# What is Stop and Wait protocol?

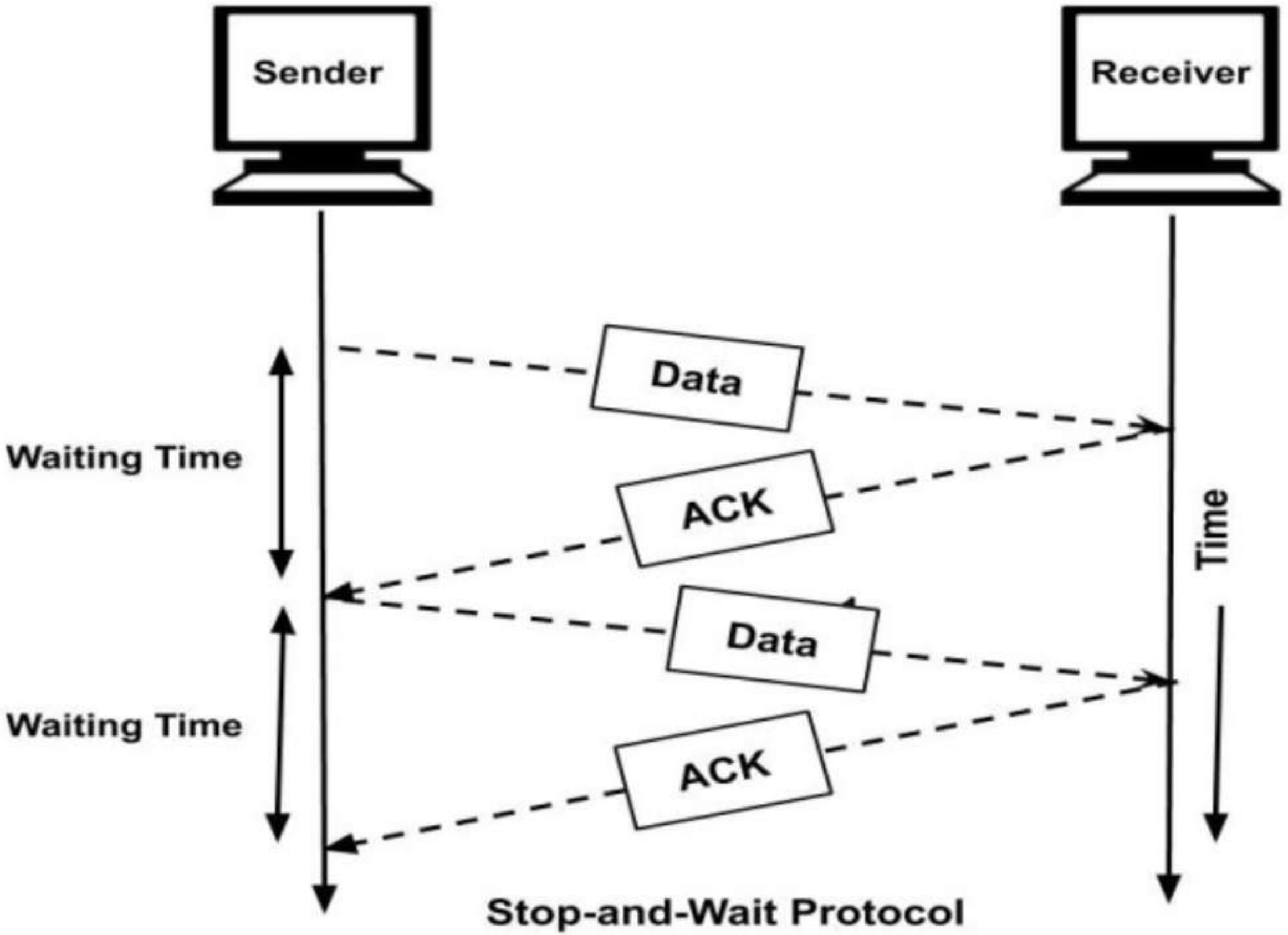
While sending the data from the sender to the receiver the flow of data needs to be controlled. Suppose a situation where the sender is sending the data at a rate higher than the receiver is able to receive and process it, then the data will get lost. The **Flow-control** methods will help in ensuring that the data doesn't get lost. The flow control method will keep a check that the senders send the data only at a rate that the receiver is able to receive and process. There are mainly two ways in which this can be achieved i.e. using **Stop-and-wait protocol** or sliding window

be achieved i.e. using **Stop-and-wait protocol** or sliding window protocol. In this blog, we are going to learn about the Stop-and-wait protocol. So, let's get started.

## Stop and Wait Protocol

It is the simplest flow control method. In this, the sender will send one frame at a time to the receiver. The sender will **stop and wait** for the acknowledgment from the receiver. This time(i.e. the time between message sending and acknowledgement receiving) is the waiting time for the sender and the sender is totally idle during this

waiting time for the sender and the sender is totally idle during this time. When the sender gets the acknowledgment(ACK), then it will send the next data packet to the receiver and wait for the acknowledgment again and this process will continue as long as the sender has the data to send. This can be understood by the diagram below:



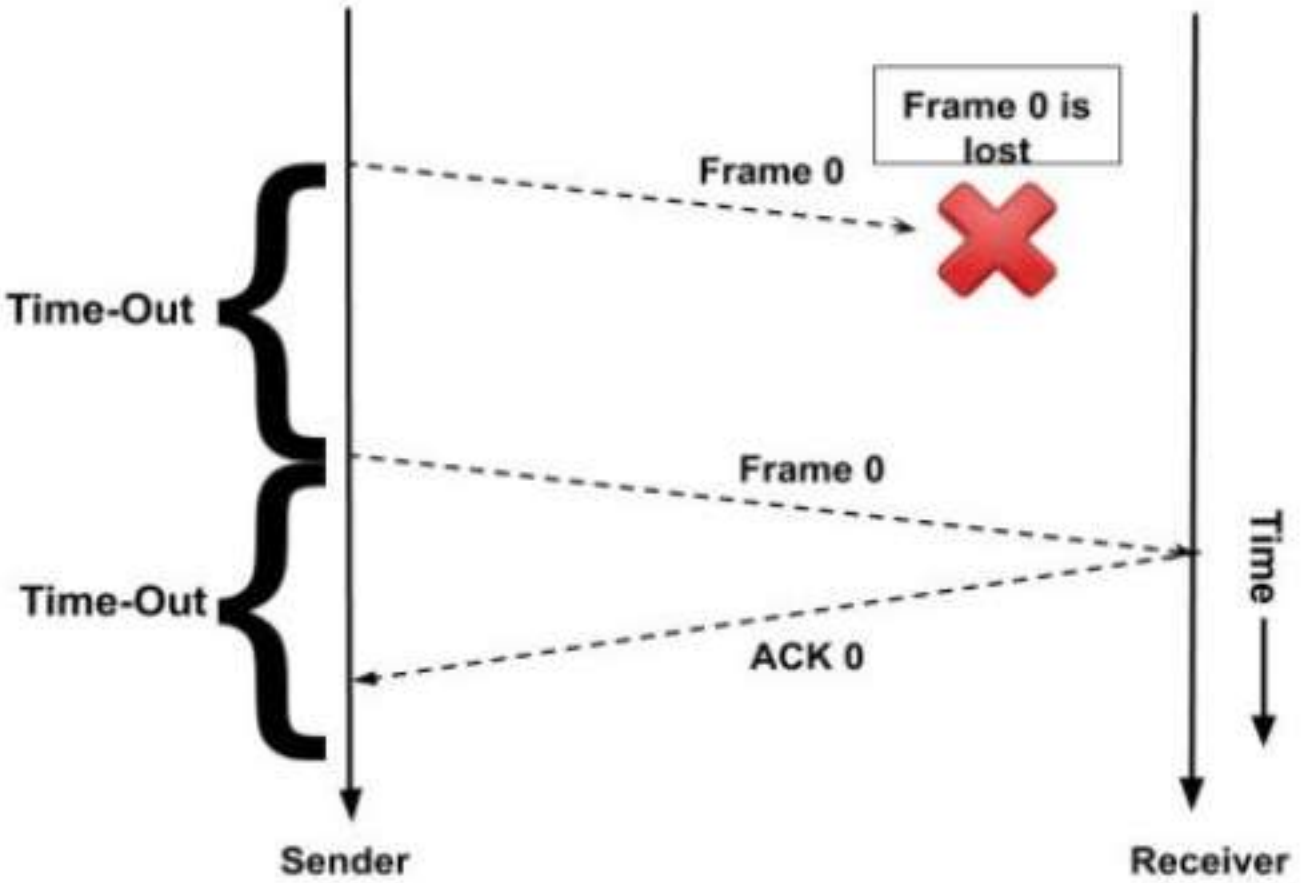

Stop-and-Wait Protocol

The above diagram explains the normal operation in a stop-and-wait protocol. Now, we will see some situations where the data or acknowledgment is lost and how the stop-and-wait protocol responds to it.

***Situation 1***

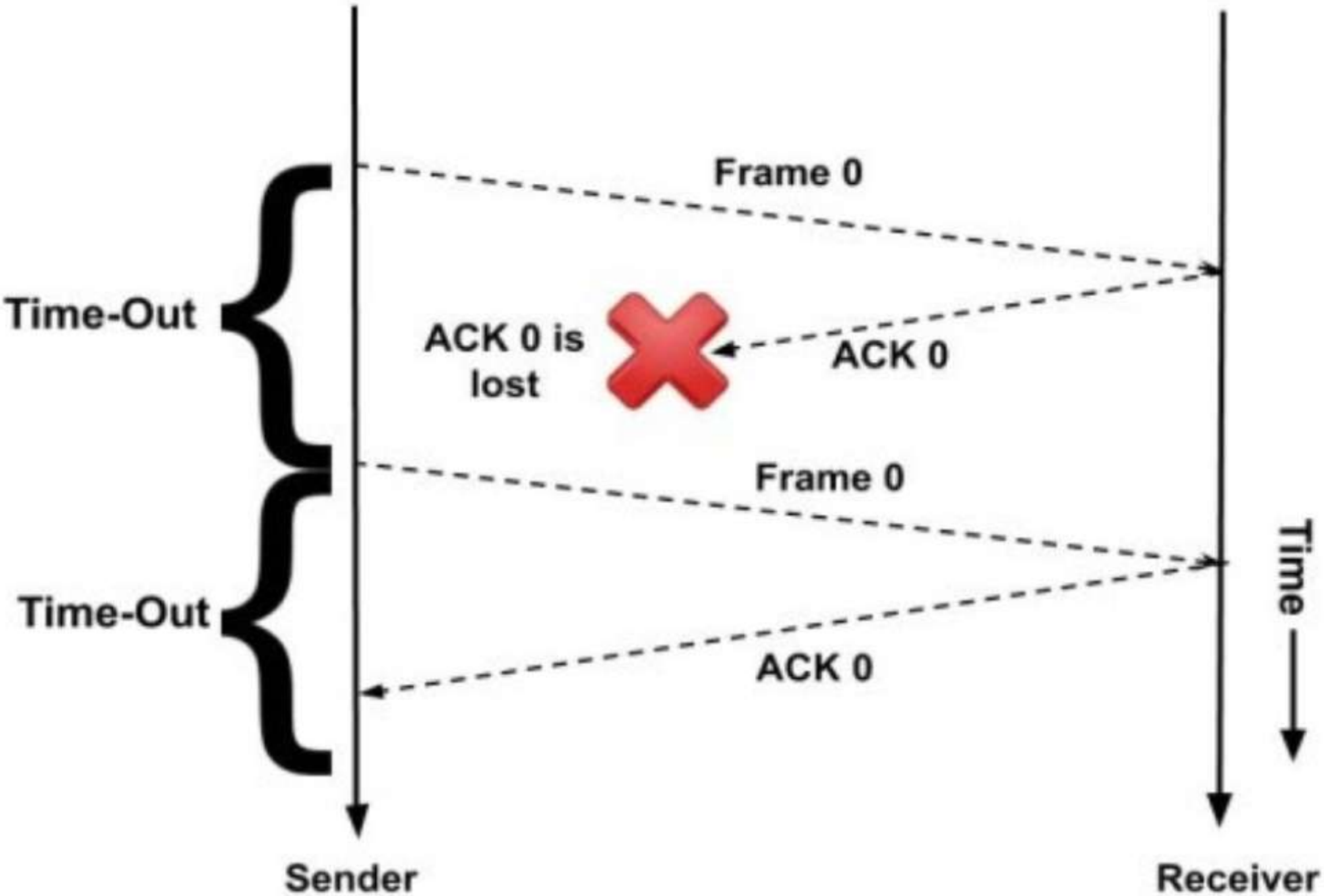
Suppose if any frame sent is not received by the receiver and is lost. So the receiver will not send any acknowledgment as it has not received any frame. Also, the sender will not send the next frame as it will wait for the acknowledgment for the previous frame which it had sent. So a deadlock situation arises here. To avoid any such situation there is a **time-out** timer. The sender waits for this fixed amount of time for the acknowledgment and if the acknowledgment is not received then it will send the frame

received then it will send the frame again.



***Situation 2***

Consider a situation where the receiver has received the data and sent the acknowledgment but the ACK is lost. So, again the sender might wait till infinite time if there is no system of **time-out** timer. So, in this case also, the time-out timer will be used and the sender will wait for a fixed amount of time for the acknowledgment and then send the frame again if the acknowledgement is not received.


Time-Out
Frame 0
ACK 0 is lost
ACK 0
Frame 0
Time-Out
ACK 0
Time
Sender
Receiver

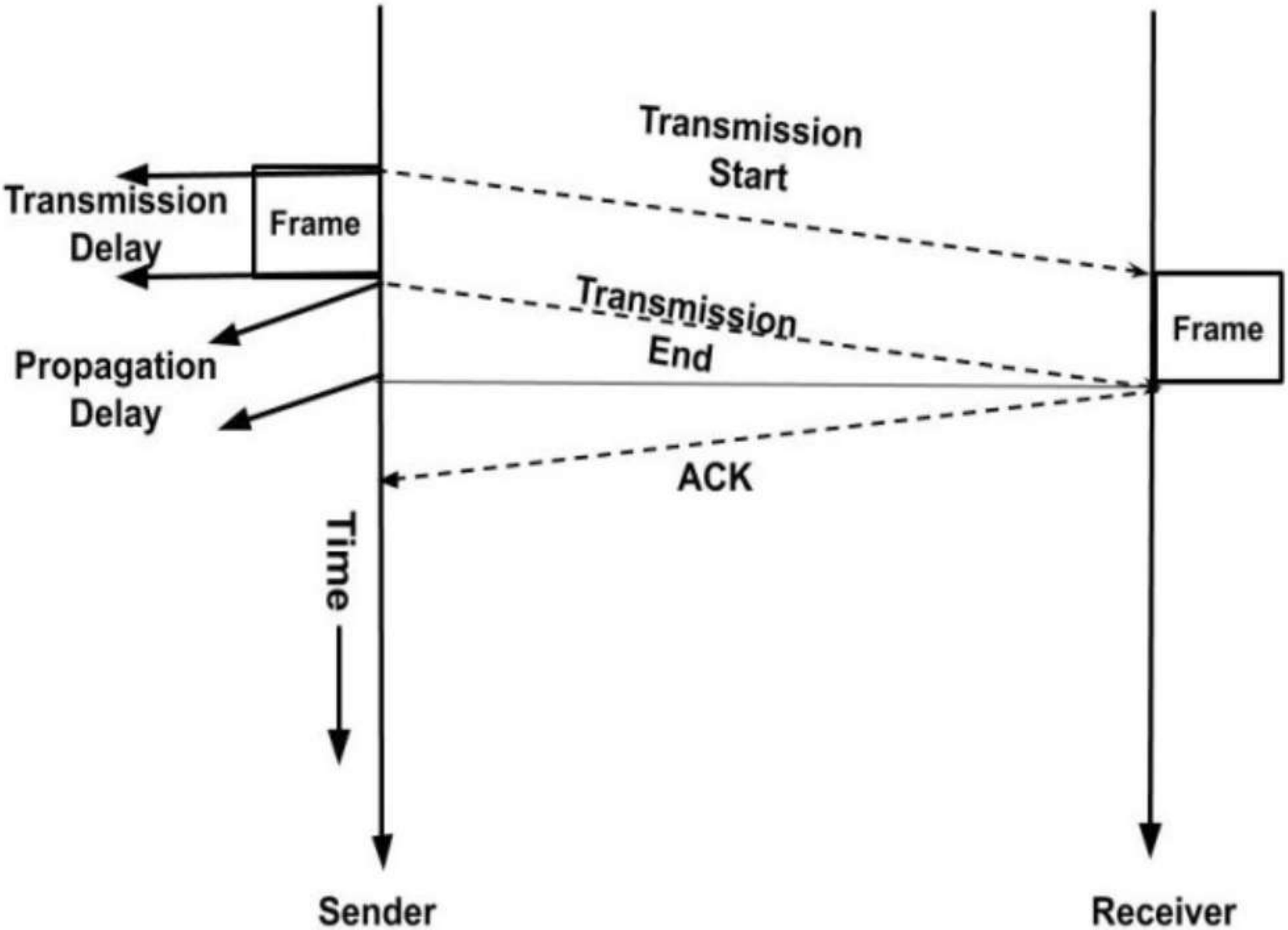
**There are two types of delays while sending these frames:**

- **Transmission Delay:** Time taken by the sender to send all the bits of the frame onto the wire is called transmission delay. This is calculated by dividing the data size(D) which has to be sent by the bandwidth(B) of the link.

$Td = D / B$

- **Propagation Delay:** Time taken by the last bit of the frame to reach from one side to the other side is called propagation delay. It is calculated by dividing the distance between the sender and receiver by the wave propagation speed.

*Tp = d / s ; where d = distance between sender and receiver, s = wave propagation speed*


Transmission
Start
Transmission
Delay
Frame
Transmission
End
Frame
Propagation
Delay
ACK
Time
Sender
Receiver

The propagation delay for sending the data frame and the acknowledgment frame is the same as distance and speed will remain the same for both frames. Hence, the total time required to send a frame is:

*Total time= Td(Transmission Delay) + Tp(Propagation Delay for data frame) + Tp(Propagation Delay for acknowledgment frame)*

*Total time=Td+ 2Tp*

The sender is doing work only for **Td** time (useful time)and for the rest **2Tp** time the sender is waiting for the acknowledgment.

## Efficiency

Efficiency = Useful Time/ Total Time

$$\eta = Td / (Td+2*Tp)$$

$$\eta = 1/(1+2a) \rightarrow (1)$$

*where* $a=Tp / Td$

# Throughput

The **number of bits that a receiver can accept in total time duration** (i.e. transmission time(Td) + 2 * propagation delay(Tp)). It is also called **effective bandwidth** or **bandwidth utilization.**

# Advantages of Stop and Wait Protocol

1. It is very simple to implement.
2. The main advantage of this protocol is the accuracy. The next frame is sent only when the first frame is acknowledged. So, there is no chance of any frame being lost.

## Disadvantages of Stop and Wait Protocol

1. We can send only one packet at a time.
2. If the distance between the sender and the receiver is large then the propagation delay would be more than the transmission delay. Hence, efficiency would become very low.

3. After every transmission, the sender has to wait for the acknowledgment and this time will increase the total transmission time. This makes the transmission process **slow**.

# Sliding Window Protocol

BY DINESH THAKUR Category: Communication Networks

- In sliding window method, multiple frames are sent by sender at a time before needing an acknowledgment.

• Multiple frames sent by source are acknowledged by receiver using a single ACK frame.

**Sliding Window**

• Sliding window refers to an imaginary boxes that hold the frames on both sender and receiver side.

• It provides the upper limit on the number of frames that can be transmitted before requiring an acknowledgment.

• Frames may be acknowledged by receiver at any point even when window is not full on receiver side.

• Frames may be transmitted by source even when window is not yet full on sender side.

• The windows have a specific size in which the frames are numbered modulo- n, which means they are numbered from 0 to n-l. For e.g. if n = 8, the frames are numbered 0, 1,2,3,4,5,6, 7, 0, 1,2,3,4,5,6, 7, 0, 1, ....

• The size of window is n-1. For e.g. In this case it is 7. Therefore, a maximum of n-l frames may be sent before an acknowledgment.

• When the receiver sends an ACK, it includes the number of next frame it expects to receive. For example in order to acknowledge the group of frames ending in frame 4, the receiver sends an ACK containing the number 5. When sender sees an ACK with number 5, it comes to know that all the frames up to number 4 have been received.


Sliding window

**Sliding Window on Sender Side**

• At the beginning of a transmission, the sender's window contains n-l frames.

• As the frames are sent by source, the left boundary of the window moves inward, shrinking the size of window. This means if window size is w, if four frames are sent by source after the last acknowledgment, then the number of frames left in window is w-4.

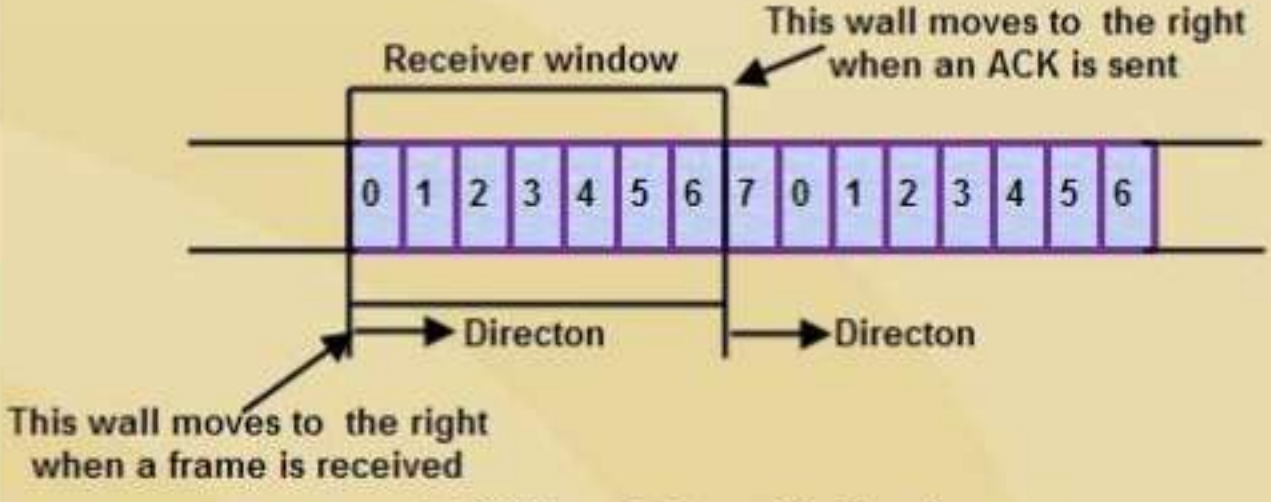
• When the receiver sends an ACK, the source's window expand i.e. (right boundary moves outward) to allow in a number of new frames equal to the number of frames acknowledged by that ACK.

• For example, Let the window size is 7 (see diagram (a)), if frames 0 through 3 have been sent and no acknowledgment has been received, then the sender's window contains three frames - 4,5,6.

• Now, if an ACK numbered 3 is received by source, it means three frames (0, 1, 2) have been received by receiver and are undamaged.

• The sender's window will now expand to include the next three frames in its buffer. At this point the sender's window will contain six frames (4, 5, 6, 7, 0, 1). (See diagram (b)).

# Sliding window on sender side




(a) Sliding window with size=7


(b) Sliding window containing 6 frames

**Sliding Window on Receiver Side**

• At the beginning of transmission, the receiver's window contains n-1 spaces for frame but not the frames.

• As the new frames come in, the size of window shrinks.

• Therefore the receiver window represents not the number of frames received but the number of frames that may still be received without an acknowledgment ACK must be sent.

• Given a window of size w, if three frames are received without an ACK being returned, the number of spaces in a window is w-3.

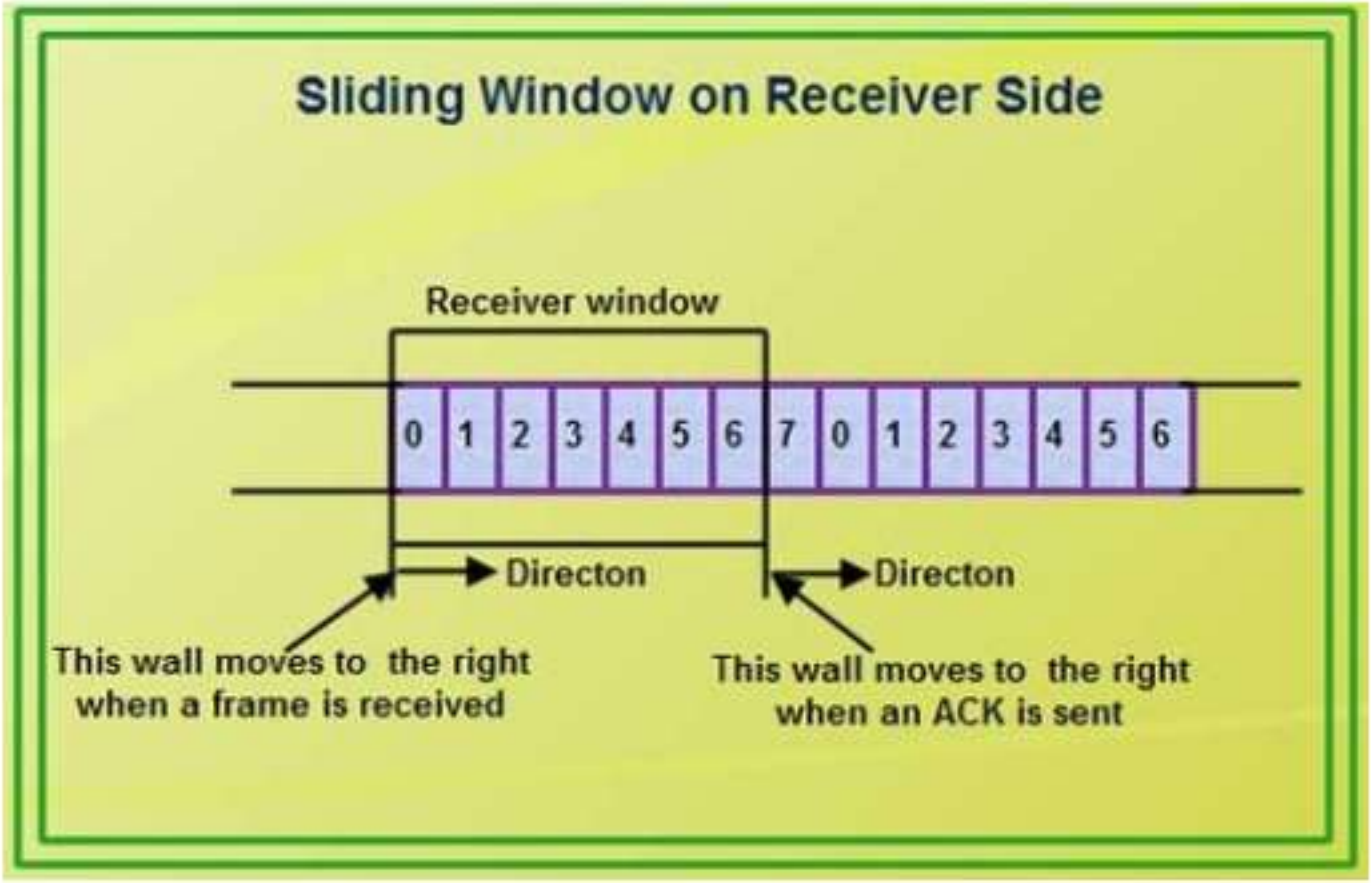
• As soon as acknowledgment is sent, window expands to include the number of frames equal to the number of frames acknowledged.

• With the arrival of the first frame, the receiving window shrinks, moving the boundary from space 0 to 1. Now, window has shrunk by one, so the receiver may accept six more frame before it is required to send an ACK.

• If frames 0 through 3 have arrived but have DOC been acknowledged, the window will contain three frame spaces.

• As receiver sends an ACK, the window of the receiver expands to include as many new placeholders as newly acknowledged frames.

• The window expands to include a number of new frame spaces equal to the number of the most recently acknowledged frame minus the number of previously acknowledged frame. For *e.g.*, If window size is 7 and if prior ACK was for frame 2 & the current ACK is for frame 5 the window expands by three (5-2).




• Therefore, the sliding window of sender shrinks from left when frames of data are sending. The sliding window of the sender expands to right when acknowledgments are received.

• The sliding window of the receiver shrinks from left when frames of data are received. The sliding window of the receiver expands to the right when acknowledgement is sent.

# Types of Sliding Window Protocol

Sliding window protocol has two types:

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

## 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.

# Go back N Protocol-

Go back N protocol is an implementation of a sliding window protocol.

The features and working of this protocol are explained in the following points-

## Point-01:

> In Go back N, sender window size is N and receiver window size is always 1.

In Go back N,

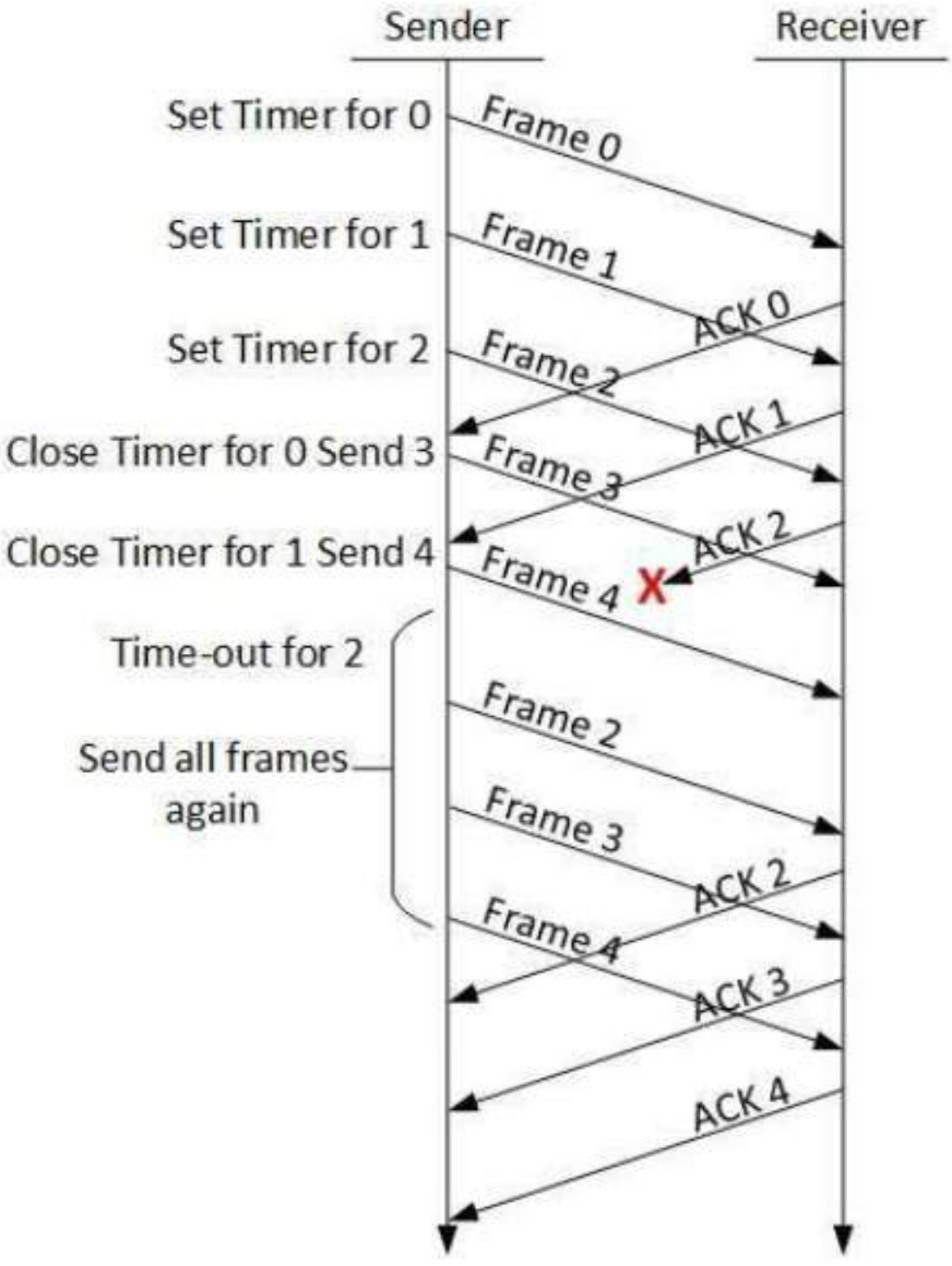
- Sender window size = N. Example in Go back 10, sender window size will be 10.
- Receiver window size is always 1 for any value of N.

- Go-Back-N ARQ

  Stop and wait ARQ mechanism does not utilize the resources at their best.When the acknowledgement is received, the sender sits idle and does nothing. In Go-Back-N ARQ method, both sender and receiver maintain a window.

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.

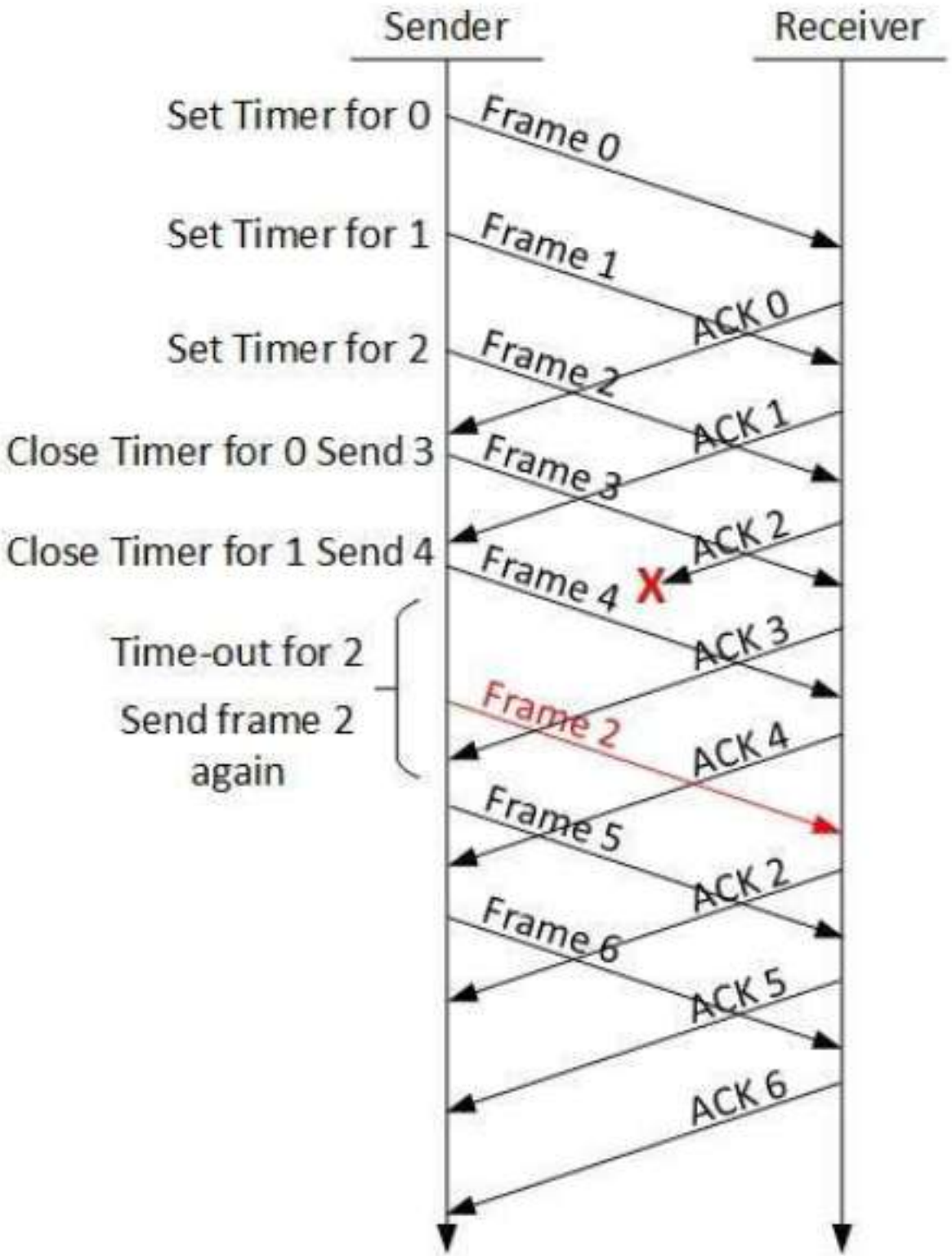

Sender
Receiver
Set Timer for 0
Frame 0
Set Timer for 1
Frame 1
ACK 0
Set Timer for 2
Frame 2
ACK 1
Close Timer for 0 Send 3
Frame 3
ACK 2
Close Timer for 1 Send 4
Frame 4
X
Time-out for 2
Frame 2
Send all frames again
Frame 3
ACK 2
Frame 4
ACK 3
ACK 4

The sending-window size enables the sender to send multiple frames without receiving the acknowledgement of the previous ones. The receiving-window enables the receiver to receive multiple frames and acknowledge them. The receiver keeps track of incoming frame's sequence number.

When the sender sends all the frames in window, it checks up to what sequence number it has received positive acknowledgement. If all frames are positively acknowledged, the sender sends next set of frames. If sender finds that it has received NACK or has not receive any ACK for a particular frame, it retransmits all the frames after which it does not receive any positive ACK.

## Selective Repeat ARQ

In Go-back-N ARQ, it is assumed that the receiver does not have any buffer space for its window size and has to process each frame as it comes. This enforces the sender to retransmit all the frames which are not acknowledged.


Sender
Receiver
Set Timer for 0
Frame 0
Set Timer for 1
Frame 1
ACK 0
Set Timer for 2
Frame 2
ACK 1
Close Timer for 0 Send 3
Frame 3
ACK 2
Close Timer for 1 Send 4
Frame 4
X
ACK 3
Time-out for 2
Send frame 2 again
Frame 2
ACK 4
Frame 5
ACK 2
Frame 6
ACK 5
ACK 6

In Selective-Repeat ARQ, the receiver while keeping track of sequence numbers, buffers the frames in memory and sends NACK for only frame which is missing or damaged.

The sender in this case, sends only packet for which NACK is received.

## Piggybacking

Instead of sending ack frame on its own, if there is an outgoing data frame in the next short interval, attach the ack to it (using "ack" field in header).
Better use of bandwidth. Ack is only a few bits (here 1 bit). Costly to have to construct an entire frame for it.