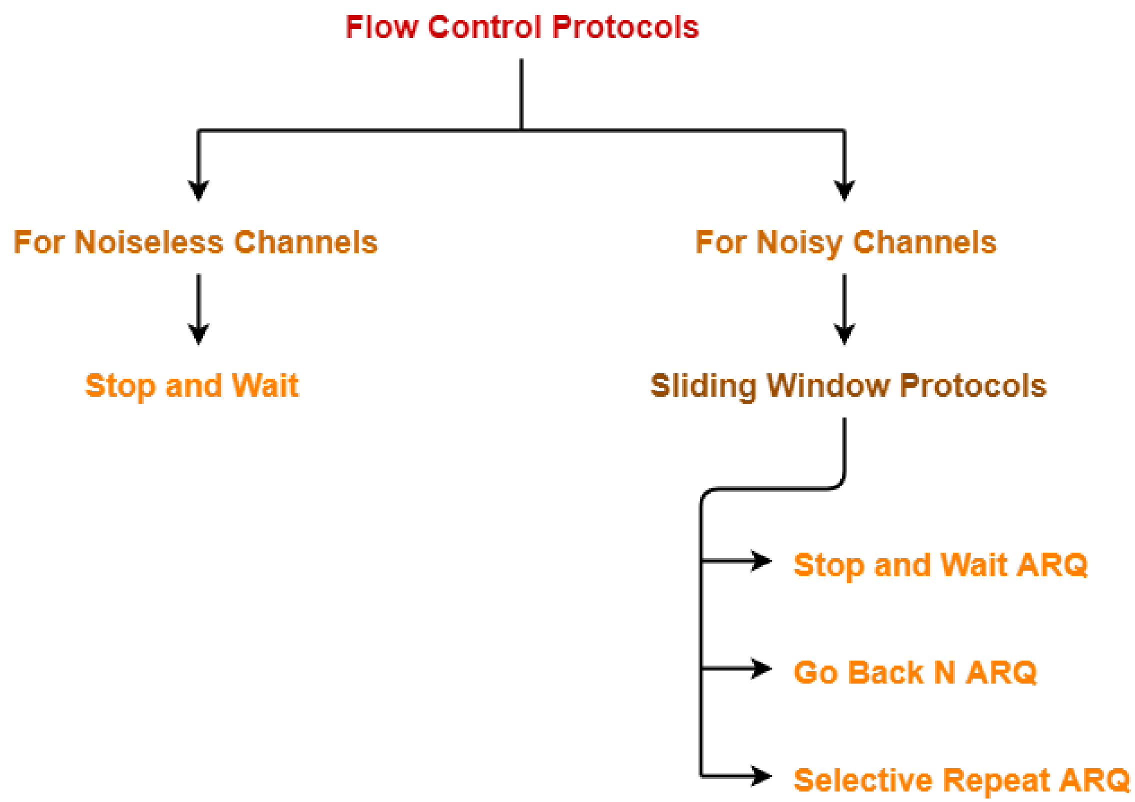
Flow Control in Computer Networks-

A set of procedures which are used for restricting the amount of data that a sender can send to the receiver.

Flow Control Protocols-



Stop and Wait Protocol-

Stop and Wait Protocol is the simplest flow control protocol.

It works under the following assumptions-

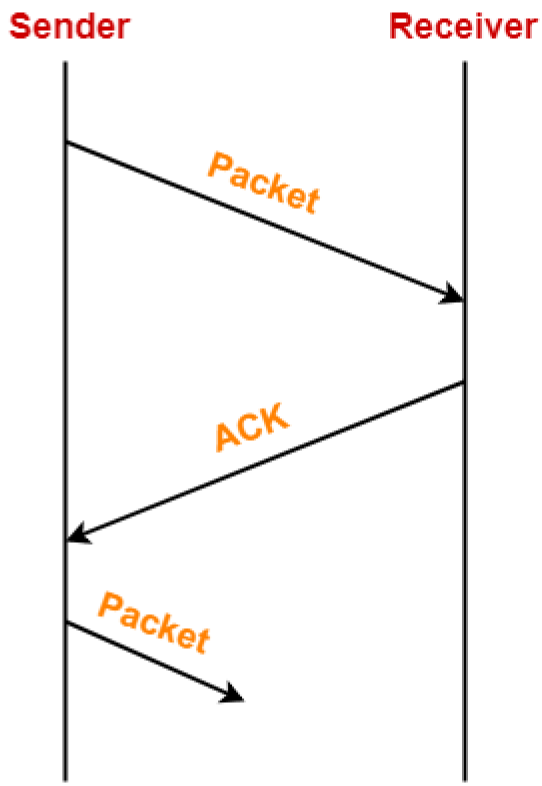
- Communication channel is perfect.
- No error occurs during transmission.

Working-

The working of a stop and wait protocol may be explained as-

- Sender sends a data packet to the receiver.
- Sender stops and waits for the acknowledgement for the sent packet from the receiver.
- Receiver receives and processes the data packet.
- Receiver sends an acknowledgement to the sender.
- After receiving the acknowledgement, sender sends the next data packet to the receiver.

These steps are illustrated below-

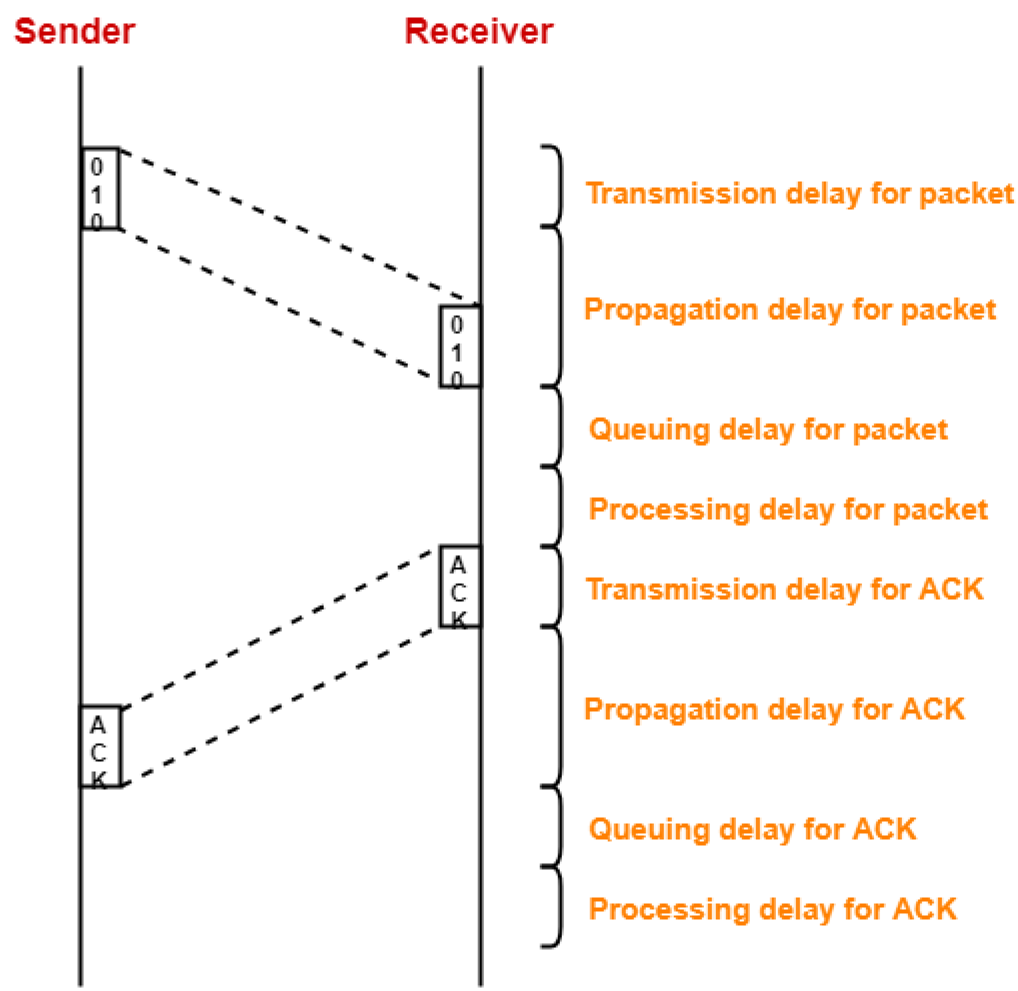


Stop and Wait Protocol

let us analyze in depth how the transmission is actually carried out-

- Sender puts the data packet on the transmission link.
- Data packet propagates towards the receiver’s end.
- Data packet reaches the receiver and waits in its buffer.
- Receiver processes the data packet.
- Receiver puts the acknowledgement on the transmission link.
- Acknowledgement propagates towards the sender’s end.
- Acknowledgement reaches the sender and waits in its buffer.
- Sender processes the acknowledgement.

These steps are illustrated below-



Stop and Wait Protocol

Total Time-

Total time taken in sending one data packet

$$= (\text{Transmission delay} + \text{Propagation delay} + \text{Queuing delay} + \text{Processing delay})_{\text{packet}}$$

$$+$$

$$(\text{Transmission delay} + \text{Propagation delay} + \text{Queuing delay} + \text{Processing delay})_{\text{ACK}}$$

Assume-

- Queuing delay and processing delay to be zero at both sender and receiver side.
- Transmission time for the acknowledgement to be zero since it’s size is very small.

Total time taken in sending one data packet

$$= (\text{Transmission delay} + \text{Propagation delay})_{\text{packet}} + (\text{Propagation delay})_{\text{ACK}}$$

We know,

- Propagation delay depends on the distance and speed.
- So, it would be same for both data packet and acknowledgement.

So, we have-

Total time taken in sending one data packet

$$= (\text{Transmission delay})_{\text{packet}} + 2 \times \text{Propagation delay}$$

Efficiency-

Efficiency of any flow control control protocol is given by-

Efficiency (η) = Useful Time / Total Time where-

- Useful time = Transmission delay of data packet = $(\text{Transmission delay})_{\text{packet}}$
- Useless time = Time for which sender is forced to wait and do nothing = 2 x Propagation delay
- Total time = Useful time + Useless time

$$\text{Efficiency } (\eta) = \frac{(\text{Transmission delay})_{\text{packet}}}{(\text{Transmission delay})_{\text{packet}} + 2 \text{ x Propagation delay}}$$

OR

$$\text{Efficiency } (\eta) = \frac{T_t}{T_t + 2T_p}$$

OR

$$\text{Efficiency } (\eta) = \frac{1}{1 + 2\left(\frac{T_p}{T_t}\right)}$$

OR

$$\text{Efficiency } (\eta) = \frac{1}{1 + 2a} \text{, where } a = \left(\frac{T_p}{T_t}\right)$$

Factors Affecting Efficiency-

We know,

Efficiency (η) = $(\text{Transmission delay})_{packet}$ / { $(\text{Transmission delay})_{packet}$ + 2 x Propagation delay }

Dividing numerator and denominator by $(\text{Transmission delay})_{packet}$, we get-

$$\text{Efficiency } (\eta) = \frac{1}{1 + 2 \times \left(\dfrac{\text{Propagation delay}}{(\text{Transmission delay})_{packet}} \right)}$$

$$\text{Efficiency } (\eta) = \frac{1}{1 + 2 \times \left(\dfrac{\text{Distance}}{\text{speed}} \right) \times \left(\dfrac{\text{Bandwidth}}{\text{Packet length}} \right)}$$

From here, we can observe-

- Efficiency (η) ∝ 1 / Distance between sender and receiver
- Efficiency (η) ∝ 1 / Bandwidth
- Efficiency (η) ∝ Transmission speed
- Efficiency (η) ∝ Length of data packet

Throughput-

- Number of bits that can be sent through the channel per second is called as its throughput.

Throughput = Efficiency (η) x Bandwidth

Round Trip Time-

Round Trip Time = 2 x Propagation delay

Advantages

- It is very simple to implement.
- The incoming packet from receiver is always an acknowledgement.

Limitations-

1. It is extremely inefficient because-

- It makes the transmission process extremely slow.
- It does not use the bandwidth entirely as each single packet and acknowledgement uses the entire time to traverse the link.

2. If the data packet sent by the sender gets lost, then-

- Sender will keep waiting for the acknowledgement for infinite time.
- Receiver will keep waiting for the data packet for infinite time.

3. If acknowledgement sent by the receiver gets lost, then-

- Sender will keep waiting for the acknowledgement for infinite time.
- Receiver will keep waiting for another data packet for infinite time.

Important Notes-

- Efficiency may also be referred by the following names-
  - Line Utilization
  - Link Utilization
  - Sender Utilization
  - Utilization of Sender
- Throughput may also be referred by the following names-
  - Bandwidth Utilization
  - Effective Bandwidth
  - Maximum data rate possible
  - Maximum achievable throughput

- Stop and Wait protocol performs better for LANs than WANs. Because-
- Efficiency of the protocol is inversely proportional to the distance between sender and receiver.
- So, the protocol performs better where the distance between sender and receiver is less.
- The distance is less in LANs as compared to WANs.

To gain better understanding about Stop and Wait Protocol,

Stop and Wait ARQ

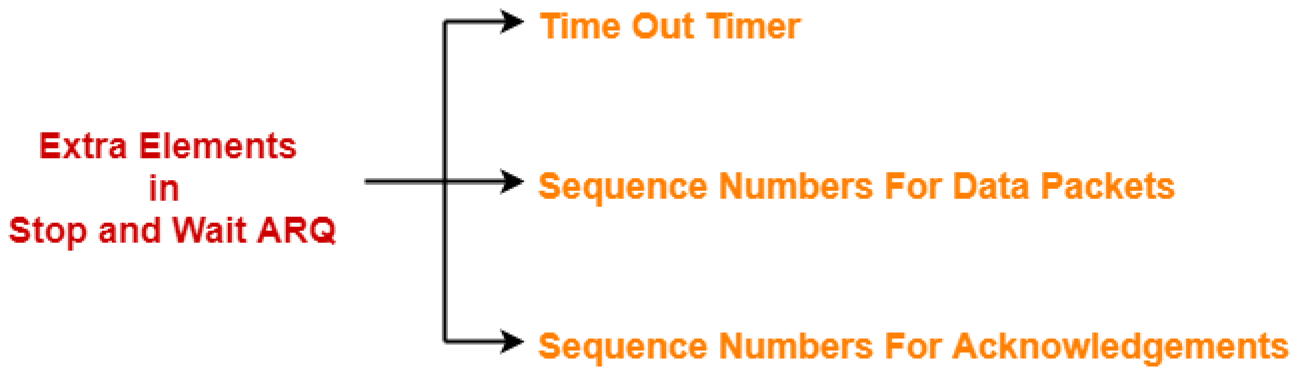
Stop and Wait ARQ is an improved and modified version of Stop and Wait protocol.

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.



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

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

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.

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.

Sliding Window Protocol

- Sliding window protocol is a flow control protocol.
- It allows the sender to send multiple frames before needing the acknowledgements.
- Sender slides its window on receiving the acknowledgements for the sent frames.
- This allows the sender to send more frames.
- It is called so because it involves sliding of sender’s window.

Maximum number of frames that sender can send without acknowledgement

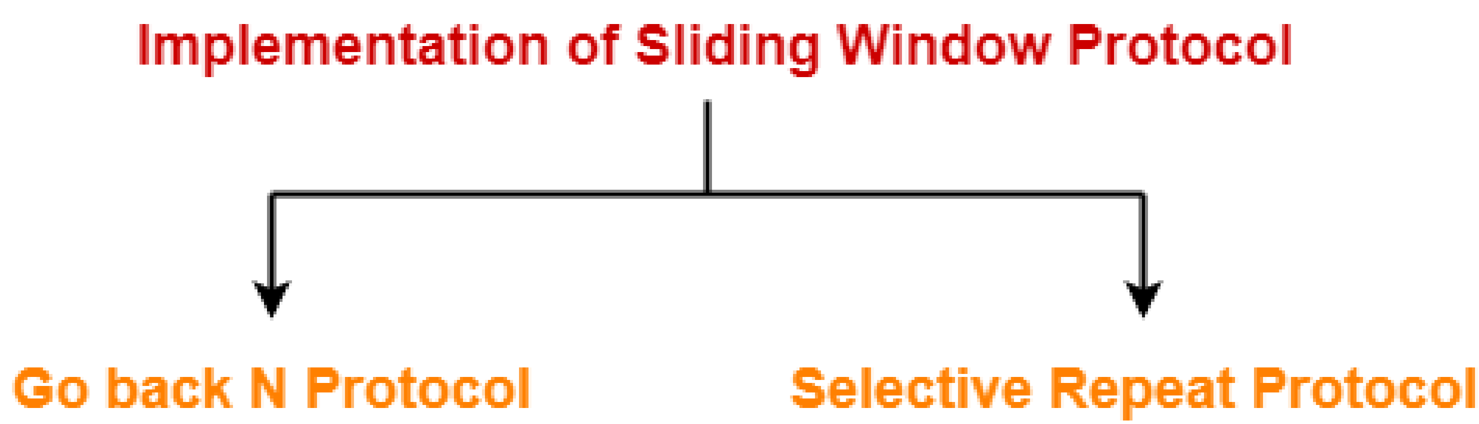
= Sender window size

Optimal Window Size-

In a sliding window protocol, optimal sender window size = 1 + 2a

Implementations of Sliding Window Protocol-

The two well known implementations of sliding window protocol are-



Efficiency-

Efficiency of any flow control protocol may be expressed as-

$$\text{Efficiency } (\eta) = \frac{\text{Number of frames sent in one window}}{\text{Total number of frames that can be sent in one window}}$$

OR

$$\text{Efficiency } (\eta) = \frac{\text{Sender Window Size in the Protocol}}{\text{Optimal Sender Window Size}}$$

OR

$$\text{Efficiency } (\eta) = \frac{\text{Sender Window Size in the Protocol}}{1 + 2a}$$

In **Stop and Wait ARQ**, sender window size = 1.

Thus, Efficiency of Stop and Wait ARQ = 1 / 1+2a

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.

Point-02:

Go back N uses cumulative acknowledgements.

In Go back N,

- Receiver maintains an acknowledgement timer.
- Each time the receiver receives a new frame, it starts a new acknowledgement timer.
- After the timer expires, receiver sends the cumulative acknowledgement for all the frames that are unacknowledged at that moment.

NOTE-

- A new acknowledgement timer does not start after the expiry of old acknowledgement timer.
- It starts after a new frame is received.

Point-03:

Go back N may use independent acknowledgements too.

- The above point does not mean that Go back N can not use independent acknowledgements.
- Go back N may use independent acknowledgements too if required.
- The kind of acknowledgement used depends on the expiry of acknowledgement timer.

Point-04:

Go back N does not accept the corrupted frames and silently discards them.

In Go back N,

- If receiver receives a frame that is corrupted, then it silently discards that frame.

- The correct frame is retransmitted by the sender after the time out timer expires.
- Silently discarding a frame means-

“Simply rejecting the frame and not taking any action”

(like not sending a NACK to the sender to send the correct frame)

Point-05:

Go back N does not accept out of order frames and silently discards them.

In Go back N,

- If receiver receives a frame whose sequence number is not what the receiver expects, then it silently discards that frame.
- All the following frames are also discarded.
- This is because receiver window size is 1 and therefore receiver can not accept out of order frames.

Point-06:

> Go back N leads to retransmission of entire window if for any frame, no ACK is received by the sender.

In Go back N,

- Receiver silently discards the frame if it founds the frame to be either corrupted or out of order.
- It does not send any acknowledgement for such frame.
- It silently discards the following frames too.

Thus,

- If for any particular frame, sender does not receive any acknowledgement, then it understands that along with that frame, all the following frames must also have been discarded by the receiver.
- So, sender has to retransmit all the following frames too along with that particular frame.
- Thus, it leads to the retransmission of entire window.
- That is why, the protocol has been named as "**Go back N**".

Point-07:

Go back N leads to retransmission of lost frames after expiry of time out timer.

in Go back N,

- Consider a frame being sent to the receiver is lost on the way.
- Then, it is retransmitted only after time out timer expires for that frame at sender's side.

Efficiency of Go back N-

Efficiency of any flow control protocol is given by-

Efficiency = Sender Window Size in Protocol / (1 + 2a)

In Go back N protocol, sender window size = N.

Thus,

Efficiency of Go back N = N / (1 + 2a)

Selective Repeat Protocol-

Selective Repeat protocol or SR 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 SR protocol, sender window size is always same as receiver window size.

In SR protocol,

- Sender window size = Receiver window size
- The size is of course greater than 1 otherwise the protocol will become **Stop and Wait ARQ**.
- If n bits are available for sequence numbers, then-

$$\text{Sender window size} = \text{Receiver window size} = 2^n/2 = 2^{n-1}$$

Point-02:

SR protocol uses independent acknowledgements only.

In SR protocol,

- Receiver acknowledges each frame independently.
- As receiver receives a new frame from the sender, it sends its acknowledgement.

Point-03:

SR protocol does not accept the corrupted frames but does not silently discard them.

In SR protocol,

- If receiver receives a frame that is corrupted, then it does not silently discard that frame.
- Receiver handles the situation efficiently by sending a negative acknowledgement (NACK).
- Negative acknowledgement allows early retransmission of the corrupted frame.
- It also avoids waiting for the time out timer to expire at the sender side to retransmit the frame.

Point-05:

SR protocol accepts the out of order frames.

In SR protocol,

- Consider receiver receives a frame whose sequence number is not what the receiver expects.
- Then, it does not discard that frame rather accepts it and keeps it in its window.

Point-06:

> SR protocol requires sorting at the receiver's side.

In SR protocol,

- Receiver window is implemented as a linked list.
- When receiver receives a new frame, it places the new frame at the end of the linked list.
- When the received frames are out of order, receiver performs the sorting.
- Sorting sorts the frames in the correct order.

Point-07:

SR protocol requires searching at the sender's side.

In SR protocol,

- Receiver does not reject the out of order frames.
- Receiver accepts the out of order frames and sort them later.
- Thus, only the missing frame has to be sent by the sender.
- For sending the missing frame, sender performs searching and finds the missing frame.
- Then, sender selectively repeats that frame.
- Thus, only the selected frame is repeated and not the entire window.
- That is why, the protocol has been named as "**Selective Repeat Protocol**".

Point-08:

SR protocol leads to retransmission of lost frames after expiry of time out timer.

In SR protocol,

- Consider a frame being sent to the receiver is lost on the way.
- Then, it is retransmitted only after time out timer expires for that frame at sender's side.

Efficiency of SR Protocol-

Efficiency of any flow control protocol is given by-

Efficiency = Sender Window Size in Protocol / (1 + 2a)

In selective repeat protocol, if sender window size = N, then-

Efficiency of SR Protocol = N / (1 + 2a)

Comparison Table-

	Stop and Wait ARQ	Go back N	Selective Repeat	Remarks
Efficiency	1 / (1+2a)	N / (1+2a)	N / (1+2a)	Go back N and Selective Repeat gives better efficiency than Stop and Wait ARQ.
Window Size	Sender Window Size = 1 Receiver Window Size = 1	Sender Window Size = N Receiver Window Size = 1	Sender Window Size = N Receiver Window Size = N	Buffer requirement in Selective Repeat is very large. If the system does not have lots of memory, then it is better to choose Go back N.

Minimum number of sequence numbers required	2	N+1	2 x N	Selective Repeat requires large number of bits in sequence number field.
Retransmissions required if a packet is lost	Only the lost packet is retransmitted	The entire window is retransmitted	Only the lost packet is retransmitted	Selective Repeat is far better than Go back N in terms of retransmissions required.

Bandwidth Requirement	Bandwidth requirement is Low	Bandwidth requirement is high because even if a single packet is lost, entire window has to be retransmitted. Thus, if error rate is high, it wastes a lot of bandwidth.	Bandwidth requirement is moderate	Selective Repeat is better than Go back N in terms of bandwidth requirement.
CPU usage	Low	Moderate	High due to searching and sorting required at sender and receiver side	Go back N is better than Selective Repeat in terms of CPU usage.

Level of difficulty in Implementation	Low	Moderate	Complex as it requires extra logic and sorting and searching	Go back N is better than Selective Repeat in terms of implementation difficulty.
Acknowledgements	Uses independent acknowledgement for each packet	Uses cumulative acknowledgements (but may use independent acknowledgements as well)	Uses independent acknowledgement for each packet	Sending cumulative acknowledgements reduces the traffic in the network but if it is lost, then the ACKs for all the corresponding packets are lost.
Type of Transmission	Half duplex	Full duplex	Full duplex	Go back N and Selective Repeat are better in terms of channel usage.

Conclusions-

- Go back N is more often used than other protocols.
- SR protocol is less used because of its complexity.
- Stop and Wait ARQ is less used because of its low efficiency.
- Depending on the context and resources availability, Go back N or Selective Repeat is employed.
- Selective Repeat and Stop and Wait ARQ are similar in terms of retransmissions.
- Go back N and Selective Repeat are similar in terms of efficiency if sender window sizes are same.
- SR protocol may be considered as a combination of advantages of Stop and Wait ARQ and Go back N.
- SR protocol is superior to other protocols but because of its complexity, it is less used.

Important Notes-

Note-01:

Protocols at data link layer like HDLC (Low level protocols) use Go back N.

This is because-

1. Bandwidth is high
2. CPU is very busy doing routing job
3. Error rate is low since out of order packets are not possible in wired medium

Note-02:

Protocols at transport layer like TCP (High level protocols) use selective repeat.

,