches a certain sequence number which helps the receiver identify the data packet. If the timer goes off before receiving the acknowledgment from the receiver, the sender retransmits the same data packet. But in this case, the receiver already has the data packet, so it discards the data and sends it back an acknowledgment. This tells the sender that the certain data packet is now received correctly.



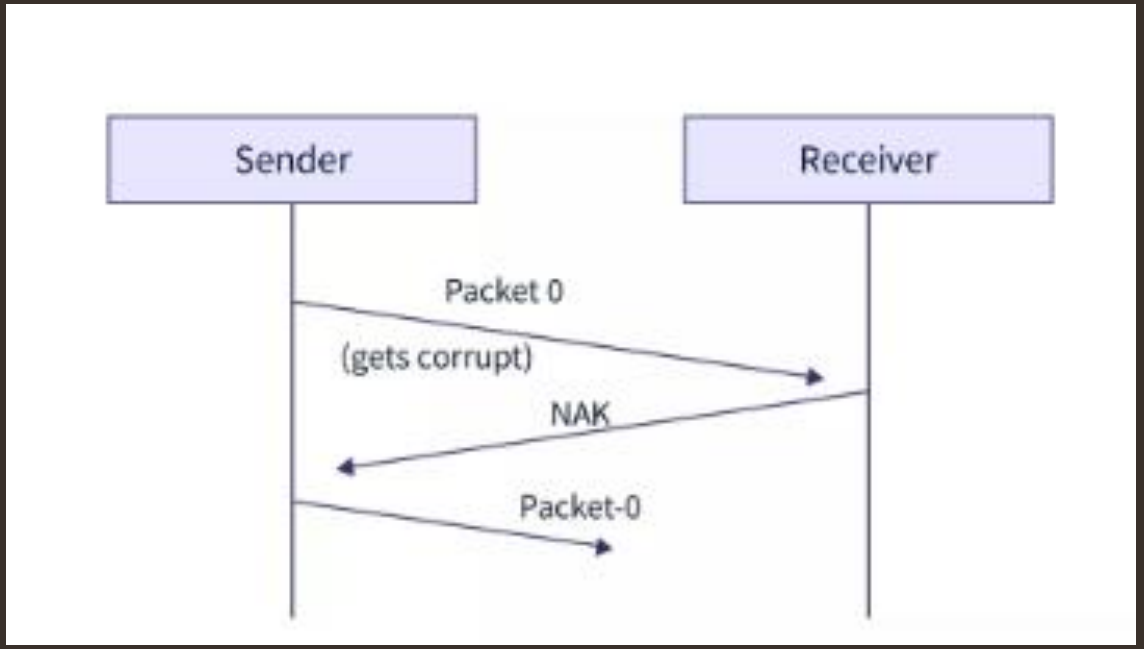
3.Problem of Delayed Acknowledgement

When the sender sends the data packet and the receiver receives the data packet but the acknowledgment from the receiver is not received. It means that the acknowledgment is lost or delayed in between the transmission. So, to overcome this type of problem, the sender uses sequence numbering. When the sender sends the data packet, it attaches a certain sequence number which helps the receiver identify the data packet. If the timer goes off before receiving the acknowledgment from the receiver, the sender retransmits the same data packet. But in this case, the receiver already has the data packet, so it discards the data and sends it back the acknowledgment again. Now, if the sender has received the previously sent acknowledgment then the newer acknowledgment is discarded by the sender. Refer to the image below for better visualization.



4. Problem of Damaged Packet

When the sender sends the data packet and the receiver receives the data packet the received data packet is corrupted. So, to overcome this type of problem, the receiver uses negative acknowledgment (NACK). So, if the sender receives a negative acknowledgment then the sender retransmits the data packet.



Advantages of Stop and Wait ARQ :

- **Simple Implementation:** Stop and Wait ARQ is a simple protocol that is easy to implement in both hardware and software. It does not require complex algorithms or hardware components, making it an inexpensive and efficient option.
- **Error Detection:** Stop and Wait ARQ detects errors in the transmitted data by using checksums or cyclic redundancy checks (CRC). If an error is detected, the receiver sends a negative acknowledgment (NAK) to the sender, indicating that the data needs to be retransmitted.
- **Reliable:** Stop and Wait ARQ ensures that the data is transmitted reliably and in order. The receiver cannot move on to the next data packet until it receives the current one. This ensures that the data is received in the correct order and eliminates the possibility of data corruption.
- **Flow Control:** Stop and Wait ARQ can be used for flow control, where the receiver can control the rate at which the sender transmits data. This is useful in situations where the receiver has limited buffer space or processing power.
- **Backward Compatibility:** Stop and Wait ARQ is compatible with many existing systems and protocols, making it a popular choice for communication over unreliable channels.

Disadvantages of Stop and Wait ARQ :

- **Low Efficiency:** Stop and Wait ARQ has low efficiency as it requires the sender to wait for an acknowledgment from the receiver before sending the next data packet. This results in a low data transmission rate, especially for large data sets.
- **High Latency:** Stop and Wait ARQ introduces additional latency in the transmission of data, as the sender must wait for an acknowledgment before sending the next packet. This can be a problem for real-time applications such as video streaming or online gaming.
- **Limited Bandwidth Utilization:** Stop and Wait ARQ does not utilize the available bandwidth efficiently, as the sender can transmit only one data packet at a time. This results in underutilization of the channel, which can be a problem in situations where the available bandwidth is limited.
- **Limited Error Recovery:** Stop and Wait ARQ has limited error recovery capabilities. If a data packet is lost or corrupted, the sender must retransmit the entire packet, which can be time-consuming and can result in further delays.
- **Vulnerable to Channel Noise:** Stop and Wait ARQ is vulnerable to channel noise, which can cause errors in the transmitted data. This can result in frequent retransmissions and can impact the overall efficiency of the protocol.

What is Sliding Window Protocol?

- Sliding windows are a method of sending multiple frames at once. It manages data packets between the two devices, ensuring delivery of data frames reliably and gradually. It’s also found in TCP (Transmission Control Protocol).
- Each frame is sent from the sequence number in this technique. The sequence numbers find use in the receiving end to locate missing data. The sequence number finds use in the sliding window technique in order to avoid duplicate data.

Working Principle of Sliding Window:

- The sender’s buffer is the sending window, and the receiver’s buffer is the receiving window in these protocols.
- Assignment of sequence numbers to frames is between the range 0 to 2n-1 if the frames’ sequence number is an n-bit field. As a result, the sending window has a size of 2n-1. As a result, we choose an n-bit sequence number to accommodate a sending window size of 2n-1.
- Modulo-n is used to number the sequence numbers. If the sending window size is set as 3, the sequence numbers will be 0, 1, 2, 0, 1, 2, 0, 1, 2 and so on.
- The receiving window’s size refers to the maximum number of frames the receiver can accept at one time. It establishes the maximum number of frames a sender can send before receiving an acknowledgement.
- Sliding Window (Sender and Receiver side):

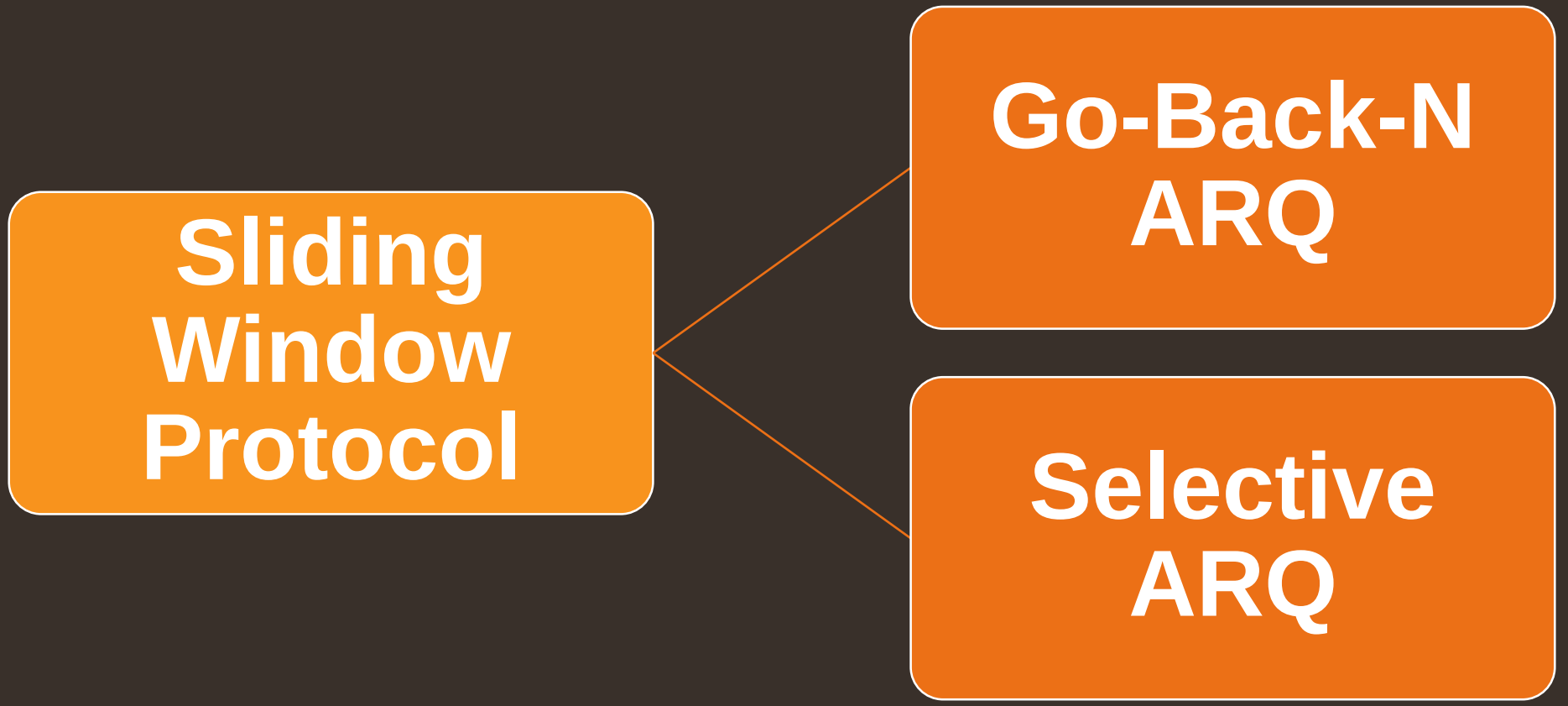
a. Sender Side:

- The sequence number of the frame occupies a field in the frame. So, the sequence number should be kept to a minimum.
- The sequence number ranges from 0 to 2k-1 if the frame header allows k bits.
- The sender maintains a list of sequence numbers that are only allowed to be sent by the sender.
- The sender window can only be 2k-1 in size.
- For example, if the frame allows 4 bits, the window's size is 2 raised to the power 4 - 1=((2^4)-1) =16-1=15.
- The sender has a buffer with the same size as the window.

b. Receiver Side:

- On the receiver side, the size of the window is always 1.
- The receiver acknowledges a frame by sending an ACK frame to the sender, along with the sequence number of the next expected frame.
- The receiver declares explicitly that it is ready to receive N subsequent frames, starting with the specified number.
- We use this scheme in order to acknowledge multiple frames.
- The receiver’s window can hold 2,3,4 frames, but the ACK frame will be held until frame 4 arrives. It will send the
- ACK along with sequence number 5 after the arrival, with which the acknowledgment of 2,3,4 will be done one at a time.
- The receiver requires a buffer size of one.

Types of Sliding Window Protocols:



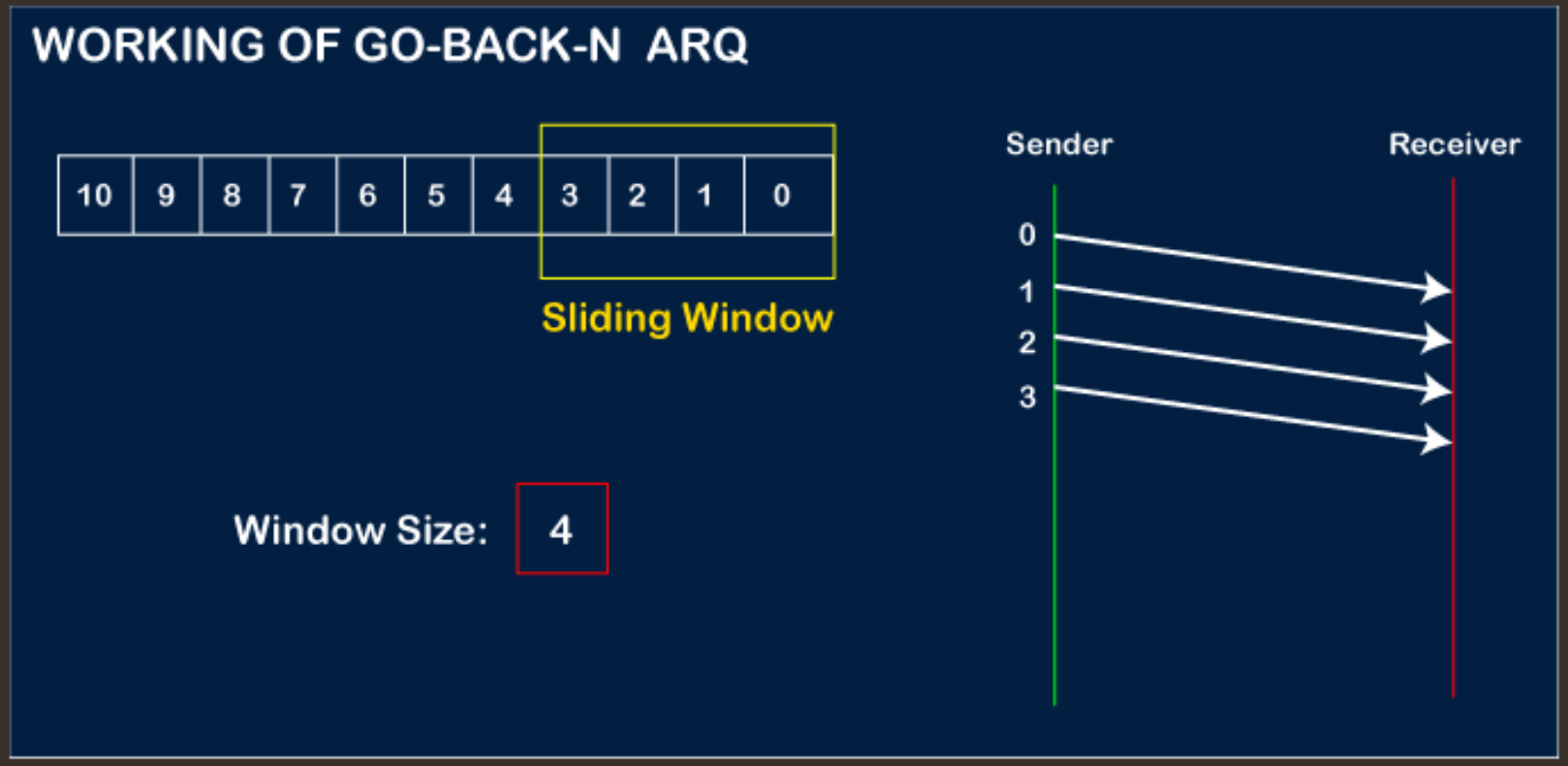
Go-Back-N ARQ

- Before understanding the working of Go-Back-N ARQ, we first look at the sliding window protocol. As we know that the sliding window protocol is different from the stop-and-wait protocol.
- In the stop-and-wait protocol, the sender can send only one frame at a time and cannot send the next frame without receiving the acknowledgment of the previously sent frame, whereas, in the case of sliding window protocol, the multiple frames can be sent at a time.
- The variations of sliding window protocol are Go-Back-N ARQ and Selective Repeat ARQ. Let's understand 'what is Go-Back-N ARQ'.

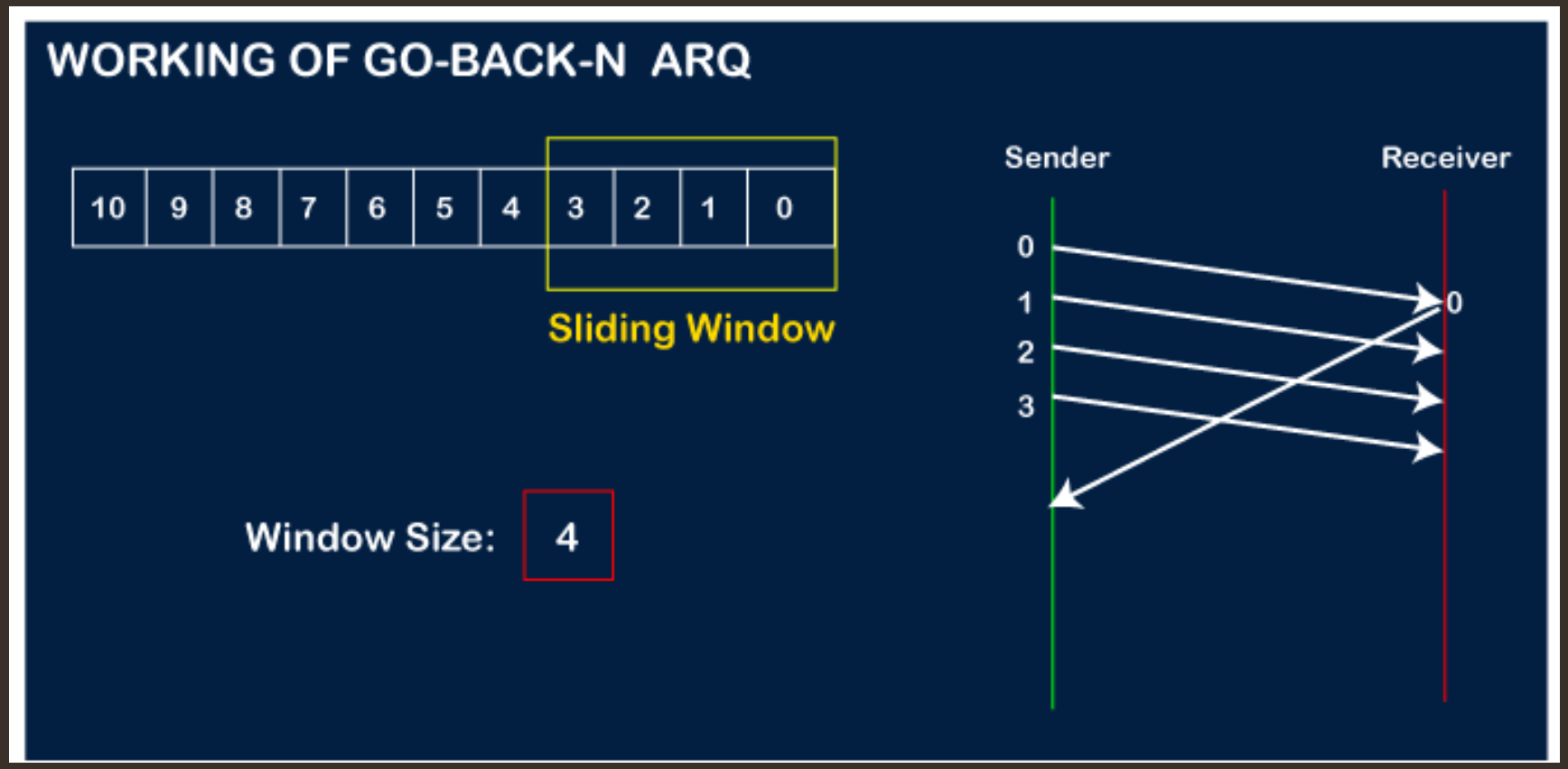
Working of Go-Back-N ARQ

- Suppose there are a sender and a receiver, and let's assume that there are 11 frames to be sent. These frames are represented as 0,1,2,3,4,5,6,7,8,9,10, and these are the sequence numbers of the frames. Mainly, the sequence number is decided by the sender's window size. But, for the better understanding, we took the running sequence numbers, i.e., 0,1,2,3,4,5,6,7,8,9,10. Let's consider the window size as 4, which means that the four frames can be sent at a time before expecting the acknowledgment of the first frame.

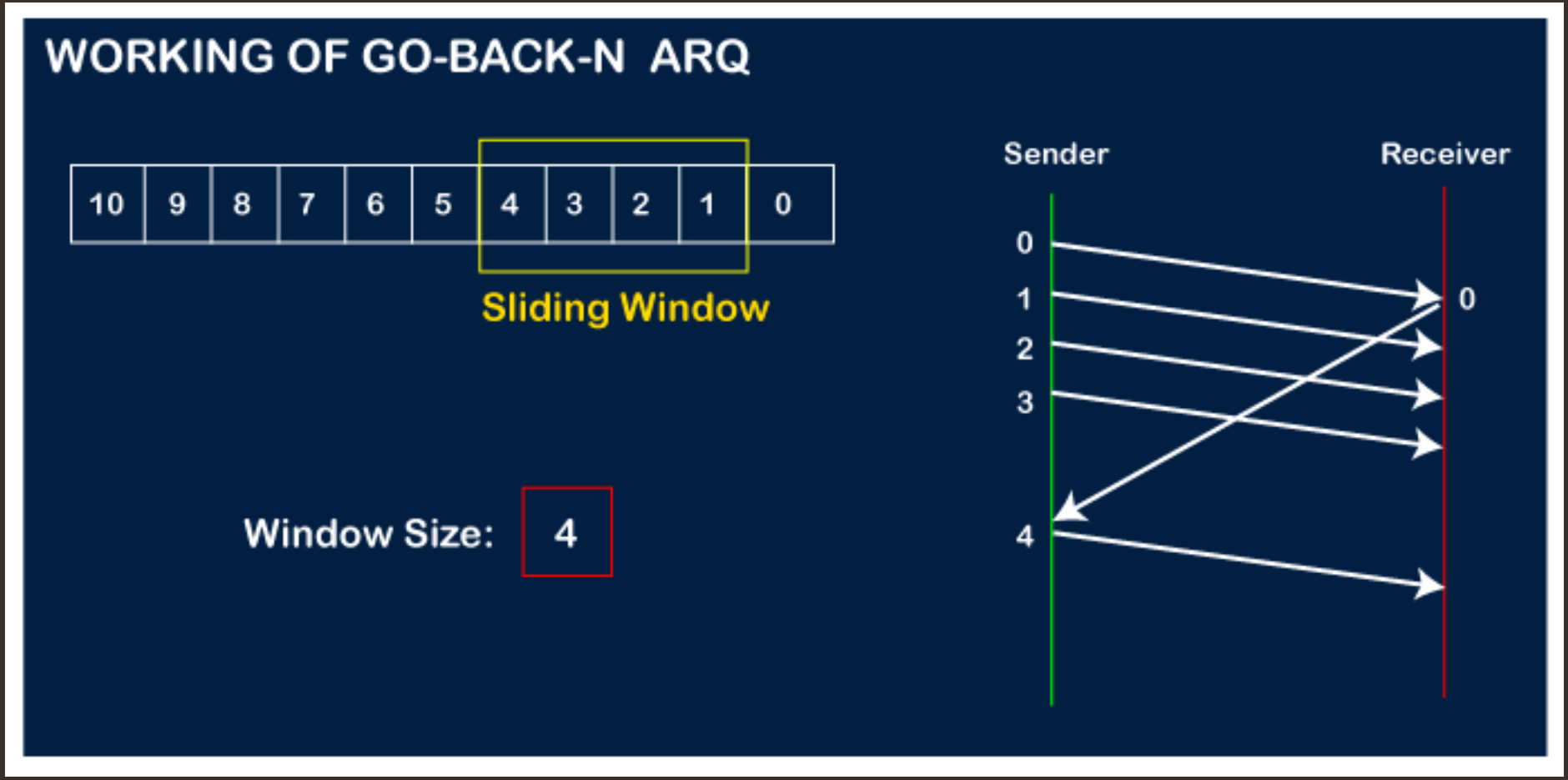
- **Step 1:** Firstly, the sender will send the first four frames to the receiver, i.e., 0,1,2,3, and now the sender is expected to receive the acknowledgment of the 0^{th} frame.



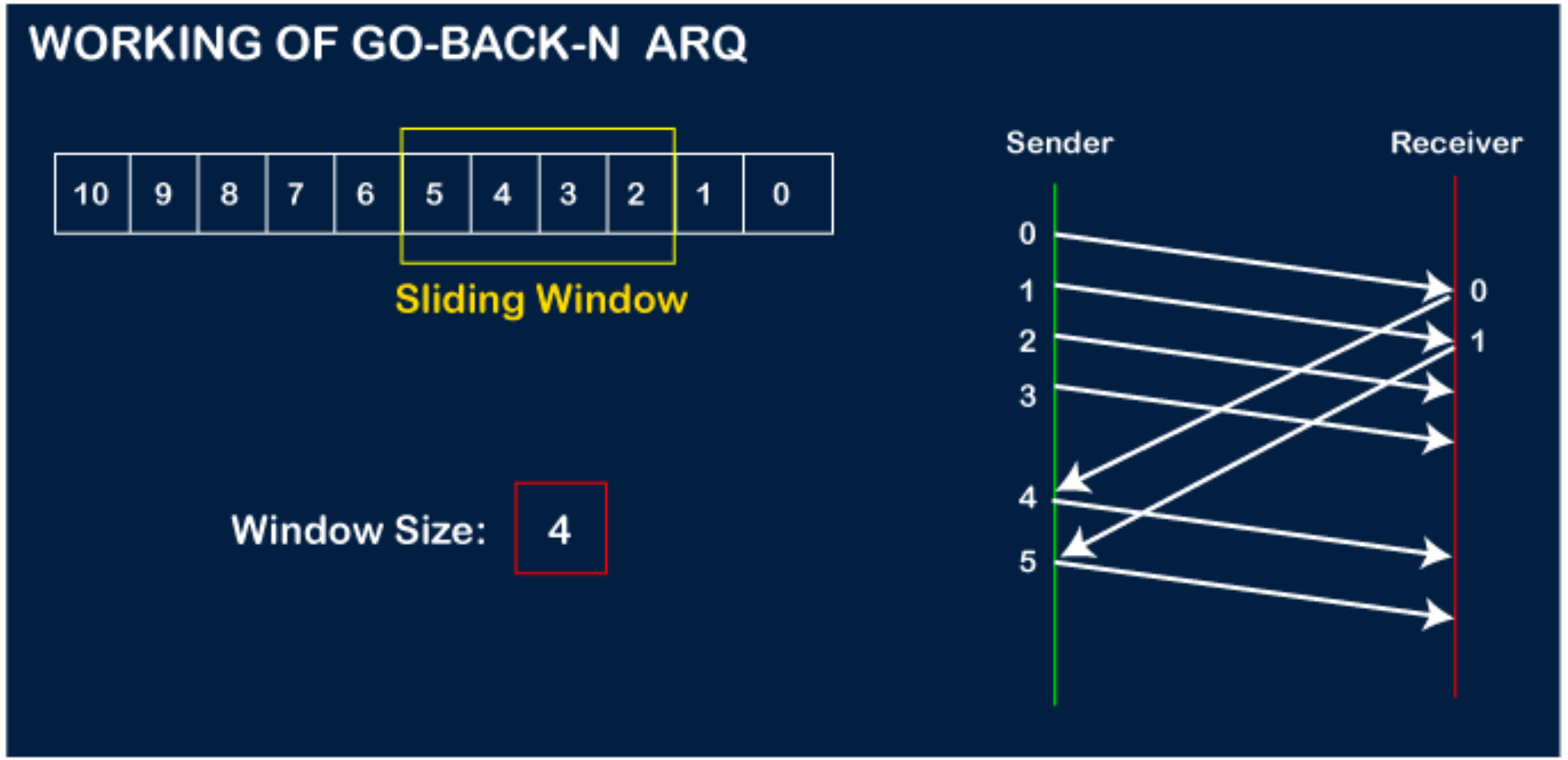
- Let's assume that the receiver has sent the acknowledgment for the 0 frame, and the receiver has successfully received it.



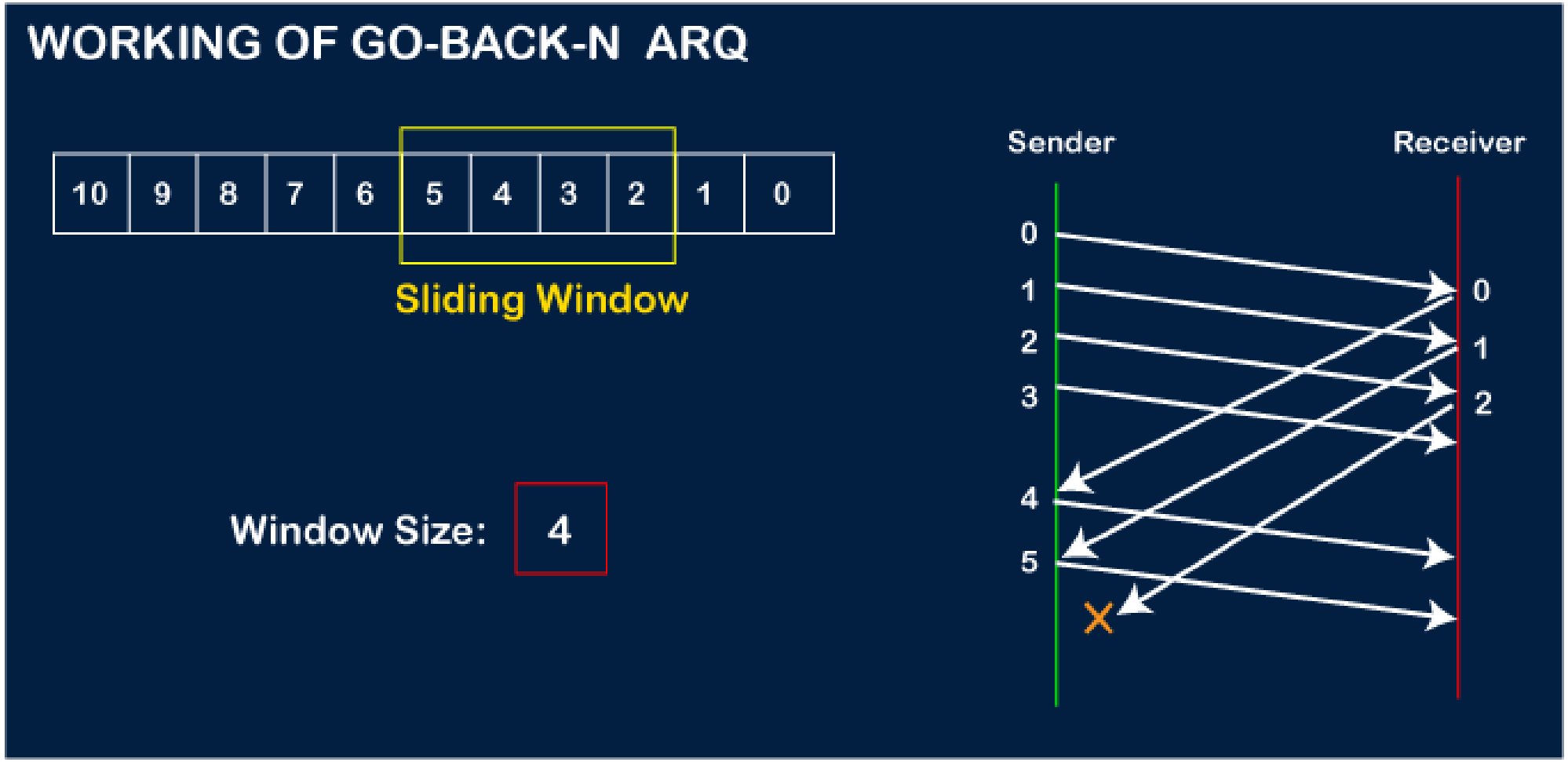
The sender will then send the next frame, i.e., 4, and the window slides containing four frames (1,2,3,4)



The receiver will then send the acknowledgment for the frame no 1. After receiving the acknowledgment, the sender will send the next frame, i.e., frame no 5, and the window will slide having four frames (2,3,4,5).



Now, let's assume that the receiver is not acknowledging the frame no 2, either the frame is lost, or the acknowledgment is lost. Instead of sending the frame no 6, the sender Go-Back to 2, which is the first frame of the current window, retransmits all the frames in the current window, i.e., 2,3,4,5.

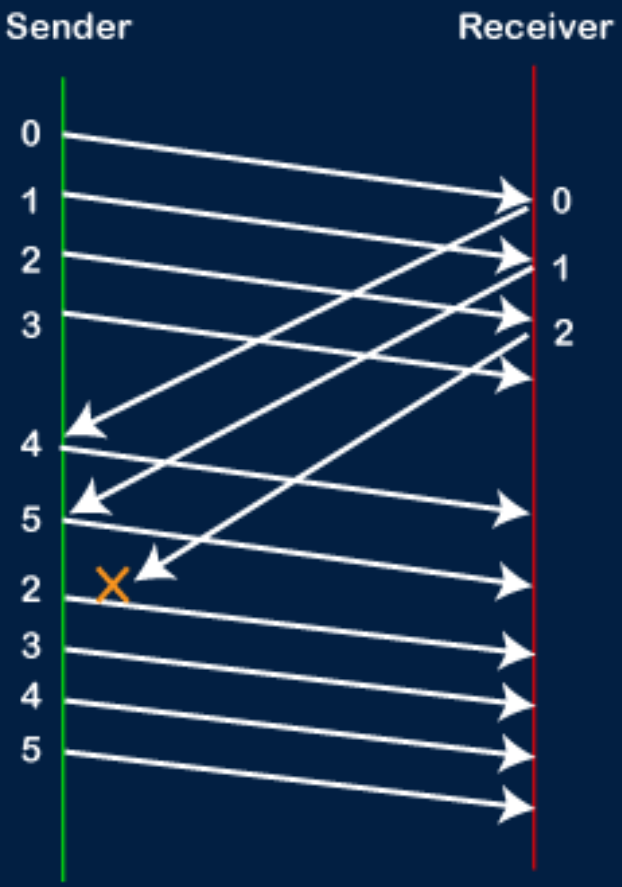


WORKING OF GO-BACK-N ARQ

Sliding Window

Go-Back to 2

Window Size: 4



Important points related to Go-Back-N ARQ:

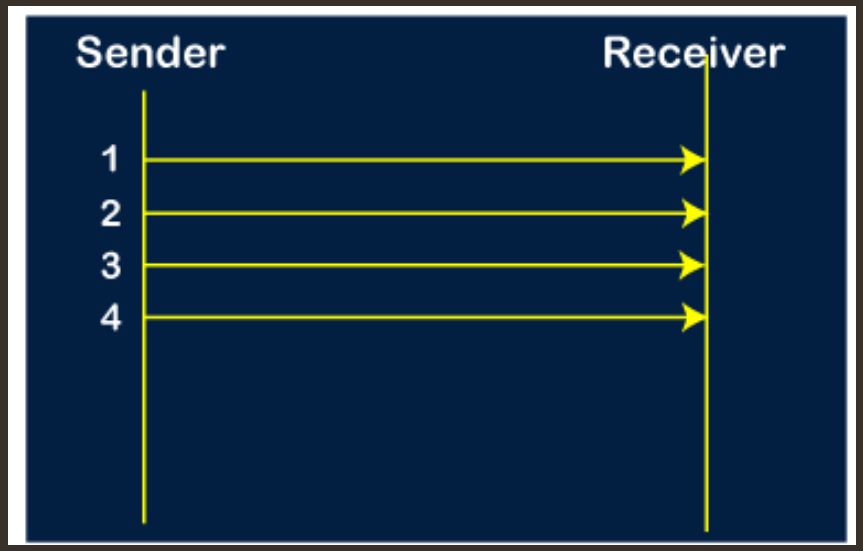
- In Go-Back-N, N determines the sender's window size, and the size of the receiver's window is always 1.
- It does not consider the corrupted frames and simply discards them.
- It does not accept the frames which are out of order and discards them.
- If the sender does not receive the acknowledgment, it leads to the retransmission of all the current window frames.

Example

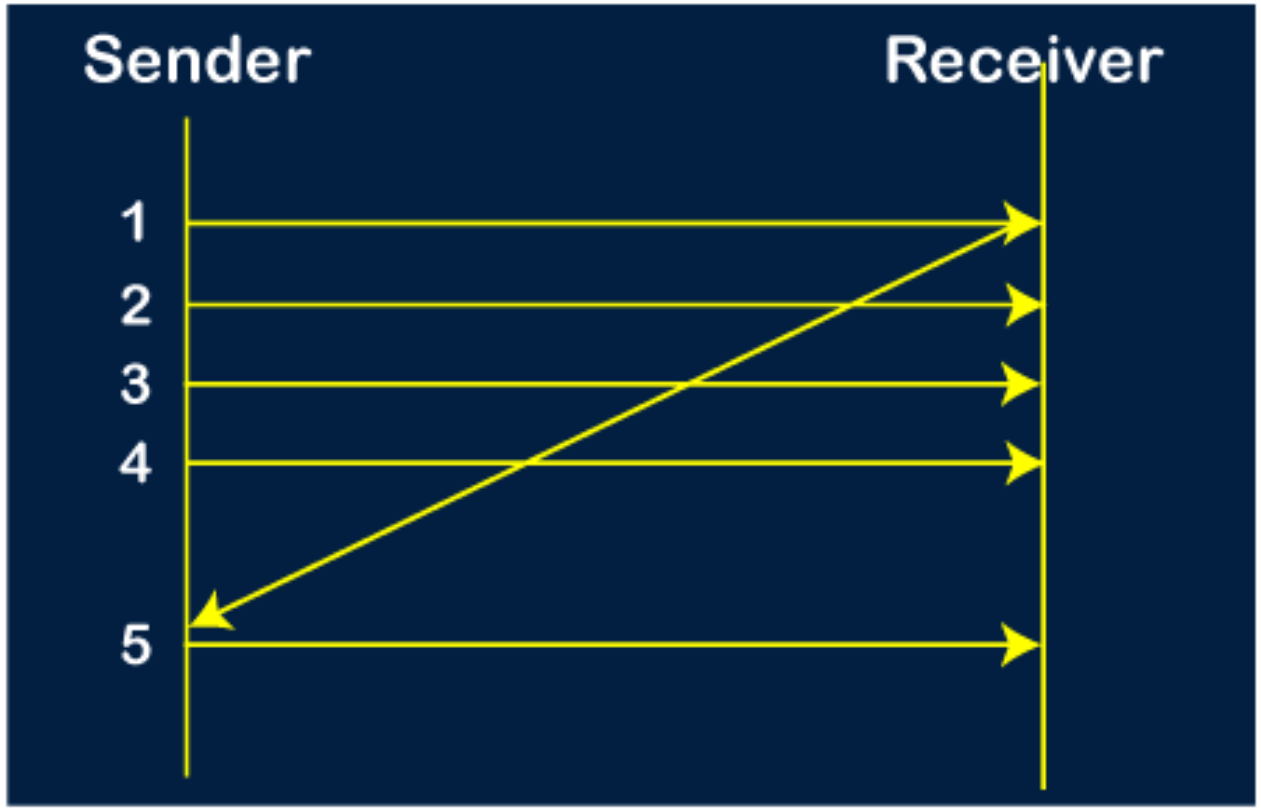
Example 1: In GB4, if every 6th packet being transmitted is lost and if we have to spend 10 packets then how many transmissions are required?

Solution: Here, GB4 means that N is equal to 4. The size of the sender's window is 4.

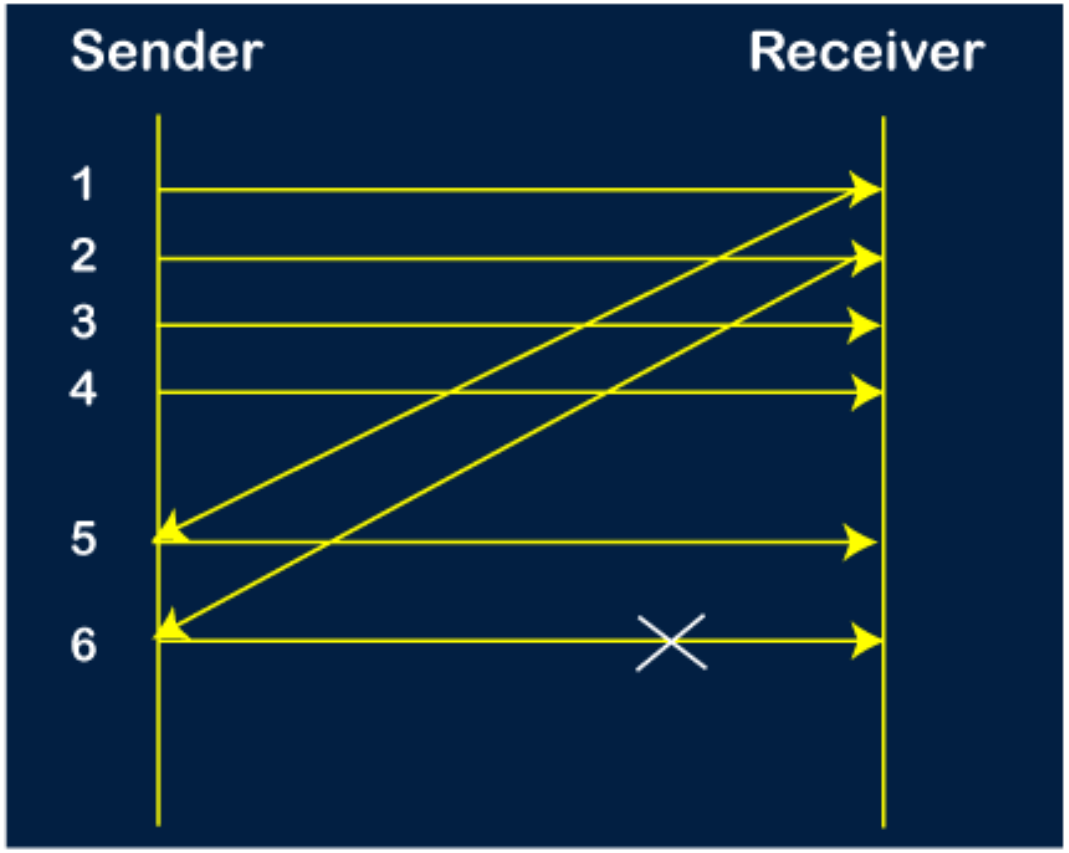
- **Step 1:** As the window size is 4, so four packets are transferred at a time, i.e., packet no 1, packet no 2, packet no 3, and packet no 4.



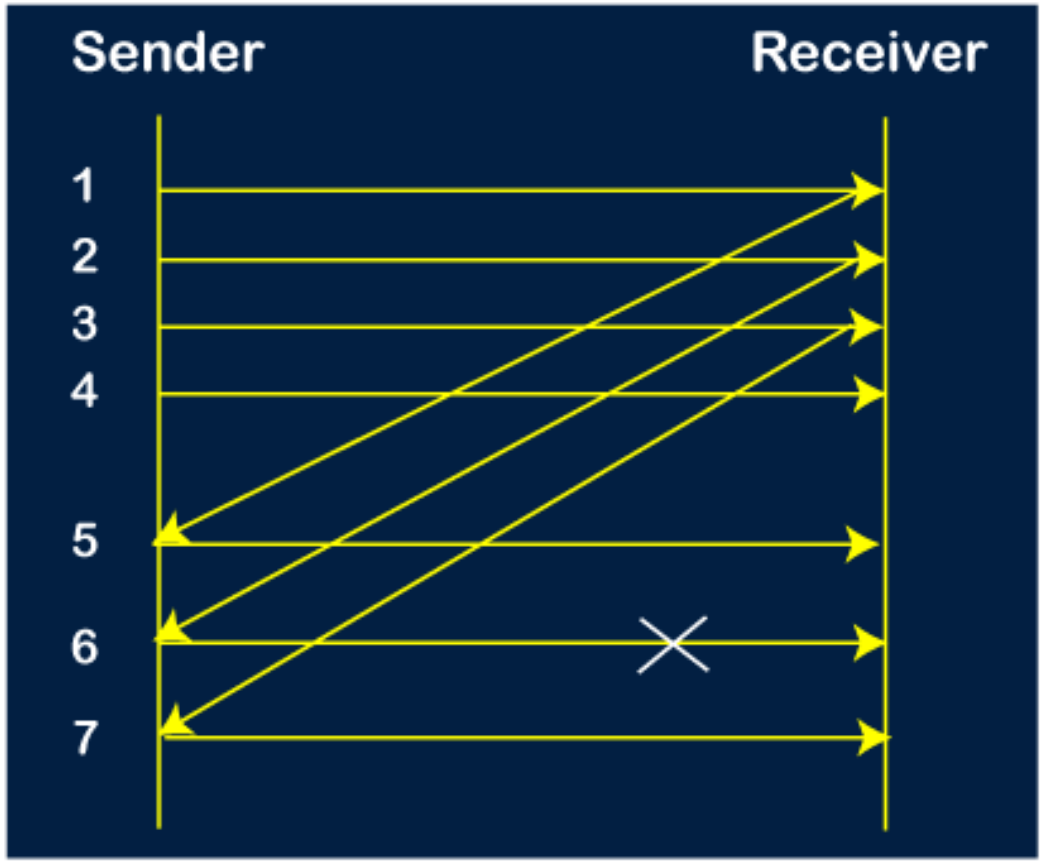
Step 2: Once the transfer of window size is completed, the sender receives the acknowledgment of the first frame, i.e., packet no1. As the acknowledgment receives, the sender sends the next packet, i.e., packet no 5. In this case, the window slides having four packets, i.e., 2,3,4,5 and excluded the packet 1 as the acknowledgment of the packet 1 has been received successfully.



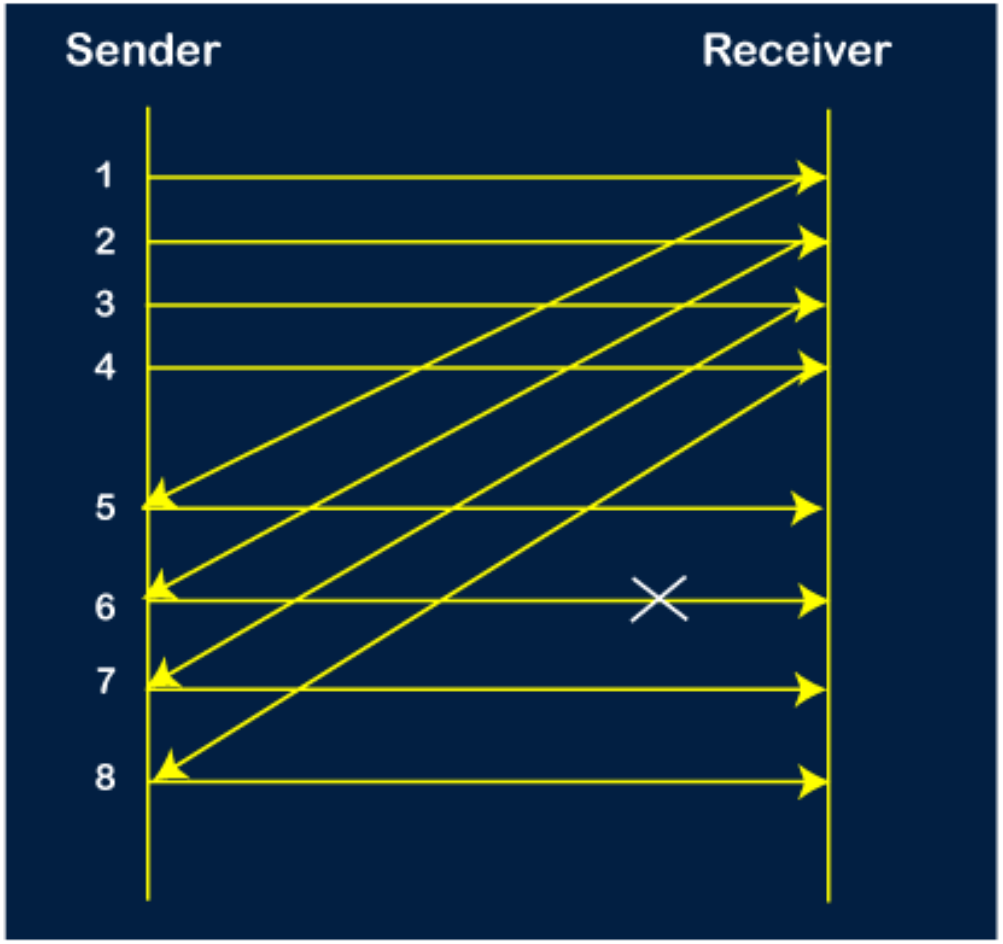
Step 3: Now, the sender receives the acknowledgment of packet 2. After receiving the acknowledgment for packet 2, the sender sends the next packet, i.e., packet no 6. As mentioned in the question that every 6^{th} is being lost, so this 6^{th} packet is lost, but the sender does not know that the 6^{th} packet has been lost.



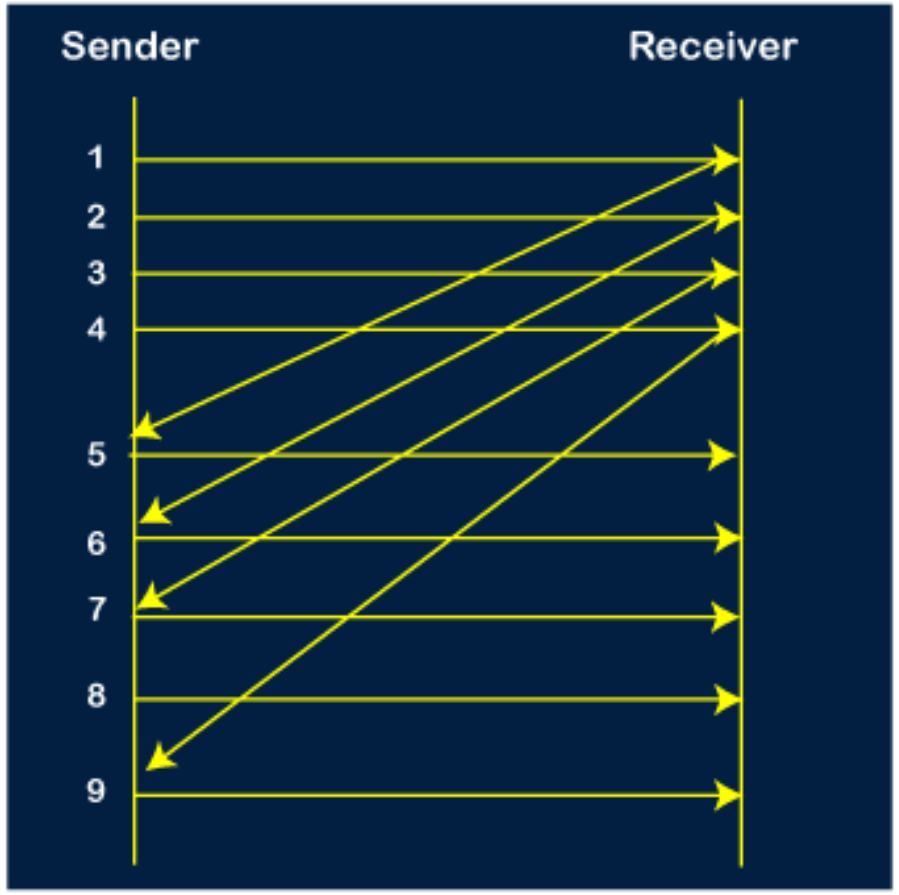
Step 4: The sender receives the acknowledgment for the packet no 3. After receiving the acknowledgment of 3rd packet, the sender sends the next packet, i.e., 7th packet. The window will slide having four packets, i.e., 4, 5, 6, 7.



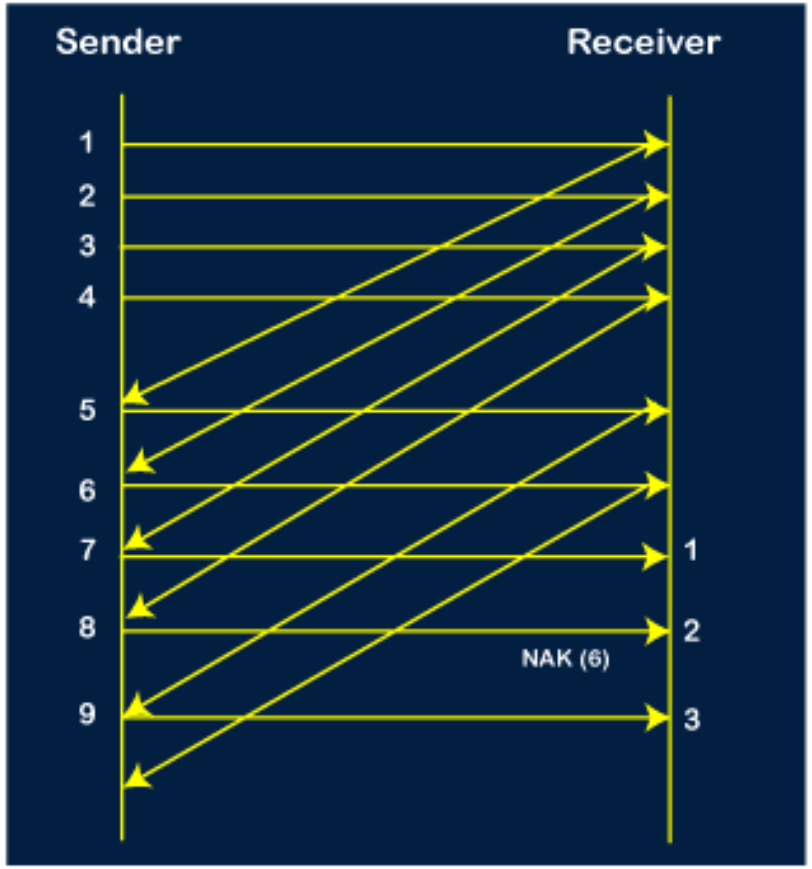
Step 5: When the packet 7 has been sent, then the sender receives the acknowledgment for the packet no 4. When the sender has received the acknowledgment, then the sender sends the next packet, i.e., the 8^{th} packet. The window will slide having four packets, i.e., 5, 6, 7, 8.



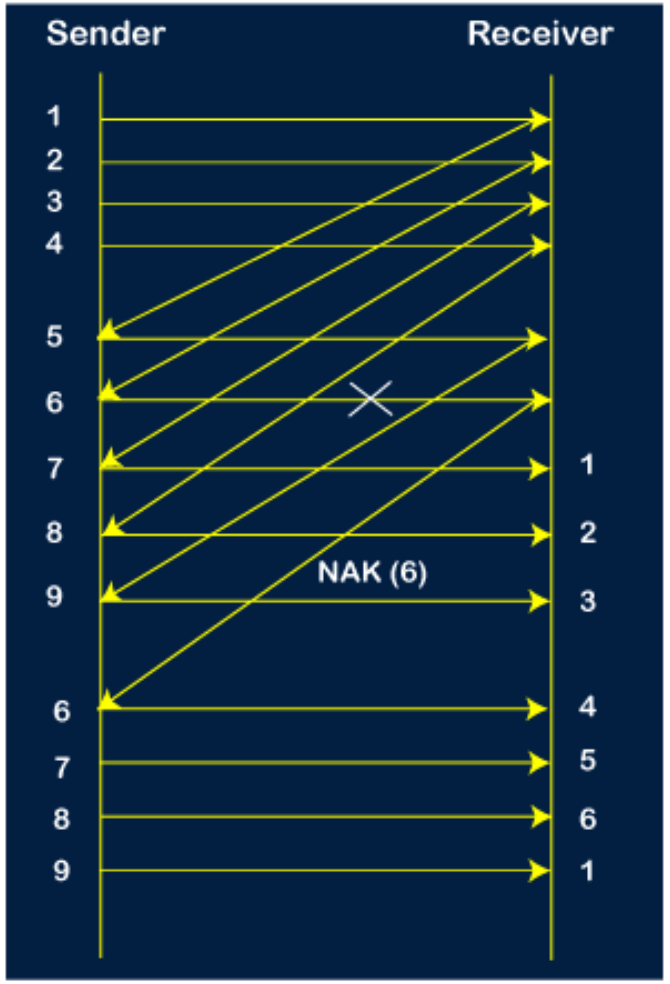
Step 6: When the packet 8 is sent, then the sender receives the acknowledgment of packet 5. On receiving the acknowledgment of packet 5, the sender sends the next packet, i.e., 9^{th} packet. The window will slide having four packets, i.e., 6, 7, 8, 9.



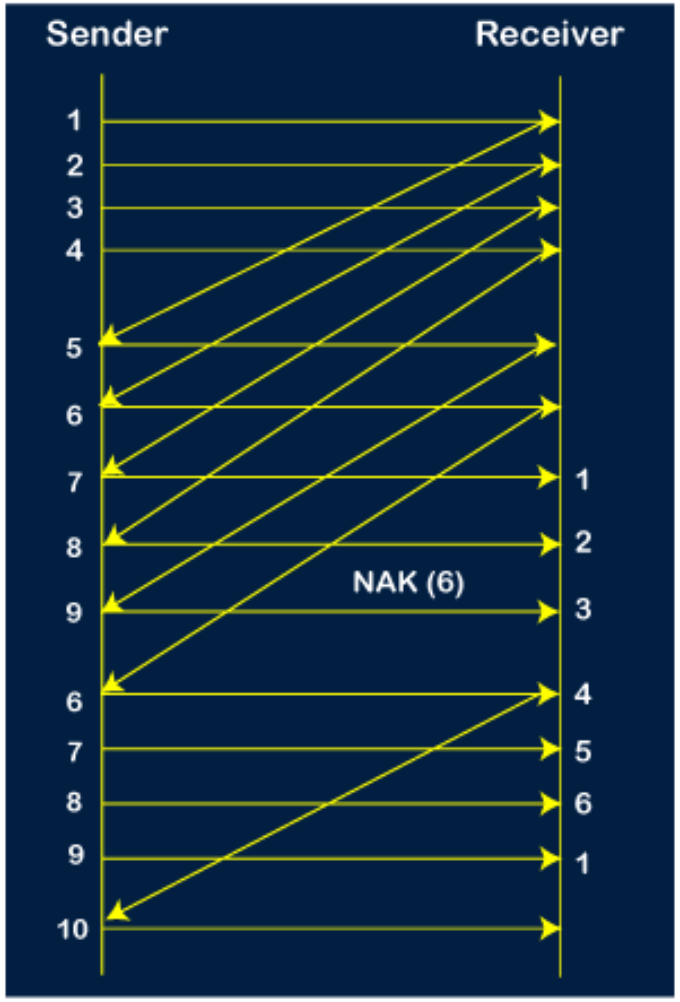
Step 7: The current window is holding four packets, i.e., 6, 7, 8, 9, where the 6^{th} packet is the first packet in the window. As we know, the 6^{th} packet has been lost, so the sender receives the negative acknowledgment NAK(6). As we know that every 6^{th} packet is being lost, so the counter will be restarted from 1. So, the counter values 1, 2, 3 are given to the 7^{th} packet, 8^{th} packet, 9^{th} packet respectively.



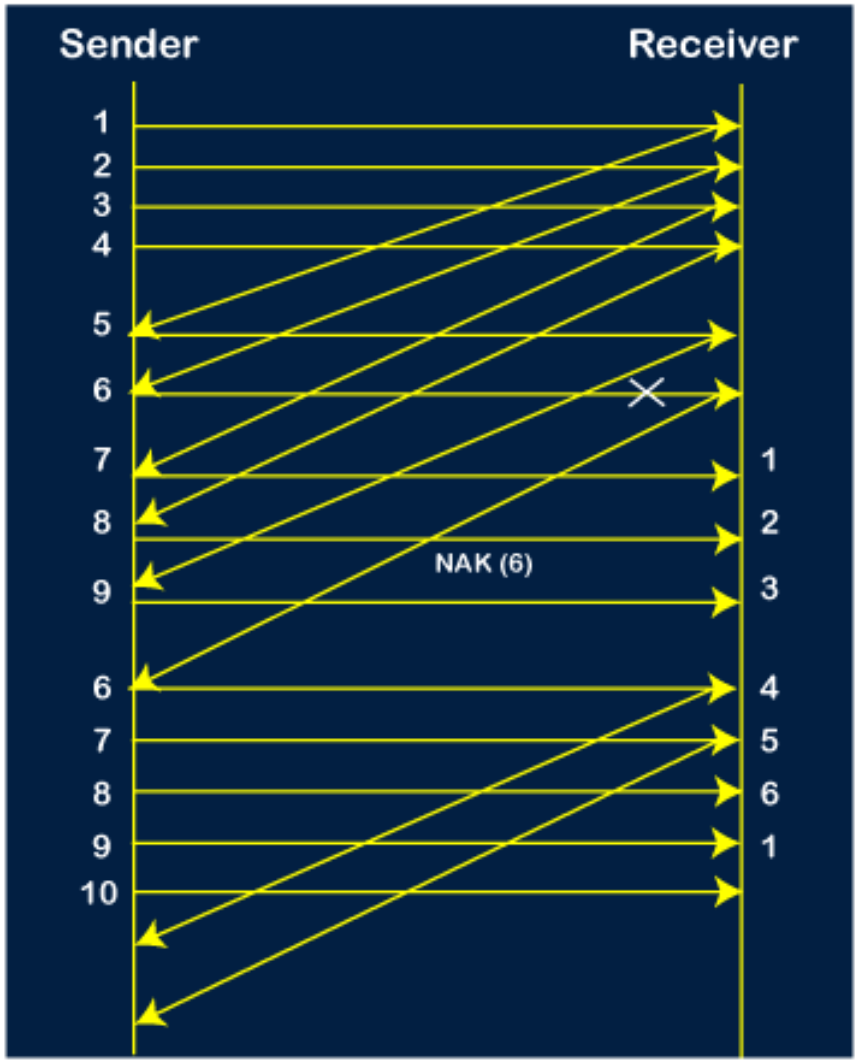
Step 8: As it is Go-BACK, so it retransmits all the packets of the current window. It will resend 6, 7, 8, 9. The counter values of 6, 7, 8, 9 are 4, 5, 6, 1, respectively. In this case, the 8^{th} packet is lost as it has a 6-counter value, so the counter variable will again be restarted from 1.



Step 9: After the retransmission, the sender receives the acknowledgment of packet 6. On receiving the acknowledgment of packet 6, the sender sends the 10^{th} packet. Now, the current window is holding four packets, i.e., 7, 8, 9, 10.

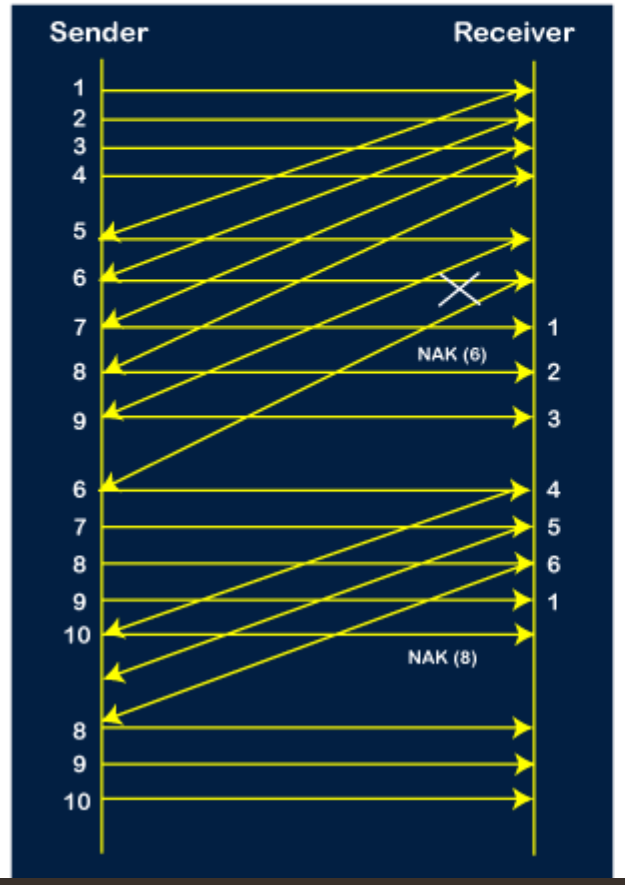


Step 10: When the 10^{th} packet is sent, the sender receives the acknowledgment of packet 7. Now the current window is holding three packets, 8, 9 and 10. The counter values of 8, 9, 10 are 6, 1, 2.



Step 11: As the 8^{th} packet has 6 counter value which means that 8^{th} packet has been lost, and the sender receives NAK (8).

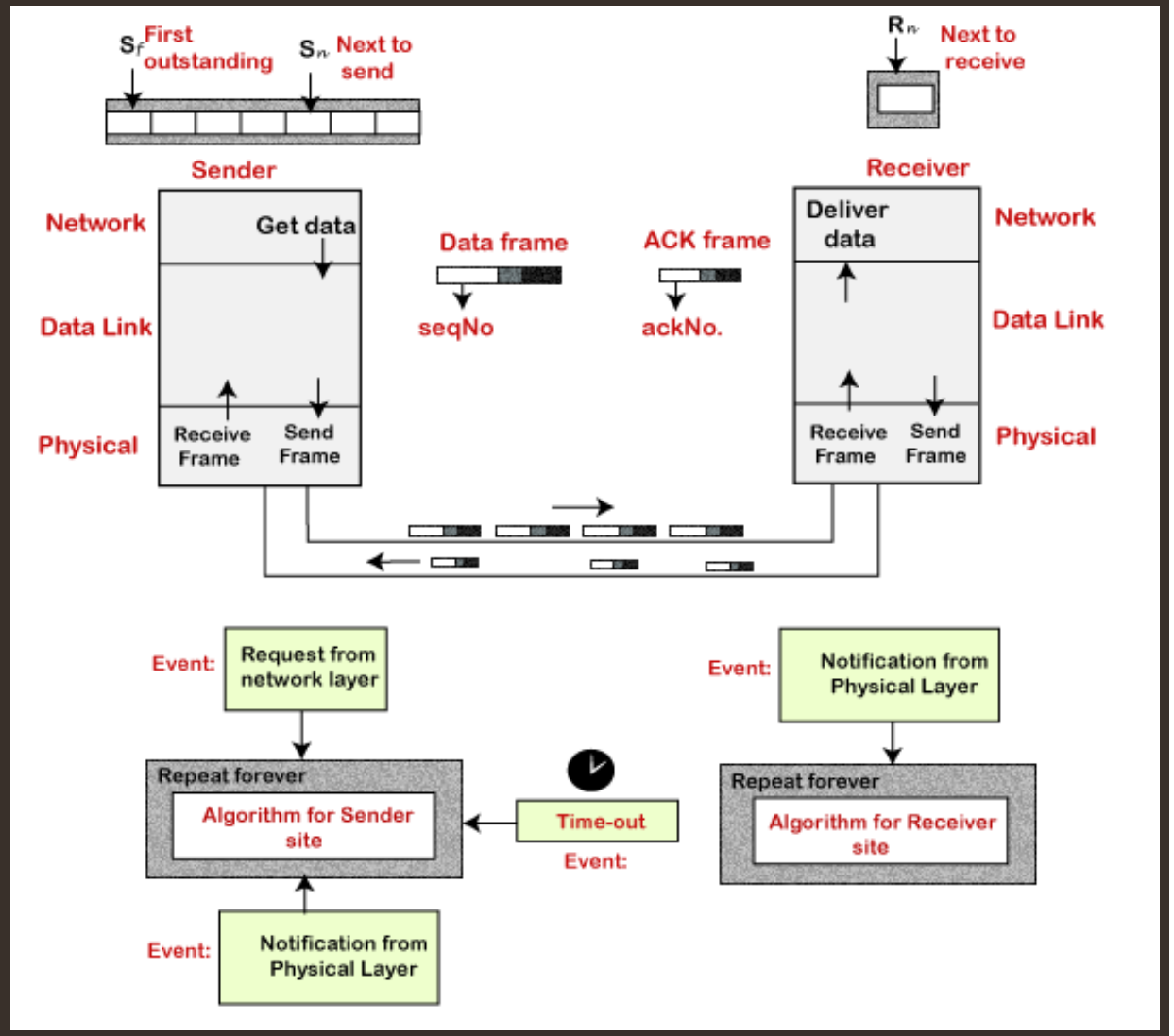
Step 12: Since the sender has received the negative acknowledgment for the 8^{th} packet, it resends all the packets of the current window, i.e., 8, 9, 10.

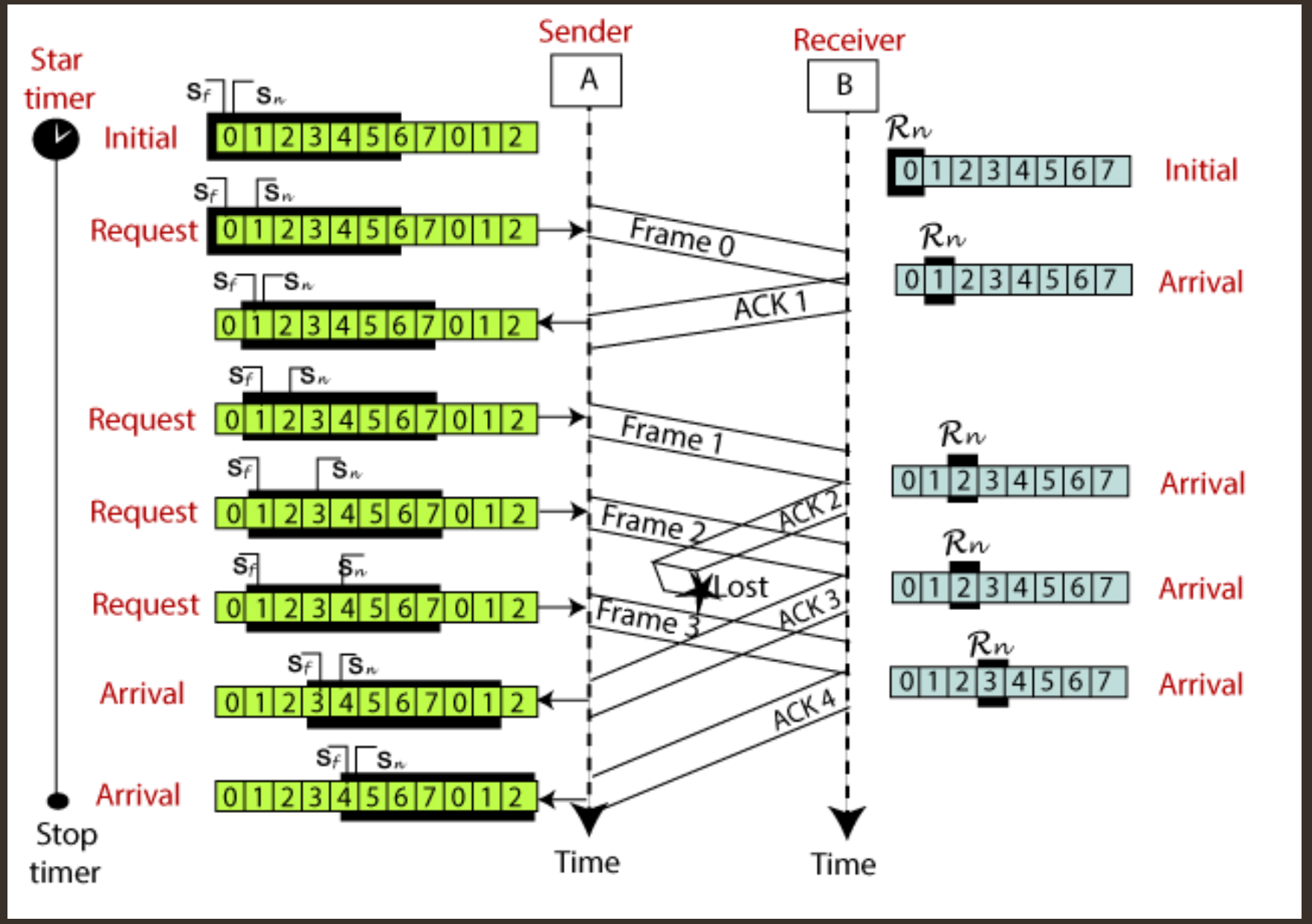


Step 13: The counter values of 8, 9, 10 are 3, 4, 5, respectively, so their acknowledgments have been received successfully.

Summary of Go-Back-N ARQ

- Go-Back-N ARQ protocol is also known as Go-Back-N Automatic Repeat Request. It is a data link layer protocol that uses a sliding window method. In this, if any frame is corrupted or lost, all subsequent frames have to be sent again.
- The size of the sender window is N in this protocol. For example, Go-Back-8, the size of the sender window, will be 8. The receiver window size is always 1.
- If the receiver receives a corrupted frame, it cancels it. The receiver does not accept a corrupted frame. When the timer expires, the sender sends the correct frame again. The design of the Go-Back-N ARQ protocol is shown below.



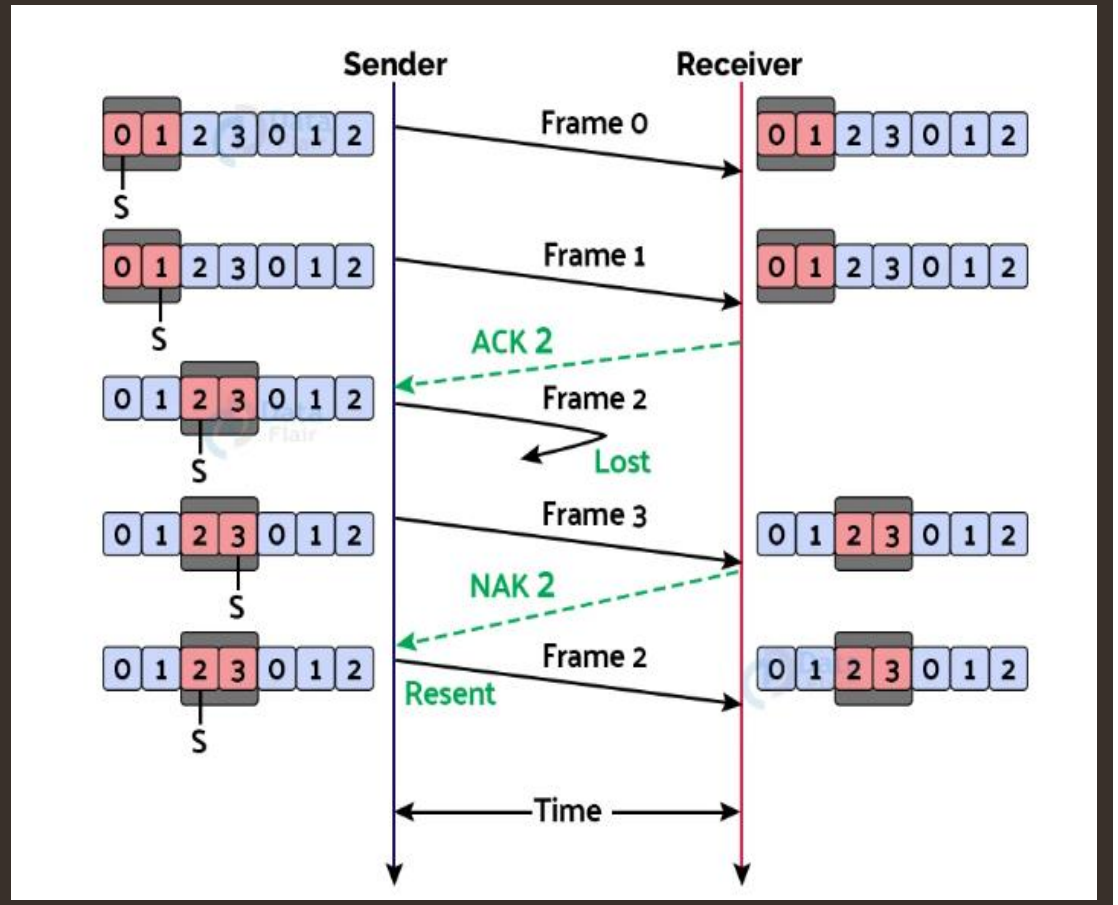

Sender
Receiver
Star timer
A
B
Initial
Request
Frame 0
ACK 1
Request
Frame 1
Request
Frame 2
ACK 2
Request
Lost
Frame 3
ACK 3
Arrival
ACK 4
Arrival
Stop timer
Time
Time
Initial
Arrival
Arrival
Arrival
Arrival

Selective Repeat ARQ

- Selective Repeat ARQ (Selective Repeat Automatic Repeat Request) is another name for Selective Repeat ARQ.
- A sliding window method is used in this data link layer protocol. If the frame has fewer errors, Go-Back-N ARQ works well.
- However, if the frame contains a lot of errors, sending the frames again will result in a lot of bandwidth loss.
- As a result, we employ the Selective Repeat ARQ method.
- The size of the sender window is always equal to the size of the receiver window in this protocol.
- The sliding window's size is always greater than 1.

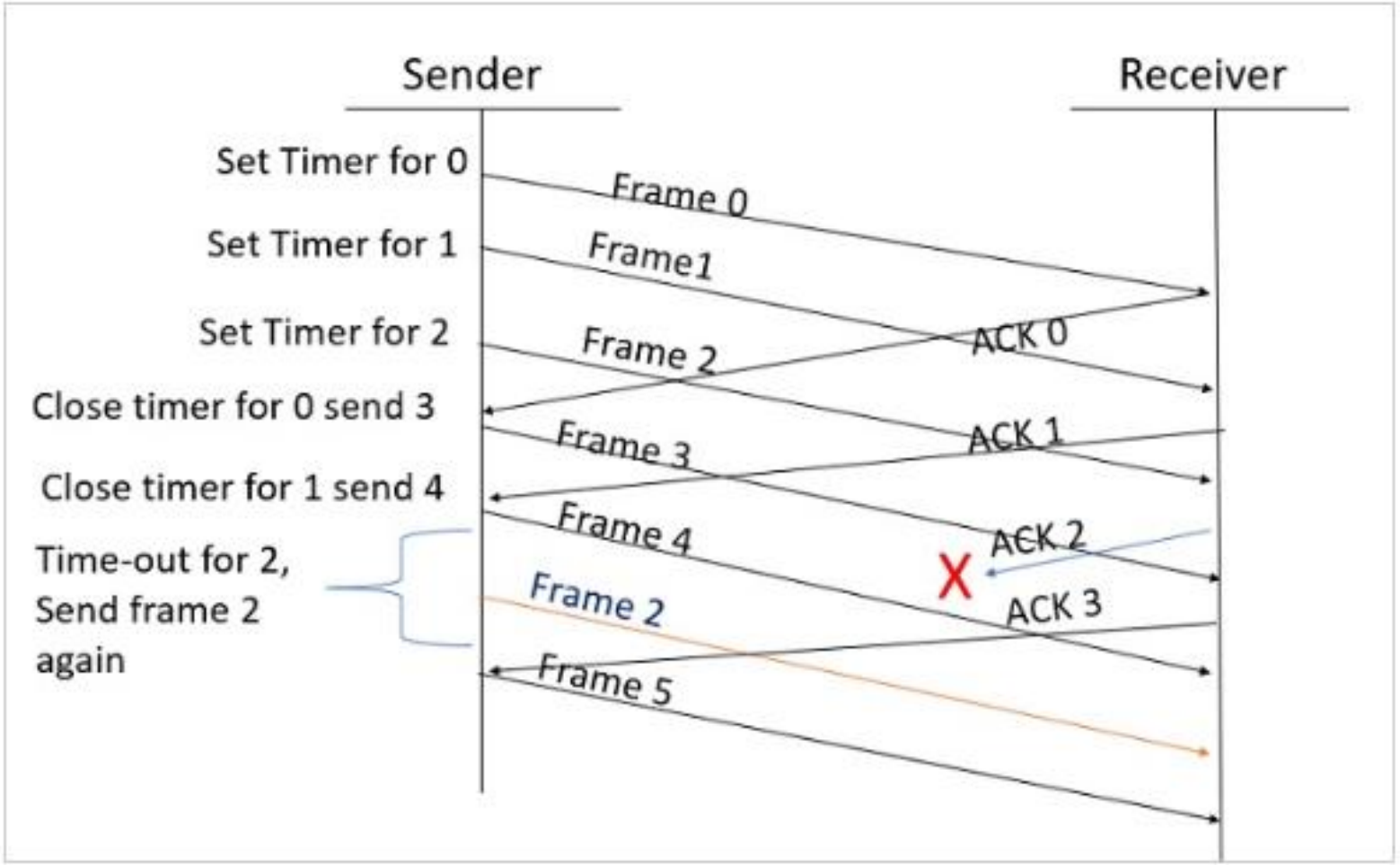
Example of Selective Repeat ARQ:

- a. First, the sender sends the contents of the first window, which are frames 0 and 1 (because the window size is 2).
- b. When the receiver receives the frames sent above, it sends an acknowledgment for frame 2 (because frame 2 is the frame it expects to receive next).
- c. The sender then sends frames 2 and 3, however, frame 2 is lost on the way. The receiver thus sends back a “NAK” signal or a non-acknowledgment to let the sender know that frame 2 has been lost, and thus the sender retransmits frame 2.



Example

Given below is an example of the Selective Repeat ARQ –



Difference between the Go-Back-N ARQ and Selective Repeat ARQ

Go-Back-N ARQ	Selective Repeat ARQ
If a frame is corrupted or lost in it,all subsequent frames have to be sent again.	In this, only the frame is sent again, which is corrupted or lost.
If it has a high error rate,it wastes a lot of bandwidth.	There is a loss of low bandwidth.
It is less complex.	It is more complex because it has to do sorting and searching as well. And it also requires more storage.
It does not require sorting.	In this, sorting is done to get the frames in the correct order.
It does not require searching.	The search operation is performed in it.
It is used more.	It is used less because it is more complex.

Advantages of Sliding Window Protocols:

1. **Efficiency:** The sliding window protocol is an efficient method of transmitting data across a network because it allows multiple packets to be transmitted at the same time. This increases the overall throughput of the network.
2. **Reliable:** The protocol ensures reliable delivery of data, by requiring the receiver to acknowledge receipt of each packet before the next packet can be transmitted. This helps to avoid data loss or corruption during transmission.
3. **Flexibility:** The sliding window protocol is a flexible technique that can be used with different types of network protocols and topologies, including wireless networks, Ethernet, and IP networks.
4. **Congestion Control:** The sliding window protocol can also help control network congestion by adjusting the size of the window based on the network conditions, thereby preventing the network from becoming overwhelmed with too much traffic.
5. Piggybacking with full-duplex lines is possible.

Disadvantages of Sliding Window Protocols:

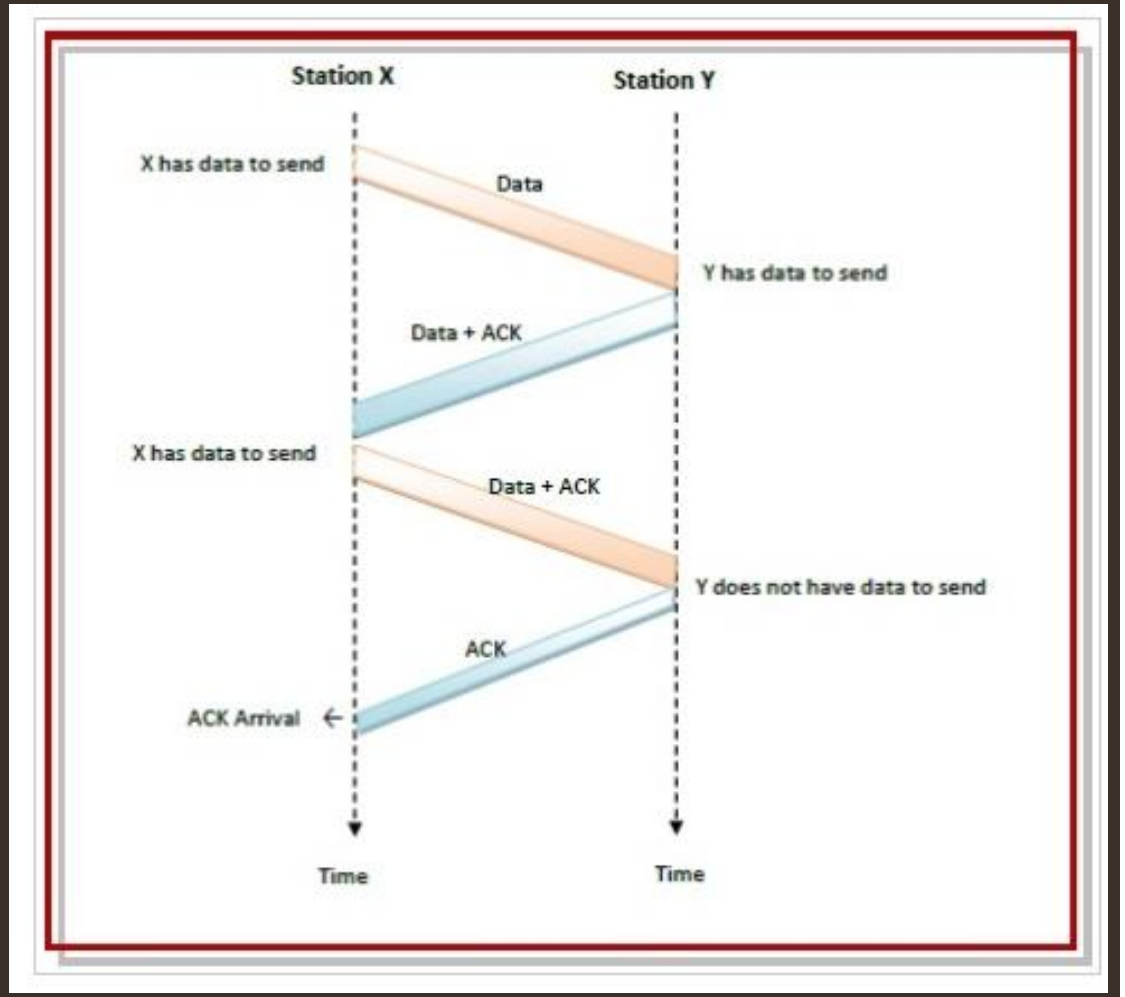
1. **Complexity:** The sliding window protocol can be complex to implement and can require a lot of memory and processing power to operate efficiently.
2. **Delay:** The protocol can introduce a delay in the transmission of data, as each packet must be acknowledged before the next packet can be transmitted. This delay can increase the overall latency of the network.
3. **Limited Bandwidth Utilization:** The sliding window protocol may not be able to utilize the full available bandwidth of the network, particularly in high-speed networks, due to the overhead of the protocol.
4. **Window Size Limitations:** The maximum size of the sliding window can be limited by the size of the receiver's buffer or the available network resources, which can affect the overall performance of the protocol.

What is Piggybacking in Networking?

- In reliable full - duplex data transmission, the technique of hooking up acknowledgments onto outgoing data frames is called piggybacking.
- Communications are mostly full – duplex in nature, i.e. data transmission occurs in both directions. A method to achieve full – duplex communication is to consider both the communication as a pair of simplex communication. Each link comprises a forward channel for sending data and a reverse channel for sending acknowledgments. However, in the above arrangement, traffic load doubles for each data unit that is transmitted. Half of all data transmission comprise of transmission of acknowledgments.
- So, a solution that provides better utilization of bandwidth is piggybacking. Here, sending of acknowledgment is delayed until the next data frame is available for transmission. The acknowledgment is then hooked onto the outgoing data frame. The data frame consists of an *ack* field. The size of the *ack* field is only a few bits, while an acknowledgment frame comprises of several bytes. Thus, a substantial gain is obtained in reducing bandwidth requirement.
- Suppose that there are two communication stations X and Y. The data frames transmitted have an acknowledgment field, *ack* field that is of a few bits length. Additionally, there are frames for sending acknowledgments, ACK frames. The purpose is to minimize the ACK frames.

Working Principle

- The three principles governing piggybacking when the station X wants to communicate with station Y are −
- If station X has both data and acknowledgment to send, it sends a data frame with the *ack* field containing the sequence number of the frame to be acknowledged.
- If station X has only an acknowledgment to send, it waits for a finite period of time to see whether a data frame is available to be sent. If a data frame becomes available, then it piggybacks the acknowledgment with it. Otherwise, it sends an ACK frame.
- If station X has only a data frame to send, it adds the last acknowledgment with it. The station Y discards all duplicate acknowledgments. Alternatively, station X may send the data frame with the *ack* field containing a bit combination denoting no acknowledgment.



Advantages of Piggybacking

1. The major advantage of piggybacking is the better use of available channel bandwidth. This happens because an acknowledgment frame needs not to be sent separately.
2. Usage cost reduction.
3. Improves latency of data transfer.
4. To avoid the delay and rebroadcast of frame transmission, piggybacking uses a very short-duration timer.

Disadvantages of Piggybacking

- The disadvantage of piggybacking is the additional complexity.
- If the data link layer waits long before transmitting the acknowledgment (blocks the ACK for some time), the frame will rebroadcast.