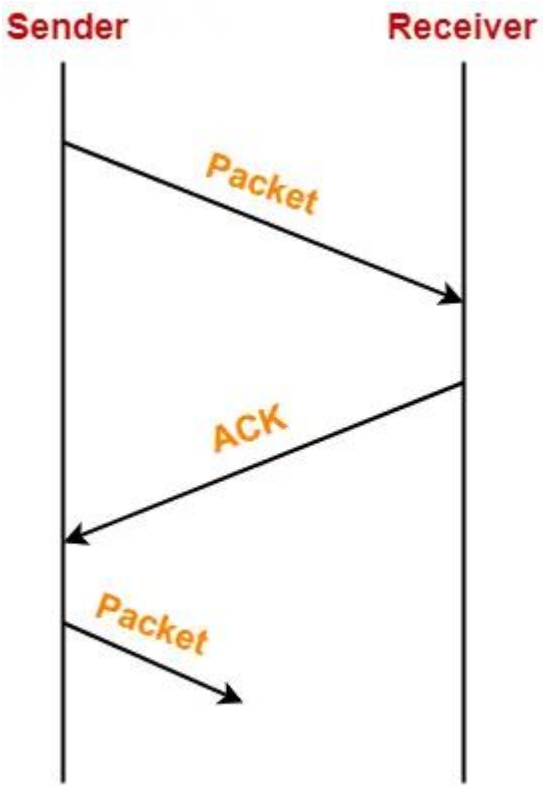
Stop and Wait Protocol-

In stop and wait protocol,

- Sender sends one data packet and then waits for its acknowledgement.
- Sender sends the next packet only after it receives the acknowledgement for the previous packet.



Stop and Wait Protocol

The main problem faced by the Stop and Wait protocol is the occurrence of deadlock due to-

1. Loss of data packet
2. Loss of acknowledgement

Stop and Wait ARQ-

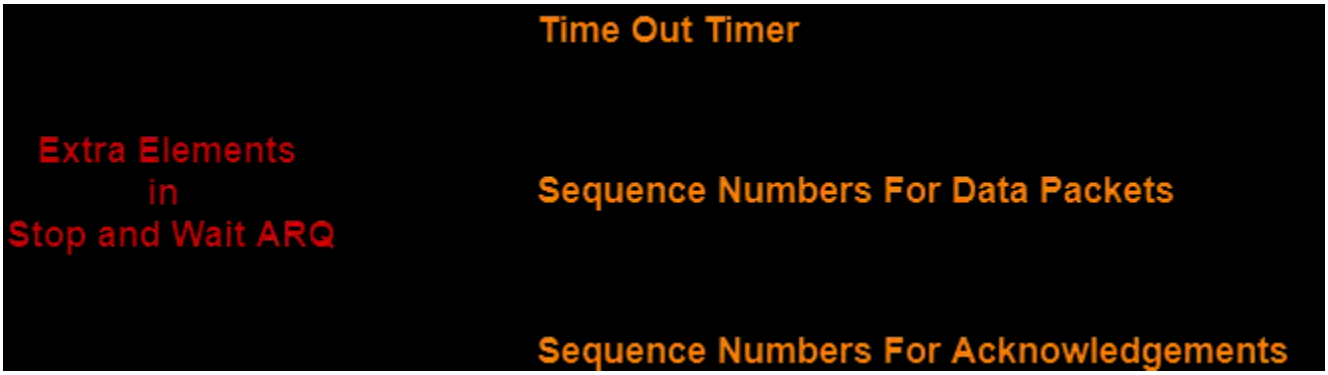
Stop and Wait ARQ is an improved and modified version of Stop and Wait protocol.

Stop and Wait ARQ assumes-

- The communication channel is noisy.
- Errors may get introduced in the data during the transmission.

Working-

- Stop and wait ARQ works similar to stop and wait protocol.
- It provides a solution to all the limitations of stop and wait protocol.
- Stop and wait ARQ includes the following three extra elements.



Thus, we can say-

> Stop and Wait ARQ
>
> = Stop and Wait Protocol + Time Out Timer + Sequence Numbers for Data Packets and Acknowledgements

Number of Sequence Numbers Required-

> **NOTE**
>
> For any sliding window protocol to work without any problem,
>
> the following condition must be satisfied-
>
> Available Sequence Numbers >= Sender Window Size + Receiver Window Size

Stop and wait ARQ is a one bit sliding window protocol where-

- Sender window size = 1
- Receiver window size = 1

Thus, in stop and wait ARQ,

Minimum number of sequence numbers required

= Sender Window Size + Receiver Window Size

= 1 + 1

= 2

Thus,

- Minimum number of sequence numbers required in Stop and Wait ARQ = 2.
- The two sequence numbers used are 0 and 1.

How Stop and Wait ARQ Solves All Problems?

1. Problem of Lost Data Packet-

- Time out timer helps to solve the problem of lost data packet.
- After sending a data packet to the receiver, sender starts the time out timer.
- If the data packet gets acknowledged before the timer expires, sender stops the time out timer.
- If the timer goes off before receiving the acknowledgement, sender retransmits the same data packet.
- After retransmission, sender resets the timer.
- This prevents the occurrence of deadlock.

2. Problem of Lost Acknowledgement-

Sequence number on data packets help to solve the problem of delayed acknowledgement.

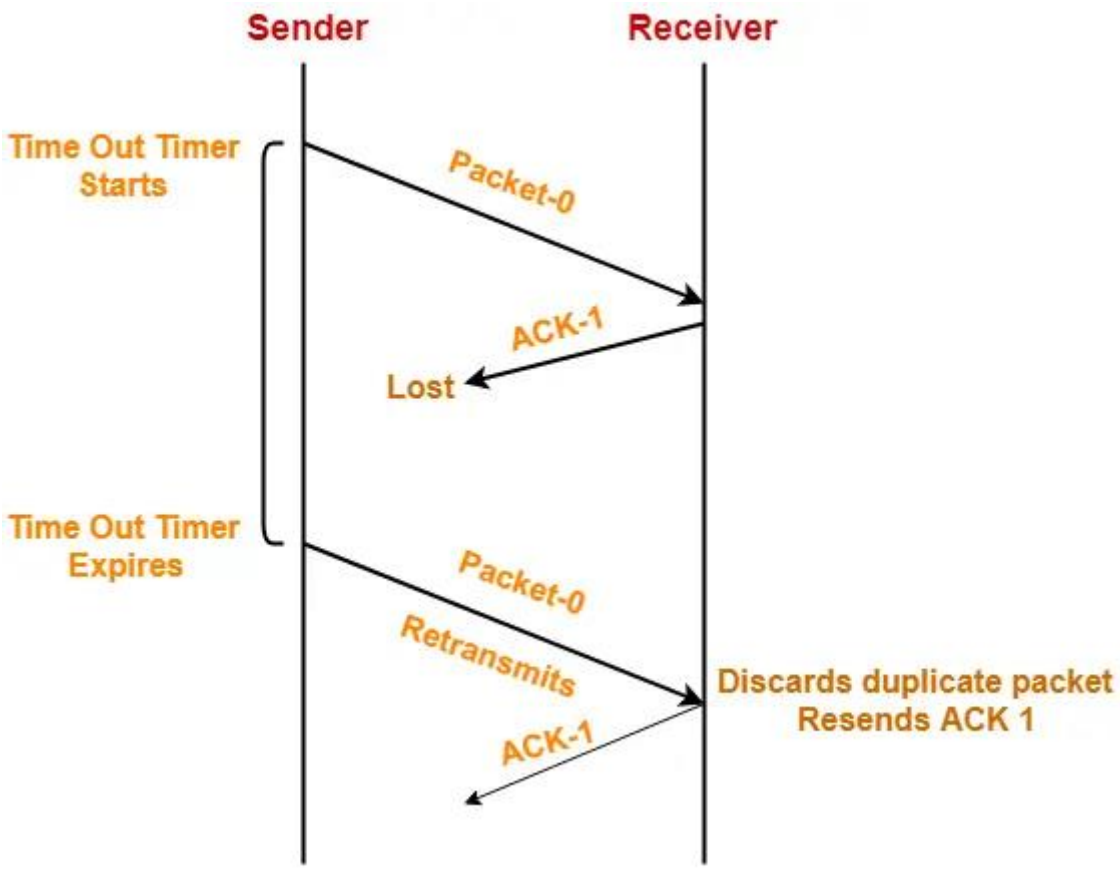
Consider the acknowledgement sent by the receiver gets lost.

Then, sender retransmits the same data packet after its timer goes off.

This prevents the occurrence of deadlock.

The sequence number on the data packet helps the receiver to identify the duplicate data packet.

Receiver discards the duplicate packet and re-sends the same acknowledgement.



Role of Sequence Number on Data Packets

Consider the above example-

Step-01:

- Sender sends a data packet with sequence number-0 to the receiver.

Step-02:

- Receiver receives the data packet correctly.
- Receiver now expects data packet with sequence number-1.
- Receiver sends the acknowledgement ACK-1.

Step-03:

- Acknowledgement ACK-1 sent by the receiver gets lost on the way.

Step-04:

- Sender receives no acknowledgement and time out occurs.
- Sender retransmits the same data packet with sequence number-0.
- This will be a duplicate packet for the receiver.

Step-05:

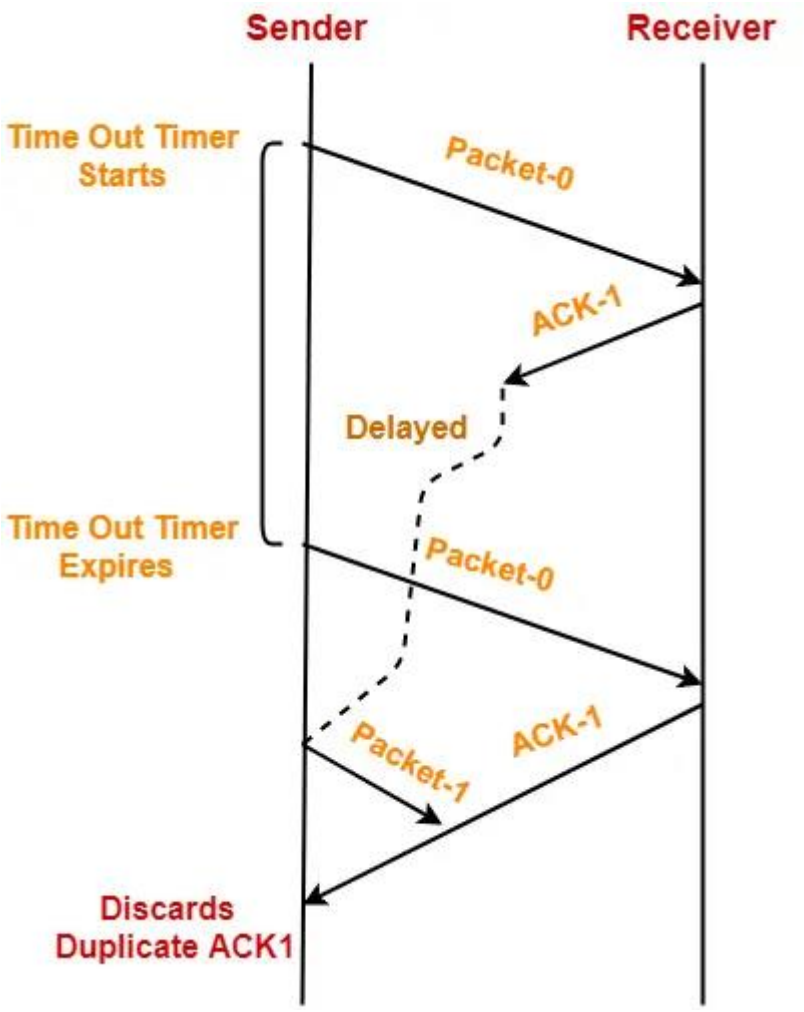
- Receiver receives the data packet and discovers it is the duplicate packet.
- It expects the data packet with sequence number-1 but receiving the data packet with sequence number-0.
- It discards the duplicate data packet and re-sends acknowledgement ACK-1.
- ACK-1 requests the sender to send a data packet with sequence number-1.
- This avoids the inconsistency of data.

Conclusion-

- Had the sequence numbers not been allotted to the data packets, receiver would have accepted the duplicate data packet thinking of it as the new data packet.
- This is how sequence numbers allotted to the data packets prove to be useful for identifying the duplicate data packets and discarding them.

3. Problem of Delayed Acknowledgement-

- Sequence number on acknowledgements help to solve the problem of delayed acknowledgement.



Role of Sequence Number on Acknowledgements

Consider the above example-

Step-01:

- Sender sends a data packet with sequence number-0 to the receiver.

Step-02:

- Receiver receives the data packet correctly.
- Receiver now expects data packet with sequence number-1.
- Receiver sends the acknowledgement ACK-1.

Step-03:

- Acknowledgement ACK-1 sent by the receiver gets delayed in reaching the sender.

Step-04:

- Sender receives no acknowledgement and time out occurs.
- Sender retransmits the same data packet with sequence number-0.
- This will be a duplicate packet for the receiver.

Step-05:

- Receiver receives the data packet and discovers it is the duplicate packet.
- It expects the data packet with sequence number-1 but receiving the data packet with sequence number-0.
- It discards the duplicate data packet and re-sends acknowledgement ACK-1.
- ACK-1 requests the sender to send a data packet with sequence number-1.

Step-06:

- Two acknowledgements ACK1 reaches the sender.
- When first acknowledgement ACK1 reaches the sender, sender sends the next data packet with sequence number 1.
- When second acknowledgement ACK1 reaches the sender, sender rejects the duplicate acknowledgement.
- This is because it has already sent the data packet with sequence number-1 and now sender expects the acknowledgement with sequence number 0 from the receiver.

Conclusion-

- Had the sequence numbers not been allotted to the acknowledgements, sender would have accepted the duplicate acknowledgement thinking of it as the new acknowledgement for the latest data packet sent by it.
- This is how sequence numbers allotted to the acknowledgements prove to be useful for identifying duplicate acknowledgements and discarding them.

4. Problem of Damaged Packet-

- If receiver receives a corrupted data packet from the sender, it sends a negative acknowledgement (NAK) to the sender.
- NAK requests the sender to send the data packet again.

Stop and Wait Protocol Vs Stop and Wait ARQ-

The following comparison table states the differences between the two protocols-

Stop and Wait Protocol	Stop and Wait ARQ
It assumes that the communication channel is perfect and noise free.	It assumes that the communication channel is imperfect and noisy.
Data packet sent by the sender can never get corrupt.	Data packet sent by the sender may get corrupt.
There is no concept of negative acknowledgements.	A negative acknowledgement is sent by the receiver if the data packet is found to be corrupt.
There is no concept of time out timer.	Sender starts the time out timer after sending the data packet.
There is no concept of sequence numbers.	Data packets and acknowledgements are numbered using sequence numbers.

Limitation of Stop and Wait ARQ-

The major limitation of Stop and Wait ARQ is its very less efficiency.

Explanation-

In stop and wait ARQ,

- Sender window size is 1.
- This allows the sender to keep only one frame unacknowledged.
- So, sender sends one frame and then waits until the sent frame gets acknowledged.
- After receiving the acknowledgement from the receiver, sender sends the next frame.

Here,

- Sender uses T_t time for transmitting the packet over the link.
- Then, sender waits for $2 \times T_p$ time.
- After $2 \times T_p$ time, sender receives the acknowledgement for the sent frame from the receiver.
- Then, sender sends the next frame.
- This $2 \times T_p$ waiting time is the actual cause of less efficiency.

Efficiency Improvement-

- The efficiency of stop and wait ARQ can be improved by increasing the window size.

- This allows the sender to keep more than one unacknowledged frame in its window.
- Thus, sender can send frames in the waiting time too.

This gives rise to the concept of sliding window protocols.